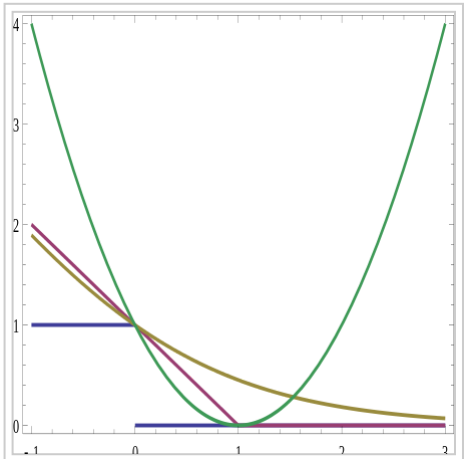
Loss functions for classification

From Wikipedia, the free encyclopedia

In machine learning and mathematical optimization, **loss functions for classification** are computationally feasible loss functions representing the price paid for inaccuracy of predictions in classification problems.[1] Given $\boldsymbol{X}$ as the vector space of all possible inputs, and $Y = \{-1,1\}$ as the vector space of all possible outputs, we wish to find a function $\boldsymbol{f : X \mapsto \mathbb{R}}$ which best maps $\boldsymbol{f(\vec{x})}$ to $\boldsymbol{y}$.[2] However, because of incomplete information, noise in the measurement, or probabilistic components in the underlying process, it is possible for the same $\boldsymbol{\vec{x}}$ to generate different $\boldsymbol{y}$.[3] As a result, the goal of the learning problem is to minimize expected risk, defined as



Plot of various loss functions. Blue is the 0–1 indicator function. Green is the square loss function. Purple is the hinge loss function. Yellow is the logistic loss function. Note that all surrogates give a loss penalty of 1 for $yf(x) = 0$

$$I[f] = \int_{X \times Y} V(f(\vec{x}), y) p(\vec{x}, y)\, d\vec{x}\, dy$$

where $\boldsymbol{V(f(\vec{x}), y)}$ represents the loss function, and $\boldsymbol{p(\vec{x}, y)}$ represents the probability distribution of the data, which can equivalently be written using Bayes' theorem as

$$p(\vec{x}, y) = p(y \mid \vec{x}) p(\vec{x}).$$

In practice, the probability distribution $\boldsymbol{p(\vec{x}, y)}$ is unknown. Consequently, utilizing a training set of $\boldsymbol{n}$ independently and identically distributed samples

$$S = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)\}$$

drawn from the data sample space, one seeks to minimize empirical risk

$$I_S[f] = \frac{1}{n} \sum_{i=1}^{n} V(f(\vec{x}_i), y_i).$$

as a proxy for expected risk.[3] (See statistical learning theory for a more detailed description.)

For computational ease, it is standard practice to write loss functions as functions of only one variable. Within classification, loss functions are generally written solely in terms of the product of the true classifier $\boldsymbol{y}$ and the predicted value $\boldsymbol{f(\vec{x})}$.[4] Selection of a loss function within this framework

$$V(f(\vec{x}), y) = \phi(-yf(\vec{x}))$$

impacts the optimal $\boldsymbol{f_S^*}$ which minimizes empirical risk, as well as the computational complexity of the learning algorithm.

Given the binary nature of classification, a natural selection for a loss function (assuming equal cost for false positives and false negatives) would be the 0–1 indicator function which takes the value of 0 if the predicted classification equals that of the true class or a 1 if the predicted classification does not match the true class. This selection is modeled by

$$V(f(\vec{x}), y) = H(-yf(\vec{x}))$$

where H indicates the Heaviside step function. However, this loss function is non-convex and non-smooth, and solving for the optimal solution is an NP-hard combinatorial optimization problem.[5] As a result, it is better to substitute continuous, convex **loss function surrogates** which are tractable for commonly used learning algorithms. In addition to their computational tractability, one can show that the solutions to the learning problem using these loss surrogates allows for the recovery of the actual solution to the original classification problem.[6] Some of these surrogates are described below.

Contents

Bounds for classification

Utilizing Bayes' theorem, it can be shown that the optimal f^* for a binary classification problem is equivalent to

$$f^*(\vec{x}) = \begin{cases} 1 & \text{if } p(1 \mid \vec{x}) > p(-1 \mid \vec{x}) \\ -1 & \text{if } p(1 \mid \vec{x}) < p(-1 \mid \vec{x}) \end{cases}$$

(when $p(1 \mid \vec{x}) \neq p(-1 \mid \vec{x})$).

