IT496: Introduction to Data Mining

Lecture 11

Evaluation - III

[Loss Functions]

Arpit Rana

28th August 2023

Loss Functions

Commonly used Loss Functions

Error Rate to Loss Function

So far, we assumed that the *optimal fit* is the hypothesis that minimizes the *error rate*: the proportion of times that **h(x)** ≠ **y** for an (x, y) example.

However, different errors may have different costs.

Example:

It is worse to classify *non-spam* as *spam* than to classify *spam* as *non-spam*.

- *error rate* = 1%, mostly classifying *spam* as <u>*non-spam*</u>,
- *error rate* = 0.5%, mostly classifying *non-spam* as <u>*spam*</u>.

Loss Function: Definition

The loss function **L(x, y, ŷ)** is defined as the amount of utility lost by predicting **h(x) = ŷ** when the correct answer is **f(x) = y**:

$$L(x, y, \hat{y}) = \mathit{Utility}(\text{result of using } y \text{ given an input } x) - \mathit{Utility}(\text{result of using } \hat{y} \text{ given an input } x)$$

$$= \mathit{Utility}(f(x)) - \mathit{Utility}(\mathbf{h}(x))$$

- Often a simplified version is used, $L(y, \hat{y})$, that is independent of x.

  For example,

  $$L(\mathit{spam}, \mathit{non\text{-}spam}) = 1,\ L(\mathit{non\text{-}spam}, \mathit{spam}) = 10$$

We choose the learner that ***minimizes*** the ***expected loss*** for ***all input-output pairs*** it will see.

Generalization Loss

Let $\mathcal{E}$ be the set of all possible input–output examples follow a prior probability distribution **P(X, Y)**.

Then the expected *generalization loss* for a hypothesis ***h*** (with respect to loss function L) is–

$$GenLoss_L(h) = \sum_{(x,y) \in \xi} L(y, h(x))P(x, y).$$

The best hypothesis **h***, is the one with the minimum expected generalization loss:

$$h^{\cdot} = \underset{h \in \mathcal{H}}{\arg\min}\ GenLoss_L(h)$$

Empirical Loss

Because *P*(X, Y) is not known in most cases, the learner can only estimate generalization loss with *empirical loss* on a set of examples D of size N.

$$EmpLoss_{L,D}(h) = \sum_{(x,y)\,\in\,D} L(y, h(x))\frac{1}{N}.$$

The estimated best hypothesis $\mathbf{h}^*$, is the one with the minimum expected empirical loss:

$$\hat{h}^{\cdot} = \underset{h\,\in\mathcal{H}}{\arg\min}\ EmpLoss_{L,D}(h)$$

There are four reasons why $\hat{h}^*$ may differ from the true function, f : *unrealizability*, *variance*, *noise*, and *computational complexity*.

Commonly Used Loss Functions

There's no *one-size-fits-all* loss function to algorithms in machine learning.

- There are various factors involved in choosing a loss function for specific problem, e.g.,
  - type of machine learning algorithm chosen,
  - ease of calculating the derivatives, and
  - to some degree the percentage of outliers in the data set.

Broadly, loss functions can be classified into two major categories — *Regression losses* and *Classification losses*.

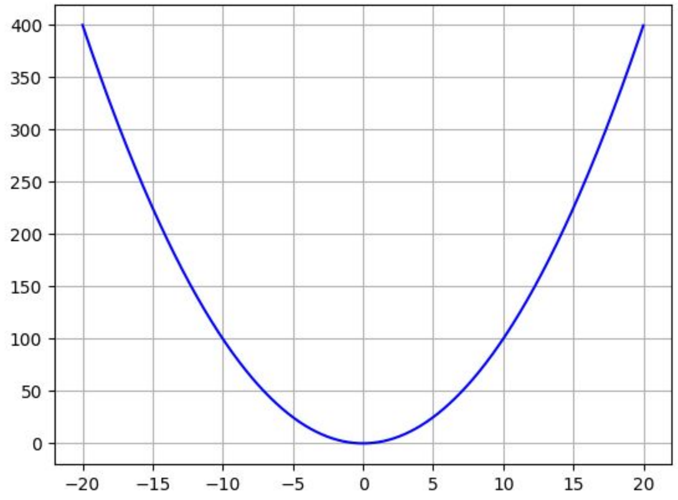
Regression Losses

Mean Squared Error (L2 loss)

This is almost every data scientist's preference when it comes to loss functions for regression. Because most variables can be modeled into a Gaussian distribution.

