New York University Tandon School of Engineering
Computer Science and Engineering

Final Exam Sample Problems

Always, Sometimes, Never. (15pts – 3pts each)

Indicate whether each of the following statements is ALWAYS true, SOMETIMES true, or NEVER true. For full credit, provide a short justification or example to explain your choice.

(a) Given a linearly separable data set, an optimal solution to the soft-margin SVM objective will be a correct separating hyperplane for the dataset..

ALWAYS **SOMETIMES** NEVER

The soft-margin SVM allows some points to be misclassified, even if the data set is linearly seperable. If we set C large enough, however, a soft-margin SVM converges to a hard margin classifier, so we will get a seperating hyperplane.

(b) Let K be a kernel gram matrix generated from a datasets $x_1, \ldots, x_n$ and a PSD kernel function k. K can always be written as BB^T for some matrix B.

ALWAYS SOMETIMES NEVER

If K is PSD, then $K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle$. So we can choose B to have its i^{th} row equal to $\phi(x_j)$.

(c) Suppose we have a random event X that happens with probability $1/2$ and a random event Y that happens with probability $1/4$. There is at least a 25% chance neither event happens.

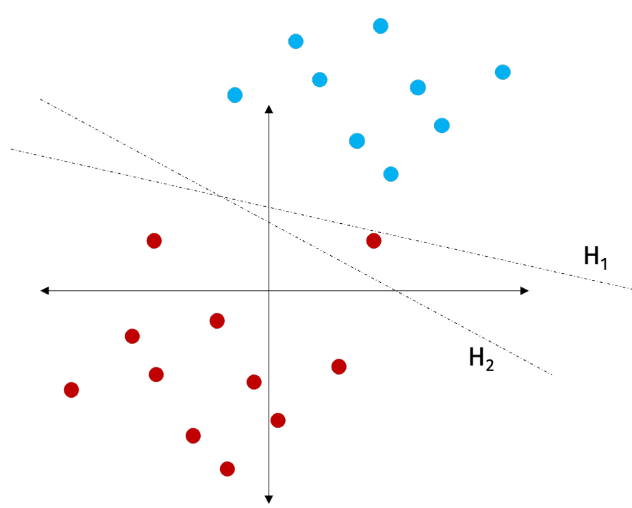
ALWAYS SOMETIMES NEVER

This is a special statement of union bound. $\Pr(X \text{ or } Y) \leq \Pr(X) + \Pr(Y) = .75$.

Short Answer (12pts – 2pts each)

Respond to each of the following questions using just a few words.

(a) In the plot below, H_1 and H_2 are hyperplanes obtained by trainging a soft-margin SVM with different values of C. Which one was trained with a larger value of C? On the same plot draw the hyperplane that you believe would be returned by a hard margin SVM.



H_1 was likely trained with a larger value of C, since its solution has fewer total misclassified points. H_1 more closely approximates the hard margin classifier. H_2 was likely trained with the smaller value of C.

(b) TRUE or FALSE. PCA is a type of linear autoencoder. TRUE

(c) Suppose you train two binary classifiers, h_1 and h_2, on the same training data, from two function classes $\mathcal{H}_1$ and $\mathcal{H}_2$ with $|\mathcal{H}_1| < |\mathcal{H}_2|$. Suppose h_1 and h_2 have the same training error. Then the PAC based generalization error bound for h_1 is:

(1) Smaller than that for h2.

(2) Larger than that for h2

(3) Equal to that for h2

(4) We can't say anything about the relationship between the two.

(1). Refer to the PAC generalization bound. Another equivalent way of writing this bound is that the generalization error, $R_{pop}(h) \leq \frac{\log(1/\delta)+\log|\mathcal{H}|}{n}$. So, if the number of training examples, n is fixed, the bound is smaller for smaller hypothesis classes.

(d) An alternative definition of a PSD kernel function that you will see in many text books is as follows: We say that k is PSD if for *any* dataset $x_1, \dots, x_n$, the kernel gram matrix K with $K_{ij} = k(x_i, x_j)$ is "positive semi-definite", where we say a matrix K is positive semidefinite if, for *all* vectors x, $x^T K x \geq 0$ (this is a definition you might have seen before in a linear algebra class, we also discussed it briefly in the class). Prove that the other definition we gave for k (i.e. that $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ for some feature transformtion ϕ implies this definition (you don't need to show the other way).