Furthermore, it can be shown that for any convex loss function $V(yf_0(\vec{x}))$, where f_0 is the function that minimizes this loss, if $f_0(\vec{x}) \neq 0$ and V is decreasing in a neighborhood of 0, then $f^*(\vec{x}) = \operatorname{sgn}(f_0(\vec{x}))$ where sgn is the sign function (for proof see [1]). Note also that $f_0(\vec{x}) \neq 0$ in practice when the loss function is differentiable at the origin. This fact confers a consistency property upon all convex loss functions; specifically, all convex loss functions will lead to consistent results with the 0–1 loss function given the presence of infinite data. Consequently, we can bound the difference of any of these convex loss function from expected risk.[1]

Simplifying expected risk for classification

Given the properties of binary classification properties, it is possible to simplify the calculation of expected risk from the integral specified above. Specifically,

$$
\begin{aligned}
I[f] &= \int_{X\times Y} V(f(\vec{x}), y) p(\vec{x}, y)\, d\vec{x}\, dy \\
&= \int_X \int_Y V(-yf(\vec{x})) p(y \mid \vec{x}) p(\vec{x})\, dy\, d\vec{x} \\
&= \int_Y V(-yf(\vec{x})) p(y \mid \vec{x})\, dy \\
&= V(-f(\vec{x})) p(1 \mid \vec{x}) + V(f(\vec{x})) p(-1 \mid \vec{x}) \\
&= V(-f(\vec{x})) p(1 \mid \vec{x}) + V(f(\vec{x}))(1 - p(1 \mid \vec{x}))
\end{aligned}
$$

The second equality follows from the properties described above. The third equality follows since $\vec{x}$ is simply data and since $\int_X p(x)\, dx = 1$. Finally, the fourth equality follows from the fact that 1 and −1 are the only possible values for y, and the fifth because $p(-1 \mid x) = 1 - p(1 \mid x)$. As a result, one can solve for the minimizers of $I[f]$ for any convex loss functions with these properties by differentiating the last equality with respect to f and setting the derivative equal to 0. Thus, minimizers for all of the loss function surrogates described below are easily obtained as functions of only $f(\vec{x})$ and $p(1 \mid x)$.[3]

Square loss

While more commonly used in regression, the square loss function can be re-written as a function $\phi(yf(\vec{x}))$ and utilized for classification. Defined as

$$V(f(\vec{x}), y) = (1 - yf(\vec{x}))^2$$

the square loss function is both convex and smooth and matches the 0–1 indicator function when $yf(\vec{x}) = 0$ and when $yf(\vec{x}) = 1$. However, the square loss function tends to penalize outliers excessively, leading to slower convergence rates (with regards to sample complexity) than for the logistic loss or hinge loss functions.[1] In addition, functions which yield high values of $f(\vec{x})$ for some $x \in X$ will perform poorly with the square loss function, since high values of $yf(\vec{x})$ will be penalized severely, regardless of whether the signs of y and $f(\vec{x})$ match.

A benefit of the square loss function is that its structure lends itself to easy cross validation of regularization parameters. Specifically for Tikhonov regularization, one can solve for the regularization parameter using leave-one-out cross-validation in the same time as it would take to solve a single problem.[7]

The minimizer of $I[f]$ for the square loss function is

$$f^*_{\text{Square}} = 2p(1 \mid x) - 1$$

This function notably equals f^* for the 0–1 loss function when $p(1 \mid x) = 1$ or $p(1 \mid x) = 0$, but predicts a value between the two classifications when the classification of $\vec{x}$ is not known with absolute certainty.

