Uncertainty Quantification in Machine Learning
From Aleatoric to Epistemic

Eyke Hüllermeier

Institute of Informatics, LMU Munich, Germany

Tutorial at ESSCaSS, Tartu, Estonia, August 15/16, 2022
The early days of AI: knowledge-based systems

**IF:**
(1) the infection is primary_bacteria, and
(2) the site of the culture is one of the sterile sites, and
(3) the suspected portal of entry is the gastrointestinal

**THEN:**
the entity of the organism is bacteroides

---

**IF:**
(1) the infection is primary_bacteria, and
(2) the site of the culture is one of the sterile sites, and
(3) the suspected portal of entry is the gastrointestinal

**THEN:**
there is a suggestive evidence (0.7) that
the entity of the organism is bacteroides

---

**Certainty-factor(h) = Belief(h) − Disbelief(h)**

*confidence, strength of evidence*
A more principled approach: graphical models

probinistic (instead of logic-based) knowledge processing
Contemporary AI: machine learning

TRAINING DATA

MODEL INDUCTION

PREDICTIVE MODEL

{pos, neg}
Uncertainty in machine learning

Machine learning is inseparably connected with uncertainty. Learning in the sense of generalising beyond the data seen so far is necessarily based on a process of induction. Models induced from data are never provably correct, but hypothetical and therefore uncertain, and the same holds true for the predictions produced by a model. Other sources of uncertainty exist: incorrect model assumptions (model misspecification), noisy or imprecise data, etc.

Trustworthy representation of uncertainty is desirable and should be considered as a key feature of any machine learning method, all the more in safety-critical application domains.
Machine learning is inseparably connected with uncertainty.
Uncertainty in machine learning

- Machine learning is inseparably connected with uncertainty.
- **Learning** in the sense of generalising beyond the data seen so far is necessarily based on a process of *induction*.
Uncertainty in machine learning

- Machine learning is inseparably connected with **uncertainty**.
- **Learning** in the sense of generalising beyond the data seen so far is necessarily based on a process of **induction**.
- **Models** induced from data are never provably correct, but hypothetical and therefore uncertain, and the same holds true for the **predictions** produced by a model.
Uncertainty in machine learning

- Machine learning is inseparably connected with **uncertainty**.
- **Learning** in the sense of generalising beyond the data seen so far is necessarily based on a process of **induction**.
- **Models** induced from data are never provably correct, but hypothetical and therefore uncertain, and the same holds true for the **predictions** produced by a model.
- Other sources of uncertainty exist: incorrect **model assumptions** (model misspecification), noisy or imprecise **data**, etc.
Uncertainty in machine learning

- Machine learning is inseparably connected with **uncertainty**.
- **Learning** in the sense of generalising beyond the data seen so far is necessarily based on a process of **induction**.
- **Models** induced from data are never provably correct, but hypothetical and therefore uncertain, and the same holds true for the **predictions** produced by a model.
- Other sources of uncertainty exist: incorrect **model assumptions** (model misspecification), noisy or imprecise **data**, etc.
- **Trustworthy representation** of uncertainty is desirable and should be considered as a key feature of any machine learning method, all the more in safety-critical application domains.
Self-awareness of ML systems

Many applications require safe and reliable predictions, and hence a certain level of self-awareness of ML systems:

▶ equip predictions with an appropriate quantification of uncertainty,
▶ reject a decision in cases of high uncertainty (abstention),
▶ deliver a credible set-valued prediction (partial abstention),
▶ ...
Self-awareness of ML systems

- Many applications require safe and reliable predictions, and hence a certain level of self-awareness of ML systems:
  - equip predictions with an appropriate quantification of uncertainty,
  - reject a decision in cases of high uncertainty (abstention),
  - deliver a credible set-valued prediction (partial abstention),
  - ...

Driver assistance systems: a safety-critical application
### Adversarial examples

<table>
<thead>
<tr>
<th>Negative sentiment</th>
<th>Positive sentiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>There is really but one thing to say about <strong>this</strong> sorry movie It should never have been made The first one one of my favourites An American Werewolf in London is a great movie with a good plot good actors and good FX But this one It stinks to heaven with a cry of helplessness</td>
<td>There is really but one thing to say about <strong>that</strong> sorry movie It should never have been made The first one one of my favourites An American Werewolf in London is a great movie with a good plot good actors and good FX But this one It stinks to heaven with a cry of helplessness</td>
</tr>
</tbody>
</table>
The adversarial example (right) is misclassified by a machine learning model trained on textual data, which changes its prediction due to a change of a single (actually unimportant) word (?).
Lack of uncertainty-awareness
Lack of uncertainty-awareness

- Predictions by EfficientNet on test images from ImageNet: For the left image, the neural network predicts “typewriter keyboard” with certainty 83.14 %, for the right image “stone wall” with certainty 87.63 %.
Levels of self-awareness and uncertainty representation

Probability distributions $p = (p(a), p(b), p(c))$ on $\Omega = \{a, b, c\}$, for example $\Omega = \{\text{home wins, draw, away wins}\}$, as points in a Barycentric coordinate system.
Levels of self-awareness and uncertainty representation

Probability distributions \( p = (p(a), p(b), p(c)) \) on \( \Omega = \{a, b, c\} \), for example \( \Omega = \{\text{home wins, draw, away wins}\} \), as points in a Barycentric coordinate system.
Supervised learning
Supervised learning

- Uncertainty occurs in various facets in machine learning, and different settings and learning problems will usually require a different handling from an uncertainty modeling point of view.
Supervised learning

- Uncertainty occurs in various facets in machine learning, and different settings and learning problems will usually require a different handling from an uncertainty modeling point of view.
- Here, we focus on the standard setting of supervised learning and predictive uncertainty.
Supervised learning

- Uncertainty occurs in various facets in machine learning, and different settings and learning problems will usually require a different handling from an uncertainty modeling point of view.
- Here, we focus on the standard setting of supervised learning and predictive uncertainty.

![Diagram of model induction process]

- Assuming a probabilistic data generating process $p(x, y) = p(x)p(y \mid x)$, probabilistic predictors (estimating $p(y \mid x)$) are natural primitives.
Supervised learning
Supervised learning

A learner is given access to a set of (i.i.d.) training data

\[ D := \{(x_1, y_1), \ldots, (x_N, y_N)\} \subset X \times Y , \]

where \( X \) is an instance space and \( Y \) the set of outcomes.
Supervised learning

- A learner is given access to a set of (i.i.d.) **training data**

  \[ \mathcal{D} := \{(x_1, y_1), \ldots, (x_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}, \]

  where \( \mathcal{X} \) is an instance space and \( \mathcal{Y} \) the set of outcomes.

- Given a **hypothesis space** \( \mathcal{H} \subset \mathcal{Y}^\mathcal{X} \) and a loss function

  \[ \ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}, \]

  the goal of the learner is to induce a hypothesis \( h^* \in \mathcal{H} \) with low **risk**

  \[ R(h) := \mathbb{E}_{(X, Y) \sim P} \ell(h(X), Y) = \int_{\mathcal{X} \times \mathcal{Y}} \ell(h(x), y) \, dP(x, y). \]
Classification and regression

In the case of regression, where the target is numerical ($Y = R$), a common loss is the squared error:

$$\ell(h(x), y) = (h(x) - y)^2$$

In the case of classification, the outcome is categorical ($Y = \{1, \ldots, K\}$), and a common choice is the 0/1 loss:

$$\ell(h(x), y) = \begin{cases} 0 & \text{if } h(x) = y \\ 1 & \text{otherwise} \end{cases}$$

Then, however, a convex, (smooth) upper approximation is typically used as a surrogate for training. If predictions are probabilities $h(x) = \hat{p} = (\hat{p}_1, \ldots, \hat{p}_K) \in \Delta_K$, the loss is defined on $\Delta_K \times Y$; a common example is the logistic loss (log-loss)

$$\ell(\hat{p}, y) = -\log(\hat{p}_y)$$
Classification and regression

In the case of **regression**, where the target is numerical ($\mathcal{Y} = \mathbb{R}$), a common loss is the squared error:

$$\ell(h(x), y) = (h(x) - y)^2$$
Classification and regression

- In the case of **regression**, where the target is numerical ($\mathcal{Y} = \mathbb{R}$), a common loss is the squared error:

  $$\ell(h(x), y) = (h(x) - y)^2$$

- In the case of **classification**, the outcome is categorical ($\mathcal{Y} = [K] := \{1, \ldots, K\}$), and a common choice is the 0/1 loss:

  $$\ell(h(x), y) = \mathbb{1}[h(x) \neq y]$$

Then, however, a convex, (smooth) upper approximation is typically used as a surrogate for training.
Classification and regression

- In the case of **regression**, where the target is numerical ($\mathcal{Y} = \mathbb{R}$), a common loss is the squared error:

  $$\ell(h(\mathbf{x}), y) = (h(\mathbf{x}) - y)^2$$

- In the case of **classification**, the outcome is categorical ($\mathcal{Y} = [K] := \{1, \ldots, K\}$), and a common choice is the 0/1 loss:

  $$\ell(h(\mathbf{x}), y) = \mathbb{I}[h(\mathbf{x}) \neq y]$$

  Then, however, a convex, (smooth) upper approximation is typically used as a surrogate for training.

- If predictions are **probabilities**

  $$h(\mathbf{x}) = \hat{\mathbf{p}} = (\hat{p}_1, \ldots, \hat{p}_K) \in \Delta_K,$$

  the loss is defined on $\Delta_K \times \mathcal{Y}$; a common example is the logistic loss (log-loss)

  $$\ell(\hat{\mathbf{p}}, y) = -\log(p_y).$$
Empirical risk minimisation

The learner's choice of a hypothesis is commonly guided by the empirical risk \( R_{\text{emp}}(h) \)...

Yet, since \( R_{\text{emp}}(h) \) is only an estimation of the true risk \( R(h) \), the (regularised) empirical risk minimiser \( \hat{h} = \arg \min_{h \in H} R_{\text{emp}}(h) + \Omega(h) \) will normally not coincide with the true risk minimiser \( h^* = \arg \min_{h \in H} R(h) \).

Correspondingly, there remains uncertainty regarding \( h^* \), the approximation quality of \( \hat{h} \)(in the sense of its proximity to \( h^* \)) and its true risk \( R(\hat{h}) \), as well as predictive uncertainty about \( \hat{y}_q \) for an individual query instance \( x_q \in X \).
Empirical risk minimisation

The learner’s choice of a hypothesis is commonly guided by the empirical risk

\[ R_{\text{emp}}(h) := \frac{1}{N} \sum_{i=1}^{N} \ell(h(x_i), y_i) \]
Empirical risk minimisation

- The learner’s choice of a hypothesis is commonly guided by the empirical risk

\[ R_{\text{emp}}(h) := \frac{1}{N} \sum_{i=1}^{N} \ell(h(x_i), y_i) . \]

- Yet, since \( R_{\text{emp}}(h) \) is only an estimation of the true risk \( R(h) \), the (regularised) empirical risk minimiser

\[ \hat{h} := \arg \min_{h \in \mathcal{H}} R_{\text{emp}}(h) + \Omega(h) \]

will normally not coincide with the true risk minimiser

\[ h^* := \arg \min_{h \in \mathcal{H}} R(h) . \]
Empirical risk minimisation

The learner’s choice of a hypothesis is commonly guided by the empirical risk

\[ R_{\text{emp}}(h) := \frac{1}{N} \sum_{i=1}^{N} \ell(h(x_i), y_i) . \]

Yet, since \( R_{\text{emp}}(h) \) is only an estimation of the true risk \( R(h) \), the (regularised) empirical risk minimiser

\[ \hat{h} := \arg \min_{h \in \mathcal{H}} R_{\text{emp}}(h) + \Omega(h) \]

will normally not coincide with the true risk minimiser

\[ h^* := \arg \min_{h \in \mathcal{H}} R(h) . \]

Correspondingly, there remains uncertainty regarding \( h^* \), the approximation quality of \( \hat{h} \) (in the sense of its proximity to \( h^* \)) and its true risk \( R(\hat{h}) \), as well as predictive uncertainty about \( \hat{y}_q \) for an individual query instance \( x_q \in \mathcal{X} \).
Problem setting and assumptions

A precise specification of the problem setting and underlying assumptions is an important prerequisite, not only for providing learning guarantees, but also for uncertainty quantification. Here, one might be quite sure about the class of the query under standard assumptions of binary classification, but much less so in a setting of novelty detection, where new classes may emerge. Likewise, assumptions such as i.i.d. data generation are really crucial (the past should be representative of the future).
Problem setting and assumptions

- A precise specification of the **problem setting** and **underlying assumptions** is an important prerequisite, not only for providing learning guarantees, but also for **uncertainty quantification**.
Problem setting and assumptions

- A precise specification of the **problem setting** and **underlying assumptions** is an important prerequisite, not only for providing learning guarantees, but also for **uncertainty quantification**.

- Here, one might be quite sure about the class of the query under standard assumptions of binary classification, but much less so in a setting of **novelty detection**, where new classes may emerge.
Problem setting and assumptions

- A precise specification of the problem setting and underlying assumptions is an important prerequisite, not only for providing learning guarantees, but also for uncertainty quantification.

- Here, one might be quite sure about the class of the query under standard assumptions of binary classification, but much less so in a setting of novelty detection, where new classes may emerge.

- Likewise, assumptions such as i.i.d. data generation are really crucial (the past should be representative of the future).
Sources of uncertainty

A query instance $x_q$ gives rise to a conditional probability on $Y$:

$$p(y|x_q) = \frac{p(x_q, y)}{p(x_q)}$$

Thus, even given full information in the form of the measure $P$ (and its density $p$), uncertainty about the actual outcome $y$ remains.

The best point predictions (minimizing expected loss) are prescribed by the pointwise Bayes predictor $f^*$:

$$f^*(x) = \arg\min_{\hat{y} \in Y} \int_Y \ell(y, \hat{y}) dP(y|x).$$
Sources of uncertainty

A query instance $x_q$ gives rise to a conditional probability on $\mathcal{Y}$:

$$p(y \mid x_q) = \frac{p(x_q, y)}{p(x_q)}$$
Sources of uncertainty

- A query instance $x_q$ gives rise to a conditional probability on $\mathcal{Y}$:

$$p(y | x_q) = \frac{p(x_q, y)}{p(x_q)}$$

- Thus, even given full information in the form of the measure $P$ (and its density $p$), uncertainty about the actual outcome $y$ remains.
Sources of uncertainty

- A query instance $x_q$ gives rise to a conditional probability on $\mathcal{Y}$:

$$p(y \mid x_q) = \frac{p(x_q, y)}{p(x_q)}$$

- Thus, even given full information in the form of the measure $P$ (and its density $p$), uncertainty about the actual outcome $y$ remains.

- The best point predictions (minimizing expected loss) are prescribed by the pointwise Bayes predictor $f^*$:

$$f^*(x) := \arg \min_{\hat{y} \in \mathcal{Y}} \int_{\mathcal{Y}} \ell(y, \hat{y}) \, dP(y \mid x).$$
Example

Let $X = \mathbb{R}$, $Y = \{-1 + 1\}$, suppose positive and negative instances distributed according to $\mathcal{N}(180, 10)$ and $\mathcal{N}(160, 10)$, respectively. Let $H = \{h_t | t \in \mathbb{R}\}$ with $h_t(x) = \begin{cases} -1 & \text{if } x < t + 1 \\ +1 & \text{if } x \geq t \end{cases}$.

If loss is 0/1, then the (pointwise) Bayes predictor is given by $f^* (x) = h^* (x) = h_{170}$. 

17 / 128
Example

Let $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \{-1 + 1\}$, suppose positive and negative instances distributed according to $N(180, 10)$ and $N(160, 10)$, respectively. Let $\mathcal{H} = \{h_t | t \in \mathbb{R}\}$ with

$$h_t(x) = \begin{cases} 
-1 & \text{if } x < t \\
1 & \text{if } x \geq t 
\end{cases}$$

If loss is 0/1, then the (pointwise) Bayes predictor is given by $f^* (x) = h^* (x) = h_{170}$.17 / 128
Example

Let $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \{-1 + 1\}$, suppose positive and negative instances distributed according to $N(180, 10)$ and $N(160, 10)$, respectively. Let $\mathcal{H} = \{h_t \mid t \in \mathbb{R}\}$ with

$$h_t(x) = \begin{cases} -1 & \text{if } x < t \\ +1 & \text{if } x \geq t \end{cases}$$

If loss is 0/1, then the (pointwise) Bayes predictor is given by

$$f^*(x) = h^*(x) = h_{170}.$$
Sources of uncertainty

The Bayes predictor $h^\ast$ does not necessarily coincide with the pointwise Bayes predictor. This discrepancy between $h^\ast$ and $f^\ast$ is connected to the uncertainty regarding the right type of model to be fit, and hence the choice of the hypothesis space $H$. We shall refer to this uncertainty as model uncertainty.

Due to model uncertainty, one cannot guarantee $h^\ast(x) = f^\ast(x)$, or, in the case of probabilistic predictions $p^\ast(y|x) = p(y|x, h^\ast)$, that $p^\ast(\cdot|x) = p(\cdot|x)$. 


Sources of uncertainty

- The **Bayes predictor** $h^*$ does not necessarily coincide with the pointwise Bayes predictor.
Sources of uncertainty

- The **Bayes predictor** $h^*$ does not necessarily coincide with the pointwise Bayes predictor.

- This discrepancy between $h^*$ and $f^*$ is connected to the uncertainty regarding the **right type of model** to be fit, and hence the choice of the hypothesis space $\mathcal{H}$. 
Sources of uncertainty

- The **Bayes predictor** $h^*$ does not necessarily coincide with the pointwise Bayes predictor.

- This discrepancy between $h^*$ and $f^*$ is connected to the uncertainty regarding the **right type of model** to be fit, and hence the choice of the hypothesis space $\mathcal{H}$.

- We shall refer to this uncertainty as **model uncertainty**.
Sources of uncertainty

- The **Bayes predictor** $h^*$ does not necessarily coincide with the pointwise Bayes predictor.

- This discrepancy between $h^*$ and $f^*$ is connected to the uncertainty regarding the **right type of model** to be fit, and hence the choice of the hypothesis space $\mathcal{H}$.

- We shall refer to this uncertainty as **model uncertainty**.

