CSEP 546 Data Mining/Machine Learning, Winter 2014: Homework 2

Due: Monday, February 3, beginning of class

1 Logistic regression [35 Points]

(Source: Hastie/Tibshirani/Friedman, Exercise 4.5) Suppose you are given two data points at $\mathbf{r} - \alpha$, $\mathbf{r} + \alpha$, which belong to the classes $\{-1, 1\}$ respectively ($\mathbf{r}, \alpha$ are arbitrary vectors). Assume that the probability of a data point $\mathbf{x}$ belonging to class y is given by,

$$P(\texttt{cl}(\mathbf{x}) = y|\mathbf{w}, w_0) = \frac{1}{1 + \mathrm{e}^{-y(\mathbf{w}^T x + w_0)}}.$$

Assuming, if you must, that $\mathbf{r}, \alpha \in \mathbb{R}^2$, answer the following questions,

1. *(5 Points)* Write down the log-likelihood of the given data under this model. We will call this $\ell(\mathbf{w}, w_0)$ hereonwards.

2. *(10 Points)* What value of w_0 maximizes $\ell(\mathbf{w}, w_0)$ for a given $\mathbf{w}$? We'll denote this maximum as $w_0^*(\mathbf{w})$. [1]

3. *(10 Points)* Consider the function $\ell'(\mathbf{w}) = \ell(\mathbf{w}, w_0^*(\mathbf{w}))$ (ℓ' is obtained by substituting $w_0^*(\mathbf{w})$ from (2) in $\ell(\mathbf{w}, w_0)$). What value of $\mathbf{w}$ maximizes ℓ' ? Is it finite ? Why or why not ? (Hint: When is the dot product maximized ?).

4. *(10 Points)* Suppose you're given a bound on $\mathbf{w}$, $\|\mathbf{w}\|_2 \leq t$. Find $\mathbf{w}$ which maximizes $\ell'(\mathbf{w})$ under this inequality bound. What does this tell you about the decision boundary ? (Hint: When is the dot product maximized ?).

5. *(Extra credit: 3 Points)* Is the maximum of $\ell'(\mathbf{w})$ equal to the maximum of $\ell(\mathbf{w}, w_0)$? What we're getting at, is whether maximizing a function $f(x, y)$ is the same as maximizing f with respect to y first and then with respect x ?

2 Kernels, separability [15 Points]

Consider a dataset with 3 points in 1d: $(x_1 = 0, y_1 = -1)$, $(x_2 = \sqrt{2}, y_2 = 1)$, and $(x_3 = 2\sqrt{2}, y_2 = -1)$. This dataset is not linearly separable. Consider mapping each point to 3d using the feature vector $\phi(x) = [1, \sqrt{2}x, x^2]^T$. (This is equivalent to using a second order polynomial kernel.)

[1] $\frac{d}{dx}\{\log(x), \mathrm{e}^x, x^n\} = \{1/x, \mathrm{e}^x, nx^{n-1}\}$.

1. *(5 points)* Are the data points linearly separable in the feature space ?

2. *(10 points)* Does your conclusion depend on the particular values of x_1, x_2, x_3 ? Explain ?

3 Programming Question

In this problem, we seek to perform a digit recognition task, where we are given an image of a handwritten digit and wish to predict what number it represents. This is a special case of an important area of language processing known as Optical Character Recognition (OCR). We will be simplifying our goal to that of a binary classification between two relatively hard-to-distinguish numbers (specifically, predicting a '3' versus a '5'). To do this, you will implement a kernelized Perceptron and linear support vector machine (SVM).

3.1 Dataset

The digit images have been taken from a Kaggle competition, `http://www.kaggle.com/c/digit-recognizer/data`. This data was originally from the MNIST database of handwritten digits, but was converted into a easier-to-use file format.

