Introduction to Machine Learning **Fall Semester**

Homework 2: Dec 7th, 2016

Due: Dec 25th (See the submission guidelines in the course web site)

Theory Questions

1. ℓ^2 **penalty for SVM.** The formulation we have seen in class for the soft margin SVM assumes we want to minimize $\sum_{i=1}^{m} \xi_i$. However, in some cases we might want a different loss function for the margin penalty. Consider the following problem:

$$\begin{aligned} &\min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{2}\sum_{i=1}^{m}\xi_i^2 \\ &\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \qquad \forall i = 1, \ldots, m \end{aligned}$$

   (a) Notice that we have dropped the $\xi_i \geq 0$ constraint in this problem. Show that these non-negativity constraints can be removed. That is, show that the optimal value of the objective will be the same whether or not these constraints are present.

   (b) What is the Lagrangian of this problem?

   (c) Minimize the Lagrangian with respect to $\mathbf{w}, b, \boldsymbol{\xi}$ by setting the derivative with respect to these variables to 0.

   (d) What is the dual problem?

2. **Upper bound on $\|\mathbf{w}\|$.** Denote by $\mathbf{w}^*, b^*$ and $\boldsymbol{\xi}^*$ the optimal solutions to the primal SVM soft margin optimization problem, and denote by $\boldsymbol{\alpha}^*$ the optimal solution to the dual problem.

   (a) Recall that we know from the duality formulation, that the primal objective value is equal to the dual objective value at the optimum. Write this equation, using $\mathbf{w}^*, b^*, \boldsymbol{\alpha}^*$ and $\boldsymbol{\xi}^*$.

   (b) Using this equality, show that $\|\mathbf{w}^*\| \leq \sqrt{Cm}$, where C is the penalty coefficient and m is the number of samples.

3. **Sequential minimal optimization.** Sequential minimal optimization (SMO) is an algorithm for solving the quadratic programming problem that arises during the training of the SVM problem. In this question we will develop a simpler variant of it. We will be interested in the soft margin SVM problem without an intercept term (that, is, $b = 0$):

$$\begin{aligned} &\min_{\mathbf{w},\boldsymbol{\xi}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\xi_i \\ &\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i) \geq 1 - \xi_i \qquad \forall i = 1, \ldots, m \\ &\quad \xi_i \geq 0 \end{aligned}$$

(a) (Do not submit.) Show that the dual problem is the same as developed in class, but without the constraint $\sum_{i=1}^{m} \alpha_i y_i = 0$.

(b) We will now try to solve the dual problem using a coordinate gradient ascent approach. That each, at every iteration, we will assume all the coordinates are fixed except for one, and optimize the changing coordinate. Assume that $\alpha_2, \ldots, \alpha_m$ are fixed. What is the optimization problem with α_1?

(c) Solve the optimization problem. Note that α_1 should satisfy the dual constraints.

(d) The SMO algorithm iterates through all the coordinates, at each step applying the optimization procedure described, until no further optimization is possible. What is the computational complexity of a single iteration? What is the space complexity?

4. **Deriving new kernels.** Given $a > 0$, $K1, K2$ positive definite kernels, f a real valued function, and p a polynomial with positive coefficients. For each of the functions K below, prove whether it is necessarily a positive definite kernel or give a counter-example:

   (a) $K(x, z) = K1(x, z) - K2(x, z)$

   (b) $K(x, z) = \frac{K1(x,z)}{K2(x,z)}$

   (c) $K(x, z) = aK1(x, z)$

   (d) $K(x, z) = -aK1(x, z)$

   (e) $K(x, z) = f(x)f(z)$

   (f) $K(x, z) = p(K1(x, z))$

5. **Kernel Perceptron.** We are given a data sample $\mathbf{x}_i \in \mathbb{R}^d$, with corresponding labels $y_i, \ldots \in \{-1, 1\}$. The Perceptron algorithm is run on the sample for T iterations, until getting the final returned hypothesis, $\mathbf{w}^* = \mathbf{w}_{T+1}$.

   (a) Show that $\mathbf{w}^*$ is a linear combination of the samples $\mathbf{x}_i$.

   (b) Change the Perceptron algorithm to use kernels. The algorithm will not (and cannot) keep $\mathbf{w}^*$ explicitly (why?). Write the pseudo-code to describe the new algorithm, and demonstrate how its result can be used to classify a new sample point, $\mathbf{x}$.