- Due to model uncertainty, one cannot guarantee

$$h^*(x) = f^*(x),$$

or, in the case of probabilistic predictions $p^*(y \mid x) := p(y \mid x, h^*)$, that

$$p^*(\cdot \mid x) = p(\cdot \mid x).$$
Example

Now, suppose positive and negative instances distributed according to $N(180, 15)$ and $N(170, 8)$, respectively. If $H = \{ h_t \mid t \in \mathbb{R} \}$, then $f^* \not\in H$ and $h^* \neq f^*$. 
Now, suppose positive and negative instances distributed according to $N(180, 15)$ and $N(170, 8)$, respectively.
Example

- Now, suppose positive and negative instances distributed according to $N(180, 15)$ and $N(170, 8)$, respectively.
- If $\mathcal{H} = \{ h_t \mid t \in \mathbb{R} \}$, then $f^* \not\in \mathcal{H}$, and $h^* \neq f^*$. 
Sources of uncertainty

Hypothesis \( \hat{h} \) produced by the learner is an estimate of \( h^* \). The quality of this estimate strongly depends on the quality and the amount of training data. We refer to the uncertainty about the discrepancy between \( \hat{h} \) and \( h^* \) as approximation uncertainty.
Sources of uncertainty

- Hypothesis $\hat{h}$ produced by the learner is an estimate of $h^*$. 
Sources of uncertainty

- Hypothesis $\hat{h}$ produced by the learner is an estimate of $h^*$. 
- The quality of this estimate strongly depends on the quality and the amount of training data.
Sources of uncertainty

- Hypothesis $\hat{h}$ produced by the learner is an estimate of $h^*$.
- The quality of this estimate strongly depends on the quality and the amount of training data.

We refer to the uncertainty about the discrepancy between $\hat{h}$ and $h^*$ as approximation uncertainty.
Example

In our example, suppose the learner checks as candidate thresholds all points half-way between two observations, and picks the one with the lowest number of errors on the training data.
Example

In our example, suppose the learner checks as candidate thresholds all points half-way between two observations, and picks the one with the lowest number of errors on the training data.
Sources of uncertainty

\[ \mathcal{F} = \mathcal{Y}^\mathcal{X} \]

**Point prediction probability**

- **Ground truth**
  - \( f^*(\mathbf{x}) \)
  - \( p(\cdot|\mathbf{x}) \)

- **Best possible**
  - \( h^*(\mathbf{x}) \)
  - \( p^*(\cdot|\mathbf{x}) = p(\cdot|\mathbf{x}, h^*) \)

- **Induced predictor**
  - \( \hat{h}(\mathbf{x}) \)
  - \( \hat{p}(\cdot|\mathbf{x}) = p(\cdot|\mathbf{x}, \hat{h}) \)

**Hypothesis space**

\[ \mathcal{H} \subset \mathcal{F} \]

**Model uncertainty**

**Approximation uncertainty**
Sources of uncertainty

A more fine-granular distinction is possible; for example, a deviation between $\hat{h}$ and $h^*$ might not only be caused by the data but also by the algorithm.
Sources of uncertainty

A more fine-granular distinction is possible; for example, a deviation between \( \hat{h} \) and \( h^* \) might not only be caused by the data but also by the algorithm.
Agenda

1. Introduction
2. **Probabilistic prediction**
3. Calibration
4. Set-valued (conformal) prediction
5. Epistemic uncertainty
6. Summary and outlook
Probabilistic models
Probabilistic models

- Probabilistic learners produce a (single) probabilistic predictor \( \hat{h} \):

\[
\mathcal{F} = \mathcal{Y}^x
\]

\( \hat{h} \)

hypothesis space
\( \mathcal{H} \subseteq \mathcal{F} \)

Captures stochastic nature of dependence \( p(y | x) \) between instances and outcomes, and hence aleatoric uncertainty. Yet, pretends full certainty about this dependence, thereby ignoring approximation and model uncertainty (epistemic uncertainty).
Probabilistic models

- Probabilistic learners produce a (single) probabilistic predictor $\hat{h}$:

  $F = \mathcal{Y}^X$

  $\hat{h}$

  hypothesis space $\mathcal{H} \subset F$

- Captures stochastic nature of dependence $p(y \mid x)$ between instances and outcomes, and hence **aleatoric uncertainty**.
Probabilistic models

- Probabilistic learners produce a (single) probabilistic predictor $\hat{h}$:

  $F = \mathcal{Y}^X$

  hypothesis space $\mathcal{H} \subset F$

  Captures stochastic nature of dependence $p(y \mid x)$ between instances and outcomes, and hence **aleatoric uncertainty**.

  Yet, pretends full certainty about this dependence, thereby ignoring approximation and model uncertainty (epistemic uncertainty).
Extension of simple ML methods

Many ML methods for classification and regression, such as decision trees, can be extended (in an ad-hoc manner) to return (pseudo-)probabilities instead of just point predictions (class-labels).
Extension of simple ML methods

Many ML methods for classification and regression, such as decision trees, can be extended (in an ad-hoc manner) to return (pseudo-)probabilities instead of just point predictions (class-labels).
Extension of simple ML methods

Many ML methods for classification and regression, such as decision trees, can be extended (in an ad-hoc manner) to return (pseudo-)probabilities instead of just point predictions (class-labels).
Extension of simple ML methods

Many ML methods for classification and regression, such as decision trees, can be extended (in an ad-hoc manner) to return (pseudo-)probabilities instead of just point predictions (class-labels).
Agenda

1. Introduction
2. **Probabilistic prediction**
   - Ad-hoc methods
   - Maximum likelihood
3. Calibration
4. Set-valued (conformal) prediction
5. Epistemic uncertainty
6. Summary and outlook
Maximum likelihood

Suppose that hypotheses \( h \) are (uniquely) identified by parameters \( \theta \in \Theta \), i.e.,
\[
H = \{ h_\theta | \theta \in \Theta \}.
\]

Learning (model induction) then comes down to parameter estimation. The maximum likelihood (ML) principle suggests to pick the parameter with the highest likelihood:
\[
\hat{\theta} = \arg \max_{\theta \in \Theta} L(\theta) = \arg \max_{\theta \in \Theta} P(D|\theta).
\]

In general, ML estimation has appealing statistical properties. Under the i.i.d. assumption, ML estimation comes down to solving
\[
\hat{\theta} = \arg \max_{\theta \in \Theta} N \prod_{i=1}^{N} P((x_i, y_i)|\theta) = \arg \max_{\theta \in \Theta} N \sum_{i=1}^{N} \log P((x_i, y_i)|\theta).
\]
Maximum likelihood

- Suppose that hypotheses $h$ are (uniquely) identified by parameters $\theta \in \Theta$, i.e.,

$$\mathcal{H} = \{h_\theta \mid \theta \in \Theta\}.$$

Learning (model induction) then comes down to parameter estimation.
Maximum likelihood

- Suppose that hypotheses $h$ are (uniquely) identified by parameters $\theta \in \Theta$, i.e.,

$$\mathcal{H} = \{ h_{\theta} \mid \theta \in \Theta \}.$$

Learning (model induction) then comes down to **parameter estimation**.

- The **maximum likelihood** (ML) principle suggests to pick the parameter with the highest likelihood:

$$\hat{\theta} = \arg \max_{\theta \in \Theta} L(\theta) = \arg \max_{\theta \in \Theta} P(D \mid \theta)$$

In general, ML estimation has appealing statistical properties. Under the i.i.d. assumption, ML estimation comes down to solving

$$\hat{\theta} = \arg \max_{\theta \in \Theta} N \prod_{i=1}^{N} P((x_i, y_i) \mid \theta) = \arg \max_{\theta \in \Theta} N \sum_{i=1}^{N} \log P((x_i, y_i) \mid \theta).$$
Maximum likelihood

- Suppose that hypotheses $h$ are (uniquely) identified by parameters $\theta \in \Theta$, i.e.,

$$\mathcal{H} = \{h_\theta \mid \theta \in \Theta\}.$$

Learning (model induction) then comes down to parameter estimation.

- The maximum likelihood (ML) principle suggests to pick the parameter with the highest likelihood:

$$\hat{\theta} = \arg \max_{\theta \in \Theta} L(\theta) = \arg \max_{\theta \in \Theta} P(D \mid \theta)$$

- In general, ML estimation has appealing statistical properties.
Maximum likelihood

- Suppose that hypotheses $h$ are (uniquely) identified by parameters $\theta \in \Theta$, i.e.,

$$
\mathcal{H} = \{ h_\theta \mid \theta \in \Theta \}.
$$

Learning (model induction) then comes down to **parameter estimation**.

- The **maximum likelihood** (ML) principle suggests to pick the parameter with the highest likelihood:

$$
\hat{\theta} = \arg \max_{\theta \in \Theta} L(\theta) = \arg \max_{\theta \in \Theta} P(\mathcal{D} \mid \theta)
$$

- In general, ML estimation has appealing statistical properties.

- Under the i.i.d. assumption, ML estimation comes down to solving

$$
\hat{\theta} = \arg \max_{\theta \in \Theta} \prod_{i=1}^{N} P((x_i, y_i) \mid \theta) = \arg \max_{\theta \in \Theta} \sum_{i=1}^{N} \log P((x_i, y_i) \mid \theta).
$$
Fisher information

The distribution of the ML estimate $\hat{\theta}$ is known to be asymptotically normal: 
\[ \sqrt{N}(\hat{\theta} - \theta) \] converges to a normal with mean 0 and covariance matrix $I_{N}(\theta)$, where $I_{N}(\theta) = \left[-E_{\theta}(\partial^{2}\ell_{N}/\partial\theta_{i}\partial\theta_{j})\right]_{1 \leq i, j \leq N}$ is the Fisher information matrix (negative Hessian of log-likelihood at $\theta$).

The Fisher information matrix allows for constructing (approximate) confidence regions for $\theta$ around the estimate $\hat{\theta}$. Obviously, the larger this region, the higher the (approximation) uncertainty about the true model $\theta$.

For a quantitative measure, the Fisher information matrix can be summarised in a scalar statistic, for example the trace (of the inverse) or the smallest eigenvalue.
Fisher information

- The distribution of the ML estimate $\hat{\theta}$ is known to be asymptotically normal: $
  \sqrt{N}(\hat{\theta} - \theta)$ converges to a normal with mean 0 and covariance matrix $I_N^{-1}(\theta)$, where

$$I_N(\theta) = -\left[\mathbb{E}_\theta \left( \frac{\partial^2 \ell_N}{\partial \theta_i \partial \theta_j} \right) \right]_{1 \leq i, j \leq N}$$

is the **Fisher information matrix** (negative Hessian of log-likelihood at $\theta$).
Fisher information

- The distribution of the ML estimate \( \hat{\theta} \) is known to be asymptotically normal: 
  \( \sqrt{N}(\hat{\theta} - \theta) \) converges to a normal with mean 0 and covariance matrix \( I_N^{-1}(\theta) \), where

  \[
  I_N(\theta) = - \left[ E_\theta \left( \frac{\partial^2 \ell_N}{\partial \theta_i \partial \theta_j} \right) \right]_{1 \leq i, j \leq N}
  \]

  is the **Fisher information matrix** (negative Hessian of log-likelihood at \( \theta \)).

- The Fisher information matrix allows for constructing (approximate) **confidence regions** for \( \theta \) around the estimate \( \hat{\theta} \).
Fisher information

- The distribution of the ML estimate $\hat{\theta}$ is known to be asymptotically normal: $\sqrt{N}(\hat{\theta} - \theta)$ converges to a normal with mean 0 and covariance matrix $I_N^{-1}(\theta)$, where

$$I_N(\theta) = -\left[ \mathbb{E}_\theta \left( \frac{\partial^2 \ell_N}{\partial \theta_i \partial \theta_j} \right) \right]_{1 \leq i, j \leq N}$$

is the Fishes information matrix (negative Hessian of log-likelihood at $\theta$).

- The Fisher information matrix allows for constructing (approximate) confidence regions for $\theta$ around the estimate $\hat{\theta}$.

- Obviously, the larger this region, the higher the (approximation) uncertainty about the true model $\theta$. 

Fisher information

- The distribution of the ML estimate $\hat{\theta}$ is known to be asymptotically normal: $\sqrt{N}(\hat{\theta} - \theta)$ converges to a normal with mean 0 and covariance matrix $I_N^{-1}(\theta)$, where

$$I_N(\theta) = -\left[ \mathbb{E}_\theta \left( \frac{\partial^2 \ell_N}{\partial \theta_i \partial \theta_j} \right) \right]_{1 \leq i, j \leq N}$$

is the **Fisher information matrix** (negative Hessian of log-likelihood at $\theta$).

- The Fisher information matrix allows for constructing (approximate) **confidence regions** for $\theta$ around the estimate $\hat{\theta}$.

- Obviously, the larger this region, the higher the (approximation) uncertainty about the true model $\theta$.

- For a quantitative measure, the Fisher information matrix can be summarised in a scalar statistic, for example the trace (of the inverse) or the smallest eigenvalue.
Neural nets and ML estimation

\[ y \in (0, 1) \]
Neural nets and ML estimation

\[ f(w_1 x_1 + w_2 x_2 + w_3 x_3) \]

\[ s = w_1 x_1 + w_2 x_2 + w_3 x_3 \]
Logistic regression

In order to learn $\theta$, we can invoke the ML principle:

$$\hat{\theta} = \arg \max_{\theta \in \mathbb{R}^d} \sum_{n=1}^{N} \log p(y_n | x_n, \theta)$$

This is equivalent to empirical risk minimisation for the logistic loss (log-loss) or cross-entropy error $\ell(s, y) = \log (1 + \exp(-2y-1) \cdot s)$. 

33 / 128
Logistic regression

In order to learn $\theta$, we can invoke the ML principle:

$$
\hat{\theta} = \arg \max_{\theta \in \mathbb{R}^d} \sum_{n=1}^{N} \log p(y_n | x_n, \theta)
$$
Logistic regression

- In order to learn $\theta$, we can invoke the ML principle:

$$
\hat{\theta} = \arg \max_{\theta \in \mathbb{R}^d} \sum_{n=1}^{N} \log p(y_n | x_n, \theta)
$$

- This is equivalent to empirical risk minimisation for the logistic loss (log-loss) or cross-entropy error

$$
\ell(s, y) = \log \left(1 + \exp \left(-(2y - 1) \cdot s\right)\right).
$$
Agenda

1. Introduction
2. Probabilistic prediction
   ▶ Ad-hoc methods
   ▶ Maximum likelihood
   ▶ Proper scoring rules
3. Calibration
4. Set-valued (conformal) prediction
5. Epistemic uncertainty
6. Summary and outlook
Proper scoring rules

The logistic (cross-entropy) loss is an example of a proper scoring rule. Consider a probabilistic prediction (for a query $x$) on a set of $K$ classes $Y = \{1, \ldots, K\}$ in the form of a probability vector $\hat{p} = (\hat{p}_1, \ldots, \hat{p}_K) \in \Delta^K$. The true distribution (conditioned on $x$) is $p = (p_1, \ldots, p_K)$, i.e., the observed class $Y$ is a random variable $Y \sim p$ with multinomial distribution. We also encode a realisation $y$ of $Y$ in terms of a vector $y$ with entry 1 on position $y = k$ (i.e., the $k$th outcome was observed), and all other entries 0. Expressed in terms of predicted probability, the log-loss can then also be written as $\ell(\hat{p}, y) = -\log \hat{p}_y$. 

35 / 128
Proper scoring rules

- The logistic (cross-entropy) loss is an example of a proper scoring rule.
Proper scoring rules

- The logistic (cross-entropy) loss is an example of a proper scoring rule.
- Consider a probabilistic prediction (for a query $x$) on a set of $K$ classes $\mathcal{Y} = [K]$ in the form of a probability vector

$$\hat{p} = (\hat{p}_1, \ldots, \hat{p}_K) \in \Delta_K.$$
Proper scoring rules

- The logistic (cross-entropy) loss is an example of a proper scoring rule.
- Consider a probabilistic prediction (for a query $x$) on a set of $K$ classes $\mathcal{Y} = [K]$ in the form of a probability vector

$$\hat{p} = (\hat{p}_1, \ldots, \hat{p}_K) \in \Delta_K.$$

- The true distribution (conditioned on $x$) is $p = (p_1, \ldots, p_K)$, i.e., the observed class $Y$ is a random variable $Y \sim p$ with multinomial distribution.
Proper scoring rules

- The logistic (cross-entropy) loss is an example of a **proper scoring rule**.
- Consider a probabilistic prediction (for a query $x$) on a set of $K$ classes $\mathcal{Y} = [K]$ in the form of a probability vector

  \[ \hat{p} = (\hat{p}_1, \ldots, \hat{p}_K) \in \Delta_K. \]

- The true distribution (conditioned on $x$) is $p = (p_1, \ldots, p_K)$, i.e., the observed class $Y$ is a random variable $Y \sim p$ with multinomial distribution.
- We also encode a realisation $y$ of $Y$ in terms of a vector $y$ with entry 1 on position $y = k$ (i.e., the $k^{th}$ outcome was observed), and all other entries 0.
Proper scoring rules

- The logistic (cross-entropy) loss is an example of a proper scoring rule.
- Consider a probabilistic prediction (for a query $x$) on a set of $K$ classes $\mathcal{Y} = [K]$ in the form of a probability vector

$$\hat{p} = (\hat{p}_1, \ldots, \hat{p}_K) \in \Delta_K.$$ 

- The true distribution (conditioned on $x$) is $p = (p_1, \ldots, p_K)$, i.e., the observed class $Y$ is a random variable $Y \sim p$ with multinomial distribution.
- We also encode a realisation $y$ of $Y$ in terms of a vector $y$ with entry 1 on position $y = k$ (i.e., the $k^{th}$ outcome was observed), and all other entries 0.
- Expressed in terms of predicted probability, the log-loss can then also be written as

$$\ell(\hat{p}, y) = -\log \hat{p}_y.$$
Proper scoring rules

A loss $\ell$ is a proper scoring rule if the expected loss minimiser coincides with the true probability $p$:

$$p = \arg \min \hat{p} \mathbb{E}_{Y \sim p} \ell(\hat{p}, Y).$$

A scoring rule is strictly proper if the minimiser is unique. Thus, given a query $x$, a learner penalised by a (strictly) proper scoring rule has an incentive to predict the true (conditional) probability $p = p(Y|x)$.

For example, the Brier score $\sum_k (\hat{p}_k - y_k)^2$ is strictly proper, because

$$K \sum_k = 1 p_k \left[ (1 - \hat{p}_k)^2 + \sum_{j \neq k} \hat{p}_j^2 \right] = \sum_k (\hat{p}_k)^2 + \sum_k p_k (1 - 2 \hat{p}_k)$$

is minimised by $\hat{p}_k = p_k$ for all $k \in [K]$. 


Proper scoring rules

A loss \( \ell \) is a proper scoring rule if the expected loss minimiser coincides with the true probability \( p \):

\[
    p = \operatorname{arg\ min}_\hat{p} \mathbb{E}_{Y \sim p} \ell(\hat{p}, Y)
\]
Proper scoring rules

- A loss $\ell$ is a proper scoring rule if the expected loss minimiser coincides with the true probability $p$:

$$p = \arg\min_{\hat{p}} \mathbb{E}_{Y \sim p} \ell(\hat{p}, Y)$$

- A scoring rule is **strictly proper** if the minimiser is unique.
Proper scoring rules

- A loss $\ell$ is a proper scoring rule if the expected loss minimiser coincides with the true probability $p$:

  $$p = \arg \min_{\hat{p}} \mathbb{E}_{Y \sim p} \ell(\hat{p}, Y)$$

- A scoring rule is **strictly proper** if the minimiser is unique.

