Neural Networks

Lecture 3: Basics of Machine Learning
Real data is noisy and uncertain

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Att 1</td>
<td>Att 2</td>
</tr>
<tr>
<td>0.25</td>
<td>red</td>
</tr>
<tr>
<td>0.25</td>
<td>red</td>
</tr>
<tr>
<td>0.99</td>
<td>green</td>
</tr>
<tr>
<td>1.02</td>
<td>green</td>
</tr>
<tr>
<td>2.05</td>
<td>?</td>
</tr>
<tr>
<td>=</td>
<td>green</td>
</tr>
</tbody>
</table>

- **Class Noise**
  - Contradictory examples
  - Mislabeled examples

- **Attribute Noise**
  - Erroneous values
  - Missing values
  - Don’t care values
Joint and marginal probability

If there are 2 or more random variables, say \(X\) and \(Y\), we can consider their **joint** probability of taking a particular pair of values, \(P(X=x, Y=y)\).
We can think of a neural network as representing a function \( f(x, w) \)

The output of this function is not direct predictions of the value \( y \)

\( f(x, w) \) provides the parameters for a distribution over \( y \)

\[
p(y|x; w) = \text{Gaussian}(y; f(x, w), 1)
\]
<table>
<thead>
<tr>
<th>Lesson</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Probability and Information Theory</td>
</tr>
<tr>
<td>3</td>
<td>Basics of Machine Learning</td>
</tr>
<tr>
<td>4</td>
<td>Deep Feed-forward networks</td>
</tr>
<tr>
<td>5</td>
<td>Back-propagation</td>
</tr>
<tr>
<td>6</td>
<td>Optimization and regularization</td>
</tr>
<tr>
<td>7</td>
<td>Convolutional Networks</td>
</tr>
<tr>
<td>8</td>
<td>Sequence Modeling: recurrent networks</td>
</tr>
<tr>
<td>9</td>
<td>Applications</td>
</tr>
<tr>
<td>10</td>
<td>Software and Practical methods</td>
</tr>
<tr>
<td>11</td>
<td>Representational Learning</td>
</tr>
<tr>
<td>12</td>
<td>Deep Generative Models</td>
</tr>
<tr>
<td>13</td>
<td>Deep Reinforcement Learning</td>
</tr>
<tr>
<td>14</td>
<td>Deep Learning and the Brain</td>
</tr>
<tr>
<td>15</td>
<td>Conclusions and Future Perspective</td>
</tr>
<tr>
<td>16</td>
<td>Project presentations</td>
</tr>
</tbody>
</table>
Machine learning

Algorithms that learn from data
Machine learning

Algorithms that learn from data

“A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$.”

Melanie Mitchell
The Task
The Task
The Task

It is very difficult to describe how we do face recognition or write a program that specifies manually how to recognise different objects
The Task

It is very difficult to describe how we do face recognition or write a program that specifies manually how to recognise different objects.

Learn from examples!
The Task

It is very difficult to describe how we do face recognition or write a program that specifies manually how to recognise different objects.

Learn from examples!

Each example is a collection of features (e.g. values of pixels) plus a label.
The Task

**Classification**: which of $k$ categories an input belongs to. We ask to produce a function

$\rightarrow \{\text{“cat”}, \text{“dog”}\}$
The Task

**Classification**: which of $k$ categories an input belongs to. We ask to produce a function

$$f : \mathbb{R}^n \rightarrow \{1, \ldots, k\}$$

$$y = f(x)$$

→  

{“cat”, “dog”}
The Task

**Classification:** which of $k$ categories an input belongs to. We ask to produce a function

$$f : \mathbb{R}^n \rightarrow \{1, \ldots, k\}$$

$$y = f(x)$$

$\rightarrow$ {“cat”, “dog”}
The Task

**Classification**: which of $k$ categories an input belongs to. We ask to produce a function

$$f : \mathbb{R}^n \rightarrow \{1,...,k\}$$

$$y = f(x) \quad \rightarrow \quad y = (p_1,...,p_k)$$

$$\rightarrow \quad \{\text{“cat”, “dog”}\}$$
The Task

Classification with missing inputs:

