EECS 189 Introduction to Machine Learning
Fall 2020

# DIS11

This discussion was released **Friday, November 13**.

## 1 Kernel Logistic Regression

We have seen in lecture how to kernelize ridge regression, and by going to the dual formulation, how to kernalize soft-margin SVMs as well. Here, we will consider how to kernelize logistic regression and compare its performance to kernelized SVMs using a real-world dataset.

(a) Imagine we have $n$ different $d$-dimensional data points in an $n \times d$ matrix $\mathbf{X}$, with associated labels in an $n$-dimensional vector $\mathbf{y}$. Let this be a binary classification problem, so each label $y_i \in \{0, 1\}$. Remember, this means that each training data point is associated with a row in the matrix $\mathbf{X}$ and the vector $\mathbf{y}$.

Recall that logistic regression associates a point $\mathbf{x}$ with a real number from 0 to 1 by computing:

$$f(\mathbf{x}) = \frac{1}{1 + \exp\{-\mathbf{w}^T\mathbf{x}\}},$$

This number can be interpreted as the estimated probability for the point $\mathbf{x}$ having a true label of +1. Since this number is $\frac{1}{2}$ when $\mathbf{w}^T\mathbf{x} = 0$, the sign of $\mathbf{w}^T\mathbf{x}$ is what predicts the label of the test point $\mathbf{x}$.

As you've seen in earlier homeworks, the loss function is defined to be

$$\sum_i -y_i \ln(f(\mathbf{x}_i)) - (1 - y_i)\ln(1 - f(\mathbf{x}_i)), \tag{1}$$

where the label of the $i$th point $\mathbf{x}_i$ is $y_i$.

**Write down the gradient-descent update step for logistic regression, with step size $\gamma$.** Assume that we are working with the raw features $\mathbf{X}$ for now, with no kernelization.

For convenience, define the logistic function $s(\cdot)$ to be

$$s(x) = \frac{1}{1 + e^{-x}}. \tag{2}$$

(b) You should have found that the update $\mathbf{w}^{(t+1)} - \mathbf{w}^{(t)}$ is a linear combination of the observations $\{\mathbf{x}_i\}$. This suggests that gradient descent for logistic regression might be compatible with the kernel trick. After all, this is the same thing that we saw when we were doing least-squares iteratively by gradient descent, and that was certainly something that we could kernelize.

When we argued for the kernelization of least-squares, we did so by means of the augmented-features view of ridge regression. That had the pedagogic advantage of helping you internalize

the importance of norm-minimization and made for an argument by which you could naturally discover the kernelization for yourself. Here, we will pursue a more direct path that has an inductive feeling to it.

Imagine that we start with some weight vector $\mathbf{w}^{(t)} = \mathbf{X}^T\mathbf{a}^{(t)}$ that is a linear combination of the $\{\mathbf{x}_i\}$. (Notice that if we start with the zero vector as our base case, this is true for the base case.) **Show that after one gradient step, $\mathbf{w}^{(t+1)}$ will remain a linear combination of the $\{\mathbf{x}_i\}$, and so is expressible as $\mathbf{X}^T\mathbf{a}^{(t+1)}$ for some "dual weight vector" $\mathbf{a}^{(t+1)}$. Then write down the gradient-descent update step for the dual weights $\mathbf{a}^{(t)}$ directly without referring to $\mathbf{w}^{(t)}$ at all.** In other words, tell us how $\mathbf{a}^{(t+1)}$ is obtained from the data, the step size, and $\mathbf{a}^{(t)}$.

(c) You should see from the previous part that the gradient-descent update step for $a^{(t)}$ can be written to depend solely on $\mathbf{X}\mathbf{X}^T$, not on the individual $\{\mathbf{x}_i\}$ in any other way. Since $\mathbf{X}\mathbf{X}^T$ is just the Gram matrix of pairwise inner-products of training point inputs, this suggests that we can use the kernel trick to quickly compute gradient steps for $\mathbf{a}^{(t)}$ so long as we can compute the inner products of any pair of featurized observations.

Now suppose that you just have access to a similarity kernel function $k(\mathbf{x}, \mathbf{z})$ that can be understood in terms of an implicit featurization $\boldsymbol{\phi}(\cdot)$ so that $k(\mathbf{x}, \mathbf{z}) = \boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{z})$. **Describe how you would compute gradient-descent updates for the dual weights $\mathbf{a}^{(t)}$ as well as how you would use the final weights together with the training data to classify a test point $\mathbf{x}$.**

Note: You do not have access to the implicit featurization $\boldsymbol{\phi}(\cdot)$. You have to use the similarity kernel $k(\cdot, \cdot)$ in your final answer.

(d) You have now derived kernel logistic regression! Next, we will study how it relates to the kernel SVM, which we will do numerically. **Complete all the parts in the associated Jupyter notebook.**

The notebook can be found here

Contributors:

- Anant Sahai
- Peter Wang
- Rahul Arya