A Unified View of Loss Functions in Supervised Learning

Shuiwang Ji
Department of Computer Science & Engineering
Texas A&M University

Linear Classifier

1. For a binary classification problem, we are given an input dataset $X = [x_1, x_2, \ldots, x_n]$ with the corresponding label $Y = [y_1, y_2, \ldots, y_n]$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{+1, -1\}$.
2. For a given sample x_i, a linear classifier computes the linear score s_i as a weighted summation of all features as:

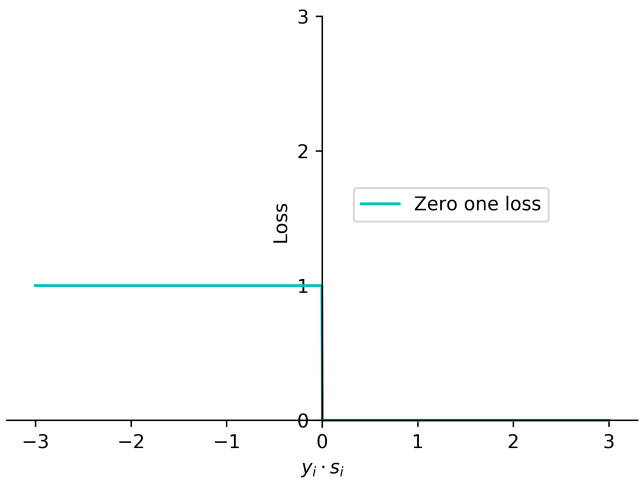
$$s_i = w^T x_i + b, \tag{1}$$

   where w is the weights and b is the bias.
3. We can predict the label of x_i based on the linear score s_i. By employing an appropriate loss function, we can train and obtain a linear classifier.
4. We study the relations between loss values and $y_i s_i$.
5. We describe and compare a variety of loss functions used in supervised learning, including zero-one loss, perceptron loss, hinge loss, log loss, exponential loss, and square loss.
6. We describe these loss functions in the context of linear classifiers, but they can also be used for nonlinear classifiers.

Zero-one Loss

1. The zero-one loss aims at measuring the number of prediction errors for classifier. For a given input x_i, the classifier makes a correct prediction if $y_i s_i > 0$. Otherwise, it makes a wrong prediction.
2. Therefore, the zero-one loss function can be described as $\frac{1}{n}\sum_{i=1}^{n} L_{0/1}(y_i, s_i)$,where $L_{0/1}$ is the zero-one loss defined as

$$L_{0/1}(y_i, s_i) = \begin{cases} 1 & \text{if } y_i s_i < 0, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

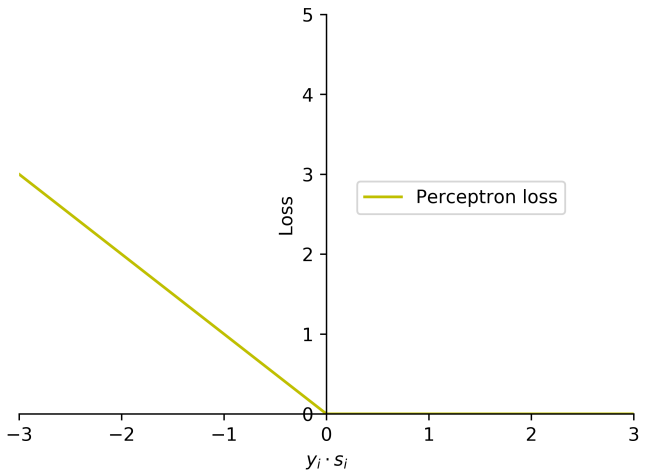


Perceptron loss

1. The zero-one loss incurs the same loss value of 1 for all wrong predictions, no matter how far a wrong prediction is from the hyperplane.
2. The perceptron loss addresses this by penalizing each wrong prediction by the extent of violation. The perceptron loss function is defined as $\frac{1}{n}\sum_{i=1}^{n} L_p(y_i, s_i)$, where L_p is perceptron loss which can be described as

$$L_p(y_i, s_i) = \max(0, -y_i s_i). \tag{3}$$

3. Note that the loss is 0 when the input example is correctly classified. The loss is proportional to a quantification of the extent of violation $(-y_i s_i)$ when the input example is incorrectly classified.