- Thus, given a query $x$, a learner penalised by a (strictly) proper scoring rule has an incentive to predict the true (conditional) probability $p = p(Y \mid x)$. 
Proper scoring rules

A loss $\ell$ is a proper scoring rule if the expected loss minimiser coincides with the true probability $p$:

$$p = \arg\min_{\hat{p}} \mathbb{E}_{Y \sim p} \ell(\hat{p}, Y)$$

A scoring rule is **strictly proper** if the minimiser is unique.

Thus, given a query $x$, a learner penalised by a (strictly) proper scoring rule has an incentive to predict the true (conditional) probability $p = p(Y \mid x)$.

For example, the **Brier score** $\sum_k (\hat{p}_k - y_k)^2$ is strictly proper, because

$$\sum_{k=1}^{K} p_k \cdot \left[ (1 - \hat{p}_k)^2 + \sum_{j \neq k} (\hat{p}_j)^2 \right] = \sum_k (\hat{p}_k)^2 + \sum_k p_k (1 - 2\hat{p}_k)$$

is minimised by $\hat{p}_k = p_k$ for all $k \in [K]$. 
Proper scoring rules

Define the scoring function on probability vectors $\hat{p}$ and $p$ as $S(\hat{p}, p) = \mathbb{E}_{Y \sim p} S(\hat{p}, Y) = \sum_k S(\hat{p}, y_k) p_k$, i.e., as the expected score (under ground-truth $p$).

A scoring rule is proper if the divergence $d(\hat{p}, p) = S(\hat{p}, p) - S(p, p)$ is nonnegative, and strictly proper if $d(\hat{p}, p) = 0$ implies $\hat{p} = p$.

e($p$) is also called entropy.
Proper scoring rules

Define the scoring function on probability vectors $\hat{p}, p$ as

$$S(\hat{p}, p) := \mathbb{E}_{Y \sim p} S(\hat{p}, Y) = \sum_k S(\hat{p}, y) p_k,$$

i.e., as the expected score (under ground-truth $p$).
Proper scoring rules

- Define the **scoring function** on probability vectors \( \hat{p}, p \) as

\[
S(\hat{p}, p) := \mathbb{E}_{Y \sim p} S(\hat{p}, Y) = \sum_k S(\hat{p}, y) p_k,
\]

i.e., as the expected score (under ground-truth \( p \)).

- A scoring rule is proper if the **divergence**

\[
d(\hat{p}, p) := S(\hat{p}, p) - S(p, p)
\]

is nonnegative, and strictly proper if \( d(\hat{p}, p) = 0 \) implies \( \hat{p} = p \).
Proper scoring rules

- Define the **scoring function** on probability vectors $\hat{p}, p$ as

$$S(\hat{p}, p) := \mathbb{E}_{Y \sim p} S(\hat{p}, Y) = \sum_k S(\hat{p}, y) p_k,$$

i.e., as the expected score (under ground-truth $p$).

- A scoring rule is proper if the **divergence**

$$d(\hat{p}, p) := S(\hat{p}, p) - S(p, p)$$

is nonnegative, and strictly proper if $d(\hat{p}, p) = 0$ implies $\hat{p} = p$.

- $e(p) := S(p, p)$ is also called **entropy**.
Proper scoring rules

For the log-loss, we obtain a decomposition into KL-divergence and information (Shannon) entropy:

$$d(\hat{p}, p) = D_{KL}(p \parallel \hat{p}) = \sum_k p_k \cdot \log(p_K \hat{p}_k)$$

$$e(p) = -\sum_k p_k \cdot \log(p_k)$$

For the Brier score, we obtain a decomposition into mean squared difference and Gini index:

$$d(\hat{p}, p) = \sum_k (\hat{p}_k - p_k)^2$$

$$e(p) = \sum_k p_k (1 - p_k)$$

The 0/1 loss is a proper but not a strictly proper scoring rule.
Proper scoring rules

For the **log-loss**, we obtain a decomposition into KL-divergence and information (Shannon) entropy:

\[
d(\hat{p}, p) = D_{KL}(p \parallel \hat{p}) = \sum_k p_k \cdot \log \left( \frac{p_k}{\hat{p}_k} \right)
\]

\[
e(p) = -\sum_k p_k \cdot \log(p_k)
\]
Proper scoring rules

- For the **log-loss**, we obtain a decomposition into KL-divergence and information (Shannon) entropy:

\[
d(\hat{p}, p) = D_{KL}(p \| \hat{p}) = \sum_k p_k \cdot \log \left( \frac{p_k}{\hat{p}_k} \right)
\]

\[
e(p) = -\sum_k p_k \cdot \log(p_k)
\]

- For the **Brier score**, we obtain a decomposition into mean squared difference and Gini index:

\[
d(\hat{p}, p) = \sum_k (\hat{p}_k - p_k)^2
\]

\[
e(p) = \sum_k p_k(1 - p_k)
\]
Proper scoring rules

- For the **log-loss**, we obtain a decomposition into KL-divergence and information (Shannon) entropy:

  \[
  d(\hat{p}, p) = D_{KL}(p \| \hat{p}) = \sum_k p_k \cdot \log \left( \frac{p_k}{\hat{p}_k} \right)
  \]

  \[
  e(p) = -\sum_k p_k \cdot \log(p_k)
  \]

- For the **Brier score**, we obtain a decomposition into mean squared difference and Gini index:

  \[
  d(\hat{p}, p) = \sum_k (\hat{p}_k - p_k)^2
  \]

  \[
  e(p) = \sum_k p_k(1 - p_k)
  \]

- The **0/1 loss** is a proper but not a strictly proper scoring rule.
Proper scoring rules

The expectation of a strictly proper score can be decomposed as follows:

$$
E_{Y \sim p} S(\hat{p}, Y) = d(\hat{p}, p) + S(p, p),
$$

where the entropy $S(p, p)$ is the unavoidable part of the loss (due to the need to predict the realisation of a random variable) and $d(\hat{p}, p)$ the extra loss.

Another decomposition is

$$
E_{YS} S(\hat{p}, Y) = E_{X \times Y} d(\hat{p}, c) + E_{X \times Y} e(c),
$$

where the expectation is taken over $X \times Y$, and $c_k = p(y = k | \hat{p})$, i.e., $c$ is the true class distribution on those instances that receive the same prediction $\hat{p}$.

$E_d(\hat{p}, c)$ is called calibration loss and $E_{e}(c)$ refinement loss.
Proper scoring rules

- The expectation of a strictly proper score can be decomposed as follows:

\[ \mathbb{E}_{Y \sim p} S(\hat{p}, Y) = S(\hat{p}, p) = d(\hat{p}, p) + S(p, p), \]

where the **entropy** \( S(p, p) \) is the unavoidable part of the loss (due to the need to predict the realisation of a random variable) and \( d(\hat{p}, p) \) the extra loss.
Proper scoring rules

- The expectation of a strictly proper score can be decomposed as follows:

\[
\mathbb{E}_{Y \sim p} S(\hat{p}, Y) = S(\hat{p}, p) = d(\hat{p}, p) + S(p, p),
\]

where the entropy \( S(p, p) \) is the unavoidable part of the loss (due to the need to predict the realisation of a random variable) and \( d(\hat{p}, p) \) the extra loss.

- Another decomposition is

\[
\mathbb{E} S(\hat{p}, Y) = \mathbb{E} d(\hat{p}, c) + \mathbb{E} e(c),
\]

where the expectation is taken over \( \mathcal{X} \times \mathcal{Y} \), and \( c_k := p(y = k | \hat{p}) \), i.e., \( c \) is the true class distribution on those instances that receive the same prediction \( \hat{p} \).
Proper scoring rules

The expectation of a strictly proper score can be decomposed as follows:

\[ \mathbb{E}_{Y \sim p} S(\hat{p}, Y) = S(\hat{p}, p) = d(\hat{p}, p) + S(p, p), \]

where the **entropy** \( S(p, p) \) is the unavoidable part of the loss (due to the need to predict the realisation of a random variable) and \( d(\hat{p}, p) \) the extra loss.

Another decomposition is

\[ \mathbb{E} S(\hat{p}, Y) = \mathbb{E} d(\hat{p}, c) + \mathbb{E} e(c), \]

where the expectation is taken over \( \mathcal{X} \times \mathcal{Y} \), and \( c_k := p(y = k \mid \hat{p}) \), i.e., \( c \) is the true class distribution on those instances that receive the same prediction \( \hat{p} \).

\( \mathbb{E} d(\hat{p}, c) \) is called **calibration loss** and \( \mathbb{E} e(c) \) **refinement loss**.
Agenda

1. Introduction
2. **Probabilistic prediction**
   - Ad-hoc methods
   - Maximum likelihood
   - Proper scoring rules
   - Bayesian learning
3. Calibration
4. Set-valued (conformal) prediction
5. Epistemic uncertainty
6. Summary and outlook
Bayesian learning

Bayesian machine learning essentially means applying the Bayesian paradigm of statistical inference to machine learning problems. Learning is based on the following instantiation of Bayes' rule, which turns a prior distribution on $H$ into a posterior in the light of observed training data:

$$p(h|D) = \frac{p(D|h)p(h)}{p(D)} \propto p(D|h)p(h)$$

The predictive posterior distribution on $Y$ is obtained via model averaging:

$$p(y|x_q) = \int_{h \in H} p(y|x_q, h) \, dP(h|D)$$
Bayesian learning

- **Bayesian machine learning** essentially means applying the Bayesian paradigm of statistical inference to machine learning problems.

\[
p(h|D) = \frac{p(D|h)p(h)}{p(D)} \propto p(D|h)p(h)
\]

The predictive posterior distribution on \(Y\) is obtained via model averaging:

\[
p(y|x_q) = \int_{h \in H} p(y|x_q,h) \, d \mathcal{P}(h|D)
\]
Bayesian learning

- **Bayesian machine learning** essentially means applying the Bayesian paradigm of statistical inference to machine learning problems.

- Learning is based on the following instantiation of [Bayes’ rule](https://en.wikipedia.org/wiki/Bayes%27_rule), which turns a prior distribution on $\mathcal{H}$ into a posterior in the light of observed training data:

$$p(h \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid h) p(h)}{p(\mathcal{D})} \propto p(\mathcal{D} \mid h) p(h)$$
Bayesian learning

- **Bayesian machine learning** essentially means applying the Bayesian paradigm of statistical inference to machine learning problems.

- Learning is based on the following instantiation of Bayes’ rule, which turns a prior distribution on $\mathcal{H}$ into a **posterior** in the light of observed training data:

\[ p(h \mid D) = \frac{p(D \mid h) p(h)}{p(D)} \propto p(D \mid h) p(h) \]

- The **predictive posterior** distribution on $\mathcal{Y}$ is obtained via **model averaging**:

\[ p(y \mid x_q) = \int_{h \in \mathcal{H}} p(y \mid x_q, h) \, dP(h \mid D) \]
Bayesian learning

Bayesian inference is very costly but can be done approximately, for example, using ensemble methods (turning the integral over $H$ into a sum) or methods such as Laplace approximation, expectation propagation, or MCMC. An alternative is to avoid averaging and commit to the hypothesis with maximum a-posteriori probability (MAP inference):

$$h_{\text{MAP}} = \arg \max_{h \in H} p(h | D).$$

There are a few exceptions where Bayesian inference is tractable and updating can be accomplished analytically (mainly due to nice properties of the Gaussian family of distributions) — one example is Gaussian processes.
Bayesian learning

- Bayesian inference is very costly but can be done approximately, for example, using ensemble methods (turning the integral over $\mathcal{H}$ into a sum) or methods such as Laplace approximation, expectation propagation, or MCMC.
Bayesian learning

- Bayesian inference is very costly but can be done approximately, for example, using ensemble methods (turning the integral over $\mathcal{H}$ into a sum) or methods such as Laplace approximation, expectation propagation, or MCMC.

- An alternative is to avoid averaging and commit to the hypothesis with maximum a-posteriori probability (MAP inference):

$$h_{\text{MAP}} = \arg \max_{h \in \mathcal{H}} p(h | \mathcal{D}).$$
Bayesian learning

- Bayesian inference is very costly but can be done approximately, for example, using ensemble methods (turning the integral over $\mathcal{H}$ into a sum) or methods such as Laplace approximation, expectation propagation, or MCMC.

- An alternative is to avoid averaging and commit to the hypothesis with maximum a-posteriori probability (MAP inference):

$$h_{MAP} = \arg \max_{h \in \mathcal{H}} p(h \mid D).$$

- There are a few exceptions where Bayesian inference is tractable and updating can be accomplished analytically (mainly due to nice properties of the Gaussian family of distributions)—one example is Gaussian processes.
Gaussian processes extend multivariate Gaussians to infinite-sized collections of real-valued variables. They can be thought of as distributions not just over random vectors but over random functions. Note that, for a finite set \( X = \{x_1, \ldots, x_m\} \), each function \( f: X \rightarrow \mathbb{R} \) can be identified by the vector \( \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_m) \end{bmatrix}^\top \). Hence, probability distributions over functions with finite domains can be represented using a finite-dimensional multivariate Gaussian distribution.
Gaussian processes

- Gaussian processes extend multivariate Gaussians to infinite-sized collections of real-valued variables.
Gaussian processes

- Gaussian processes extend multivariate Gaussians to **infinite-sized collections** of real-valued variables.

- They can be thought of as distributions not just over random vectors but over **random functions**.
Gaussian processes

- Gaussian processes extend multivariate Gaussians to infinite-sized collections of real-valued variables.

- They can be thought of as distributions not just over random vectors but over random functions.

- Note that, for a finite set \( \mathcal{X} = \{x_1, \ldots, x_m\} \), each function \( f : \mathcal{X} \rightarrow \mathbb{R} \) can be identified by the vector

\[
[f(x_1), \ldots, f(x_m)]^T.
\]

Hence, probability distributions over functions with finite domains can be represented using a finite-dimensional multivariate Gaussian distribution.
Gaussian processes

A stochastic process is a collection of random variables \( \{ f(x) | x \in X \} \) indexed by a set \( X \), called the index set. A Gaussian process is a stochastic process such that any finite subcollection of random variables has a multivariate Gaussian distribution.
Gaussian processes

- A **stochastic process** is a collection of random variables \( \{ f(x) \mid x \in \mathcal{X} \} \) indexed by a set \( \mathcal{X} \), called the index set.
Gaussian processes

- A **stochastic process** is a collection of random variables \( \{ f(x) \mid x \in \mathcal{X} \} \) indexed by a set \( \mathcal{X} \), called the index set.
- A **Gaussian process** is a stochastic process such that any finite subcollection of random variables has a multivariate Gaussian distribution.

\[
\mathcal{X} = \mathbb{R}^2
\]

\[
\begin{bmatrix}
  f(x_1) \\
  f(x_2) \\
  f(x_3)
\end{bmatrix}
\sim \mathcal{N}
\begin{pmatrix}
  \mu_1 & \sigma_{1,2} & \sigma_{1,3} \\
  \sigma_{2,1} & \mu_2 & \sigma_{2,3} \\
  \sigma_{3,1} & \sigma_{3,2} & \mu_3
\end{pmatrix}
\]

\[
\begin{bmatrix}
  \mu_1 \\
  \sigma_{1,2} \\
  \sigma_{1,3}
\end{bmatrix}
\]

\[
\begin{bmatrix}
  \mu_2 \\
  \sigma_{2,1} \\
  \sigma_{2,3}
\end{bmatrix}
\]

\[
\begin{bmatrix}
  \mu_3 \\
  \sigma_{3,1} \\
  \sigma_{3,2}
\end{bmatrix}
\]
Illustration for the one-dimensional case
Gaussian processes

In particular, a collection of random variables \( \{ f(x) | x \in X \} \) is said to be drawn from a Gaussian process with mean function \( m \) and covariance function \( k \) if for any finite set of elements \( x_1, \ldots, x_m \in X \), the associated finite set of random variables \( f(x_1), \ldots, f(x_m) \) have distribution

\[
\begin{pmatrix}
  f(x_1) \\
  f(x_2) \\
  \vdots \\
  f(x_m)
\end{pmatrix}
\sim \mathcal{N}
\begin{pmatrix}
  m(x_1) \\
  m(x_2) \\
  \vdots \\
  m(x_m)
\end{pmatrix},
\begin{pmatrix}
  k(x_1, x_1) & \cdots & k(x_1, x_m) \\
  \vdots & \ddots & \vdots \\
  k(x_m, x_1) & \cdots & k(x_m, x_m)
\end{pmatrix}
\]

This is denoted \( f \sim \text{GP}(m, k) \).

Note that the above properties imply

\[
m(x) = \mathbb{E}(f(x))
\]

\[
k(x, x') = \mathbb{E}((f(x) - m(x))(f(x') - m(x')))
\]
Gaussian processes

In particular, a collection of random variables \( \{ f(x) \mid x \in \mathcal{X} \} \) is said to be drawn from a Gaussian process with mean function \( m \) and covariance function \( k \) if for any finite set of elements \( x_1, \ldots, x_m \in \mathcal{X} \), the associated finite set of random variables \( f(x_1), \ldots, f(x_m) \) have distribution

\[
\begin{bmatrix}
  f(x_1) \\
  f(x_2) \\
  \vdots \\
  f(x_m)
\end{bmatrix}
\sim \mathcal{N}
\left(
\begin{bmatrix}
  m(x_1) \\
  m(x_2) \\
  \vdots \\
  m(x_m)
\end{bmatrix},
\begin{bmatrix}
  k(x_1, x_1) & \cdots & k(x_1, x_m) \\
  k(x_2, x_1) & \cdots & k(x_2, x_m) \\
  \vdots & \cdots & \vdots \\
  k(x_m, x_1) & \cdots & k(x_m, x_m)
\end{bmatrix}
\right)
\]
Gaussian processes

In particular, a collection of random variables \( \{ f(x) \mid x \in \mathcal{X} \} \) is said to be drawn from a Gaussian process with **mean function** \( m \) and **covariance function** \( k \) if for any finite set of elements \( x_1, \ldots, x_m \in \mathcal{X} \), the associated finite set of random variables \( f(x_1), \ldots, f(x_m) \) have distribution

\[
\begin{bmatrix}
  f(x_1) \\
  f(x_2) \\
  \vdots \\
  f(x_m)
\end{bmatrix}
\sim \mathcal{N}
\begin{bmatrix}
  m(x_1) \\
  m(x_2) \\
  \vdots \\
  m(x_m)
\end{bmatrix},

\begin{bmatrix}
  k(x_1, x_1) & \cdots & k(x_1, x_m) \\
  \vdots & \ddots & \vdots \\
  k(x_m, x_1) & \cdots & k(x_m, x_m)
\end{bmatrix}
\]

This is denoted \( f \sim \mathcal{GP}(m, k) \).
Gaussian processes

In particular, a collection of random variables \( \{f(x) | x \in \mathcal{X}\} \) is said to be drawn from a Gaussian process with **mean function** \( m \) and **covariance function** \( k \) if for any finite set of elements \( x_1, \ldots, x_m \in \mathcal{X} \), the associated finite set of random variables \( f(x_1), \ldots, f(x_m) \) have distribution

\[
\begin{bmatrix}
    f(x_1) \\
    f(x_2) \\
    \vdots \\
    f(x_m)
\end{bmatrix} \sim \mathcal{N}
\begin{bmatrix}
    m(x_1) \\
    m(x_2) \\
    \vdots \\
    m(x_m)
\end{bmatrix},
\begin{bmatrix}
    k(x_1, x_1) & \cdots & k(x_1, x_m) \\
    \vdots & \ddots & \vdots \\
    k(x_m, x_1) & \cdots & k(x_m, x_m)
\end{bmatrix}
\]

This is denoted \( f \sim \mathcal{GP}(m, k) \).

