MTAT.03.227 Machine Learning
Practice session 8

November 4-6, 2019

Exercise 1. Cross-entropy loss

Cross entropy loss, also known as log-loss, can be used to define the loss function in machine learning and optimization.

$$CE = \frac{1}{n}\sum_{i=1}^{n}(-y_i \log \hat{p}(x_i) - (1-y_i)\log(1-\hat{p}(x_i)))$$

In the lecture we saw that in *logistic regression* we could use the log-loss a cost function to minimize in order to find the best-fitting logistic curve on 2 class data.

In the below table you are given scores from 2 different models and the true labels (n=8):

True label	Model 1	Model 2
1	0.9	0.5
1	0.8	0.6
0	0.5	0.4
0	0.1	0.3
1	0.3	0.2
0	0.1	0.5
0	0.2	0.8
1	0.7	0.2

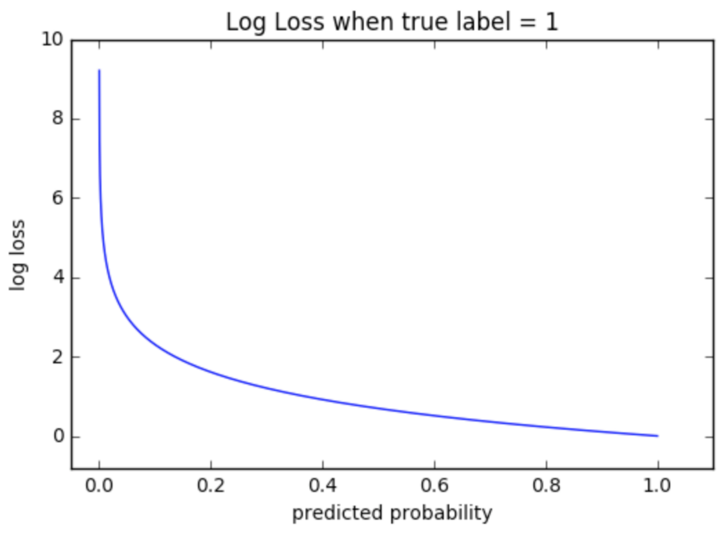
(a) Just by looking at the table, what do you think, which of the models has lower cross-entropy loss?

Answer: Model 1 should have lower cross-entropy loss, as it has closer probability estimate to the true label than Model 2 for most of the instances.

(b) Check your answer by calculating the losses for both of these models.

Answer: loss of the first model is about 0.377, loss of the second model is about 0.949 (if natural logarithm was used).

(c) In the below graph you are given the log-loss values in case of fixed true class $y = 1$ when changing the assigned score from 0 to 1. How would the plot look like in case of $y = 0$? *Answer:* see Jupyter notebook of this practice session.



(d) What is the difference between entropy and cross-entropy measures?

Hint: entropy is defined as: $E = -\sum_{i=0}^{n} p(X_i) \cdot log(p(X_i))$

Answer: one of the main difference is that entropy is a measure of single probability distribution (X in the formula), while cross-entropy measures difference between two distributions y and $\hat{p}(x)$.

Exercise 2. Gradient descent

In logistic regression, our goal is to minimize the cross-entropy loss:

$$\arg\min_{\boldsymbol{w},b} \frac{1}{n} \sum_{i=1}^{n} \left(-y_i \log \hat{p}(x_i) - (1-y_i) \log(1-\hat{p}(x_i))\right)$$

One way to do this is to use gradient descent (GD) or stochastic gradient descent (SGD) algorithms.

The goal of the algorithms is to minimize function $f(\theta)$. In SGD, we use the fact that the function to optimize can be written in form:

$$f(\theta) = \sum_{i=1}^{n} f_i(\theta)$$

The following pseudocodes show how these algorithms work.

Algorithm 1 Gradient Descent

```
1: procedure GD
2:     initialize $\theta$
3:     while not converged do
4:         $\theta \leftarrow \theta - \eta \nabla f(\theta)$
       return $\theta$
```

Algorithm 2 Stochastic Gradient Descent

```
1: procedure SGD
2:     initialize $\theta$
3:     while not converged do
4:         randomly shuffle the indices $1 \ldots n$
5:         for each index $i$ do
6:             $\theta \leftarrow \theta - \eta \nabla f_i(\theta)$
       return $\theta$
```

(a) What is θ in logistic regression? What is $f(\theta)$? What is $\nabla f(\theta)$?

Answer: θ is a vector of logistic regression parameters: bias b and weights w_i, where each w_i correspond to one feature. $f(\theta)$ is a cross-entropy of the model that uses parameters θ. $\nabla f(\theta)$ is a gradient of cross-entropy with respect to θ.
Logistic regression formula is the following:

$$\hat{p}(x) = \frac{1}{1 + \exp\left(-b - \sum_{i=0}^{m} w_i \cdot x_i\right)} \tag{1}$$

(b) What is $f_i(\theta)$? What is $\nabla f_i(\theta)$?

Answer: $f_i(\theta)$ is a cross-entropy of the single i-th instance. $\nabla f_i(\theta)$ is a correspondent gradient with respect to θ.

(c) What is the difference between GD and SGD?

Answer: GD uses gradient of the total cross-entropy computed at all training instances, while SGD uses gradient of the cross-entropy on the single instance. It means that SGD updates weights once per instance, while GD - once per iteration over whole dataset.

Hint: Slide 68 of lecture 8

Exercise 3. Bayes-optimality

Bayes optimal model is

$$p^*(x) = \frac{P(X = x|Y = 1)}{P(X = x|Y = 0) + P(X = x|Y = 1)}$$

Let's assume that we know that in our data the positive and negative classes are normally distributed with same standard deviation and different means. We also assume that the classes are balanced. We observe an instance x (vertical dashed line).

