Introduction to Machine Learning **Fall Semester**

Homework 3: Dec 21st, 2016

Due: Jan 4th, 2017 (See the submission guidelines in the course web site)

Theory Questions

1. **VC-dimension of Neural Networks - Lower bound.** Let $\mathcal{C}$ be the class of hypotheses implementable by neural networks (NN) as described in recitation 7 - that is, there are L layers (including the output node, not including the input layer); each layer has exactly d nodes (except, of course, the output node); and the activation function is *sign*.

   Show that $VCdim(\mathcal{C}) \geq d$.

2. **VC-dimension of Neural Networks - Upper bound.** We will finish what we started in recitation 7. Denote by $\mathcal{H}$ the family of hyperplane separators in $\mathbb{R}^d$. We have seen that the output function of any single node i in the t-th layer, $f_{i,t}$, is a member of $\mathcal{H}$.

   (a) Seen as a whole function, layer t implements a function from $\mathbb{R}^d$ to $\mathbb{R}^d$. Denoting this function by $f^{(t)}$:
   $$f^{(t)} := \mathbf{z}_{t+1} = h(\mathbf{W}^{(t+1)}\mathbf{z}_t - \mathbf{b}^{(t+1)})$$
   Express the hypothesis class of functions of the form in terms of $\mathcal{H}$. Give a bound on the growth function of this class, for $m \geq d+1$.

   (b) Express $\mathcal{C}$ in terms of $\mathcal{H}$. Give a bound on the growth function of $\mathcal{C}$, for $m \geq d+1$.

   (c) Let N by the number of parameters in a multilayer NN as defined above. Express N in terms of d and L.

   (d) (Bonus:) Show that $2^m \leq (em)^N \Rightarrow m \leq 2N\log_2(eN)$.

   (e) We are finally in a position to derive a bound for the VC-dimension. Show $\pi_{\mathcal{C}}(m) \leq (em)^N$, and use this to show that $VCdim(\mathcal{C}) \leq 2N\log_2(eN)$.

3. **SGD with projection.** In the context of convex optimization, sometimes we would like to limit our solution to a convex set $\mathcal{K} \subseteq \mathbb{R}^d$; that is,
   $$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in \mathcal{K} \end{aligned}$$
   for a convex function f and a convex set $\mathcal{K}$. In this scenario, each step in the gradient descent algorithm might result in a point outside $\mathcal{K}$. Therefore, we add an additional projection step. The projection operator finds the closest point in the set, i.e.:
   $$\Pi_{\mathcal{K}}(\mathbf{y}) := \arg\min_{\mathbf{x}\in\mathcal{K}} \|\mathbf{x}-\mathbf{y}\|$$
   A modified iteration in the *gradient descent with projection* therefore consists of:
   $$\begin{aligned} \mathbf{y}_{t+1} &= \mathbf{x}_t - \eta_t \nabla f(\mathbf{x}_t) \\ \mathbf{x}_{t+1} &= \Pi_{\mathcal{K}}(\mathbf{y}_{t+1}) \end{aligned}$$

(a) Suppose we want to solve the SVM primal problem, as formulated in the context of SGD:

$$\frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m}\max(0, 1 - y_i\mathbf{w}\cdot\mathbf{x}_i)$$

However, we want to find a solution with a bounded norm; i.e. $\mathbf{w} \in \mathcal{K}$, where $\mathcal{K} = \{\mathbf{x}|\|\mathbf{x}\| \leq R\}$. Modify the SGD algorithm to include a projection step. How do you calculate the projection?

(b) (Bonus:) Let $\mathcal{K}$ be a convex set, $\mathbf{y} \in \mathbb{R}^d$ and $\mathbf{x} = \Pi_{\mathcal{K}}(\mathbf{y})$. Prove that for any $\mathbf{z} \in \mathcal{K}$, we have $\|\mathbf{y} - \mathbf{z}\| \geq \|\mathbf{x} - \mathbf{z}\|$.

(c) Prove that Theorem 1.1 in Lesson 7 scribe still holds (Hint: Does the equation after Equation (12) still hold as is?).