The data have also undergone some preprocessing. They have been filtered to just those datapoints whose labels are 3 or 5, which have then been relabeled to 1 and -1 respectively. Then, 1000-point samples have been created, named *validation.csv* and *test.csv* on the class website. Each row in these files represents an image. The first column is the label of the image, and the remaining columns give the grayscale values for each pixel.

3.2 Perceptron [50 Points]

1. *(10 points)* Write the prediction rule of kernelized Perceptron on the t-th iteration of its training procedure in terms of a kernel function $k(u, v) = \phi(u) \cdot \phi(v)$, where u and v are data points and ϕ is a projection to a feature space. That is, give an expression for sign $\left(w^{(t)} \cdot \phi(x)\right)$ in terms of k and previously misclassified data points, where $w^{(t)}$ is the weight vector estimated on the t-th iteration, and x is a data point whose label we want to predict. You will use this expression to implement Perceptron. Ignore the intercept, as it will appear naturally when we apply kernel functions. You may cite the lecture slides.

2. Implement Perceptron. Initially start all the weights at 0. Use one pass over the data. Write it so that it takes a kernel function as a parameter.

3. *(10 Points)* Consider the kernel $k_p^1(u, v) = u \cdot v + 1$. ($k_p^1$ is what the standard dot product would give us, if we had added a constant term 1.) Run Perceptron on the **validation** set with this kernel, and plot the average loss $\bar{L}$ as a function of the number of steps T, where

$$\bar{L}(T) = \frac{1}{T} \sum_{j=1}^{T} \mathbb{I}(\hat{y}^{(t)} \neq y^{(t)})$$

Here $\hat{y}^t$ is the label that Perceptron predicts for datapoint t as it runs, and $\mathbb{I}$ is an indicator function, which is 1 if its condition is true and 0 otherwise. Only show the average loss every 100 steps, e.g. [100, 200, 300, ...].

4. *(10 Points)* For a positive integer d, the polynomial kernel $k_p^d(u, v) = (u \cdot v + 1)^d$ maps x into a feature space of all polynomials of degree up to d. For the set $d = [1, 3, 5, 7, 10, 15, 20]$, run Perceptron for a single pass over the **validation** set with k_p^d, and plot the average loss over the validation set $\bar{L}(1000)$ as a function of d.

5. *(10 Points)* For $\sigma > 0$, the Gaussian kernel $k_G^\sigma(u, v) = \exp\left(-\frac{\|u-v\|^2}{2\sigma^2}\right)$ is a map to *all* polynomials of x, where σ is a tuning constant that roughly corresponds to the "window size" of the distribution. The Gaussian kernel, also known as the radial basis function (RBF) kernel, is one of the most popular kernel functions because it performs well on a variety of data, making it an ideal choice as a default kernel.

   Tuning on the validation set has produced a value of $\sigma = 1000$. For the best d in the previous step, run Perceptron with both k_p^d and k_G^{1000} for a single pass over the **testing** set. (Use a fresh run of the algorithm; do not reuse the set of mistakes from the validation set.) For each of these two kernels, plot the average loss $\bar{L}(T)$ as a function of the number of steps on the same graph. As above, show the average loss for every 100 steps.

3.3 SVM [Extra Credits: 15 Points]

In this problem we consider only linear SVMs—that is, we do not use kernels. Although kernels would perform better, their implementation is quite complicated. Linear SVMs, on the other hand, can be trained by stochastic gradient descent, though this method is not the best known.

1. *(5 points)* Write the stochastic gradient descent update rules for both intercept and non-intercept weights in linear SVMs.

2. Implement linear SVMs. Initially start all the weights at 0. Use one pass over the data.

3. *(5 points)* Run SVM on the **testing** set with $\eta = 10^{-5}$ and $C = 1$. (Use a fresh run of the algorithm; do not use weights learned from the validation set.) Plot the average loss for every 100 steps.

4. *(5 points)* Since C should matter, describe how the expected behavior of linear SVMs changes as C increases and decreases. What is the relationship to Perceptron?