Note that the above properties imply

\[
m(x) = \mathbb{E}(f(x))
\]

\[
k(x, x') = \mathbb{E}((f(x) - m(x))(f(x') - m(x')))
\]
Illustration for the one-dimensional case
Gaussian process regression

GP regression with a zero-mean process prior and (a) $N = 10$, (b) $N = 20$, and (c) $N = 40$ training examples. The blue line denotes the mean of the posterior predictive distribution, and the green shaded region denotes the 95% confidence region based on the model’s variance estimates.
Agenda

1. Introduction
2. Probabilistic prediction
3. **Calibration**
4. Set-valued (conformal) prediction
5. Epistemic uncertainty
6. Summary and outlook
Remarks on probabilistic learning

Learning probabilistic predictors remains challenging, especially for classification. Many methods (decision trees, SVM+Platt, NN+softmax, etc.) produce scores that are good enough for correct classification but do not qualify as accurate probability estimates.

Posterior probabilities produced by logistic regression and GP classification should be right in theory; however, the former comes with strong assumptions, and the latter is analytically intractable and needs to approximate the posterior.

Other Bayesian classification methods, such as Naïve Bayes, do not converge to true class probabilities in realistic settings.
Remarks on probabilistic learning

- Learning probabilistic predictors remains challenging, especially for classification.
Remarks on probabilistic learning

- Learning probabilistic predictors remains challenging, especially for classification.

- Many methods (decision trees, SVM+Platt, NN+softmax, etc.) produce scores that are good enough for correct classification but do not qualify as accurate probability estimates.
Remarks on probabilistic learning

- Learning probabilistic predictors remains challenging, especially for classification.

- Many methods (decision trees, SVM+Platt, NN+softmax, etc.) produce scores that are good enough for correct classification but do not qualify as accurate probability estimates.

- Posterior probabilities produced by logistic regression and GP classification should be right in theory; however, the former comes with strong assumptions, and the latter is analytically intractable and needs to approximate the posterior.
Remarks on probabilistic learning

- Learning probabilistic predictors remains challenging, especially for classification.

- Many methods (decision trees, SVM+Platt, NN+softmax, etc.) produce scores that are good enough for correct classification but do not quality as accurate probability estimates.

- Posterior probabilities produced by logistic regression and GP classification should be right in theory; however, the former comes with strong assumptions, and the latter is analytically intractable and needs to approximate the posterior.

- Other Bayesian classification methods, such as Naïve Bayes, do not converge to true class probabilities in realistic settings.
Many ML methods naturally yield predictions in the form of scores $s_1, \ldots, s_K \in S \subseteq \mathbb{R}$, for the $K$ classes, but these scores (e.g., the distance from a linear hyperplane) are not probabilities. Other methods do produce probability estimates $s = h(x) = (s_1, \ldots, s_K) \in [0, 1]^K$, but again, these may not match the true probabilities — they might rather be pseudo-probabilities that are not well "calibrated".

Yet, calibration is a prerequisite for uncertainty-awareness and important for prediction, decision-making, cost-sensitive classification, etc.
Calibration

Many ML methods naturally yield predictions in the form of scores

\[ s_1, \ldots, s_K \in \mathbb{S} \subseteq \mathbb{R}, \]

for the \( K \) classes, but these scores (e.g., the distance from a linear hyperplane) are not probabilities.
Calibration

- Many ML methods naturally yield predictions in the form of scores

\[ s_1, \ldots, s_K \in S \subseteq \mathbb{R}, \]

for the \( K \) classes, but these scores (e.g., the distance from a linear hyperplane) are not probabilities.

- Other methods do produce probability estimates

\[ s = h(x) = (s_1, \ldots, s_K) \in [0, 1]^K, \]

but again, these may not match the true probabilities— they might rather be pseudo-probabilities that are not well “calibrated”.
Calibration

- Many ML methods naturally yield predictions in the form of scores

\[ s_1, \ldots, s_K \in S \subseteq \mathbb{R}, \]

for the \( K \) classes, but these scores (e.g., the distance from a linear hyperplane) are not probabilities.

- Other methods do produce probability estimates

\[ s = h(x) = (s_1, \ldots, s_K) \in [0, 1]^K, \]

but again, these may not match the true probabilities—they might rather be pseudo-probabilities that are not well “calibrated”.

- Yet, calibration is a prerequisite for uncertainty-awareness and important for prediction, decision-making, cost-sensitive classification, etc.
Probability estimation with decision trees

\[ \hat{p} = \frac{7}{8} \quad \hat{p} = 0 \]

\[ \hat{p} = 0 \quad \hat{p} = \frac{8}{9} \]

\[ \hat{p} = 1 \quad \hat{p} = \frac{1}{7} \]

\[ \hat{p} = \frac{6}{7} \quad \hat{p} = \frac{1}{4} \]
Calibration

Consider the binary case ($K = 2$) with probabilistic predictions $\hat{p} = (\hat{p} - \hat{p}, \hat{p} + \hat{p}) = (1 - \hat{p} + \hat{p}, \hat{p} + \hat{p}) \in \Delta^2$, where $\hat{p} + \hat{p}$ denotes the (predicted) probability for the positive class.

We say that a probabilistic predictor is calibrated if, for all $\alpha \in [0, 1]$, $P(y = +1 | \hat{p} + \hat{p} = \alpha) = \alpha$.

Broadly speaking, averaged over all instances $x$ for which the learner predicts $\hat{p} + \hat{p} = h(x) = \alpha$, the fraction of positives is indeed $\alpha$. In other words, a predicted probability vector is supposed to match empirical frequencies, at least in the long run.
Consider the **binary case** \((K = 2)\) with probabilistic predictions

\[
\hat{p} = (\hat{p}_-, \hat{p}_+) = (1 - \hat{p}_+, \hat{p}_+) \in \Delta_2,
\]

where \(\hat{p}_+\) denotes the (predicted) probability for the positive class.
Calibration

- Consider the **binary case** \((K = 2)\) with probabilistic predictions

\[
\hat{p} = (\hat{p}_-, \hat{p}_+) = (1 - \hat{p}_+, \hat{p}_+) \in \Delta_2,
\]

where \(\hat{p}_+\) denotes the (predicted) probability for the positive class.

- We say that a probabilistic predictor is **calibrated** if, for all \(\alpha \in [0, 1]\),

\[
P(y = +1 | \hat{p}_+ = \alpha) = \alpha.
\]
Consider the **binary case** \((K = 2)\) with probabilistic predictions

\[
\hat{p} = (\hat{p}_-, \hat{p}_+) = (1 - \hat{p}_+, \hat{p}_+) \in \Delta_2,
\]

where \(\hat{p}_+\) denotes the (predicted) probability for the positive class.

We say that a probabilistic predictor is **calibrated** if, for all \(\alpha \in [0, 1]\),

\[
P(y = +1 | \hat{p}_+ = \alpha) = \alpha.
\]

Broadly speaking, averaged over all instances \(x\) for which the learner predicts \(\hat{p}_+ = h(x) = \alpha\), the fraction of positives is indeed \(\alpha\).
Calibration

- Consider the **binary case** \((K = 2)\) with probabilistic predictions

\[
\hat{p} = (\hat{p}_-, \hat{p}_+) = (1 - \hat{p}_+, \hat{p}_+) \in \Delta_2,
\]

where \(\hat{p}_+\) denotes the (predicted) probability for the positive class.

- We say that a probabilistic predictor is **calibrated** if, for all \(\alpha \in [0, 1]\),

\[
P(y = +1 | \hat{p}_+ = \alpha) = \alpha.
\]

- Broadly speaking, averaged over all instances \(x\) for which the learner predicts \(\hat{p}_+ = h(x) = \alpha\), the fraction of positives is indeed \(\alpha\).

- In other words, a predicted probability vector is supposed to match empirical frequencies, at least in the long run.
Calibration

Examples of miscalibration: bias toward $1/2$ (left), systematic underestimation (right).
Examples of miscalibration: bias toward 1/2 (left), systematic underestimation (right).
Reliability diagram

As ground-truth probabilities are not observed, binning is needed in practice. There is a trade-off in the choice of the width of the bins (more data per bin vs. more fine-granular assessment).
As ground-truth probabilities are not observed, binning is needed in practice.
As ground-truth probabilities are not observed, binning is needed in practice.

There is a trade-off in the choice of the width of the bins (more data per bin vs. more fine-granular assessment).
Calibration and scaling functions

Different post-processing (post-hoc) methods have been proposed for the purpose of calibration, i.e., to construct a calibration function $C: S \rightarrow [0, 1]$, such that $\hat{p} + = C(s)$ is a well-calibrated probability estimate for instances $x$ assigned score $s = h(x)$.

NB: To make the distinction between (post-hoc) calibration and scaling, $C$ is often called scaling function in the case where $S \neq [0, 1]$.

For learning $C$, a set of calibration data $D_{cal} = \{(s_1, y_1), \ldots, (s_N, y_N)\} \subset S \times \{0, 1\}$ is used. This data should be different from the training data (used to learn the scoring classifier $h$). Otherwise, there is a risk of introducing a bias.
Different post-processing (post-hoc) methods have been proposed for the purpose of calibration, i.e., to construct a calibration function

\[ C : \mathbb{S} \rightarrow [0,1], \]

such that \( \hat{p}_+ = C(s) \) is a well-calibrated probability estimate for instances \( x \) assigned score \( s = h(x) \).
Calibration and scaling functions

- Different post-processing (post-hoc) methods have been proposed for the purpose of calibration, i.e., to construct a calibration function

\[ C : \mathbb{S} \rightarrow [0, 1], \]

such that \( \hat{p}_+ = C(s) \) is a well-calibrated probability estimate for instances \( x \) assigned score \( s = h(x) \).

- NB: To make the distinction between (post-hoc) calibration and scaling, \( C \) is often called scaling function in the case where \( \mathbb{S} \neq [0, 1] \).
Calibration and scaling functions

- Different **post-processing** (post-hoc) methods have been proposed for the purpose of calibration, i.e., to construct a **calibration function**

\[ C : S \rightarrow [0, 1], \]

such that \( \hat{p}_+ = C(s) \) is a well-calibrated probability estimate for instances \( x \) assigned score \( s = h(x) \).

- NB: To make the distinction between (post-hoc) calibration and scaling, \( C \) is often called **scaling function** in the case where \( S \neq [0, 1] \).

- For learning \( C \), a set of **calibration data** is used:

\[ D_{cal} = \{(s_1, y_1), \ldots, (s_N, y_N)\} \subset S \times \{0, 1\} \]
Calibration and scaling functions

Different post-processing (post-hoc) methods have been proposed for the purpose of calibration, i.e., to construct a **calibration function**

\[ C : S \rightarrow [0,1], \]

such that \( \hat{p}_+ = C(s) \) is a well-calibrated probability estimate for instances \( x \) assigned score \( s = h(x) \).

**NB:** To make the distinction between (post-hoc) calibration and scaling, \( C \) is often called **scaling function** in the case where \( S \neq [0,1] \).

For learning \( C \), a set of **calibration data** is used:

\[ \mathcal{D}_{cal} = \{ (s_1, y_1), \ldots, (s_N, y_N) \} \subset S \times \{0,1\} \]

This data should be different from the training data (used to learn the scoring classifier \( h \)). Otherwise, there is a risk of introducing a bias.
Empirical binning

Binning offers a first obvious approach: Partition $S$ into bins (intervals) $B_1, \ldots, B_M$, and define $C(s) = \hat{p}_{J(s)}$, where $J(s)$ denotes the index of the bin of $s$ (i.e., $s \in B_{J(s)}$).

$\hat{p}_1, \ldots, \hat{p}_M$ are chosen so as to minimise the estimated calibration error (ECE)

$$ECE = \sum_{m=1}^{M} |B_m| N |\hat{p}_m - \bar{p}_m|,$$

where $\bar{p}_m = \frac{1}{\sum_{n=1}^{N} J(s_n \in B_m)} \sum_{n=1}^{N} y_n k$ is the average proportion of positives in bin $B_m$.

Binning is nonparametric and hence flexible, easy to train, and can directly minimise calibration error, albeit at the cost of (increased) grouping loss.
**Empirical binning**

- **Binning** offers a first obvious approach: Partition $\mathcal{S}$ into bins (intervals) $B_1, \ldots, B_M$, and define $C(s) = \hat{p}_{J(s)}$, where $J(s)$ denotes the index of the bin of $s$ (i.e., $s \in B_{J(s)}$).
Empirical binning

- **Binning** offers a first obvious approach: Partition $\mathbb{S}$ into bins (intervals) $B_1, \ldots, B_M$, and define $C(s) = \hat{p}_{J(s)}$, where $J(s)$ denotes the index of the bin of $s$ (i.e., $s \in B_{J(s)}$).
- $\hat{p}_1, \ldots, \hat{p}_M$ are chosen so as to minimise the estimated calibration error (ECE)

\[
\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} |\hat{p}_m - \bar{p}_m|,
\]

where

\[
\bar{p}_m = \frac{\sum_{n=1}^{N} \mathbb{1}[s_n \in B_m] \mathbb{1}[y_n = +1]}{\sum_{n=1}^{N} \mathbb{1}[s_n \in B_m]}
\]

is the average proportion of positives in bin $B_m$. 
Empirical binning

- **Binning** offers a first obvious approach: Partition $\mathcal{S}$ into bins (intervals) $B_1, \ldots, B_M$, and define $C(s) = \hat{p}_{J(s)}$, where $J(s)$ denotes the index of the bin of $s$ (i.e., $s \in B_{J(s)}$).
- $\hat{p}_1, \ldots, \hat{p}_M$ are chosen so as to minimise the estimated calibration error (ECE)

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{N} |\hat{p}_m - \bar{p}_m|,$$

where

$$\bar{p}_m = \frac{\sum_{n=1}^{N} [s_n \in B_m] [y_n = +1]}{\sum_{n=1}^{N} [s_n \in B_m]}$$

is the average proportion of positives in bin $B_m$.

- Binning is nonparametric and hence flexible, easy to train, and can directly minimise calibration error, albeit at the cost of (increased) grouping loss.
Another method is Platt scaling, which essentially applies logistic regression to predicted scores $s \in \mathbb{R}$, i.e., it fits a calibration function $C_{\alpha, \beta}(s) = \frac{1}{1 + \exp(-\alpha \cdot s - \beta)}$, minimising log-loss on $D_{\text{cal}}$ (including regularisation, or cross-validated training).

Platt scaling is fast and easy to implement, but restricted to sigmoidal calibration functions (pushing scores from the center toward the extremes, hence coming with a risk of over-confidence).
Another method is **Platt scaling**, which essentially applies logistic regression to predicted scores $s \in \mathbb{R}$, i.e., it fits a calibration function $C$ such that

$$C_{\alpha, \beta}(s) = \frac{1}{1 + \exp(-\alpha \cdot s - \beta)},$$

minimising log-loss on $\mathcal{D}_{\text{cal}}$ (including regularisation, or cross-validated training).
Another method is **Platt scaling**, which essentially applies logistic regression to predicted scores $s \in \mathbb{R}$, i.e., it fits a calibration function $C$ such that

$$C_{\alpha,\beta}(s) = \frac{1}{1 + \exp(-\alpha \cdot s - \beta)},$$

minimising log-loss on $D_{cal}$ (including regularisation, or cross-validated training).

Platt scaling is fast and easy to implement, but restricted to **sigmoidal calibration functions** (pushing scores from the center toward the extremes, hence coming with a risk of over-confidence).
Platt scaling

\[ \hat{p} = C(s) \]
Beta calibration

Beta calibration is specifically designed for probabilistic classifiers and fits a function $C_{\alpha,\beta,\gamma}(s) = \frac{1}{1 + \exp(-\alpha \cdot \log(s) - \beta \log(1-s) - \gamma)}$, again minimizing log-loss on $D_{\text{cal}}$.

Although still restricted in a parametric way, Beta calibration is more flexible than Platt scaling and includes inverse sigmoids and the identity map (which helps prevent over-calibration and unnecessary adjustments).
Beta calibration

- **Beta calibration** (? ) is specifically designed for probabilistic classifiers and fits a function

\[ C_{\alpha, \beta, \gamma}(s) = \frac{1}{1 + \exp(-\alpha \cdot \log(s) - \beta \log(1 - s) - \gamma)}, \]

again minimising log-loss on \( D_{\text{cal}} \).
Beta calibration

- **Beta calibration** is specifically designed for probabilistic classifiers and fits a function

\[ C_{\alpha,\beta,\gamma}(s) = \frac{1}{1 + \exp(-\alpha \cdot \log(s) - \beta \log(1 - s) - \gamma)} , \]

again minimising log-loss on \( D_{cal} \).

- Although still restricted in a parametric way, Beta calibration is more flexible than Platt scaling and includes inverse sigmoids and the identity map (which helps prevent over-calibration and unnecessary adjustments).
Isotonic regression combines the nonparametric character of binning with Platt scaling's guarantee of monotonicity. Isotonic regression minimizes

$$\sum_{i=1}^{n} w_n (C(s_n) - y_n)^2$$

subject to the constraint that $C$ is isotonic:

$$C(s) \leq C(t) \text{ for } s < t.$$ 

Note that $C$ is evaluated only at a finite number of points; in-between, one may (linearly) interpolate or assume a piecewise constant function. Isotonic regression is more expensive in terms of training time and memory consumption.
Isotonic regression

- Isotonic regression combines the nonparametric character of binning with Platt scaling’s guarantee of monotonicity.
Isotonic regression

- **Isotonic regression** combines the nonparametric character of binning with Platt scaling’s guarantee of monotonicity.