This is differentiable.

$$MSE = \frac{1}{n} \sum_{x \in S} (h(x) - y)^2$$



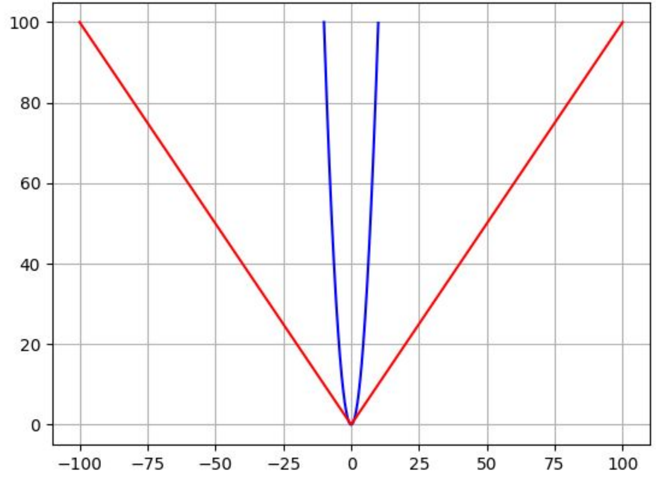
Mean Absolute Error (L1 loss)

This is one of the most simple yet *robust* loss functions used for regression models.

Regression problems may have variables that are not strictly *Gaussian* in nature due to the presence of outliers (values that are very different from the rest of the data).

L1 loss is an ideal option in such cases.

$$MAE = \frac{1}{n}\sum_{x\in S} |h(x) - y|$$



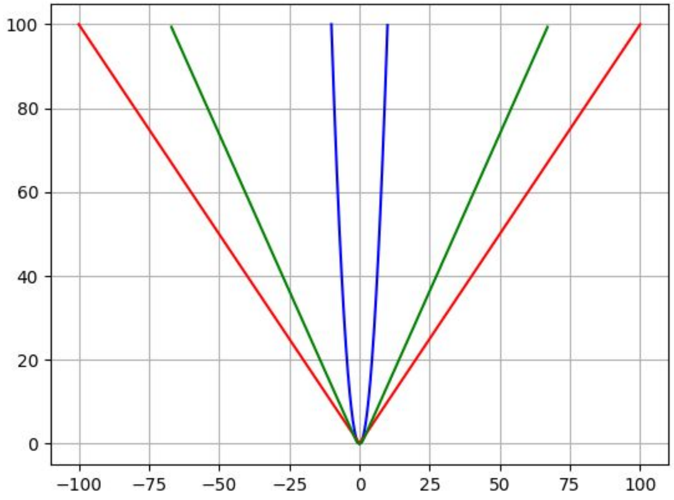
MAE (red), MSE (blue) loss functions

Huber Loss

Now we know that the MSE is great for outliers while the MAE is great for ignoring them. But what about something in the middle?

The Huber Loss offers the best of both worlds by balancing the MSE and MAE together. We can define it using the following piecewise function:

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta\, |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$



