Hinge Loss

In Support Vector Machines

Chandan Kumar Sahu and Maitrey Sharma

School of Computer Sciences,
National Institute of Science Education and Research, Bhubaneshwar,
Homi Bhabha National Institute

February 7, 2023

Part I: A Recap of Support Vector Machines

Part II: Understanding Loss Functions

Part III: Hinge Loss in SVM

Part I

A Recap of Support Vector Machines

Support Vector Machines (SVMs)

What are SVMs?

- **Classification** is one of the most important and recurring problems to tackle in machine learning. The problem becomes more difficult when the data is multidimensional.
- **SVM** is a type of supervised machine learning (ML) algorithm that aims to create a decision boundary (or hyperplane in multiple dimensions) that can separate n-dimensional data points into classes so that any new data point can be easily classified into the correct category.
- There are many other classification algorithms like ***Decision Trees***, ***k-Nearest Neighbours*** etc., but SVMs are the most robust, especially when the data is linearly separable. This is why SVMs are also referred to as **non-probabilistic binary linear classifiers**. They are the most popular and widely used algorithms in ML.
- In fact, SVMs can even be used to perform non-linear classification using the kernel trick.

Support Vector Machines (SVMs)

Difference between Classification and Regression using SVMs

- SVMs are primarily used as a classification algorithm, but they can be used for **regression** as well.
- In classifiers based on SVM, for a data embedded in p-dimensional vector space, we seek a $(p-1)$-dimensional **hyperplane**, which will separate the data points. [2], [3]
- In Support Vector Regression (SVR), the hyperplane is the best-fit hypersurface that contains the maximum number of points.

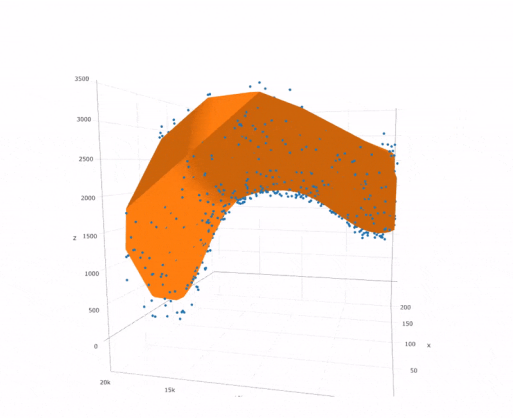

Figure. Support Vector Regression Hyperplane

LINEAR SVM

- Consider a training dataset of n points of the form
$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n),$$
where y_i are either -1 or +1 each, indicating the class to which the point $\mathbf{x}_i$ belongs. Each $\mathbf{x}_i$ is a p-dimensional real vector.
- Any hyperplane in this space can be written as the set of points $\mathbf{x}$ satisfying
$$\mathbf{w}^{\mathrm{T}}\mathbf{x} - b = 0 \tag{1}$$

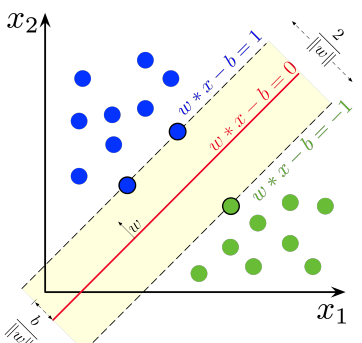

Figure. Maximum-margin hyperplane (red line); margins are trained with samples from two classes. Samples on the margin are called the **support vectors**.

Linear SVM

Hard-margin

- If the training data is linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible.
- The distance between these two hyperplanes is called the **margin**, and the maximum-margin hyperplane is the hyperplane that lies halfway between them. With a normalized or standardized dataset, these hyperplanes can be described by the equations

$$\mathbf{w}^T\mathbf{x} - b = +1 \quad \text{(anything on or above this boundary is of one class, with label +1)}$$
$$\mathbf{w}^T\mathbf{x} - b = -1 \quad \text{(anything on or below this boundary is of the other class, with label -1).}$$

- Geometrically, the distance between these two hyperplanes is $\frac{2}{||\mathbf{w}||}$. To maximize, this margin, we need to minimize $||\mathbf{w}||$.
- To prevent data points falling inside the margin, we add following constraint:

$$y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1, \qquad \forall\ 1 \leq i \leq n \tag{2}$$

- These $\mathbf{x}_i$'s are called as the **hard-margin support vectors**.