4. **SVM with multiple classes.** One limitation of the standard SVM is that it can only handle binary classification. Here is one extension to handle multiple classes. Let $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^d$ and now let $y_1, \ldots, y_m \in [K]$, where $[K] = \{1, 2, \ldots, K\}$. We will try to find a separate classifier $\mathbf{w}_j$ for each one of the classes $j \in [K]$, and we will focus on the case of no bias ($b = 0$). Define the following loss function:

$$\ell(\mathbf{w}_1, \ldots, \mathbf{w}_K, \mathbf{x}_i, y_i) = \max_{j\in[K]}\left(\mathbf{w}_j \cdot \mathbf{x}_i - \mathbf{w}_{y_i} \cdot \mathbf{x}_i + \mathbb{1}(j \neq y_i)\right)$$

Define the following multiclass SVM problem:

$$f = \sum_{j\in[K]}\frac{1}{2}\|\mathbf{w}_j\|^2 + \frac{C}{m}\sum_{i=1}^{m}\ell(\mathbf{w}_1, \ldots, \mathbf{w}_K, \mathbf{x}_i, y_i)$$

After learning $\mathbf{w}_j$, classification of a new point $\mathbf{x}$ is done by $\arg\max_{j\in[K]} \mathbf{w}_j \cdot \mathbf{x}$. The rationale of the loss function is that we want the "score" of the true label, $\mathbf{w}_{y_i} \cdot \mathbf{x}_i$, to be larger by at least 1 than the "score" of each other label, $\mathbf{w}_j \cdot \mathbf{x}_i$. Therefore, we pay a loss if $\mathbf{w}_{y_i} \cdot \mathbf{x}_i - \mathbf{w}_j \cdot \mathbf{x}_i \leq 1$, for $j \neq y_i$.

(a) (Bonus:) Show that when $K = 2$, f reduces to the standard SVM with binary classification. That is, denote by $\mathbf{w}_1^*(C), \mathbf{w}_2^*(C)$ the solution of the multiclass problem with $K = 2$, with a penalty C. Denote by $\mathbf{w}^*(C')$ best solution of the standard SVM, with a penality C' (see 3a). Show that for every C, there is a C', so that multiclass classifer defined by $\mathbf{w}_1^*(C), \mathbf{w}_2^*(C)$ and the standard SVM classifier defined by $\mathbf{w}^*(C')$ are the same. Show the correspondence C and C', and between $\mathbf{w}_1^*(C), \mathbf{w}_2^*(C)$ and $\mathbf{w}^*(C')$ (i.e., an equation showing the relationship between them).

(b) Calculate the subgradient of ℓ, with respect to each of the $\mathbf{w}_j$, for a given $\mathbf{x}_i, y_i$. Use it to derive a Stochastic (Sub)gradient Descent algorithm to find $\mathbf{w}_j, j \in [K]$ that minimize f (use a constant step size, η).

(c) Convert your SGD algorithm to use a kernel K (Hint: Recall the Kernel Perceptron algorithm). The algorithm will not (and cannot) keep $\mathbf{w}$ explicitly. Write the pseudocode to describe the new algorithm, and demonstrate how its result can be used to classify a new sample point, $\mathbf{x}$.

5. **Binary decision trees.** Show that any binary classifier $h : \{0,1\}^d \to \{0,1\}$ can be implemented as a decision tree of height at most $d+1$, with internal nodes of the form $(x_i = 0?)$ for some $i \in \{1,\ldots,d\}$. Conclude that the VC-dimension of the class of decision trees over the domain $\{0,1\}^d$ is 2^d.

Programming Assignment

In this exercise, we will study the performance of the multiclass SVM on the MNIST dataset, with which you are now *very* familiar. We divide the data into *training set*, *validation set* and *test set*, as in the previous HW. Under

`~schweiger/courses/ML2016-7/hw3.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. Cross validation is time consuming, so it's OK if it takes time.

Here we will investigate how well we can classify a digit to $0, 1, \ldots, 9$. 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).

6. **Linear Multiclass SVM.** Implement the SGD algorithm you developed above, in the version without a kernel.

   (a) Train the algorithm on the *training set*. By now you are experts on choosing parameters, so find the best C, η and choose a sufficient number of iterations, T. You can use any searching scheme you like, as long as it is justified and well explained. Supply plots to demonstrate your search strategy. For each plot, show the training error as well as the validation error. As usual, validate the accuracy for each parameter set on the *validation set*.

   (b) Each weight vector $\mathbf{w}_j$, $j \in [K]$, can be viewed as a matrix of weights, with which we multiply each respective pixel in the input image. For your selected classifiers, show each $\mathbf{w}_j$, as a 28×28 image, for example with `imshow(reshape(image, (28, 28)), interpolation='nearest')`. Do these images reflect the digits they should classify?

   (c) Calculate the accuracy of your classifier, applied on the *test set*.

7. **Kernel Multiclass SVM.** Implement the SGD algorithm you developed above, in the kernel version. (It is recommended to first use the linear kernel and see you get comparable results, for debugging purposes.)

   (a) Use the quadratic kernel, and train the algorithm on the *training set*, while validating on the *validation set*. As in 1(a), find good C, η and choose the number of iterations, T.

   (b) Calculate the accuracy of your classifier, applied on the *test set*.

   (c) (**Bonus <10pts**). Can you do better with another kernel? Describe your attempts and the results you got, with suitable plots. No need to submit code for this part.