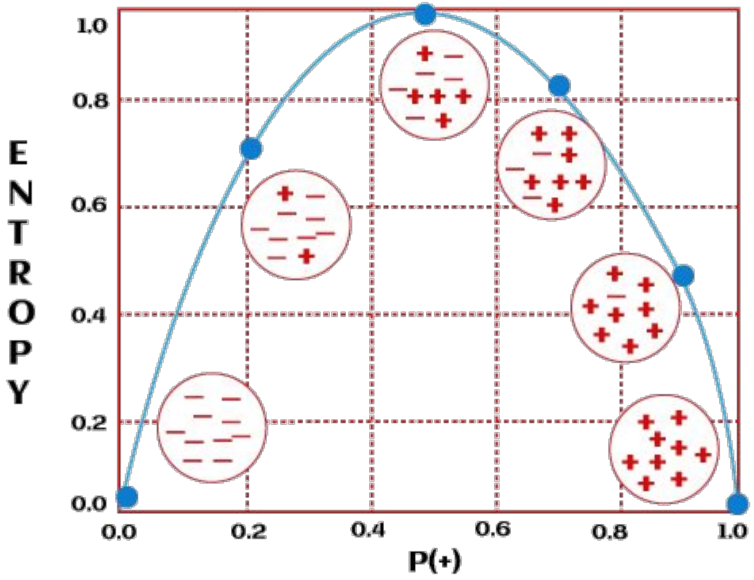
MAE (red), MSE (blue), and Huber (green) Loss functions

Classification Losses

Entropy

Entropy is the measure of uncertainty of a variable: the more uncertain it is, the higher the entropy is, the higher the surprise is, and the lower the probability is.

$$Entropy = -\sum_{x} p(x)\log_e(p(x))$$



Here,

x is a class

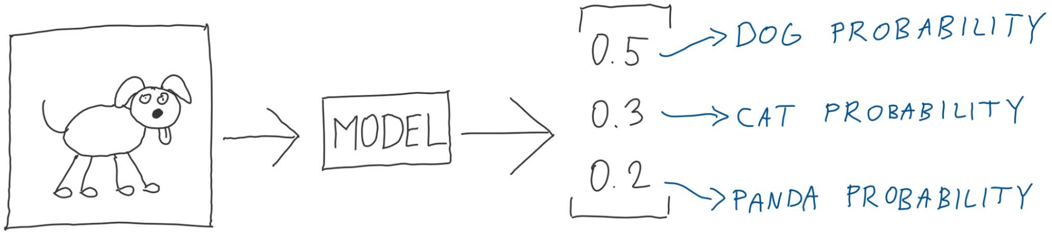
$p(x)$ is the probability of class x in the dataset

Source: Image is taken from the web.

Cross Entropy Loss Function

Which class is on the image — *dog*, *cat*, or *panda*? It can only be one of them (multi-class classification).

Let's have an image of a dog.



The prediction is a ***probability vector***, meaning it represents predicted probabilities of all classes, summing up to 1.

Source: Towards Data Science Blog by Vlastimil Martinek

Cross Entropy Loss Function

- Cross-Entropy

  The general formula, used for calculating loss among two probability vectors: the target and the prediction.

  The more we are away from our target, the more the error grows — similar idea to the square error.

$$L = -\sum_{x} p(x) \log\left(q(x)\right)$$

Here,

x is a class
$p(x)$ is the probability of class x in target
$q(x)$ is the probability of class x in prediction

Cross Entropy Loss Function

The target for multi-class classification is a *one-hot vector*, meaning it has 1 on a single position and 0’s everywhere else.

$$\begin{array}{cc} \text{TARGET} & \text{PREDICTION} \\ \begin{bmatrix}1\\0\\0\end{bmatrix} & \begin{bmatrix}0.5\\0.3\\0.2\end{bmatrix}\end{array}$$

We will start by calculating the loss for each class separately and then summing them. The loss for each separate class is computed like this:

$$\text{Loss for class } X = -\underbrace{p(X)}_{\substack{\text{probability}\\ \text{of class } X\\ \text{in TARGET}}} \cdot \log \underbrace{q(X)}_{\substack{\text{probability}\\ \text{of class } X\\ \text{in PREDICTION}}}$$

Source: Towards Data Science Blog by Vlastimil Martinek

Cross Entropy Loss Function

$$\text{Loss For CAT} = -p(\text{CAT}) \cdot \log q(\text{CAT})$$
$$= -0 \cdot \log q(\text{CAT})$$
$$= 0$$

$$\text{Loss For PANDA} = -p(\text{PANDA}) \cdot \log q(\text{PANDA})$$
$$= -0 \cdot \log q(\text{PANDA})$$
$$= 0$$