Hinge loss

The hinge loss function is defined as

$$V(f(\vec{x}), y) = \max(0, 1 - yf(\vec{x})) = |1 - yf(\vec{x})|_+.$$

The hinge loss provides a relatively tight, convex upper bound on the 0–1 indicator function. Specifically, the hinge loss equals the 0–1 indicator function when $\operatorname{sgn}(f(\vec{x})) = y$ and $|yf(\vec{x})| \geq 1$. In addition, the empirical risk minimization of this loss is equivalent to the classical formulation for support vector machines (SVMs). Correctly classified points lying outside the margin boundaries of the support vectors are not penalized, whereas points within the margin boundaries or on the wrong side of the hyperplane are penalized in a linear fashion compared to their distance from the correct boundary.[5]

While the hinge loss function is both convex and continuous, it is not smooth (that is not differentiable) at $yf(\vec{x}) = 1$. Consequently, the hinge loss function cannot be used with gradient descent methods or stochastic gradient descent methods which rely on differentiability over the entire domain. However, the hinge loss does have a subgradient at $yf(\vec{x}) = 1$, which allows for the utilization of subgradient descent methods.[5] SVMs utilizing the hinge loss function can also be solved using quadratic programming.

The minimizer of $I[f]$ for the hinge loss function is

$$f^*_{\text{Hinge}}(\vec{x}) = \begin{cases} 1 & \text{if } p(1 \mid \vec{x}) > p(-1 \mid \vec{x}) \\ -1 & \text{if } p(1 \mid \vec{x}) < p(-1 \mid \vec{x}) \end{cases}$$

when $p(1 \mid x) \neq 0.5$, which matches that of the 0–1 indicator function. This conclusion makes the hinge loss quite attractive, as bounds can be placed on the difference between expected risk and the sign of hinge loss function.[1]

Logistic loss

The logistic loss function is defined as

$$V(f(\vec{x}), y) = \frac{1}{\ln 2} \ln(1 + e^{-yf(\vec{x})})$$

This function displays a similar convergence rate to the hinge loss function, and since it is continuous, gradient descent methods can be utilized. However, the logistic loss function does not assign zero penalty to any points. Instead, functions that correctly classify points with high confidence (i.e., with high values of $|f(\vec{x})|$) are penalized less. This structure leads the logistic loss function to be sensitive to outliers in the data.

The minimizer of $I[f]$ for the logistic loss function is

$$f^*_{\text{Logistic}} = \ln\left(\frac{p(1 \mid x)}{1 - p(1 \mid x)}\right).$$

This function is undefined when $p(1 \mid x) = 1$ or $p(1 \mid x) = 0$ (tending toward ∞ and –∞ respectively), but predicts a smooth curve which grows when $p(1 \mid x)$ increases and equals 0 when $p(1 \mid x) = 0.5$.[3]

Cross entropy loss

Using the alternative label convention $t = (1 + y)/2$ so that $t \in \{0, 1\}$, the cross entropy loss is defined as

$$V(f(\vec{x}), t) = -t \ln(f(\vec{x})) - (1 - t) \ln(1 - f(\vec{x}))$$

The cross entropy loss is closely related to the Kullback-Leibler divergence between the empirical distribution and the predicted distribution. This function is not naturally represented as a product of the true label and the predicted value, but is convex and can be minimized using stochastic gradient descent methods. The cross entropy loss is ubiquitous in modern deep neural networks.

See also

- Statistical learning theory
- Loss function
- Support vector machine

References

1. Rosasco, L.; De Vito, E. D.; Caponnetto, A.; Piana, M.; Verri, A. (2004). "Are Loss Functions All the Same?" (PDF). *Neural Computation*. **16** (5): 1063–1076. doi:10.1162/089976604773135104. PMID 15070510.
2. Shen, Yi (2005), *Loss Functions For Binary Classification and Class Probability Estimation* (PDF), University of Pennsylvania, retrieved 6 December 2014
3. Rosasco, Lorenzo; Poggio, Tomaso (2014), *A Regularization Tour of Machine Learning*, MIT-9.520 Lectures Notes, Manuscript
4. Masnadi-Shirazi, Hamed; Vasconcelos, Nuno, *On the Design of Loss Functions for Classification: theory, robustness to outliers, and SavageBoost* (PDF), Statistical Visual Computing Laboratory, University of California, San Diego, retrieved 6 December 2014
5. Piyush, Rai (13 September 2011), *Support Vector Machines (Contd.), Classification Loss Functions and Regularizers* (PDF), Utah CS5350/6350: Machine Learning, retrieved 6 December 2014
6. Ramanan, Deva (27 February 2008), *Lecture 14* (PDF), UCI ICS273A: Machine Learning, retrieved 6 December 2014
7. Rifkin, Ryan M.; Lippert, Ross A. (1 May 2007), *Notes on Regularized Least Squares*, MIT Computer Science and Artificial Intelligence Laboratory