Support vector machines (intuitive understanding) — Part#1

Comparison with logistic regression and hinge loss

Ravindra Kompella Follow
Oct 17, 2017 · 5 min read

Most of the material online covered on this topic was heavily treated with mathematics and lot of finer details, one can get easily lost understanding the broader concept. Here is an attempt to bring an intuitive understanding to most of the details in SVM with a very little mathematical treatment. The only basic assumption made is that the reader is already aware of some math fundamentals, logistic regression along with basic terms and concepts of machine learning. I plan to cover this topic "**Support vector machines (intuitive understanding)"** in 3 parts. In Part # 1, we will look at the loss function for SVM.

Lets start with what we understand about logistic regression and how SVM differs from logistic regression.

In logistic regression, a line L1 defines a probability distribution over the input space. A line L1 is said to be better than line L2, if the the distribution defined by L1 is low at class '-1' points and high at class '+1' points, on average, compared to the distribution defined by line L2.

Few major things of SVM that are conceptually different from a logistic regression —

Part#1: Loss function

Part#2: Maximum margin classification—At a very fundamental level, in SVM, a line L1 is said to be a better classifier than line L2, if the "margin" of L1 is larger i.e., L1 is farther from both classes.

Part#3: Feature transformation using Kernel trick

Lets go over one by one —

Loss function: First, lets start with Loss function —

Lets take an example of a simple binary classification task. Then for the given input features "X" and target "y", the goal of the SVM algorithm is to predict a value ('predicted y') close to the target ('actual y') for each observation. To do this —

1. We are interested in a equation that could calculate 'predicted y'. This equation depends on some weighted values of input X. It can be written as :

'predicted y' = f (weighted values of X).

Lets denote our weights as *w*.

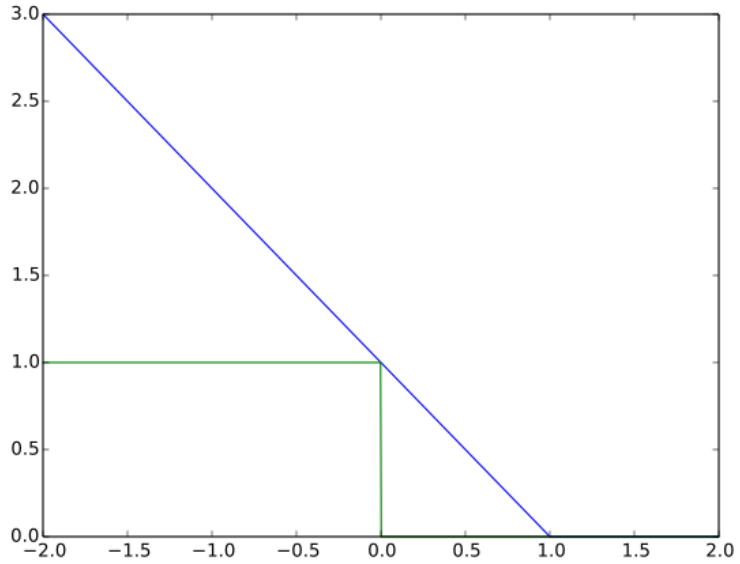
1. We can start the classification by drawing a random decision boundary (aka., this is same as predicting some random values for 'predicted y')—and this is exactly what is happening when we initialize the weights in the above equation with random values.

2. Identify the cost/loss function. This job of the loss function is to quantify the error between the 'predicted y' and the 'actual y'. To put this in a simple way, this defines the amount by which you want to penalize the mis-classified observations. So intuitively, the farther the 'predicted y' from the 'actual y', more should be the penalty and vice-versa. Given the amount of total error (= sum of losses of for each observation) we made in a particular iteration, we try to reduce this error by adjusting the weights for the next iteration. This continues till we can no longer can minimize the Total error / cost**.** This total error / cost function is as below:

Total error / cost = Sum of all losses for each observation in that iteration.

4. The final weights which we arrive at, at the end of all iterations forms our final model used for predicting on unseen data.

So, for SVM, a loss function called as 'hinge loss' is used—refer to the below plot (*from wikipedia)*.