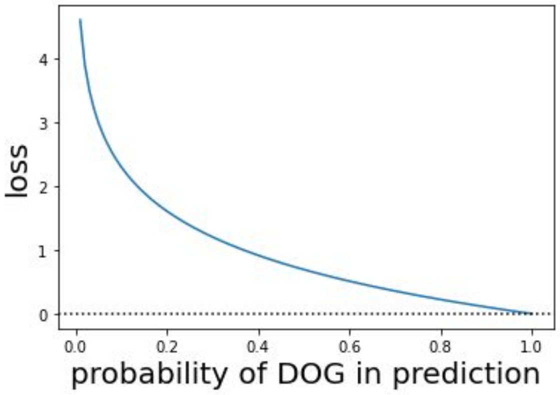
$$\text{Loss For DOG} = -p(\text{DOG}) \cdot \log q(\text{DOG})$$
$$= -1 \cdot \log 0.5$$
$$= 0.693\ldots$$

$$\text{Cross-entropy} = \text{Loss For DOG} + \text{Loss For CAT} + \text{Loss For PANDA}$$
$$= 0.693 + 0 + 0$$
$$= 0.693$$

Source: Towards Data Science Blog by Vlastimil Martinek

Cross Entropy Loss Function

Let's see how would the loss behave if the predicted probability was different:



This shows a similar concept to square error — the further away we are from the target, the faster the error grows.

If we want the loss for the whole dataset, we would just sum up the losses of the individual images.

Source: Towards Data Science Blog by Vlastimil Martinek

Cross Entropy Loss

- Categorical Cross-Entropy (Multi-class Classification)

  If our target is a one-hot vector, we can indeed forget targets and predictions for all the other classes and compute only the loss for the hot class.

  This is the negative natural logarithm of our prediction.

$$L = -\log\left(q(x)\right)$$

x is the class that is 1 in our target.

This is called *categorical cross-entropy* — a special case of cross-entropy, where our target is a one-hot vector.

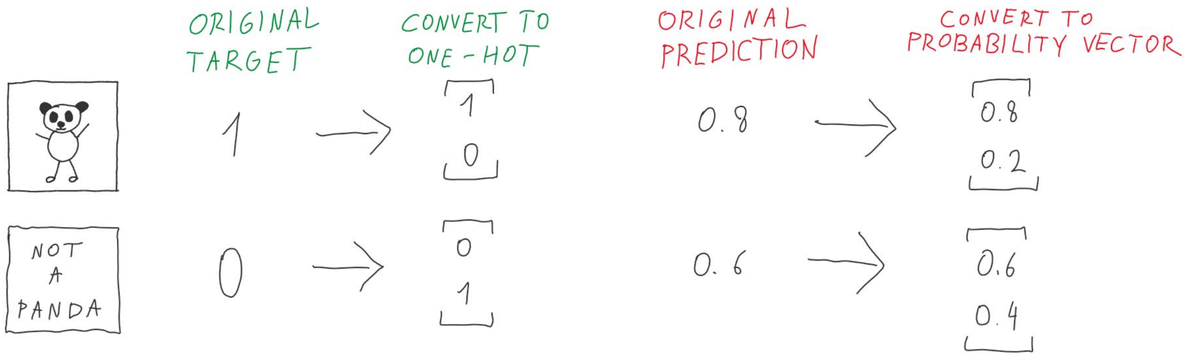
Cross Entropy Loss

- Binary Cross-Entropy (for Binary Classification)

  We use binary cross-entropy — a specific case of cross-entropy where target is 0 or 1.

  It can be computed with the cross-entropy formula if we convert the target to a one-hot vector like [0,1] or [1,0] and the predictions respectively.

  Suppose we want to predict whether the image contains a panda or not.



