Kernelization of ML algorithms by loss function

Bilal Ahmed, Virgil Pavlu
December 8, 2014

1 Representer Theorem

We have seen that the dual perceptron and the support vector machine (SVM) have identical forms for the final weight vector i.e., $w^* = \sum_{i=1}^{N} \alpha_i y_i x_i$. We have also seen that both these algorithms can work with kernels, that allows us to work efficiently in high dimensional spaces enabling us to learn complex non-linear decision boundaries and use these learning methods to work with other types of data such as strings, trees, etc. But, is the use of kernels limited to only these algorithms or is it possible to kernelize other learning methods as well? The answer to this is provided by the Representer theorem, which "roughly" states that:

If a learning algorithm can posed as a minimization problem of the form:

$$\min_{w} \ \text{Loss}(y, f(w, x)) + \lambda \text{Penalty}(w) \tag{1}$$

where, w are the model parameters, $f(w, x)$ represents the classifier output, y is the actual label and λ is a regularization parameter. Then under some "weak" conditions on the loss and penalty functions, the solution has the form $w^* = \sum_i \alpha_i y_i x_i$, i.e., a linear combination of the training instances.

This is a very powerful result that allows us to apply the kernel trick to a broader range of learning algorithms. In the following sections we will see how the Representer theorem can be applied to SVMs, ridge regression and Logistic regression.

2 Loss functions

3 Support Vector Machines

Recall, the primal SVM problem:

$$\begin{aligned} \min_{w,\xi_i} \quad & \frac{1}{2}\|w\|^2 + \sum_{i=1}^{N} \xi_i \\ \text{subject to:} & \\ \forall i: \quad & y_i(w^T x_i + b) \ \geq \ 1 - \xi_i \\ & \xi_i^t \ \geq \ 0 \end{aligned} \tag{2}$$

From the objective function we can see that the loss function for a particular training instance is represented by ξ_i. The constraints can be combined to show that $\xi_i = max(0, 1 - y_i(w^T x_i + b))$, which is also known as the Hinge loss for the i^{th} instance. Therefore, we can represent (2) equivalently as:

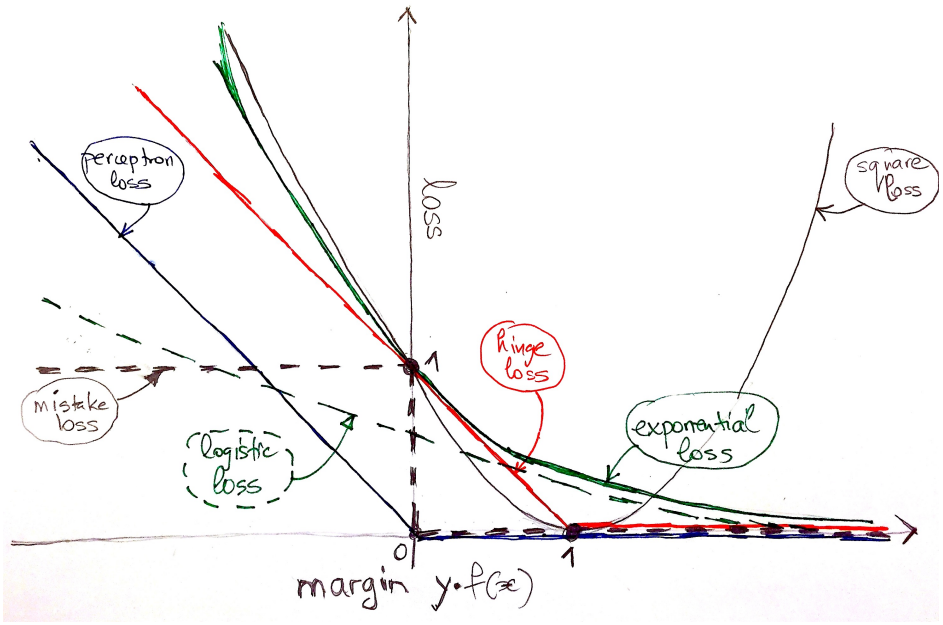

Figure 1: Various loss objectives as a function of the margin. What is the corresponding machine learning algorithm for each loss?

$$\min_{w} \quad \frac{1}{2}\|w\|^2 + \sum_{i=1}^{N} \max(0, 1 - y_i(w^T x_i + b)) \tag{3}$$

According to the Representer theorem (1), the primal SVM (3) will have a solution of the form $w^* = \sum_i \alpha_i y_i x_i$. Recall, that this is exactly the solution that we recovered for the optimal w^* by solving the Lagrangian of (2).