Square loss

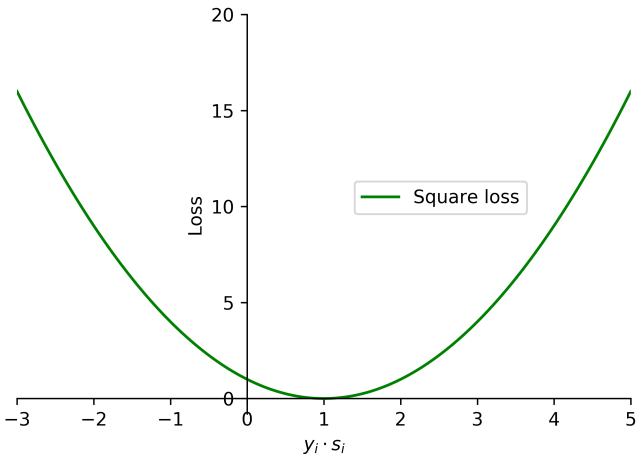
1. The square loss function is commonly used for regression problems.
2. It can also be used for binary classification problems as

$$\frac{1}{n}\sum_{i=1}^{n} L_s(y_i, s_i), \tag{4}$$

where L_s is the square loss, defined as

$$L_s(y_i, s_i) = (1 - y_i s_i)^2. \tag{5}$$

3. Note that the square loss tends to penalize wrong predictions excessively. In addition, when the value of $y_i s_i$ is large and the classifier is making correct predictions, the square loss incurs a large loss value.



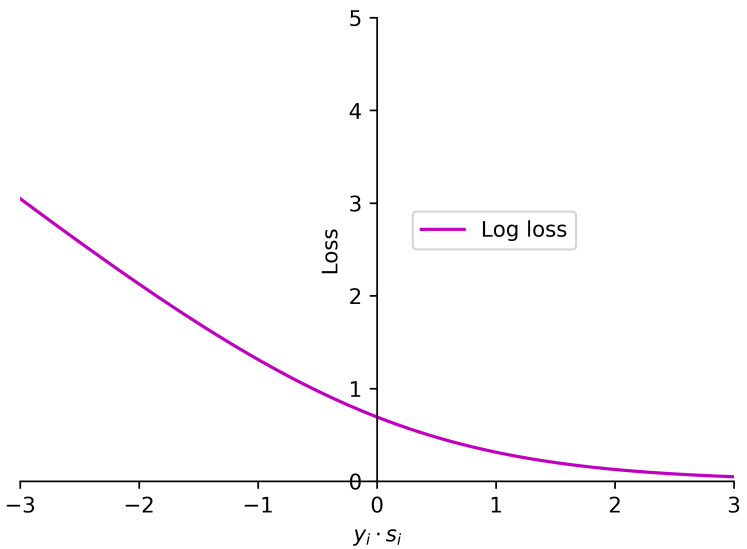
Log loss (cross entropy)

1. Logistic regression employs the log loss (cross entropy) to train classifiers.
2. The loss function used in logistic regression can be expressed as

$$\frac{1}{n}\sum_{i=1}^{n} L_{\log}(y_i, s_i), \tag{6}$$

where $L_{\log}$ is the log loss, defined as

$$L_{\log}(y_i, s_i) = \log(1 + e^{-y_i s_i}). \tag{7}$$



Hinge loss (support vector machines)

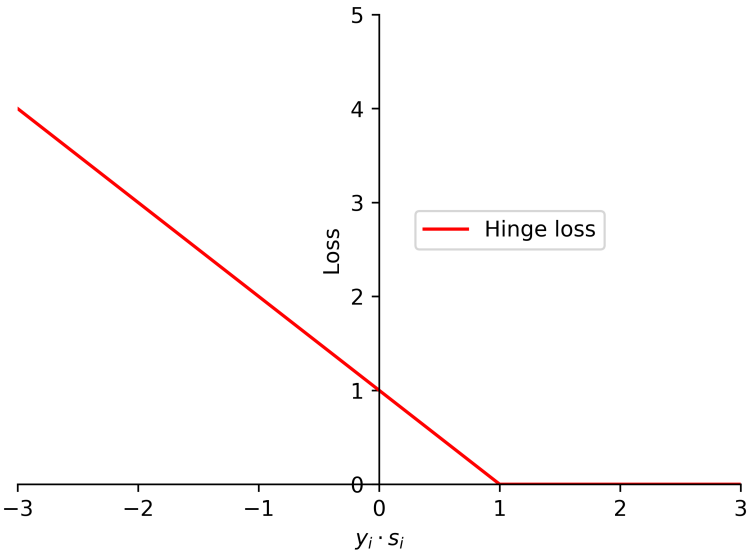
1. The support vector machines employ hinge loss to obtain a classifier with "maximum-margin".
2. The loss function in support vector machines is defined as follows:

$$\frac{1}{n}\sum_{i=1}^{n} L_h(y_i, s_i), \tag{8}$$

where L_h is the hinge loss:

$$L_h(y_i, s_i) = \max(0, 1 - y_i s_i). \tag{9}$$

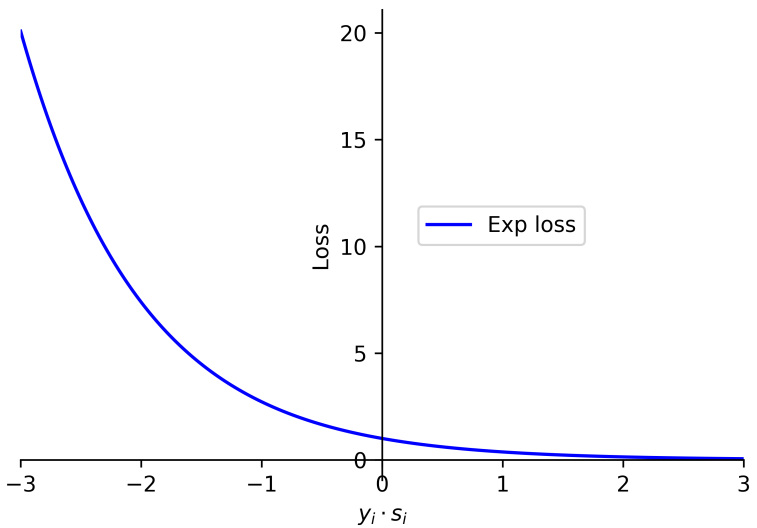
3. Different with the zero-one loss and perceptron loss, a data may be penalized even if it is predicted correctly.



Exponential Loss

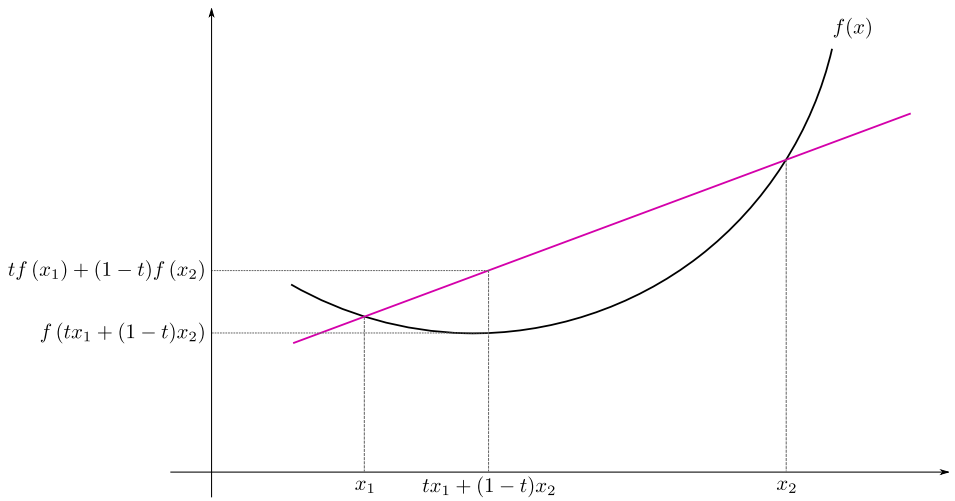
1. The log term in the log loss encourages the loss to grow slowly for negative values, making it less sensitive to wrong predictions.
2. There is a more aggressive loss function, known as the exponential loss, which grows exponentially for negative values and is thus very sensitive to wrong predictions. The AdaBoost algorithm employs the exponential loss to train the models.
3. The exponential loss function can be expressed as $\frac{1}{n}\sum_{i=1}^{n} L_{\text{exp}}(y_i, s_i)$, where L_{exp} is the exponential loss, defined as

$$L_{\text{exp}}(y_i, s_i) = e^{-y_i s_i}. \tag{10}$$



Convexity

1. Mathematically, a function $f(\cdot)$ is convex if
$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2), \text{for } t \in [0,1].$$
2. A function $f(\cdot)$ is strictly convex if
$$f(tx_1 + (1-t)x_2) < tf(x_1) + (1-t)f(x_2), \text{for } t \in (0,1),\ x_1 \neq x_2.$$
3. Intuitively, a function is convex if the line segment between any two points on the function is not below the function.
4. A function is strictly convex if the line segment between any two distinct points on the function is strictly above the function, except for the two points on the function itself.