We must learn 1 function for classifying x with a different subset of inputs
The Task

Classification with missing inputs:

We must learn 1 function for classifying $x$ with a different subset of inputs

a) $2^d$ different classification functions
The Task

Classification with missing inputs:

We must learn 1 function for classifying \( x \) with a different subset of inputs

a) \( 2^d \) different classification functions

b) \( p(y|x_1, x_2, \ldots, x_d) \) & marginalize missing variables
The Task

**Regression**: predict a numerical value given some input. We ask to produce a function
The Task

**Regression**: predict a numerical value given some input. We ask to produce a function

\[ f : \mathbb{R}^n \rightarrow \mathbb{R} \]

\[ y = f(x) \]
The Task

**Regression**: predict a numerical value given some input. We ask to produce a function

\[ f : \mathbb{R}^n \rightarrow \mathbb{R} \]

\[ y = f(x) \quad y = P(x;w) \]
The Task

**Transcription**: convert unstructured representation of some data into discrete, textual form
The Task

Machine translation
The Task

Structured output

Anomaly detection

Synthesis and sampling

Imputation of missing values

Denoising

Density estimation

...
The Performance

Quantitative measure to evaluate the abilities of the algorithm.

**Accuracy**: proportion of examples for which the model produces the *correct* output.

**Error rate**: proportion of examples for which the model produces the *incorrect* output. $E(0\cdot1 \text{ loss})$

**Density estimation**: $E(\log(p_{\text{model}}))$
The *Performance*

Encode the desired behaviour require some choices…

**Regression**: should we penalize more frequent medium-size errors or rare but large-sized mistakes?
The Experience

Most machine learning algorithms experience a fixed dataset (collection of samples)
Types of ML depending of experience

- Supervised: each sample is associated to a label/target

{“cat”, “dog”}
Types of ML depending of experience

- Supervised: each sample is associated to a label/target

\[ P(y|x) \]

\[ f : X \rightarrow Y \]

\{“cat”, “dog”\}

\[ P(y|x) \]
Types of ML depending of experience

- Supervised: each sample is associated to a label/target

\[ P(y|x) \]

\[ f : X \rightarrow Y \]

"dog"
Types of ML depending of experience

- Supervised
- Unsupervised: each sample contains features but no label

\[ P(x) \]
Types of ML depending of experience

- **Supervised**
- **Unsupervised**: each sample contains features but no label

\[
p(x) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1}).
\]

\[
p(y \mid x) = \frac{p(x, y)}{\sum_{y'} p(x, y')}. 
\]
Types of ML depending of experience

• Supervised
• Unsupervised
• Reinforcement: experience via interacting with environment
Example: linear regression

\[ \hat{y} = w^\top x \]

\[ \text{MSE}_{\text{test}} = \frac{1}{m} \| \hat{y}^{(\text{test})} - y^{(\text{test})} \|_2^2 \]

\[ \nabla_w \text{MSE}_{\text{train}} = 0 \]
Data  Model  Cost function  Optimization
Model generalization

Training and test set

Data generating process (iid) \( X \sim P_{\text{data}} \)

Machine learning algorithm \( P_{\text{model}} \)
**Model generalization**

**Underfitting:** model is not able to obtain sufficiently low error in the training set

**Overfitting:** gap between training and test error is too large

**Capacity:** ability to fit a wide variety of functions
Model generalization

![Diagram showing the relationship between model capacity and error, with underfitting and overfitting zones. The diagram illustrates the generalization gap between training error and generalization error.](image)
Model generalization

We can control the model capacity by adding or removing functions in the hypothesis space but also by giving preference to certain functions.