- Isotonic regression minimises

\[
\sum_{i=n}^{N} w_n (C(s_n) - y_n)^2
\]

subject to the constraint that \( C \) is isotonic: \( C(s) \leq C(t) \) for \( s < t \).
Isotonic regression

**Isotonic regression** combines the nonparametric character of binning with Platt scaling’s guarantee of monotonicity.

- Isotonic regression minimises

\[
\sum_{i=n}^{N} w_n (C(s_n) - y_n)^2
\]

subject to the constraint that \( C \) is isotonic: \( C(s) \leq C(t) \) for \( s < t \).

- Note that \( C \) is evaluated only at a finite number of points; in-between, one may (linearly) interpolate or assume a piecewise constant function.
Isotonic regression

- **Isotonic regression** combines the nonparametric character of binning with Platt scaling’s guarantee of monotonicity.
- Isotonic regression minimises

\[
\sum_{i=n}^{N} w_n (C(s_n) - y_n)^2
\]

subject to the constraint that \( C \) is isotonic: \( C(s) \leq C(t) \) for \( s < t \).

- Note that \( C \) is evaluated only at a finite number of points; in-between, one may (linearly) interpolate or assume a piecewise constant function.
- Isotonic regression is more expensive in terms of training time and memory consumption.
Pair-adjacent violators algorithm (PAVA)

Let the scores observed for calibration be sorted (and without ties), such that
\[ s_1 < s_2 < \ldots < s_N. \]

We then seek values \( c_1 \leq c_2 \leq \ldots \leq c_N \) which minimize
\[ N \sum_{n=1}^{N} w_n \left( c_n - y_n \right)^2. \]

Initialise one block \( B_n \) for each observation \((s_n, y_n)\); the value of the block is \( c(B_n) = y_n \) and the width is \( w(B_n) = 1. \)

A merge operation combines two blocks \( B' \) and \( B'' \) into a new block \( B \) with width \( w(B) = w(B') + w(B'') \) and value
\[ c = \frac{w(B') c(B') + w(B'') c(B'')} {w(B')} \].
Pair-adjacent violators algorithm (PAVA)

- Let the scores observed for calibration be sorted (and without ties), such that

\[ s_1 < s_2 < \ldots < s_N. \]

We then seek values \( c_1 \leq c_2 \leq \ldots \leq c_N \) which minimize

\[
\sum_{n=1}^{N} w_n (c_n - y_n)^2.
\]
Pair-adjacent violators algorithm (PAVA)

- Let the scores observed for calibration be sorted (and without ties), such that
  \[ s_1 < s_2 < \ldots < s_N. \]
  We then seek values \( c_1 \leq c_2 \leq \ldots \leq c_N \) which minimize
  \[
  \sum_{n=1}^{N} w_n (c_n - y_n)^2.
  \]
- **Initialise** one block \( B_n \) for each observation \((s_n, y_n)\); the value of the block is \( c(B_n) = y_n \) and the width is \( w(B_n) = 1 \).
Pair-adjacent violators algorithm (PAVA)

- Let the scores observed for calibration be sorted (and without ties), such that
  \[ s_1 < s_2 < \ldots < s_N. \]
  We then seek values \( c_1 \leq c_2 \leq \ldots \leq c_N \) which minimize
  \[ \sum_{n=1}^{N} w_n (c_n - y_n)^2. \]

- Initialise one block \( B_n \) for each observation \( (s_n, y_n) \); the value of the block is \( c(B_n) = y_n \) and the width is \( w(B_n) = 1. \)

- A merge operation combines two blocks \( B' \) and \( B'' \) into a new block \( B \) with width \( w(B) = w(B') + w(B'') \) and value
  \[ c = \frac{w(B')c(B') + w(B'')c(B'')}{w(B') + w(B'')}. \]
Pair-adjacent violators algorithm (PAVA)

PAVA iterates the following steps (the description is somewhat simplified to avoid notational overload):

1. Find the first violating pair, namely, adjacent blocks $B_i$ and $B_{i+1}$ such that $c_i > c_{i+1}$; if there is no such pair, then stop.

2. Merge $B_i$ and $B_{i+1}$ into a new block $B^\prime$.

3. If $c(B^\prime) < c(B_{i-1})$ for the left neighbor block $B_{i-1}$, merge also these blocks and continue doing so until no more violations are encountered.

4. Continue with (1).
Pair-adjacent violators algorithm (PAVA)

PAVA iterates the following steps (the description is somewhat simplified to avoid notational overload):

1. Find the first violating pair, namely, adjacent blocks $B_i$ and $B_{i+1}$ such that $c_i > c_{i+1}$; if there is no such pair, then stop.
2. Merge $B_i$ and $B_{i+1}$ into a new block $B$.
3. If $c(B) < c(B_{i-1})$ for the left neighbor block $B_{i-1}$, merge also these blocks and continue doing so until no more violations are encountered.
4. Continue with (1).
Pair-adjacent violators algorithm (PAVA)
Pair-adjacent violators algorithm (PAVA)

Note that, in the case of binary classification, the target values are all in \{0, 1\}. 

\[
\frac{65}{128}
\]
Pair-adjacent violators algorithm (PAVA)

Note that, in the case of binary classification, the target values are all in \( \{0, 1\} \):
Multi-class calibration

Calibration methods also exist for the multi-class case (i.e., classification problems with more than two classes). However, this makes the problem conceptually more difficult and is still a topic of ongoing research.

Some conceptual ideas do not generalize, for example, isotonic regression (which assumes a ranking on scores, and rankings are inherently bipartite).

Different definitions of calibration (leading to increasingly difficult problems) can be distinguished for more than two classes:

- **Confidence calibration**: Calibration of the highest predicted probability
- **Class-wise calibration**: Calibration of the marginal probabilities
- **Multi-class calibration**: Calibration of the entire vector of predicted probabilities

See recent survey by [?](http://example.com).
Multi-class calibration

- Calibration methods also exist for the **multi-class case** (i.e., classification problems with more than two classes).
Multi-class calibration

- Calibration methods also exist for the **multi-class case** (i.e., classification problems with more than two classes).

- Then, however, the problem becomes conceptually more difficult (and is still a topic of ongoing research).
Multi-class calibration

- Calibration methods also exist for the multi-class case (i.e., classification problems with more than two classes).

- Then, however, the problem becomes conceptually more difficult (and is still a topic of ongoing research).

- Some concepts do not immediately generalise, for example isotonic regression (which assumes a ranking on scores, and rankings are inherently bipartite).
Multi-class calibration

- Calibration methods also exist for the **multi-class case** (i.e., classification problems with more than two classes).

- Then, however, the problem becomes conceptually more difficult (and is still a topic of ongoing research).

- Some concepts do not immediately generalise, for example isotonic regression (which assumes a ranking on scores, and rankings are inherently bipartite).

- Different definitions of calibration (leading to increasingly difficult problems) can be distinguished for more than two classes:
  - **Confidence calibration**: Calibration of the highest predicted probability
  - **Class-wise calibration**: Calibration of the marginal probabilities
  - **Multi-class calibration**: Calibration of the entire vector of predicted probabilities
Multi-class calibration

- Calibration methods also exist for the multi-class case (i.e., classification problems with more than two classes).

- Then, however, the problem becomes conceptually more difficult (and is still a topic of ongoing research).

- Some concepts do not immediately generalise, for example isotonic regression (which assumes a ranking on scores, and rankings are inherently bipartite).

- Different definitions of calibration (leading to increasingly difficult problems) can be distinguished for more than two classes:
  - **Confidence calibration**: Calibration of the highest predicted probability
  - **Class-wise calibration**: Calibration of the marginal probabilities
  - **Multi-class calibration**: Calibration of the entire vector of predicted probabilities

- See recent survey by ?.
Agenda

1. Introduction
2. Probabilistic prediction
3. Calibration
4. Set-valued (conformal) prediction
5. Epistemic uncertainty
6. Summary and outlook
Conformal prediction is a framework for reliable prediction that is rooted in classical frequentist statistics and hypothesis testing. Instead of point predictions, CP makes set-valued predictions covering the true outcome with high probability.
Conformal prediction

- **Conformal prediction** (?) is a framework for reliable prediction that is rooted in classical frequentist statistics and hypothesis testing.
Conformal prediction

- **Conformal prediction** (??) is a framework for reliable prediction that is rooted in classical frequentist statistics and **hypothesis testing**.
- Instead of point predictions, CP makes **set-valued predictions** covering the true outcome with high probability.
Conformal prediction

- **Conformal prediction** (?) is a framework for reliable prediction that is rooted in classical frequentist statistics and **hypothesis testing**.
- Instead of point predictions, CP makes **set-valued predictions** covering the true outcome with high probability.

\[ P(y \in \{2, 3, 9\}) \geq 0.9 \]
Conformal prediction

Given a sequence of training observations $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$, and a new query $x_{N+1}$ with unknown outcome $y_{N+1}$, each candidate, i.e., the hypothesis $y_{N+1} = y$ is tested for all $y \in Y$:

$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N), (x_{N+1}, y)$

Only those outcomes $y$ for which this hypothesis can be rejected at a predefined level of confidence are excluded, while those for which the hypothesis cannot be rejected are collected to form the prediction set or prediction region $Y \subseteq Y$. 
Conformal prediction

- Given a sequence of training observations

  \[(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N), (x_{N+1}, \text{●})\]

  and a new query \(x_{N+1}\) with unknown outcome \(y_{N+1}\), \(\text{●}\) is hypothetically replaced by each candidate, i.e., the hypothesis \(y_{N+1} = y\) is tested for all \(y \in \mathcal{Y}\):

  \[(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N), (x_{N+1}, y)\]
Conformal prediction

- Given a sequence of training observations

\[(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N), (x_{N+1}, \bullet)\]

and a new query \(x_{N+1}\) with unknown outcome \(y_{N+1}\), \(\bullet\) is hypothetically replaced by each candidate, i.e., the hypothesis \(y_{N+1} = y\) is tested for all \(y \in \mathcal{Y}\):

\[(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N), (x_{N+1}, y)\]

- Only those outcomes \(y\) for which this hypothesis can be rejected at a predefined level of confidence are excluded, while those for which the hypothesis cannot be rejected are collected to form the prediction set or prediction region \(\mathcal{Y} \subseteq \mathcal{Y}\).
Nonconformity function

In conformal prediction, the "strangeness" of a pattern \((x_{N+1}, y)\) is captured in terms of a nonconformity score. Moreover, hypothesis testing is done in a nonparametric way.

Consider any nonconformity function \(f: \mathbb{X} \times \mathbb{Y} \rightarrow \mathbb{R}\) that assigns scores \(\alpha = f(x, y)\) to input/output tuples. The higher the score, the more "strange" the pattern \((x, y)\), i.e., the less the data point \((x, y)\) conforms to what one would expect to observe.
Nonconformity function

- In conformal prediction, the “strangeness” of a pattern $\left(x_{N+1}, y\right)$ is captured in terms of a **nonconformity score**.
Nonconformity function

- In conformal prediction, the “strangeness” of a pattern \((x_{N+1}, y)\) is captured in terms of a nonconformity score.

- Moreover, hypothesis testing is done in a nonparametric way.
Nonconformity function

- In conformal prediction, the “strangeness” of a pattern \((x_{N+1}, y)\) is captured in terms of a nonconformity score.
- Moreover, hypothesis testing is done in a nonparametric way.
- Consider any nonconformity function

\[
f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}
\]

that assigns scores \(\alpha = f(x, y)\) to input/output tuples.
In conformal prediction, the “strangeness” of a pattern \((x_{N+1}, y)\) is captured in terms of a **nonconformity score**.

Moreover, hypothesis testing is done in a **nonparametric way**.

Consider any **nonconformity function**

\[
f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}
\]

that assigns scores \(\alpha = f(x, y)\) to input/output tuples.

The higher the score, the more “strange” the pattern \((x, y)\), i.e., the less the data point \((x, y)\) conforms to what one would expect to observe.
Nonconformity function

conforming

nonconforming
Nonconformity function
Nonconformity function

Example of a nonconformity score based on nearest neighbors:

\[ f(x, y) = \sum_{i=1}^{k} d_{+i} + \sum_{i=1}^{k} d_{-i} \]

where \( d_{+i} \) is the distance from the \( i \)th nearest neighbor labeled \( y \), and \( d_{-i} \) the distance from the \( i \)th nearest neighbor labeled differently.
Nonconformity function

Example of a nonconformity score based on nearest neighbors:

\[
f(x, y) = \frac{\sum_{i=1}^{k} d_i^+}{\sum_{i=1}^{k} d_i^-},
\]

where \(d_i^+\) is the distance from the \(i^{th}\) nearest neighbor labeled \(y\), and \(d_i^-\) the distance from the \(i^{th}\) nearest neighbor labeled differently.
Applying this function to the sequence of observations, with a specific (though hypothetical) choice of $y = y_{N+1}$, yields a sequence of scores $\alpha_1, \alpha_2, \ldots, \alpha_N, \alpha_{N+1}$, where $\alpha_i = f(x_i, y_i)$. Denote by $\sigma$ the permutation of $\{1, \ldots, N+1\}$ that sorts the scores in increasing order, i.e., such that $\alpha_{\sigma(1)} \leq \ldots \leq \alpha_{\sigma(N+1)}$. Under the assumption that the hypothetical choice of $y_{N+1}$ is in agreement with the true data-generating process, and that this process has the property of exchangeability, every permutation $\sigma$ has the same probability of occurrence.
Applying this function to the sequence of observations, with a specific (though hypothetical) choice of $y = y_{N+1}$, yields a sequence of scores

$$\alpha_1, \alpha_2, \ldots, \alpha_N, \alpha_{N+1},$$

where $\alpha_i = f(x_i, y_i)$. 
Conformal prediction

Applying this function to the sequence of observations, with a specific (though hypothetical) choice of \( y = y_{N+1} \), yields a sequence of scores

\[
\alpha_1, \alpha_2, \ldots, \alpha_N, \alpha_{N+1},
\]

where \( \alpha_i = f(x_i, y_i) \).

Denote by \( \sigma \) the permutation of \( \{1, \ldots, N+1\} \) that sorts the scores in increasing order, i.e., such that

\[
\alpha_{\sigma(1)} \leq \ldots \leq \alpha_{\sigma(N+1)}.
\]
Conformal prediction

- Applying this function to the sequence of observations, with a specific (though hypothetical) choice of $y = y_{N+1}$, yields a sequence of scores

$$\alpha_1, \alpha_2, \ldots, \alpha_N, \alpha_{N+1},$$

where $\alpha_i = f(x_i, y_i)$.

- Denote by $\sigma$ the permutation of $\{1, \ldots, N + 1\}$ that sorts the scores in increasing order, i.e., such that

$$\alpha_{\sigma(1)} \leq \cdots \leq \alpha_{\sigma(N+1)}.$$

- Under the assumption that the hypothetical choice of $y_{N+1}$ is in agreement with the true data-generating process, and that this process has the property of **exchangeability**, every permutation $\sigma$ has the same probability of occurrence.
Exchangeability

A (finite) sequence of random variables $X_1, X_2, \ldots, X_N$ is exchangeable, if the following holds for any permutation $\sigma$: 

$$N \rightarrow \sigma(N):$$ 

the joint probability distribution of the random variables is the same as the joint distribution of the permuted sequence $X_\sigma(1), X_\sigma(2), \ldots, X_\sigma(N)$.

This means that the cumulative distribution function $F_{X_1, \ldots, X_N}: \mathbb{R}^N \rightarrow [0, 1]$ is symmetric in its arguments.

Exchangeability is weaker than the i.i.d. assumption (the latter implies the former but not the other way around).
Exchangeability

A (finite) **sequence of random variables** \(X_1, X_2, \ldots, X_N\) is exchangeable, if the following holds for any permutation \(\sigma : [N] \rightarrow [N]\): the joint probability distribution of the random variables is the same as the joint distribution of the permuted sequence \(X_{\sigma(1)}, X_{\sigma(2)}, \ldots, X_{\sigma(N)}\).
Exchangeability

- A (finite) **sequence of random variables** $X_1, X_2, \ldots, X_N$ is exchangeable, if the following holds for any permutation $\sigma : [N] \rightarrow [N]$: the joint probability distribution of the random variables is the same as the joint distribution of the permuted sequence $X_{\sigma(1)}, X_{\sigma(2)}, \ldots, X_{\sigma(N)}$.

- This means that the cumulative distribution function

  $$F_{X_1, \ldots, X_N} : \mathbb{R}^N \rightarrow [0, 1]$$

  is **symmetric** in its arguments.
Exchangeability

- A (finite) sequence of random variables $X_1, X_2, \ldots, X_N$ is exchangeable, if the following holds for any permutation $\sigma : [N] \rightarrow [N]$: the joint probability distribution of the random variables is the same as the joint distribution of the permuted sequence $X_{\sigma(1)}, X_{\sigma(2)}, \ldots, X_{\sigma(N)}$.

- This means that the cumulative distribution function

$$F_{X_1, \ldots, X_N} : \mathbb{R}^N \rightarrow [0, 1]$$

is symmetric in its arguments.

- Exchangeability is weaker than the i.i.d. assumption (the latter implies the former but not the other way around).
Conformal prediction

Consequently, the probability that $\alpha_{N+1}$ is among the $\epsilon$% highest nonconformity scores should be low. This notion can be captured by the $p$-values associated with the candidate $y$, defined as

$$p(y) = \frac{\# \{i | \alpha_i \geq \alpha_{N+1}\}}{N+1}$$

According to what we said, the probability that $p(y) < \epsilon$ (i.e., $\alpha_{N+1}$ is among the $\epsilon$% highest $\alpha$-values) is upper-bounded by $\epsilon$. Thus, the hypothesis $y_{N+1} = y$ can be rejected for those candidates $y$ for which $p(y) < \epsilon$. 
Conformal prediction

Consequently, the probability that $\alpha_{N+1}$ is among the $\epsilon$% highest nonconformity scores should be low.
Conformal prediction

- Consequently, the probability that $\alpha_{N+1}$ is among the $\epsilon\%$ highest nonconformity scores should be low.

- This notion can be captured by the $p$-values associated with the candidate $y$, defined as

$$p(y) := \frac{\#\{i \mid \alpha_i \geq \alpha_{N+1}\}}{N + 1}$$
Conformal prediction

- Consequently, the probability that $\alpha_{N+1}$ is among the $\epsilon\%$ highest nonconformity scores should be low.

- This notion can be captured by the $p$-values associated with the candidate $y$, defined as

$$p(y) := \frac{\# \{ i \mid \alpha_i \geq \alpha_{N+1} \}}{N + 1}$$

- According to what we said, the probability that $p(y) < \epsilon$ (i.e., $\alpha_{N+1}$ is among the $\epsilon\%$ highest $\alpha$-values) is upper-bounded by $\epsilon$. 