4 Regularized Least Squares Classification

Regularized Least Squares classification adapts the regularized linear regression framework to the task of classification. To this end it uses an objective function that minimizes the square of the difference between the actual label of the instance and the classifier output. Let (x_i, y_i), $i \in \{1, 2, \ldots, N\}$ represent the training data where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. The objective that we want to minimize in this case is:

$$\min_{w} \quad \frac{\lambda}{2}\|w\|^2 + \sum_{i=1}^{N} (w^T x_i - y_i)^2 \tag{4}$$

Applying the Representer theorem to this objective function we obtain its dual version:

$$\min_{\alpha} \quad \frac{\lambda}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) + \sum_{j=1}^{N} (\sum_{i=1}^{N} \alpha_i y_i K(x_i, x_j) - y_i)^2 \tag{5}$$

where, K(.,.) is the value of the kernel function.

5 Regularized Logistic Regression

Let (x_i, y_i), $i \in \{1, 2, \ldots, N\}$ represent the training data where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. Then the probability of an instance belonging to a particular class is given as:

$$P(y_i|w, x_i) = \frac{1}{1 + e^{-y_i w^T x_i}} \tag{6}$$

where, w are the model parameters. The maximum likelihood solution can be obtained by minimizing the negative log-likelihood. For the regularized case, we add an L2 penalty on the norm of w. The final objective function for RLR is:

$$\min_w \quad \frac{\lambda}{2}\|w\|^2 + \sum_{i=1}^{N} \ln(1 + e^{-y_i w^T x_i}) \tag{7}$$

Comparing (7) to (1), we can see that the loss function that RLR minimizes is given by the second term in the objective also known as log loss. Based on the Representer theorem we can safely replace $w = \sum_i \alpha_i y_i x_i$ to get the dual version of RLR:

$$\min_\alpha \quad \frac{\lambda}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) + \sum_{j=1}^{N} \ln(1 + \exp(-y_j \sum_{i=1}^{N} \alpha_i y_i K(x_i, x_j)) \tag{8}$$

where, K(.,.) is the value of the kernel function.

6 Dual perceptron

Recall the perceptron training algorithm PERCEPTRON-PRIMAL-PROBLEM:

- assume all training points have been processed so that $y_i = 1$;
- start with vector w=0
- **repeat until no mistakes made**
  for all mistakes x (that is when $wx < 0$), add x to the plane normal w, to cause a plane correction towards x : $w^{new} = w + x$

6.0.1 Duality intuition (without representer theorem)

. The primal variables are vector w. We can dualize the perceptron, just like we did with the SVM, by observing that when its finished, the w must be the sum of the x-s added to it, which is the form

$w = \sum_i m_i x_i$

where m_i is the count of times/rounds when x_i was a mistake. Lets write the prediction for any point z as $wz = \sum_i m_i x_i z = \sum_i m_i < x_i * z >$, and thus a mistake on x_j is detected by $\sum_i m_i < x_i * x_j > < 0$.

We can then rewrite the perceptron into the PERCEPTRON-DUAL-PROBLEM:

- assume all training points are not processed, so that $y_i \in \{-1, 1\}$
- start with vector of counts $m = (m_1, m_2, ...m_n)$=0
- **repeat until no mistakes made**
  for all mistakes x_t tested as $y_t \sum_i m_i < x_i * x_t > \leq 0$, simulate adding x to the plane normal w, by increasing x_t count in w: $m_t = m_t + y_t$

The vector m are the dual variables. We can use a kernel with the DUAL-PERCEPTRON, because like with the SVM, both training and testing is written with formulas only containing dot product $< x * z >$, never x or z single variables; thus we can solve the dual for a kernel $k(x, z)$ replacing the dot product.

6.0.2 HW Questions

- Why in the dual perceptron we are not processing data to invert negative-labeled points (like we do for the primal perceptron), so that all datapoints have positive labels?
- What is the perceptron primal loss function? Write the primal perceptron as an optimization problem.
- Apply the representer theorem and substitute $w = \sum_i y_i \alpha_i x_i$ in the objective to obtain the dual from