Source: Towards Data Science Blog by Vlastimil Martinek

Cross Entropy Loss

- Binary Cross-Entropy (for Binary Classification)

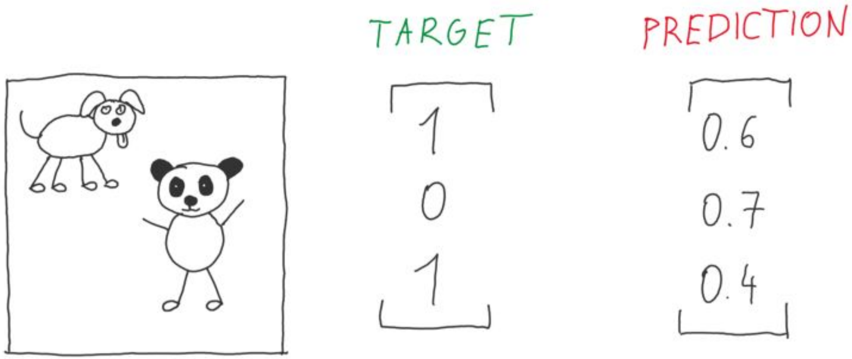
  We can compute it even without this conversion, with the simplified formula.

$$L = -(p(x)\log(q(x)) + (1 - p(x))\log(1 - q(x)))$$

Cross Entropy Loss

- Multi-label Classification (Multiple Binary Class Cross-Entropy)

  Our target can represent multiple (or even zero) classes at once.



Notice our target and prediction are not a probability vector. It's possible that there are all classes in the image, as well as none of them.

Source: Towards Data Science Blog by Vlastimil Martinek

Cross Entropy Loss

- Multi-label Classification (Multiple Binary Class Cross-Entropy)

  We can look at this problem as multiple binary classification subtasks.

$$
\begin{aligned}
\text{Binary Cross-entropy}_{DOG} &= -\big(p(x)\cdot\log q(x) + (1-p(x))\cdot\log(1-q(x))\big)\\
&= -\big(1 \cdot \log 0.6 + (1-1) \cdot \log(1-0.6)\big)\\
&= -\big(\log 0.6 + 0\big)\\
&= 0.510\ldots
\end{aligned}
$$

$$
\begin{aligned}
\text{Binary Cross-entropy}_{PANDA} &= -\big(p(x)\cdot\log q(x) + (1-p(x))\cdot\log(1-q(x))\big)\\
&= -\big(1 \cdot \log 0.4 + (1-1) \cdot \log(1-0.4)\big)\\
&= -\big(\log 0.4 + 0\big)\\
&= 0.916\ldots
\end{aligned}
$$

$$
\begin{aligned}
\text{Binary Cross-entropy}_{CAT} &= -\big(p(x)\cdot\log q(x) + (1-p(x))\cdot\log(1-q(x))\big)\\
&= -\big(0 \cdot \log 0.7 + (1-0) \cdot \log(1-0.7)\big)\\
&= -\big(0 + \log 0.3\big)\\
&= 1.20\ldots
\end{aligned}
$$

$$
\begin{aligned}
\text{Total Loss} &= \text{Binary Cross-entropy}_{DOG}\\
&+ \text{Binary Cross-entropy}_{CAT}\\
&+ \text{Binary Cross-entropy}_{PANDA}\\
&= 2.631\ldots
\end{aligned}
$$

Source: Towards Data Science Blog by Vlastimil Martinek

Cross Entropy Loss

- Multi-label Classification (Multiple Binary Class Cross-Entropy)

  Our target can represent multiple (or even zero) classes at once.

  We compute the binary cross-entropy for each class separately and then sum them up for the complete loss.

$$L = \sum_{x} L_{binary}(x)$$

Here, $L_{binary}(x)$ is the binary cross-entropy loss for class x.

Other Classification Losses

There are other commonly used loss functions available for classification problems.

- **Hinge Loss**: Specifically used in SVM
- **Kullback Leibler Divergence Loss (KL Loss)**: Most commonly used for Neural Networks

$$
\begin{aligned}
D_{KL}(P||Q) &= \sum_x P(x) \log \frac{P(x)}{Q(x)} \\
&= -\sum_x P(x) \log \frac{Q(x)}{P(x)} \\
&= -\left(\sum_x P(x) \log Q(x) - \sum_x P(x) \log P(x) \right) \\
&= H(P, Q) - H(P)
\end{aligned}
$$

Next lecture

Optimization

29th August 2023