Conformal prediction

Consequently, the probability that $\alpha_{N+1}$ is among the $\epsilon \%$ highest nonconformity scores should be low.

This notion can be captured by the $p$-values associated with the candidate $y$, defined as

$$p(y) := \frac{\#\{i \mid \alpha_i \geq \alpha_{N+1}\}}{N + 1}$$

According to what we said, the probability that $p(y) < \epsilon$ (i.e., $\alpha_{N+1}$ is among the $\epsilon \%$ highest $\alpha$-values) is upper-bounded by $\epsilon$.

Thus, the hypothesis $y_{N+1} = y$ can be rejected for those candidates $y$ for which $p(y) < \epsilon$. 
By construction, the set-valued prediction $Y_{n+1} = Y(x_{n+1})$ is guaranteed to cover the true outcome $y_{n+1}$ with a pre-specified probability of $1 - \epsilon$ (for example 95%). The error bounds are valid by construction, regardless of the nonconformity function. However, the choice of this function has an important influence on the efficiency of conformal prediction, that is, the size of prediction regions: The more suitably the nonconformity function $f$ is chosen, the smaller these sets will be.
By construction, the set-valued prediction $Y = Y(x_{n+1})$ is guaranteed to cover the true outcome $y_{N+1}$ with a pre-specified probability of $1 - \epsilon$ (for example 95%).
Conformal prediction

- By construction, the set-valued prediction $Y = Y(x_{n+1})$ is guaranteed to cover the true outcome $y_{N+1}$ with a pre-specified probability of $1 - \epsilon$ (for example 95%).

- The error bounds are valid by construction, regardless of the nonconformity function.
Conformal prediction

- By construction, the set-valued prediction $Y = Y(x_{n+1})$ is guaranteed to cover the true outcome $y_{N+1}$ with a pre-specified probability of $1 - \epsilon$ (for example 95%).

- The error bounds are valid by construction, regardless of the nonconformity function.

- However, the choice of this function has an important influence on the efficiency of conformal prediction, that is, the size of prediction regions: The more suitably the nonconformity function $f$ is chosen, the smaller these sets will be.
Inductive conformal prediction

Conformal prediction as outlined above realises transductive inference, i.e., inference directly targeting a query instance (known by the learner) without inducing a general model beforehand. This is computationally costly, as it possibly requires refitting a model to the data in each iteration. Inductive conformal prediction (ICP) is an alternative that is computationally less expensive. In the split-conformal prediction variant, ICP splits the training data $D$ into ▶ proper training data $D_T$ of size $N - M$, ▶ calibration data $D_C = \{ (x_j, y_j) \}_{M_{j=1}}$ of size $M < N$. 


Inductive conformal prediction

- Conformal prediction as outlined above realises **transductive inference**, i.e., inference directly targeting a query instance (known by the learner) without inducing a general model beforehand.
Inductive conformal prediction

- Conformal prediction as outlined above realises **transductive inference**, i.e., inference directly targeting a query instance (known by the learner) without inducing a general model beforehand.

- This is computationally costly, as it possibly requires refitting a model to the data in each iteration.
Inductive conformal prediction

- Conformal prediction as outlined above realises transductive inference, i.e., inference directly targeting a query instance (known by the learner) without inducing a general model beforehand.

- This is computationally costly, as it possibly requires refitting a model to the data in each iteration.

- **Inductive conformal prediction** (ICP) is an alternative that is computationally less expensive.
Inductive conformal prediction

- Conformal prediction as outlined above realises transductive inference, i.e., inference directly targeting a query instance (known by the learner) without inducing a general model beforehand.

- This is computationally costly, as it possibly requires refitting a model to the data in each iteration.

- **Inductive conformal prediction** (ICP) is an alternative that is computationally less expensive.

- In the **split-conformal prediction** variant, ICP splits the training data $\mathcal{D}$ into
  - proper training data $\mathcal{D}_T$ of size $N - M$,
  - calibration data $\mathcal{D}_C = \{(x_j, y_j)\}_{j=1}^M$ of size $M < N$. 
Inductive conformal prediction

Imagine, for example, a probabilistic classifier $h$ trained on the proper training data, and nonconformity of $(x, y)$ is defined as $\alpha = 1 - p(y)$, where $p = h(x)$.

The scores on the calibration data define an empirical CDF ($\hat{F}(\alpha) = \text{probability (relative frequency) of nonconformity scores} \leq \alpha$):

With high probability, nonconformity of a "real" data point is $\leq \alpha_0$. 

Inductive conformal prediction

Imagine, for example, a probabilistic classifier $h$ is trained on the proper training data, and nonconformity of $(x, y)$ is defined as $\alpha = 1 - p(y)$, where $p = h(x)$. 
Inductive conformal prediction

Imagine, for example, a probabilistic classifier $h$ is trained on the proper training data, and nonconformity of $(x, y)$ is defined as $\alpha = 1 - p(y)$, where $p = h(x)$.

The scores on the calibration data define an empirical CDF ($\hat{F}(\alpha) = \text{probability (relative frequency) of nonconformity scores} \leq \alpha$):
Inductive conformal prediction

Imagine, for example, a probabilistic classifier $h$ is trained on the proper training data, and nonconformity of $(x, y)$ is defined as $\alpha = 1 - p(y)$, where $p = h(x)$.

The scores on the calibration data define an empirical CDF ($\hat{F}(\alpha) = \text{probability (relative frequency) of nonconformity scores } \leq \alpha$):

With high probability, nonconformity of a “real” data point is $\leq \alpha_0$. 
Imagine, for example, a probabilistic classifier \( h \) is trained on the proper training data, and nonconformity of \((x, y)\) is defined as \( \alpha = 1 - p(y) \), where \( p = h(x) \).

The scores on the calibration data define an empirical PDF (\( \hat{f}(\alpha) \) = probability (relative frequency) of nonconformity scores):

With high probability, nonconformity of a "real" data point is \( \leq \alpha_0 \).
Imagine, for example, a probabilistic classifier $h$ is trained on the proper training data, and nonconformity of $(x, y)$ is defined as $\alpha = 1 - p(y)$, where $p = h(x)$. 
Inductive conformal prediction

- Imagine, for example, a probabilistic classifier $h$ is trained on the proper training data, and nonconformity of $(x, y)$ is defined as $\alpha = 1 - p(y)$, where $p = h(x)$.
- The scores on the calibration data define an empirical PDF ($\hat{f}(\alpha) = \text{probability (relative frequency) of nonconformity scores } \alpha$):

![Diagram showing the empirical PDF $\hat{f}(\alpha)$ and the threshold $\alpha_0$. The probability $1 - \epsilon$ is highlighted in red.](image)
Inductive conformal prediction

- Imagine, for example, a probabilistic classifier $h$ is trained on the proper training data, and nonconformity of $(x, y)$ is defined as $\alpha = 1 - p(y)$, where $p = h(x)$.
- The scores on the calibration data define an empirical PDF ($\hat{f}(\alpha) =$ probability (relative frequency) of nonconformity scores $\alpha$):

![Histogram of nonconformity scores](image)

- With high probability, nonconformity of a “real” data point is $\leq \alpha_0$. 

Inductive conformal prediction

A predictive model $h$ is trained on the proper training set $D_T$. Using this model, the nonconformity scores $\alpha_1, \ldots, \alpha_M$ are computed for the examples in the calibration set — for example, if $h$ predicts probabilities, scores could be defined as $\alpha_j = f(x_j, y_j) = 1 - p(y_j | h, x_j)$.

Define $\hat{q}$ to be the $\lceil (M + 1)(1 - \epsilon) \rceil / M$ empirical quantile of $\alpha_1, \ldots, \alpha_M$.

For a query $x_q \in X$, construct the prediction set as follows:

$$Y(x_q) = \{ y \in Y | f(x_q, y) \leq \hat{q} \}$$

Assuming $D$ to be an i.i.d. sample, and $(x_q, y_q)$ sampled from the same distribution, the following estimate is provably correct:

$$1 - \epsilon \leq P(y_q \in Y(x_q)) \leq 1 - \epsilon + \frac{M + 1}{81 / 128}.$$
Inductive conformal prediction

- A predictive model $h$ is trained on the proper training set $\mathcal{D}_T$. 

Using this model, the nonconformity scores $\alpha_1, \ldots, \alpha_M$ are computed for the examples in the calibration set. For example, if $h$ predicts probabilities, scores could be defined as $\alpha_j = f(x_j, y_j) = 1 - p(y_j | h, x_j)$. Define $\hat{q}$ to be the $\lceil (M+1)(1-\epsilon) \rceil / M$ empirical quantile of $\alpha_1, \ldots, \alpha_M$.

For a query $x_q \in X$, construct the prediction set as follows: $Y(x_q) = \{ y \in Y | f(x_q, y) \leq \hat{q} \}$. Assuming $\mathcal{D}$ to be an i.i.d. sample, and $(x_q, y_q)$ sampled from the same distribution, the following estimate is provably correct:

$$1 - \epsilon \leq P(y_q \in Y(x_q)) \leq 1 - \epsilon + \frac{1}{M+1} \frac{81}{128}.$$
Inductive conformal prediction

- A predictive model $h$ is trained on the proper training set $\mathcal{D}_T$.
- Using this model, the nonconformity scores $\alpha_1, \ldots, \alpha_M$ are computed for the examples in the calibration set — for example, if $h$ predicts probabilities, scores could be defined as $\alpha_j = f(x_j, y_j) = 1 - p(y_j \mid h, x_j)$. 

Define $\hat{q}$ to be the $\lceil (M + 1)(1 - \epsilon) \rceil / M$ empirical quantile of $\alpha_1, \ldots, \alpha_M$.

For a query $x_q \in X$, construct the prediction set as follows:

$Y(x_q) = \{ y \in Y \mid f(x_q, y) \leq \hat{q} \}$

Assuming $\mathcal{D}$ to be an i.i.d. sample, and $(x_q, y_q)$ sampled from the same distribution, the following estimate is provably correct:

$1 - \epsilon \leq P(y_q \in Y(x_q)) \leq 1 - \epsilon + \frac{1}{M + 1}$
Inductive conformal prediction

- A predictive model $h$ is trained on the proper training set $D_T$.
- Using this model, the nonconformity scores $\alpha_1, \ldots, \alpha_M$ are computed for the examples in the calibration set — for example, if $h$ predicts probabilities, scores could be defined as $\alpha_j = f(x_j, y_j) = 1 - p(y_j \mid h, x_j)$.
- Define $\hat{q}$ to be the $\lceil (M + 1)(1 - \epsilon) \rceil / M$ empirical quantile of $\alpha_1, \ldots, \alpha_M$. 

For a query $x_q \in X$, construct the prediction set as follows:

$$Y(x_q) = \{y \in Y \mid f(x_q, y) \leq \hat{q}\}$$

Assuming $D$ to be an i.i.d. sample, and $(x_q, y_q)$ sampled from the same distribution, the following estimate is provably correct:

$$1 - \epsilon \leq P(y_q \in Y(x_q)) \leq 1 - \epsilon + \frac{1}{M + 1}$$
A predictive model $h$ is trained on the proper training set $D_T$.

Using this model, the nonconformity scores $\alpha_1, \ldots, \alpha_M$ are computed for the examples in the calibration set—for example, if $h$ predicts probabilities, scores could be defined as $\alpha_j = f(x_j, y_j) = 1 - p(y_j | h, x_j)$.

Define $\hat{q}$ to be the $\lceil (M + 1)(1 - \epsilon) \rceil / M$ empirical quantile of $\alpha_1, \ldots, \alpha_M$.

For a query $x_q \in X$, construct the prediction set as follows:

$$Y(x_q) = \left\{ y \in Y \mid f(x_q, y) \leq \hat{q} \right\}$$
Inductive conformal prediction

- A predictive model $h$ is trained on the proper training set $\mathcal{D}_T$.
- Using this model, the nonconformity scores $\alpha_1, \ldots, \alpha_M$ are computed for the examples in the calibration set—for example, if $h$ predicts probabilities, scores could be defined as $\alpha_j = f(x_j, y_j) = 1 - p(y_j \mid h, x_j)$.
- Define $\hat{q}$ to be the $\lceil(M + 1)(1 - \epsilon)\rceil / M$ empirical quantile of $\alpha_1, \ldots, \alpha_M$.
- For a query $x_q \in \mathcal{X}$, construct the prediction set as follows:

  $$Y(x_q) = \left\{ y \in \mathcal{Y} \mid f(x_q, y) \leq \hat{q} \right\}$$

- Assuming $\mathcal{D}$ to be an i.i.d. sample, and $(x_q, y_q)$ sampled from the same distribution, the following estimate is provably correct:

  $$1 - \epsilon \leq P(y_q \in Y(x_q)) \leq 1 - \epsilon + \frac{1}{M + 1}$$
Example

Recall our example with two class-conditional normals. Define nonconformity as negative margin, i.e.,

\[ s(x, y) = \begin{cases} 
  t - x & \text{if } y = +1 \\
  x - t & \text{if } y = -1 
\end{cases} \]

where \( t \) is the threshold produced by the learner.
Recall our example with two class-conditional normals.
Recall our example with two class-conditional normals.

Define nonconformity as negative margin, i.e.,

$$s(x, y) = \begin{cases} 
  t - x & \text{if } y = +1 \\
  x - t & \text{if } y = -1 
\end{cases}$$

where $t$ is the threshold produced by the learner.
Example

Training data:

(175, 48, -1), (174, 16, -1), (167, 50, +1), (156, 16, +1), (180, 02, +1), (156, 79, -1), (171, 94, +1), (162, 72, -1), (171, 94, +1), (183, 75, +1), (161, 31, -1), (173, 17, -1), (205, 09, +1), (183, 35, +1), (179, 54, -1), (169, 76, -1), (173, 54, -1), (160, 74, -1), (166, 05, +1), (157, 42, +1), (187, 68, +1).

Threshold produced by the learner:

t = 171.77

CDF of nonconformity (on calibration data):

83 / 128
Example

- Training data: (175.48, −1), (174.16, −1), (167.50, +1), (156.16, +1), (180.02, +1), (156.79, −1), (171.94, +1), (162.72, −1), (171.94, +1), (183.75, +1), (161.31, −1), (173.17, −1), (205.09, +1), (183.35, +1), (179.54, −1), (169.76, −1), (173.54, −1), (160.74, −1), (166.05, +1), (157.42, +1), (187.68, +1).
Example

- Training data: (175.48, −1), (174.16, −1), (167.50, +1), (156.16, +1), (180.02, +1), (156.79, −1), (171.94, +1), (162.72, −1), (171.94, +1), (183.75, +1), (161.31, −1), (173.17, −1), (205.09, +1), (183.35, +1), (179.54, −1), (169.76, −1), (173.54, −1), (160.74, −1), (166.05, +1), (157.42, +1), (187.68, +1).

- Threshold produced by the learner: $t = 171.77$
Example

- Training data: (175.48, −1), (174.16, −1), (167.50, +1), (156.16, +1), (180.02, +1), (156.79, −1), (171.94, +1), (162.72, −1), (171.94, +1), (183.75, +1), (161.31, −1), (173.17, −1), (205.09, +1), (183.35, +1), (179.54, −1), (169.76, −1), (173.54, −1), (160.74, −1), (166.05, +1), (157.42, +1), (187.68, +1).

- Threshold produced by the learner: $t = 171.77$

- CDF of nonconformity (on calibration data):
The 90th percentile is 8.85, so for $\epsilon = 0.1$, CF yields the following prediction for a new query $x_q$:

\[
Y(x) = \begin{cases} 
-1 & \text{if } x_q < 162.92 \\
-1, +1 & \text{if } 162.92 \leq x_q \leq 180.62 \\
+1 & \text{if } x_q > 180.62 
\end{cases}
\]
The 90\textsuperscript{th} percentile is 8.85, so for $\epsilon = 0.1$, CF yields the following prediction for a new query $x_q$:

$$Y(x) = \begin{cases} 
\{-1\} & \text{if } x_q < 162.92 \\
\{-1, +1\} & \text{if } 162.92 \leq x_q \leq 180.62 \\
\{+1\} & \text{if } x_q > 180.62
\end{cases}$$
Remarks

The above validity property is also called marginal coverage. The randomisation is over the entire data generation and prediction procedure; thus, the coverage of $1 - \varepsilon$ is neither guaranteed for a fixed sequence on previous data (coverage can be higher or lower), conditioned on the query $x_q$.

There are various extensions of CP, also for conditional coverage. Besides, instead of controlling coverage, there are variants for controlling more general notions of risk.

Uncertainty quantification with conformal prediction is agnostic to the underlying model, agnostic to the underlying data distribution (i.e., distribution-free), valid for the finite sample case (not only asymptotically).
The above validity property is also called *marginal coverage*. The randomisation is over the entire data generation and prediction procedure; thus, the coverage of $1 - \epsilon$ is neither guaranteed

- for a fixed sequence on previous data (coverage can be higher or lower),
- conditioned on the query $x_q$. 

There are various extensions of CP, also for *conditional coverage*. Besides, instead of controlling coverage, there are variants for controlling more general notions of *risk*. Uncertainty quantification with conformal prediction is

- agnostic to the underlying model,
- agnostic to the underlying data distribution (i.e., distribution-free),
- valid for the finite sample case (not only asymptotically).
The above validity property is also called **marginal coverage**. The randomisation is over the entire data generation and prediction procedure; thus, the coverage of $1 - \epsilon$ is neither guaranteed
- for a fixed sequence on previous data (coverage can be higher or lower),
- conditioned on the query $x_q$.

There are various extensions of CP, also for **conditional coverage**. Besides, instead of controlling coverage, there are variants for controlling more general notions of **risk**.
The above validity property is also called **marginal coverage**. The randomisation is over the entire data generation and prediction procedure; thus, the coverage of $1 - \epsilon$ is neither guaranteed
- for a fixed sequence on previous data (coverage can be higher or lower),
- conditioned on the query $x_q$.

There are various extensions of CP, also for **conditional coverage**. Besides, instead of controlling coverage, there are variants for controlling more general notions of **risk** (?)

Uncertainty quantification with conformal prediction is
- agnostic to the underlying model,
- agnostic to the underlying data distribution (i.e., distribution-free),
- valid for the finite sample case (not only asymptotically).
Agenda

1. Introduction
2. Probabilistic prediction
3. Calibration
4. Set-valued (conformal) prediction
5. **Epistemic uncertainty**
6. Summary and outlook
Aleatoric versus epistemic uncertainty

Aleatoric (aka statistical) uncertainty refers to the notion of randomness, that is, the variability in the outcome of an experiment which is due to inherently random effects.

Epistemic (aka systematic) uncertainty refers to uncertainty caused by a lack of knowledge, i.e., to the epistemic state of the agent.