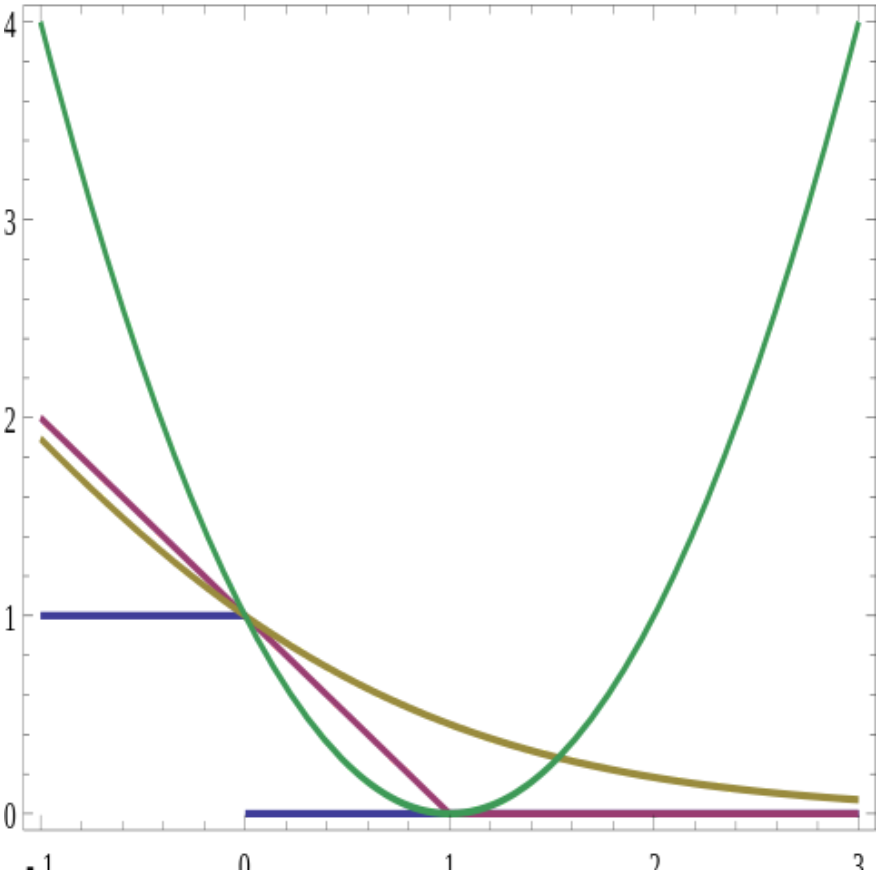
Plot of hinge loss on y-axis and 'predicted y' on x-axis.

Note that from the above plot (blue line), it can be seen that the loss equals 0 , when

- the 'predicted y' ≥ 1 and
- 'actual y' and the 'predicted y' have the same sign (meaning 'predicted y' predicts the right class)

But when 'actual y' , 'predicted y' have opposite sign, the hinge loss increases linearly with y (one-sided error).

SVM uses hinge loss where as logistic regression using logistic loss function for optimizing the cost function and arriving at the weights. The way the hinge loss is different from logistic loss can be understood from the plot below (*from wikipedia—Purple is the hinge loss, Yellow is the logistic loss function*).



Plot of various loss functions—Purple is the hinge loss function. Yellow is the logistic loss function.

Note that the yellow line gradually curves downwards unlike purple line where the loss becomes 0 for values 'predicted y' ≥1. By looking at the plots above, this nature of curves brings out few major differences between logistic loss and hinge loss —

- Note that the logistic loss diverges faster than hinge loss. So, in general, it will be more sensitive to outliers—why? Because, assuming there is an outlier in our data, the logistic loss (due to its diverging nature) will be very high compared to the hinge loss for that outlier. This means greater adjustments to our weights.
- Note that the logistic loss does not go to zero even if the point is classified sufficiently confidently—The horizontal axis being the confidence of 'predicted y' value, if we take a value like '1.5' on x-axis, then the corresponding logistic loss (yellow line) still shows some loss (close to 0.2 from the above plot and hence still not very confident of our prediction), whereas the hinge loss is '0' (which means there is no loss and we are more confident of our prediction). This nature of logistic loss might lead to minor degradation in accuracy.
- Logistic regression has a more probabilistic interpretation.

Given this understanding of the hinge loss function for a SVM, lets add a regularization term (L2 norm) to the cost. The intuition behind the regularization term is that we increase the cost penalty if the values for the weights are high. So while trying to minimize the cost, we not only adjust the weights, we also try to minimize the value of the weights and thereby reduce over fitting to the training data and make the model less sensitive to outliers.

So with the added regularization term, the total cost function finally looks like:

Total cost = $||w^2||/2$ + C*(Sum of all losses for each observation)

where 'C' is the hyper-parameter that controls the amount of regularization.

Depending on the value of 'C' that we choose, we can have a hard margin classifier and a soft margin classifier.

- If C is chosen to be sufficiently small such that the second term in the total cost can be ignored, then we call this a hard-margin classifier. In such case we minimize only the first term ($||w^2||/2$) on every iteration to arrive at the adjusted weights.
- If C is chosen sufficiently large, the second term cannot be ignored and we arrive at soft margin classifier.

In the next part we will look at what exactly causes SVM hyper-plane to distance itself (aka., maximize the margin) from the data points in two classes in order to achieve the notion of a maximum margin classifier. We will also look at what is the value of the margin.