(a) What class label should we predict for x and why?

Answer: we should predict positive label, because probability of X being equal to x given positive class (b) is larger than correspondent probability given negative class (a).

(b) What do a and b show in this plot?

Answer: $a = P(X = x|Y = 0)$, $b = P(X = x|Y = 1)$.

(c) For the instance x find the probability of being positive, expressed in terms of a and b. (*Hint: slide 46 of lecture 8*)

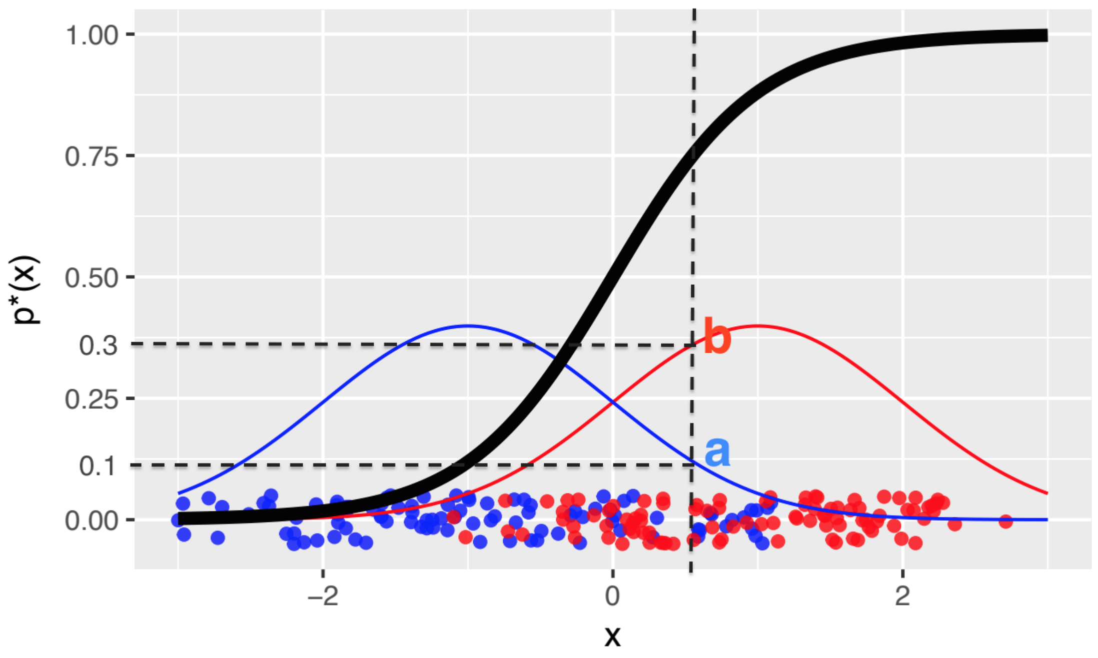
Answer: $P(Y = 1|X = x) = \frac{P(X=x|Y=1)}{P(X=x|Y=0)+P(X=x|Y=1)} = \frac{b}{a+b}$

(d) The predicted probabilities in this setting are equal to the ones given by the Bayes-optimal model. Why can't we learn the Bayes-optimal model from the training data?

Answer: we can't explore true class probability distribution from the training data.

(e) Where is the decision boundary in this plot?

Answer: Decision boundary is at x = 0, where we have an intersection of red and blue curves. At this point, we have that P(X=x | Y = 1) = P(X=x | Y = 0).



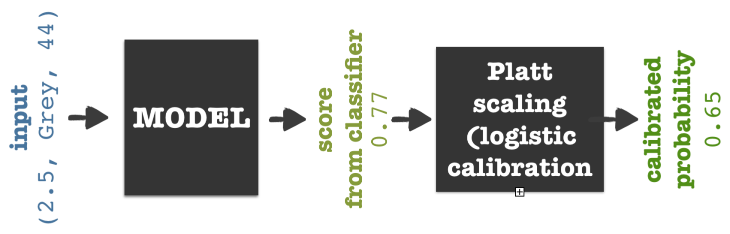
Exercise 4. Calibration

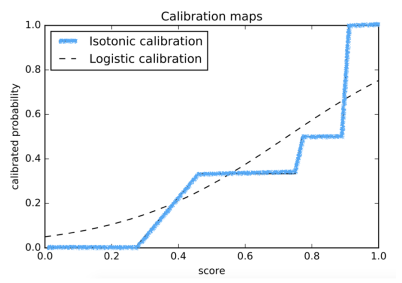
In the lecture we learned what it means for a model to be calibrated.

In practice, the process of calibration is usually the following:

- train a model on your training set
- predict scores with the trained model on a calibration set
- using the predicted scores and true labels, learn a calibration map (e.g using isotonic calibration, Platt scaling, etc).
- in future predictions you apply the model and then the learned calibration map on top of the model's output to get a calibrated probability

The idea is illustrated in the below figure.





(a) If you would have the probability scores and true class labels, how can you check whether the model is calibrated or not? Can you come up with a plot, such that this check could be done visually?

Hint: read this material: https://changhsinlee.com/python-calibration-plot/

Answer: such plot is called calibration (reliability) plot. You can find out more via the provided link. X axis of this plot is predicted probability, Y axis - fraction of positive instances among all instances that have correspondent predicted probability. Well-calibrated model has a diagonal calibration plot (started from (0, 0) to (1, 1)).