As opposed to aleatoric uncertainty, epistemic uncertainty can in principle be reduced on the basis of additional information.
Aleatoric versus epistemic uncertainty

- **Aleatoric** (aka statistical) uncertainty refers to the notion of *randomness*, that is, the variability in the outcome of an experiment which is due to inherently random effects.
Aleatoric versus epistemic uncertainty

- **Aleatoric** (aka statistical) uncertainty refers to the notion of *randomness*, that is, the variability in the outcome of an experiment which is due to inherently random effects.

- **Epistemic** (aka systematic) uncertainty refers to uncertainty caused by a lack of knowledge, i.e., to the epistemic state of the agent.
Aleatoric versus epistemic uncertainty

- **Aleatoric** (aka statistical) uncertainty refers to the notion of *randomness*, that is, the variability in the outcome of an experiment which is due to inherently random effects.

- **Epistemic** (aka systematic) uncertainty refers to uncertainty caused by a *lack of knowledge*, i.e., to the epistemic state of the agent.

- As opposed to aleatoric uncertainty, epistemic uncertainty can in principle be reduced on the basis of additional information.
Aleatoric versus epistemic uncertainty

"Not knowing the chance of mutually exclusive events and knowing the chance to be equal are two quite different states of knowledge"

Ronald Fisher (1890-1962)
Aleatoric versus epistemic uncertainty in ML
Both types of uncertainty also play an important role in ML, where the learner’s state of knowledge strongly depends on the amount of data seen so far...
Both types of uncertainty also play an important role in ML, where the learner’s state of knowledge strongly depends on the amount of data seen so far ...
Aleatoric versus epistemic uncertainty in ML

Both types of uncertainty also play an important role in ML, where the learner’s state of knowledge strongly depends on the amount of data seen so far ...
Aleatoric versus epistemic uncertainty in ML
Aleatoric versus epistemic uncertainty in ML

... but also on the underlying model assumptions:
Uncertainty-aware learning

Uncertainty representation: How should the learner represent its (model, predictive) uncertainty, i.e., which mathematical formalisms should be used?

Learning and inference: How to make the learner accomplish the task? How to make sure that the uncertainty representation is accurate (and why to trust it more than the actual prediction)?

Uncertainty quantification: How to quantify the learner's uncertainty in terms of numbers? How to measure and disentangle the different types of uncertainty (aleatoric, epistemic, total)?
Uncertainty-aware learning

- **Uncertainty representation:** How should the learner represent its (model, predictive) uncertainty, i.e., which mathematical formalisms should be used?
Uncertainty-aware learning

- **Uncertainty representation:** How should the learner represent its (model, predictive) uncertainty, i.e., which mathematical formalisms should be used?

- **Learning and inference:** How to make the learner accomplish the task? How to make sure that the uncertainty representation is accurate (and why to trust it more than the actual prediction)?
Uncertainty-aware learning

- **Uncertainty representation**: How should the learner represent its (model, predictive) uncertainty, i.e., which mathematical formalisms should be used?

- **Learning and inference**: How to make the learner accomplish the task? How to make sure that the uncertainty representation is accurate (and why to trust it more than the actual prediction)?

- **Uncertainty quantification**: How to quantify the learner’s uncertainty in terms of numbers? How to measure and disentangle the different types of uncertainty (aleatoric, epistemic, total)?
Aleatoric versus epistemic uncertainty in ML

The distinction between aleatoric and epistemic uncertainty can be very difficult: Is the data-generating process completely random or only very complicated?

Predict the next number: 116, 304, 194, 341, 224, 654, 609, 625, 533, 91, 205, 35, 527, 611, 128, 235, 348, 912, 582, 52, 672, 20, 856, 904, 628, 273, 615, 105, 610, 862, 384, 705, 73, 794, 775, 156, ??

$x \leftarrow x \times 237 \mod 971$

$92 / 128$
Aleatoric versus epistemic uncertainty in ML

- The distinction between aleatoric and epistemic uncertainty can be very difficult: Is the data-generating process completely random or only very complicated?
The distinction between aleatoric and epistemic uncertainty can be very difficult: Is the data-generating process completely random or only very complicated?

**Predict the next number:** 116, 304, 194, 341, 224, 654, 609, 625, 533, 91, 205, 35, 527, 611, 128, 235, 348, 912, 582, 52, 672, 20, 856, 904, 628, 273, 615, 105, 610, 862, 384, 705, 73, 794, 775, 156, ??
Aleatoric versus epistemic uncertainty in ML

The distinction between aleatoric and epistemic uncertainty can be very difficult: Is the data-generating process completely random or only very complicated?

Predict the next number: 116, 304, 194, 341, 224, 654, 609, 625, 533, 91, 205, 35, 527, 611, 128, 235, 348, 912, 582, 52, 672, 20, 856, 904, 628, 273, 615, 105, 610, 862, 384, 705, 73, 794, 775, 156, ??

\[ x \leftarrow x \times 237 \mod 971 \]
Uncertainty quantification
Uncertainty quantification

- In the case of neural networks, where hypotheses $h = h_\theta$ are identified by network weights $\theta$, epistemic uncertainty essentially corresponds to uncertainty about these weights.
In the case of neural networks, where hypotheses $h = h_\theta$ are identified by network weights $\theta$, epistemic uncertainty essentially corresponds to uncertainty about these weights.

- Fixed weights $\theta$ lead to a fixed probability $p(\cdot | x, \theta)$. 
The Bayesian approach

- A Bayesian learner maintains a probability distribution over the hypothesis space (probabilistic predictors).
- The less concentrated that distribution, the higher the learner’s epistemic uncertainty.
The Bayesian approach

- A Bayesian learner maintains a probability distribution over the hypothesis space (probabilistic predictors).
- The less concentrated that distribution, the higher the learner’s epistemic uncertainty.
Posterior predictive distribution

hypothesis space $\mathcal{H}$

$p(h | D)$
Uncertainty quantification

How to measure uncertainty, i.e., quantify the amount of uncertainty contained in a prediction? A well-known uncertainty measure is the Shannon entropy, which, in the case of discrete probability $p: Y \rightarrow [0, 1]$, is given by

$$H[Y] = H[p] = -\sum_{y \in Y} p(y) \log_2 p(y).$$

What we seek is a decomposition $TU(x) = AU(x) + EU(x)$ of total uncertainty into aleatoric uncertainty and epistemic uncertainty.
Uncertainty quantification

- How to measure uncertainty, i.e., quantify the amount of uncertainty contained in a prediction?

A well-known uncertainty measure is the Shannon entropy, which, in the case of discrete probability $p: Y \rightarrow [0, 1]$, is given by:

$$H[Y] = H[p] = -\sum_{y \in Y} p(y) \log_2 p(y).$$

What we seek is a decomposition $TU(x) = AU(x) + EU(x)$ of total uncertainty into aleatoric and epistemic uncertainty.
Uncertainty quantification

- How to measure uncertainty, i.e., quantify the amount of uncertainty contained in a prediction?

- A well-known uncertainty measure is the **Shannon entropy**, which, in the case of discrete probability $p : \mathcal{Y} \rightarrow [0, 1]$, is given by

$$H[ Y ] = H[ p ] = - \sum_{y \in \mathcal{Y}} p(y) \log_2 p(y).$$
Uncertainty quantification

- How to measure uncertainty, i.e., quantify the amount of uncertainty contained in a prediction?
- A well-known uncertainty measure is the **Shannon entropy**, which, in the case of discrete probability $p : \mathcal{Y} \rightarrow [0, 1]$, is given by

  $$H[Y] = H[p] = - \sum_{y \in \mathcal{Y}} p(y) \log_2 p(y).$$

- What we seek is a **decomposition**

  $$\text{TU}(x) = \text{AU}(x) + \text{EU}(x)$$

  total uncertainty  aleatoric uncertainty  epistemic uncertainty
Uncertainty quantification

One idea is to model epistemic uncertainty as mutual information between outcomes and hypotheses ($H[Y] \triangleq I(Y; \Theta)$ total uncertainty $= I(Y; \Theta) + H[Y|\Theta]$ aleatoric

Intuitively, epistemic uncertainty thus captures the amount of information about the model parameters $\theta$ that would be gained through knowledge of the true outcome $y$. Total uncertainty $= \text{entropy of the predictive posterior distribution, in the case of discrete } Y \text{ given by } TU(x) = H[p(y|x)] = - \sum_{y \in Y} p(y|x) \log_2 p(y|x)$. 

One idea is to model **epistemic uncertainty** as **mutual information** between outcomes and hypotheses (?):

\[
H[Y] = I(Y; \Theta) + H[Y \mid \Theta]
\]

- \(H[Y]\) is total uncertainty
- \(I(Y; \Theta)\) is epistemic uncertainty
- \(H[Y \mid \Theta]\) is aleatoric uncertainty

Intuitively, epistemic uncertainty thus captures the amount of information about the model parameters \(\theta\) that would be gained through knowledge of the true outcome \(y\). Total uncertainty is the entropy of the predictive posterior distribution, in the case of discrete \(Y\) given by

\[
TU(x) = H[p(y \mid x)] = -\sum_{y \in Y} p(y \mid x) \log_2 p(y \mid x).
\]
One idea is to model **epistemic uncertainty** as **mutual information** between outcomes and hypotheses (\(I\)):

\[
H[Y] = I(Y; \Theta) + H[Y | \Theta]
\]

Intuitively, epistemic uncertainty thus captures the amount of information about the model parameters \(\theta\) that would be gained through knowledge of the true outcome \(y\).
Uncertainty quantification

- One idea is to model **epistemic uncertainty** as **mutual information** between outcomes and hypotheses $(\cdot)$:

\[
H[Y] = I(Y; \Theta) + H[Y | \Theta]
\]

- Intuitively, epistemic uncertainty thus captures the amount of information about the model parameters $\theta$ that would be gained through knowledge of the true outcome $y$.

- **Total uncertainty** = entropy of the predictive posterior distribution, in the case of discrete $Y$ given by

\[
TU(x) = H[p(y | x)] = - \sum_{y \in Y} p(y | x) \log_2 p(y | x).
\]
Uncertainty quantification

This uncertainty also includes the (epistemic) uncertainty about the network weights \( \theta \), but fixing a set of weights, i.e., considering a distribution \( p(y|x, \theta) \), removes the epistemic uncertainty. Therefore, the expectation over the entropies of these distributions,

\[
E_{p(\theta|D)} H[p(y|x, \theta)] = -\int p(\theta|D) \left[ \sum_{y \in Y} p(y|x, \theta) \log_2 p(y|x, \theta) \right] d\theta,
\]

is a measure of the aleatoric uncertainty (conditional entropy).

Finally, the epistemic uncertainty is obtained as the difference \( EU(x) = H[p(y|x)] - E_{p(\theta|D)} H[p(y|x, \theta)] \), which equals the mutual information between \( y \) and \( \theta \).
Uncertainty quantification

- This uncertainty also includes the (epistemic) uncertainty about the network weights $\theta$, but fixing a set of weights, i.e., considering a distribution $p(y \mid x, \theta)$, removes the epistemic uncertainty.
Uncertainty quantification

- This uncertainty also includes the (epistemic) uncertainty about the network weights $\theta$, but fixing a set of weights, i.e., considering a distribution $p(y \mid x, \theta)$, removes the epistemic uncertainty.

- Therefore, the expectation over the entropies of these distributions,

$$
\mathbb{E}_{p(\theta \mid D)} H[p(y \mid x, \theta)] =
$$

$$
= - \int p(\theta \mid D) \left( \sum_{y \in \mathcal{Y}} p(y \mid x, \theta) \log_2 p(y \mid x, \theta) \right) d\theta,
$$

is a measure of the **aleatoric uncertainty** (conditional entropy).
Uncertainty quantification

- This uncertainty also includes the (epistemic) uncertainty about the network weights $\theta$, but fixing a set of weights, i.e., considering a distribution $p(y \mid x, \theta)$, removes the epistemic uncertainty.

- Therefore, the expectation over the entropies of these distributions,

$$\mathbb{E}_{p(\theta \mid D)} H[p(y \mid x, \theta)] =$$

$$= - \int p(\theta \mid D) \left( \sum_{y \in Y} p(y \mid x, \theta) \log_2 p(y \mid x, \theta) \right) d\theta ,$$

is a measure of the **aleatoric uncertainty** (conditional entropy).

- Finally, the **epistemic uncertainty** is obtained as the difference

$$\text{EU}(x) := H[p(y \mid x)] - \mathbb{E}_{p(\theta \mid D)} H[p(y \mid x, \theta)] ,$$

which equals the **mutual information** between $y$ and $\theta$. 
Example: coin flipping

- Tossing a coin with bias $p$, task is to predict the next outcome, hypothesis space equipped with Dirichlet distribution
Remarks

- Is a uniform distribution on $\Delta = \mathbb{P}(\mathcal{Y})$ (set of all distributions $p$ on $\mathcal{Y}$) an adequate representation of **complete ignorance** (full epistemic uncertainty)?

- **Averaging** the (conditional) entropies over all $p$ is meaningful only if all $p$ are indeed **known** to be equally likely (aleatoric uncertainty is always $1/2$).

- But this is certainly not the case, as only one $p^*$ can be the ground truth.
Remarks

- One may also question the additive decomposition $TU = AU + EU$ itself.

- In the beginning, total uncertainty should be full ($TU = 1$), and so should epistemic uncertainty ($EU = 1$) — but this implies $AU = 0$.

- This suggests a role of $AU$ as a lower bound on (the true) aleatoric uncertainty.

- Indeed, epistemic uncertainty partially comprises aleatoric uncertainty (high $EU$ implies high uncertainty about $AU$, showing interaction between both).
Ensemble methods for uncertainty quantification

\[ \hat{y}_1 = h_1(x) \quad \hat{y}_2 = h_2(x) \quad \ldots \quad \hat{y}_M = h_M(x) \]
Ensemble methods for uncertainty quantification

Ensemble can be seen as an approximation of a distribution.

\[ \hat{y}_1 = h_1(x) \quad \hat{y}_2 = h_2(x) \quad \ldots \quad \hat{y}_M = h_M(x) \]
Ensemble methods for uncertainty quantification

Ensemble can be seen as an approximation of a distribution.

Intuitively, diversity is an indicator of epistemic uncertainty.
Bayesian agents: Ensemble-based approximation

Based on an ensemble of hypotheses $h_1, \ldots, h_M$, producing respective predictions $p_1, \ldots, p_M$, an approximation of conditional entropy can be obtained by

$$AU(x) \approx - \sum_{i=1}^{M} \sum_{y \in Y} p_i(y|x) \log_2 p_i(y|x),$$

an approximation of total uncertainty (Shannon entropy) by

$$TU(x) \approx - \sum_{y \in Y} \left( \frac{1}{M} \sum_{i=1}^{M} p_i(y|x) \right) \log_2 \left( \frac{1}{M} \sum_{i=1}^{M} p_i(y|x) \right),$$

and an approximation of epistemic uncertainty (mutual information) by the difference.
Bayesian agents: Ensemble-based approximation

Based on an ensemble of hypotheses $h_1, \ldots, h_M$, producing respective predictions $p_1, \ldots, p_M$, an approximation of conditional entropy can be obtained by

$$\text{AU}(x) := -\frac{1}{M} \sum_{i=1}^{M} \sum_{y \in \mathcal{Y}} p_i(y \mid x) \log_2 p_i(y \mid x),$$

an approximation of total uncertainty (Shannon entropy) by

$$\text{TU}(x) := -\sum_{y \in \mathcal{Y}} \left( \frac{1}{M} \sum_{i=1}^{M} p_i(y \mid x) \right) \log_2 \left( \frac{1}{M} \sum_{i=1}^{M} p_i(y \mid x) \right),$$

and an approximation of epistemic uncertainty (mutual information) by the difference.
Bayesian agents: Ensemble-based approximation

Epistemic uncertainty thus defined is equivalent to Jensen-Shannon divergence of the distributions $p_i(y|x)$, $i = 1, \ldots, M$. Low divergence high divergence

$105 / 128$
Bayesian agents: Ensemble-based approximation

- Epistemic uncertainty thus defined is equivalent to **Jensen-Shannon divergence** of the distributions $p_i(y \mid x)$, $i = 1, \ldots, M$.
Bayesian agents: Ensemble-based approximation
For neural networks, it has been shown that techniques such as Dropout (?) and DropConnect (?) can be interpreted as (implicit) ensemble methods, and can hence be used to implement this approach.
Bayesian agents: Ensemble-based approximation

- For neural networks, it has been shown that techniques such as **Dropout** (?) and **DropConnect** (?) can be interpreted as (implicit) ensemble methods, and can hence be used to implement this approach.

- Of course, any other ensemble technique could be used as well.
Bayesian agents: Ensemble-based approximation

We proposed an implementation based on Random Forests, using decision trees that predict probabilities in terms of (Laplace-corrected) relative frequencies. Empirically, there are no significant performance differences between neural networks and random forests.
Bayesian agents: Ensemble-based approximation

We proposed an implementation based on **Random Forests**, using decision trees that predict probabilities in terms of (Laplace-corrected) relative frequencies (?).
Bayesian agents: Ensemble-based approximation

- We proposed an implementation based on **Random Forests**, using decision trees that predict probabilities in terms of (Laplace-corrected) relative frequencies (?).

  - Empirically, there are no significant performance differences between neural networks and random forests.
Direct (epistemic) uncertainty prediction

- Consider a (level-1) loss for **probabilistic predictions**:

  \[ \ell_1 : \mathbb{P}(\mathcal{Y}) \times \mathcal{Y} \rightarrow \mathbb{R} \]

- Recall that ERM yields good (unbiased) predictors if \( \ell_1 \) is a (strictly) **proper scoring rule**, which incentivises the learner to predict the true \( p(y | x) \).

- **Question**: Can we do the same on the **epistemic level**, i.e., train a predictor

  \[ h : \mathcal{X} \rightarrow \mathbb{P}(\mathbb{P}(\mathcal{Y})) \]

  by minimising a **level-2 loss**

  \[ \ell_2 : \mathbb{P}(\mathbb{P}(\mathcal{Y})) \times \mathcal{Y} \rightarrow \mathbb{R} , \]

  such that the predictor represents its epistemic uncertainty in a faithful way?
Predicting a Dirichlet distribution
A Dirichlet distribution $\text{Dir}(\alpha)$ is specified by means of $K \geq 2$ positive real-valued parameters, i.e., a vector $\alpha = (\alpha_1, \ldots, \alpha_K) \in \mathbb{R}_+^K$. The probability density function is defined on the $K$-simplex $\Delta^K = \{ \theta = (\theta_1, \ldots, \theta_K)^\top | \theta_1, \ldots, \theta_K \geq 0, \sum_{k=1}^K \theta_k = 1 \}$ and given as follows:

$$p(\theta | \alpha) = p(\theta_1, \ldots, \theta_K | \alpha) = \frac{1}{B(\alpha)} \prod_{k=1}^K \theta^{\alpha_k - 1}_k,$$

where the normalisation constant is the multivariate Beta function. In Bayesian statistics, the Dirichlet distribution is commonly used as the conjugate prior of the multinomial distribution.
A Dirichlet distribution Dir(\(\alpha\)) is specified by means of \(K \geq 2\) positive real-valued parameters, i.e., a vector \(\alpha = (\alpha_1, \ldots, \alpha_K) \in \mathbb{R}_+^K\).