**Regularization:** modification of algorithm intended to decrease the generalization error but not the training error.

\[ J(w) = \text{MSE}_{\text{train}} + \lambda w^T w \]

Underfitting (Excessive \( \lambda \)) | Appropriate weight decay (Medium \( \lambda \)) | Overfitting (\( \lambda \to 0 \))
Validation and testing

Performance is evaluated in a test set (not used for training)
Validation and testing

Performance is evaluated in a test set (not used for training)
Final Accuracy = Average(Round 1, Round 2, ...)
Cross-Validation and Testing

Full data

Cross-validation

Test

Choose the best hyper-parameters with cross-validation

Train 1  Validation 1  ...  Train n  Validation n

Hyper-parameters

Train

Test

Publish hyper-parameters and parameters
Statistical inference

Deducing statistical properties of underlying distribution from analysis of data. Select a statistical model...

\[ X = \{ x^{(1)}, \ldots, x^{(m)} \} \]

\[ p_{\text{data}}(x) \]

\[ p_{\text{model}}(x; \theta) \]

\[ \theta_{\text{ML}} = \arg \max_{\theta} p_{\text{model}}(X; \theta) \]
Maximum likelihood

\[ \theta_{ML} = \arg \max_{\theta} p_{\text{model}}(X; \theta) \]

= arg max \[ \prod_{i=1}^{m} p_{\text{model}}(x^{(i)}; \theta) \] (iid)

\[ \theta_{ML} = \arg \max_{\theta} \sum_{i=1}^{m} \log p_{\text{model}}(x^{(i)}; \theta) \] (monotonicity of log)

\[ \theta_{ML} = \arg \max_{\theta} \mathbb{E}_{x \sim \hat{p}_{\text{data}}} \log p_{\text{model}}(x; \theta) \] (def. of Expectation)
Maximum likelihood

$$\theta_{ML} = \arg \max_{\theta} \mathbb{E}_{x \sim \hat{p}_{\text{data}}} \log p_{\text{model}} (x; \theta)$$

$$D_{KL} (\hat{p}_{\text{data}} || p_{\text{model}}) = \mathbb{E}_{x \sim \hat{p}_{\text{data}}} \left[ \log \hat{p}_{\text{data}} (x) - \log p_{\text{model}} (x) \right]$$

Minimize $$- \mathbb{E}_{x \sim \hat{p}_{\text{data}}} \left[ \log p_{\text{model}} (x) \right]$$

Maximize $$\mathbb{E}_{x \sim \hat{p}_{\text{data}}} \log p_{\text{model}} (x; \theta)$$
Maximum likelihood

Unconditional
(Unsupervised)

\[ \theta_{ML} = \arg \max_{\theta} p_{model}(X; \theta) \]
\[ \theta_{ML} = \arg \max_{\theta} \sum_{i=1}^{m} \log p_{model}(x^{(i)}; \theta) \]

Conditional
(Supervised)

\[ \theta_{ML} = \arg \max_{\theta} P(Y | X; \theta) \]
\[ \theta_{ML} = \arg \max_{\theta} \sum_{i=1}^{m} \log P(y^{(i)} | x^{(i)}; \theta) \]
The figure illustrates a cost function $J(w)$ in a 2D space. The function is depicted as a parabolic curve, indicating that the cost decreases as the weight $w$ moves towards the global minimum $J_{\text{min}}(w)$. The gradient points towards the direction of decreasing cost, guiding the optimization process. The starting point is labeled as the 'Initial weight'. The dashed line indicates the path taken by the optimization algorithm, converging towards the global minimum.
Gradient descent

\[ J(\theta) = \mathbb{E}_{x, y \sim \hat{p}_{\text{data}}} L(x, y, \theta) = \frac{1}{m} \sum_{i=1}^{m} L(x^{(i)}, y^{(i)}, \theta) \]

\[ L(x, y, \theta) = - \log p(y \mid x; \theta) \]

\[ \nabla_\theta J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L(x^{(i)}, y^{(i)}, \theta) \]

\[ \theta \leftarrow \theta - \epsilon g \]
Shallow Machine Learning

0 or 1 abstraction layer (feature transformation)
The 3 options for $\phi(x)$
The 3 options for $\phi(x)$