Linear SVM

Soft-margin

- Sometimes, the margin separating the data points is so small that the model becomes prone to **overfitting** or being too sensitive to **outliers**.
- The solution to this problem is to employ a **larger** margin or a **soft** margin that may include sacrificing some data points to **misclassifications**. This will help the model to generalise better.
- Soft margin can also be used when the data is not linearly separable at all. Here, we can try to minimize the number of misclassification.
- Because now, we have to deal with misclassifications or errors in our model, we need a quantifiable framework to deal with it. This is where the loss or cost functions come into the picture.
- In the next part, we will explore the lost and cost functions and will then see how a soft-margin SVM can be defined using the **hinge loss function**.

Part II

Understanding Loss Functions

Loss and Cost functions

- In mathematical optimization and decision theory, a **loss function** or **cost function** (sometimes also called an **error function**) is a function that maps an event or values of one or more variables onto a real number intuitively representing some **cost** associated with the event. [6]
- When solving an optimization problem, the primary goal is to minimize the cost function.
- These functions present a quantifiable way of understanding how well our model is generalizing.
- **Loss functions** capture the difference between the actual and predicted values for a single record.
- On the other hand, **cost functions** aggregate the difference for the entire training dataset.

Part III

Hinge Loss in SVMs

Intuition of Hinge Loss

- The hinge loss is a loss function used for training classifiers, most notably the SVM. It is used for the de-facto **maximum-margin** classification for SVMs. [3], [4]
- The x-axis represents the distance from the boundary of any single instance, and the y-axis represents the loss size, or penalty, that the function will incur depending on its distance.

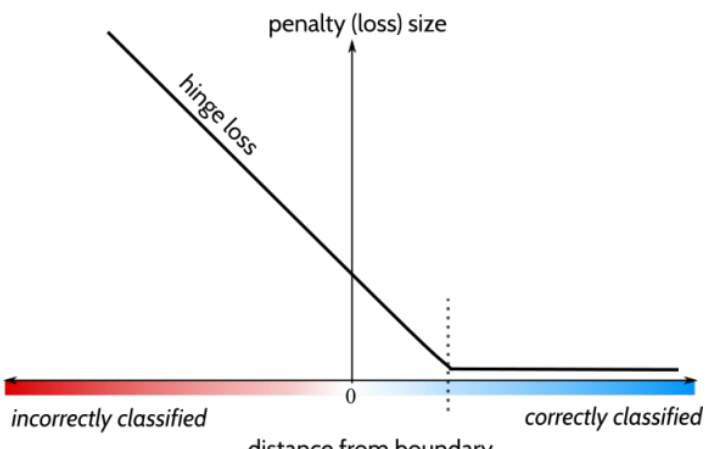

Figure. Hinge loss, visualized

A Qualitative Description of Hinge Loss

- When a datapoint's distance from the boundary is greater than or equal to 1, the loss size is 0.
- If the distance from the boundary is less than 1, the we incur a loss. At 0 distance (the data point is on the boundary), then the loss size is 1.
- Correctly classified points will have a small loss size, while incorrectly classified instances will have a high loss size.
- High hinge loss indicates datapoints being on the wrong side of the boundary, and hence are misclassified; while a positive distance calls for low (or zero) hinge loss and correct classification. [1]

Mathematical Description of Hinge Loss

For an intended output of $t = \pm 1$ and a classifier score y, the hinge loss of the prediction y is defined as

$$L(y) = \max(0, 1 - t \cdot y) \tag{3}$$

Note that y should be **raw output** of the classifier's decision function, not the predicted class label. For instance, in linear SVMs, $y = \mathbf{w}^{\mathbf{T}} \cdot \mathbf{x} + b$, where $(\mathbf{w}, b)$ are the parameters of the hyperplane and $\mathbf{x}$ is the vector composed of input variables.