Programming Assignment

In this exercise, we will study the performance of various hyperplane algorithms on the MNIST dataset, with which you are now familiar. As before, we will have a held out *test set* that will be used to assess the performance of the selected best hypotheses. However, we will divide the rest of the data to a *training set* and *validation set*. For algorithms in which we have to select external parameters (e.g. C for SVM), we will use use the training set to train classifier for various parameter values and use the validation set to check how well they do.

Under

`~schweiger/courses/ML2016-7/hw2.py`

(in the file system accessible from `nova`) you will find the code to load the training, validation and test sets. It is recommended to use `numpy` and `scipy` where possible. Your program should run in under an hour or so (it can be faster though).

Here we will investigate how well we can classify a digit as either 0 or 8. We will only use linear classifiers with $b = 0$, and we will mean-center each coordinate of our data points (this is done in the script above).

1. **Perceptron.** Implement the Perceptron algorithm. Do not forget to normalize the samples to have unit length (i.e., $\|\mathbf{x}_i\| = 1$).

   (a) Use only the first $n = 5, 10, 50, 100, 500, 1000, 5000$ samples as an input to Perceptron. For each n, run Perceptron 100 times, each time with a different random order of inputs, and calculate the accuracy of the classifier on the *test set* of each run. You should therefore have 100 accuracy measurement per n. Print a table showing, for each n, the mean accuracy across the 100 runs, as well as the 5% and 95% percentiles of the accuracies obtained.

   (b) The weight vector $\mathbf{w}$, returned by Percepton, can be viewed as a matrix of weights, with which we multiply each respective pixel in the input image. Run Perceptron on the entire *training set*, and show $\mathbf{w}$, as a 28×28 image, for example with `imshow(reshape(image, (28, 28)), interpolation='nearest')`. Give an intuitive interpretation of this image.

   (c) Calculate the accuracy of the classifier trained on the full *training set*, applied on the *test set*.

   (d) Choose one (or two) of the samples in the *test set* that was misclassified by Perceptron (with the full training set) and show it as an image (show the unscaled images). Can you explain why it was misclassified?

2. **SVM.** Now, we will see how well SVM works in this problem. Will we first try an existing implementation: Use the `LinearSVC` class from `sklearn.SVM`. Use the `loss='hinge'` and `fit_intercept=False` flags.

   (a) Train a linear SVM on the *training set*, while using cross-validation on the *validation set* to select the best penalty constant, C. It is highly recommended to search C on the log scale, e.g., perform a grid search $C = 10^{-10}, 10^{-9}, \ldots, 10^{10}$, and increase the resolution until you are satisfied.

   Plot the accuracy of the resulting hyperplane on the *training set* and on the *validation set*, as a function of C. What is the best C you found?

(b) Explain the behavior of training error as a function of C.

(c) Show the $\mathbf{w}$ for the best C as an image (see `LinearSVC` class documentation on how to extract it).

(d) What is the accuracy of the linear SVM with the best C on the *test set*?

3. **SGD.** Now, we will try our own implementation of SVM, using the stochastic gradient descent implementation discussed in class. Namely, at each iteration $t = 1, \ldots$ we sample i uniformly; and if $y_i \mathbf{w}_t \cdot \mathbf{x}_i < 1$, we update:

$$\mathbf{w}_{t+1} = (1 - \eta_t)\mathbf{w}_t + \eta_t C y_i \mathbf{x}_i$$

where $\eta_t = \eta_0/t$, and η_0 is a constant. Implement an SGD function that accepts the samples and their labels, C, η_0 and T, and runs for a specified number of iterations T.

(a) Train the SGD SVM classifier on the *training set*. Use cross-validation on the *validation set* to find the best η_0, assuming $T = 1000$ and $C = 1$. For each possible η_0, assess the performance of η_0 by averaging the accuracy on the *validation set* across 10 runs. Plot the average accuracy on the *validation set*, as a function of η_0.

(b) Now, cross-validate on the *validation set* to find the best C given the best η_0 you found above. For each possible C, average the accuracy on the *validation set* across 10 runs. Plot the average accuracy on the *validation set*, as a function of C.

(c) Using the best C, η_0 you found, train the SGD SVM classifier, but for $T = 20000$. Show the resulting $\mathbf{w}$ as an image.

(d) What is the accuracy of the best classifier on the *test set*?

4. **Bonus (<10pts).** Can you find an SVM classifier (possibly with a kernel) that gives better accuracy? Validate parameters on the validation set and measure your success on the test set. Describe your attempts and the results you got, with suitable plots. No need to submit code for this part.