1. **Generic** (if $\phi(x)$ high-dimensional $\rightarrow$ high-capacity to fit training set but low generalization. E.g. kernel machines)
The 3 options for $\phi(x)$

1. **Generic** (if $\phi(x)$ high-dimensional $\rightarrow$ high-capacity to fit training set but low generalization. E.g. kernel machines)

2. **Manual** (engineer $\phi(x)$ for each separate task. Decades of experience in speech recognition or computer vision)
The 3 options for $\phi(x)$

1. **Generic** (if $\phi(x)$ high-dimensional $\rightarrow$ high-capacity to fit training set but low generalization. E.g. kernel machines)

2. **Manual** (engineer $\phi(x)$ for each separate task. Decades of experience in speech recognition or computer vision)

3. **Learn** (use parameters to learn $\phi(x; \theta)$ from a broad class of functions. Optimization is non-convex but benefits are overwhelming, including advantages of 1 and 2.)
What is deep learning?

many layers of adaptive non-linear processing to model complex relationships among data
Challenges

The curse of dimensionality

Local constancy prior
What is the problem with high-dimensional data?

The curse of dimensionality: sample size to estimate a function of several variables grows exponentially with number of variables (D)
What is the problem with high-dimensional data?

Our intuition fails in high dimensional spaces: \textbf{empty space phenomena}!
What is the problem with high-dimensional data?

Our intuition fails in high dimensional spaces: \textbf{we are sampling from tails!}

<table>
<thead>
<tr>
<th>$n$</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probability</td>
<td>0.04550</td>
<td>0.13534</td>
<td>0.54942</td>
<td>0.94734</td>
<td>0.99995</td>
<td>1.00000</td>
</tr>
</tbody>
</table>

Tail probability of the multivariate Gaussian distributions for different dimensions ( $P(|x| > 2)$ )
What is the problem with high-dimensional data?

Our intuition fails in high dimensional spaces: 
**distances have weak discriminative power!**

Euclidean distance between random vectors (i.i.d.) in high dimensions is approximately constant (k-NN is risky in very high dim.)
What “priors” are used in ML for generalizing to unseen data?

- Priors (kNN, SVM, RF, ...): smoothness
  \[ f^*(x) \approx f^*(x + \epsilon) \]

- To distinguish \( O(k) \) regions in input space, these methods need \( O(k) \) samples
What “priors” are used in ML for generalizing to unseen data?

• It is possible to distinguish $O(2^k)$ regions using $O(k)$ samples if one allows dependencies between the regions via additional assumptions.

• Priors (ANN): smoothness + compositionality

$$y = f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$$

• Deep and distributed representations provide an exponential gain to counter the curse of dimensionality in central AI tasks.
Geometric intuition for non-local generalization with rectifying units
<table>
<thead>
<tr>
<th>Lesson</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Probability and Information Theory</td>
</tr>
<tr>
<td>3</td>
<td>Basics of Machine Learning</td>
</tr>
<tr>
<td>4</td>
<td>Deep Feed-forward networks</td>
</tr>
<tr>
<td>5</td>
<td>Back-propagation</td>
</tr>
<tr>
<td>6</td>
<td>Optimization and regularization</td>
</tr>
<tr>
<td>7</td>
<td>Convolutional Networks</td>
</tr>
<tr>
<td>8</td>
<td>Sequence Modeling: recurrent networks</td>
</tr>
<tr>
<td>9</td>
<td>Applications</td>
</tr>
<tr>
<td>10</td>
<td>Software and Practical methods</td>
</tr>
<tr>
<td>11</td>
<td>Representational Learning</td>
</tr>
<tr>
<td>12</td>
<td>Deep Generative Models</td>
</tr>
<tr>
<td>13</td>
<td>Deep Reinforcement Learning</td>
</tr>
<tr>
<td>14</td>
<td>Deep Learning and the Brain</td>
</tr>
<tr>
<td>15</td>
<td>Conclusions and Future Perspective</td>
</tr>
<tr>
<td>16</td>
<td>Project presentations</td>
</tr>
</tbody>
</table>