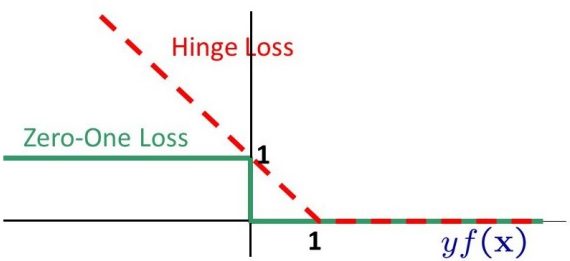

Figure. Hinge loss vs One-zero loss: the y-axis represents the Hinge loss (red) and zero-one loss (green) for a fixed $t = 1$, and x-axis represents the prediction value.

The plot shows that the Hinge loss penalizes predictions $y > 1$, corresponding to the notion of a margin in a support vector machine. [1] [4]

Soft margin in SVM using Hinge Loss

- We define the hinge loss function for our soft margin by

$$L(y) = \max(0, 1 - y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i - b)) \tag{4}$$

  $L(y) = 0$ if the constraint in equation (2) is satisfied, i.e., if $\mathbf{x}_i$ lies on the correct side of the margin. For data on the wrong side of the margin, the function's value is proportional to the distance from the margin. [5]
- The goal of optimization is to minimize

$$\lambda||\mathbf{w}||^2 + \left[\frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i - b)\right]$$

  where the parameter $\lambda > 0$ determines the trade-off between increasing the margin size and ensuring that the $\mathbf{x}_i$ lie on the correct side of the margin.
- If the margin is high, then $||\mathbf{w}||$ is less and some misclassification is allowed for the second term.

Optimisations to Hinge Loss

- The convexity of the hinge loss function makes it easy for optimizers to be applied. Common optimizers like **gradient descent** can be used for it. [4]
- However, the loss function is not differentiable. But it has a sub-gradient with respect to model parameters $\mathbf{w}$ of a linear SVM, with score function $y = \mathbf{w} \cdot \mathbf{x}$ that is given by [4]

$$\frac{dL}{dw_i} = \begin{cases} t \cdot x_i, & \text{if } t \cdot y < 1 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Optimisations to Hinge Loss

- Since the derivative of the hinge loss at $ty = 1$ is undefined, smoothed versions may be preferred for optimization, such as used by Rennie and Srebro [5]:

$$L(y) = \begin{cases} \frac{1}{2} - ty, & \text{if } ty < 0, \\ \frac{1}{2}(1 - ty)^2, & \text{if } 0 < ty < 1, \\ 0, & \text{if } 1 \leq y \end{cases} \tag{6}$$

  or the quadratically smoothed

$$L_\gamma(y) = \begin{cases} \frac{1}{2\gamma}\max(0, 1 - ty)^2, & \text{if } ty \geq 1 - \gamma \\ 1 - \frac{\gamma}{2} - ty, & \text{otherwise} \end{cases} \tag{7}$$

  suggested by Zhang [7]. The modified Huber loss L is a special case of this loss function with $\gamma = 2$, specifically $L(t, y) = 4l_2(y)$.

Optimisations to Hinge Loss

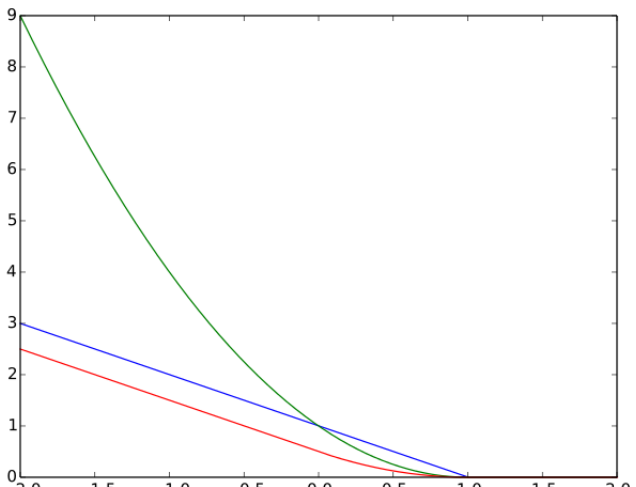

Figure. Plot of three variants of the hinge loss as a function of $z = ty$: the "ordinary" variant (blue), its square (green), and the piece-wise smooth version by Rennie and Srebro (red) [5]. The y-axis is the $L(y)$ hinge loss, and the x-axis is the parameter t. (refer [4])