The probability density function is defined on the \(K\) simplex

\[
\Delta_K = \left\{ \boldsymbol{\theta} = (\theta_1, \ldots, \theta_K)^\top \mid \theta_1, \ldots, \theta_K \geq 0, \sum_{k=1}^{K} \theta_k = 1 \right\}
\]

and given as follows:

\[
p(\boldsymbol{\theta} \mid \alpha) = p(\theta_1, \ldots, \theta_K \mid \alpha) = \frac{1}{\mathbb{B}(\alpha)} \prod_{k=1}^{K} \theta_k^{\alpha_k-1},
\]

where the normalisation constant is the multivariate Beta function.
The Dirichlet distribution

- A **Dirichlet distribution** $\text{Dir}(\alpha)$ is specified by means of $K \geq 2$ positive real-valued parameters, i.e., a vector $\alpha = (\alpha_1, \ldots, \alpha_K) \in \mathbb{R}_+^K$.

- The probability density function is defined on the $K$ simplex

$$\Delta_K = \left\{ \theta = (\theta_1, \ldots, \theta_K)^\top \mid \theta_1, \ldots, \theta_K \geq 0, \sum_{k=1}^K \theta_k = 1 \right\}$$

and given as follows:

$$p(\theta \mid \alpha) = p(\theta_1, \ldots, \theta_K \mid \alpha) = \frac{1}{\mathbb{B}(\alpha)} \prod_{k=1}^K \theta_k^{\alpha_k-1},$$

where the normalisation constant is the multivariate Beta function.

- In Bayesian statistics, the Dirichlet distribution is commonly used as the conjugate prior of the **multinomial distribution**.
The Dirichlet distribution

- Dirichlet distribution with parameters $\alpha = (\alpha_1, \alpha_2) = (1, 1), (5, 5), (3, 6), (3, 12)$. 

[Graph showing the probability of positive class for different Dirichlet distributions]
Several authors proposed to do empirical loss minimisation for a level-2 loss of the form

$$
\ell_2(Q, y) = \mathbb{E}_{\theta \sim Q} \ell_1(\theta, y),
$$

where $Q$ is the level-2 prediction for a query instance $x$. 
Direct epistemic uncertainty prediction

Several authors proposed to do **empirical loss minimisation** for a **level-2 loss** of the form

\[ \ell_2(Q, y) = \mathbb{E}_{\theta \sim Q} \ell_1(\theta, y), \]

where \( Q \) is the level-2 prediction for a query instance \( x \).

Thus, an individual prediction \( Q \) is penalised in terms of the **expected** level-1 loss, with the expectation taken over the realisations of \( \theta \).
Several authors proposed to do empirical loss minimisation for a level-2 loss of the form

\[ \ell_2(Q, y) = \mathbb{E}_{\theta \sim Q} \ell_1(\theta, y), \]

where \( Q \) is the level-2 prediction for a query instance \( x \).

Thus, an individual prediction \( Q \) is penalised in terms of the expected level-1 loss, with the expectation taken over the realisations of \( \theta \).

Examples of level-1 loss include cross entropy (?) and Brier score (?).
Direct epistemic uncertainty prediction

Several authors proposed to do **empirical loss minimisation** for a **level-2 loss** of the form

\[ \ell_2(Q, y) = \mathbb{E}_{\theta \sim Q} \ell_1(\theta, y), \]

where \( Q \) is the level-2 prediction for a query instance \( x \).

Thus, an individual prediction \( Q \) is penalised in terms of the **expected** level-1 loss, with the expectation taken over the realisations of \( \theta \).

Examples of level-1 loss include cross entropy (\( ? \)) and Brier score (\( ? \)).

Besides, a **regularised version** has been proposed:

\[ \ell_2(Q, y) = \mathbb{E}_{\theta \sim Q} \ell_1(\theta, y) + \lambda d_{KL}(Q, Q_0) \]
Informally, we define a level-2 loss function $\ell_2$ as **appropriate** if the following holds for the empirical loss minimiser

$$Q^{(N)} = \arg\min_Q \frac{1}{N} \sum_{n=1}^{N} \ell_2 \left( Q, y^{(n)} \right)$$

on any i.i.d. observational data sequence $y^{(1)}, y^{(2)}, \ldots$ with $y^{(i)} \sim \theta^*$:

(A1) The learner’s uncertainty gradually decreases (in expectation) with increasing sample size $N$, in terms of a suitable uncertainty measure $U$.

(A2) In the limit $N \to \infty$, all epistemic uncertainty disappears and $Q^{(N)} \to \delta_{\theta^*}$. 
A negative result

- We formally proved that a loss minimisation approach using a level-2 loss as specified above does not lead to an appropriate level-2 loss (?).
A negative result

- We formally proved that a loss minimisation approach using a level-2 loss as specified above does not lead to an appropriate level-2 loss (?).

- The results are general in the sense that $Q$ can be any level-2 distribution, not necessarily restricted to Dirichlet distributions.
A negative result

- We formally proved that a loss minimisation approach using a level-2 loss as specified above does not lead to an appropriate level-2 loss (?).

- The results are general in the sense that $Q$ can be any level-2 distribution, not necessarily restricted to Dirichlet distributions.

- Moreover, the results do not depend on the underlying uncertainty measure $U$, as long as $U$ is not constant, maximal for the uniform distribution and minimal for Dirac measures.
A negative result

- We formally proved that a loss minimisation approach using a level-2 loss as specified above does not lead to an appropriate level-2 loss (?).

- The results are general in the sense that $Q$ can be any level-2 distribution, not necessarily restricted to Dirichlet distributions.

- Moreover, the results do not depend on the underlying uncertainty measure $U$, as long as $U$ is not constant, maximal for the uniform distribution and minimal for Dirac measures.

- The results reveal the following problem of the loss minimisation approach: The quality of a (level-2) prediction $Q$ cannot be judged solely in the context of (level-0) observations $y$. 
Generalised uncertainty calculi

- Credal sets
- Coherent lower/upper probabilities
  - 2-monotone capacities
  - Random sets
  - Comonotonic clouds
  - P-boxes
  - Probabilities

- Clouds
  - Possibilities

"generalizes"
Sets versus distributions

- Two basic ways to represent uncertainty:
  - (sub-)sets of possible alternatives/outcomes,
  - (probability) distributions
Sets versus distributions

- Sets are easier to specify and appear to provide weaker information (support of a distribution, sets as approximations, e.g., confidence intervals in statistics).
- More importantly, sets and distributions have different semantics (chance vs. imprecision), suggesting different operations (probability calculus vs. constraint processing, additivity vs. “maxitivity”).
- Moreover, sets appear to be more suitable for representing a lack of knowledge. Naturally, the larger a set, the weaker the knowledge.
- The (un-)suitability of probability distributions to represent ignorance has been discussed quite controversially in the literature:
  - precise knowledge about a random event versus complete ignorance,
  - lack of invariance under reparametrisation (e.g., $X$ versus $X^3$),
  - (“non-informative”) priors in Bayesian inference,
  - ...
Credal uncertainty representation

Probabilistic learner
\[ h : \mathcal{X} \rightarrow \mathbb{P}(\mathcal{Y}) \]

Bayesian learner
\[ h : \mathcal{X} \rightarrow \mathbb{P}(\mathbb{P}(\mathcal{Y})) \]

Credal learner
\[ h : \mathcal{X} \rightarrow 2^{\mathbb{P}(\mathcal{Y})} \]

---

**ENSEMBLE-BASED UNCERTAINTY QUANTIFICATION: BAYESIAN VERSUS CREDAL INFEERENCE**

A PREPRINT

Mohammad Hossein Shaker  
Department of Computer Science  
Paderborn University  
Paderborn, Germany  
mhshaker@mail.upb.de

Eyke Hüllermeier  
Institute of Informatics  
University of Munich (LMU)  
Munich, Germany  
eyke@lmu.de

December 13, 2021
Credal machine learning
Credal machine learning

- Analyse the measures **empirically** in the context of **credal machine learning**.
A new measure

They also propose and axiomatically justify a new measure of total predictive uncertainty, more tailored to the ML setting, as well as its decomposition into aleatoric and epistemic uncertainty.

In the case of binary classification, where a credal prediction is of the form $Q = [a, b] \subseteq [0, 1]$, the measure is given as follows:

$$TP(a, b) = \min(1 - a, b)$$

$$\text{total} = \min(a, 1 - b) + (b - a)$$

The total uncertainty is composed of the aleatoric (AP) and epistemic (EP) uncertainties.
They also propose and axiomatically justify a new measure of total predictive uncertainty, more tailored to the ML setting, as well as its decomposition into aleatoric and epistemic uncertainty.
A new measure

- They also propose and axiomatically justify a new measure of **total predictive uncertainty**, more tailored to the ML setting, as well as its decomposition into aleatoric and epistemic uncertainty.

- In the case of **binary classification**, where a credal prediction is of the form

  \[ Q = [a, b] \subseteq [0, 1], \]

  the measure is given as follows:

\[
TP(a, b) = \min(1 - a, b) = \underbrace{\min(a, 1 - b)}_{\text{total}} + \underbrace{(b - a)}_{\text{epistemic (EP)}} + \underbrace{\text{aleatoric (AP)}}_{\text{total}}
\]
Epistemic uncertainty representation

- A credal set $Q$ captures both aleatoric and epistemic uncertainty (lack of knowledge), also called conflict and nonspecificity, respectively.

- How to quantify the total uncertainty represented by a credal set, and how to disaggregate it into aleatoric and epistemic uncertainty?

$$TU(Q) = AU(Q) + EU(Q)$$
Measures of total, aleatoric, and epistemic uncertainty

A well-founded generalisation of entropy and natural measure of total uncertainty is the upper entropy:

\[ S^*(Q) = \max_{q \in Q} S(q) \]

A well-founded measure of epistemic uncertainty is the generalised Hartley measure:

\[ GH(Q) = \sum_{A \subseteq Y} m_Q(A) \log(|A|) \]

which extends the Hartley measure \( H(A) = \log(|A|) \) from sets to graded sets.

Although an equally well-justified measure of aleatoric uncertainty (conflict) in the form of an extension of Shannon entropy has not been found so far (\( ? \)), the lower entropy is a natural measure of irreducible uncertainty:

\[ S^*(Q) = \min_{q \in Q} S(q) \]
A well-founded generalisation of entropy and natural measure of total uncertainty is the **upper entropy**:

\[ S^* (Q) := \max_{q \in Q} S(q) \]
Measures of total, aleatoric, and epistemic uncertainty

- A well-founded generalisation of entropy and natural measure of total uncertainty is the upper entropy:

  \[ S^*(Q) := \max_{q \in Q} S(q) \]

- A well-founded measure of epistemic uncertainty is the generalised Hartley measure

  \[ GH(Q) := \sum_{A \subseteq \mathcal{Y}} m_Q(A) \log(|A|), \]

  which extends the Hartley measure \( H(A) := \log(|A|) \) from sets to graded sets.
Measures of total, aleatoric, and epistemic uncertainty

- A well-founded generalisation of entropy and natural measure of total uncertainty is the **upper entropy**:
  \[ S^*(Q) := \max_{q \in Q} S(q) \]

- A well-founded measure of **epistemic uncertainty** is the **generalised Hartley measure**
  \[ \text{GH}(Q) := \sum_{A \subseteq Y} m_Q(A) \log(|A|), \]
  which extends the Hartley measure \( H(A) := \log(|A|) \) from sets to graded sets.

- Although an equally well-justified measure of **aleatoric uncertainty** (conflict) in the form of an extension of Shannon entropy has not been found so far (?), the **lower entropy** is a natural measure of **irreducible uncertainty**:
  \[ S_*(Q) := \min_{q \in Q} S(q) \]
Disaggregation

There is no additive decomposition 
\[\text{TU}(Q) = \text{AU}(Q) + \text{EU}(Q)\]
such that all three measures behave well. 

Idea: Fix two "good" measures and derive the third one in terms of the difference. 

\[S^*(Q) = (S^*(Q) - \text{GH}(Q)) + \text{GH}(Q)\]
\[S^*(Q) = S^*(Q) + (S^*(Q) - S^*(Q))\]

We provide a critical discussion of such decompositions in the paper and isolate potential deficiencies.
There is no additive decomposition

\[ TU(Q) = AU(Q) + EU(Q) \]

such that all three measures behave well.
Disaggregation

- There is no \textit{additive decomposition}

\[
TU(Q) = AU(Q) + EU(Q)
\]

such that all three measures behave well.

- Idea: Fix two “good” measures and \textbf{derive} the third one in terms of the \textit{difference}.

\[
S^*(Q) = \left( S^*(Q) - GH(Q) \right) + GH(Q)
\]

\[
S^*(Q) = S^*(Q) + \left( S^*(Q) - S^*(Q) \right)
\]

We provide a \textit{critical discussion} of such decompositions in the paper and isolate potential deficiencies.
Disaggregation

- There is no additive decomposition

\[ TU(Q) = AU(Q) + EU(Q) \]

such that all three measures behave well.

- Idea: Fix two “good” measures and derive the third one in terms of the difference.

\[ S^*(Q) = (S^*(Q) - GH(Q)) + GH(Q) \]

\[ S^*(Q) = S_*(Q) + (S^*(Q) - S_*(Q)) \]

- We provide a critical discussion of such decompositions in the paper and isolate potential deficiencies.
Empirical evaluation

Ensemble-based construction of credal predictions.

Accuracy-rejection curves: Allow the learner to reject the $r\%$ presumably most uncertain test cases and measure accuracy on the remaining ones.

If allowed to reject 50% of the cases, the red learner manages to increase accuracy from 0.25 to 0.75.
Empirical evaluation

- **Ensemble-based construction** of credal predictions.
Empirical evaluation

- **Ensemble-based construction** of credal predictions.

- **Accuracy-rejection curves**: Allow the learner to reject the $r \%$ presumably most uncertain test cases and measure accuracy on the remaining ones.

If allowed to reject 50% of the cases, the red learner manages to increase accuracy from 0.25 to 0.75.
Results convey a clear picture: Formally justified measures show strong performance, whereas the "derived" measures perform very poorly. Newly proposed measure yields the only decomposition of total into aleatoric and epistemic uncertainty, such that all three measures produce meaningful results. Thus, the potential deficiencies of existing measures and decompositions isolated and highlighted in the paper could be confirmed empirically.
Results

- Results convey a clear picture: Formally justified measures show strong performance, whereas the “derived” measures perform very poorly.
Results

- Results convey a clear picture: Formally justified measures show strong performance, whereas the “derived” measures perform very poorly.

- Newly proposed measure yields the only decomposition of total into aleatoric and epistemic uncertainty, such that all three measures produce meaningful results.

Results

- Results convey a clear picture: Formally justified measures show strong performance, whereas the “derived” measures perform very poorly.

- Newly proposed measure yields the only decomposition of total into aleatoric and epistemic uncertainty, such that all three measures produce meaningful results.

- Thus, the potential deficiencies of existing measures and decompositions isolated and highlighted in the paper could be confirmed empirically.
Summary and Outlook

Uncertainty is of major importance in ML and attracting more and more attention, also due to practical applications. We highlighted the benefits of distinguishing between different types of uncertainty, notably aleatoric and epistemic uncertainty. Currently, uncertainty quantification for ML is developing very dynamically. Most approaches so far neglect model uncertainty, assuming instead that the model is correctly specified, although model misspecification is a common problem in practice. Related to this is the "closed world" assumption, which is often violated in practice, e.g., in the case of OOD data. Usefulness of generalized uncertainty calculi?
Summary and Outlook

- **Uncertainty** is of major importance in ML and attracting more and more attention, also due to practical applications.
Summary and Outlook

- **Uncertainty** is of major importance in ML and attracting more and more attention, also due to practical applications.

- We highlighted the benefits of distinguishing between different types of uncertainty, notably **aleatoric and epistemic uncertainty**.
Summary and Outlook

- **Uncertainty** is of major importance in ML and attracting more and more attention, also due to practical applications.
- We highlighted the benefits of distinguishing between different types of uncertainty, notably **aleatoric and epistemic uncertainty**.
- Currently, **uncertainty quantification** for ML is developing very dynamically.
Summary and Outlook

- **Uncertainty** is of major importance in ML and attracting more and more attention, also due to practical applications.

- We highlighted the benefits of distinguishing between different types of uncertainty, notably **aleatoric and epistemic uncertainty**.

- Currently, **uncertainty quantification** for ML is developing very dynamically.

- Most approaches so far neglect **model uncertainty**, assuming instead that the model is correctly specified, although **model misspecification** is a common problem in practice.
Summary and Outlook

- **Uncertainty** is of major importance in ML and attracting more and more attention, also due to practical applications.
- We highlighted the benefits of distinguishing between different types of uncertainty, notably **aleatoric and epistemic uncertainty**.
- Currently, **uncertainty quantification** for ML is developing very dynamically.
- Most approaches so far neglect **model uncertainty**, assuming instead that the model is correctly specified, although **model misspecification** is a common problem in practice.
- Related to this is the “**closed world” assumption**, which is often violated in practice, e.g., in the case of OOD data.
Summary and Outlook

- **Uncertainty** is of major importance in ML and attracting more and more attention, also due to practical applications.

- We highlighted the benefits of distinguishing between different types of uncertainty, notably **aleatoric and epistemic uncertainty**.

- Currently, **uncertainty quantification** for ML is developing very dynamically.

- Most approaches so far neglect **model uncertainty**, assuming instead that the model is correctly specified, although **model misspecification** is a common problem in practice.

- Related to this is the “**closed world**” assumption, which is often violated in practice, e.g., in the case of OOD data.

- Usefulness of **generalized uncertainty calculi**?
Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods

Eyke Hüllermeier & Willem Waegeman

Machine Learning 110, 457–506 (2021) | Cite this article

Abstract

The notion of uncertainty is of major importance in machine learning and constitutes a key element of machine learning methodology. In line with the statistical tradition, uncertainty has long been perceived as almost synonymous with standard probability and probabilistic predictions. Yet, due to the steadily increasing relevance of machine learning for practical applications and related issues such as safety requirements, new problems and challenges have recently been identified by machine learning scholars, and these problems may call for new methodological developments. In particular, this includes the importance of distinguishing between (at least) two different types of uncertainty, often referred to as aleatoric and epistemic. In this paper, we provide an introduction to the topic of uncertainty in machine learning as well as an overview of attempts so far at handling uncertainty in general and formalizing this distinction in particular.