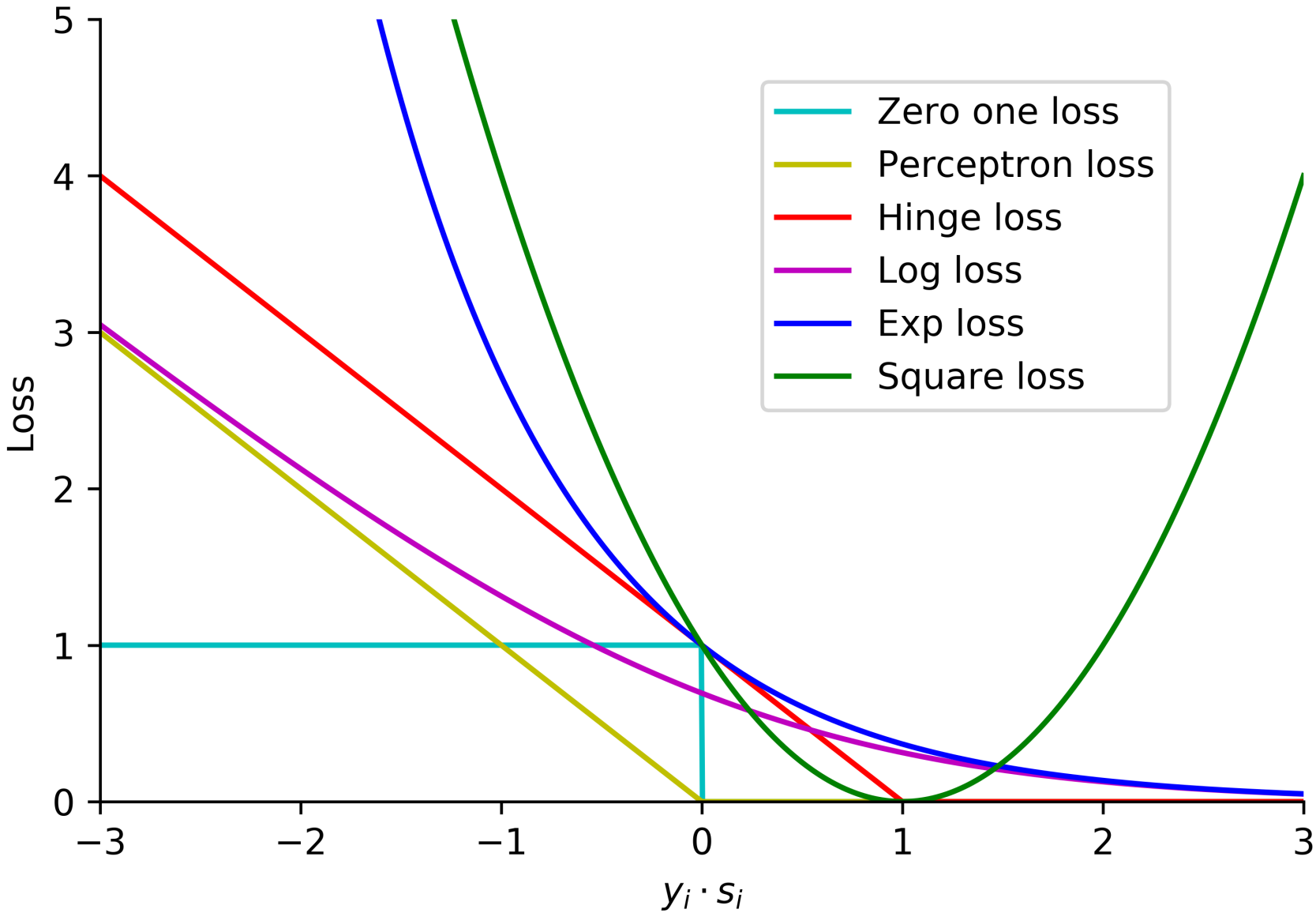


https://en.wikipedia.org/wiki/Convex_function

Comparison of loss functions

1. In the zero-one loss, if a data sample is predicted correctly ($y_i s_i > 0$), it results in zero penalties; otherwise, there is a penalty of one. For any data sample that is not predicted correctly, it receives the same loss.
2. For the perceptron loss, the penalty for each wrong prediction is proportional to the extent of violation. For other losses, a data sample can still incur penalty even if it is classified correctly.
3. The log loss is similar to the hinge loss but it is a smooth function which can be optimized with the gradient descent method.
4. While log loss grows slowly for negative values, exponential loss and square loss are more aggressive.
5. Note that, in all of these loss functions, square loss will penalize correct predictions severely when the value of $y_i s_i$ is large.
6. In addition, zero-one loss is not convex while the other loss functions are convex. Note that the hinge loss and perceptron loss are not strictly convex.

Comparison of different loss functions in a unified view



THANKS!