Comparing Hinge Loss with other penalization methods

There are many other types of loss functions used in modern ML. Some of them are mentioned below along with their minimizing functions: [3]

Loss Function	$L[y, f(x)]$	Minimizing Function
Binomial Deviance	$\log[1 + e^{-yf(x)}]$	$f(x) = \log \frac{\Pr(Y = +1\|x)}{\Pr(Y = -1\|x)}$
SVM Hinge Loss	$[1 - yf(x)]_+$	$f(x) = \text{sign}[\Pr(Y = +1\|x) - \frac{1}{2}]$
Squared Error	$[y - f(x)]^2 = [1 - yf(x)]^2$	$f(x) = 2\Pr(Y = +1\|x) - 1$
"Huberised" Square Hinge Loss	$-4yf(x)$, $yf(x) < -1$ $[1 - yf(x)]_+^2$ otherwise	$f(x) = 2\Pr(Y = +1\|x) - 1$

Figure. The population minimizers are shown for the different loss functions in Figure 7. Logistic regression uses the binomial log-likelihood or deviance. Linear discriminant analysis uses squared-error loss. The SVM hinge loss estimates the mode of the posterior class probabilities, whereas the others estimate a linear transformation of these probabilities.

Comparing Hinge Loss with other penalization methods

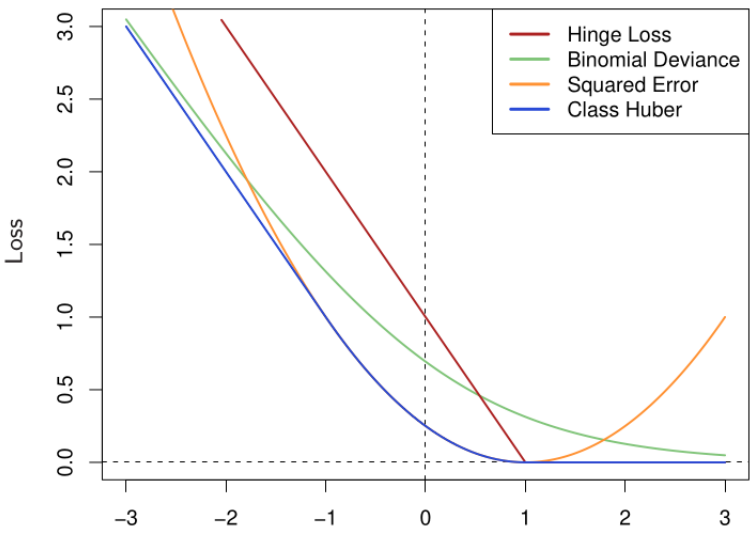

Figure. The support vector loss function (hinge loss), compared to the negative log-likelihood loss (binomial deviance) for logistic regression, squared-error loss, and a "Huberized" version of the squared hinge loss. All are shown as a function of y_f rather than f, because of the symmetry between the $y = +1$ and $y = 1$ case. The deviance and Huber have the same asymptotes as the SVM loss, but are rounded in the interior. All are scaled to have the limiting left-tail slope of -1. (refer [3])

References I

[1] *A definitive explanation to the Hinge Loss for Support Vector Machines*. https://towardsdatascience.com/a-definitive-explanation-to-hinge-loss-for-support-vector-machines-ab6d8d3178f1.

[2] Deisenroth, Aldo A. Faisal, and Cheng Soon Ong. *Mathematics for Machine Learning*. https://mml-book.github.io/.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. https://hastie.su.domains/ElemStatLearn/.

[4] *Hinge loss - Wikipedia*. https://en.wikipedia.org/wiki/Hinge_loss#:~:text=In%20machine%20learning%2C%20the%20hinge,support%20vector%20machines%20(SVMs)..

[5] Jason DM Rennie and Nathan Srebro. "Loss functions for preference levels: Regression with discrete ordered labels". In: *Proceedings of the IJCAI multidisciplinary workshop on advances in preference handling*. Vol. 1. AAAI Press, Menlo Park, CA. 2005.

[6] Lorenzo Rosasco et al. "Are loss functions all the same?" In: *Neural computation* 16.5 (2004), pp. 1063–1076.

References II

[7] Tong Zhang. "Solving large scale linear prediction problems using stochastic gradient descent algorithms". In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 116.