As discussed earlier, we know that $K = \Phi\Phi^T$ for some matrix Φ. so $x^T K x = x^T \Phi\Phi^T x = \|\Phi^T x\|_2^2$. The squared two norm of a vector is always ≥ 0, so our definition implies that $x^T K x \geq 0$ as desired.

(e) What is the runtime complexity of computing a single stochastic gradient (involving one data point $\mathbf{x}$ and label y) for a neural network with d parameters? This can be done using backpropagation, which has linear complexity, so $O(d)$.

Alternating Minimization

When finding a k-rank approximation (e.g. for semantic embedding), we are given a matrix $\mathbf{M} \in \mathbb{R}^{n \times d}$ and our goal is to learn two matrices $\mathbf{W} \in \mathbb{R}^{n \times k}$ and $\mathbf{Y} \in \mathbb{R}^{k \times d}$ such that $\mathbf{M} \approx \mathbf{WY}$. If we want to minimize the Frobenius norm loss $\|\mathbf{M} - \mathbf{WY}\|_F^2$, we can find $\mathbf{W}$ and $\mathbf{Y}$ using an SVD. However, there is another approach called *alternating minimization* that works well in practice and more easily generalizes to other loss functions (e.g. $L1$ norm, losses with regularization, etc).

The approach is as follows. Suppose we have a loss function $L(\mathbf{M}, \mathbf{W}, \mathbf{Y})$, e.g. $L(\mathbf{M}, \mathbf{W}, \mathbf{Y}) = \|\mathbf{M} - \mathbf{WY}\|_F^2$ or $L(\mathbf{M}, \mathbf{W}, \mathbf{Y}) = \sum_{i,j} |\mathbf{M}_{ij} - (\mathbf{WY})_{ij}|^2$. We can run the following iteration, which produces a sequence of approximate solutions $W^{(0)}, Y^{(0)}, W^{(1)}, Y^{(1)}, \dots, W^{(t)}, Y^{(t)}$.

- Randomly initialize $\mathbf{W}^{(0)}$ and $\mathbf{Y}^{(0)}$
- For $t = 1, \dots, T$
  - $\mathbf{Y}^{(t)} = \arg\min_{\mathbf{Y}} L(\mathbf{M}, \mathbf{W}^{(t-1)}, \mathbf{Y})$
  - $\mathbf{W}^{(t)} = \arg\min_{\mathbf{W}} L(\mathbf{M}, \mathbf{W}, \mathbf{Y}^{(t)})$
- Return $\mathbf{W}^{(t)}$, $\mathbf{Y}^{(t)}$.

In words, we start by keeping $\mathbf{W}$ fixed, and only optimizing over $\mathbf{Y}$, then keeping $\mathbf{Y}$ fixed and only optimizing over $\mathbf{W}$. This process repeats for T steps, at which point we have hopefully converged on a good solution.

(a) (4pts) Show that $L(\mathbf{M}, \mathbf{W}^{(t)}, \mathbf{Y}^{(t)}) \leq L(\mathbf{M}, \mathbf{W}^{(t-1)}, \mathbf{Y}^{(t-1)})$. In other words, our loss decreases at every iteration, which implies that the alternating minimization processes converges to a local minimum.

By the definition of $\mathbf{Y}^{(t)}$, we have that $L(\mathbf{M}, \mathbf{W}^{(t-1)}, \mathbf{Y}^{(t)}) \leq L(\mathbf{M}, \mathbf{W}^{(t-1)}, \mathbf{Y})$ for any other choice of $\mathbf{Y}$. In particular:

$$L(\mathbf{M}, \mathbf{W}^{(t-1)}, \mathbf{Y}^{(t)}) \leq L(\mathbf{M}, \mathbf{W}^{(t-1)}, \mathbf{Y}^{(t-1)}).$$

Similarly, By the definition of $\mathbf{W}^{(t)}$, we have that $L(\mathbf{M}, \mathbf{W}^{(t)}, \mathbf{Y}^{(t)}) \leq L(\mathbf{M}, \mathbf{W}, \mathbf{Y}^{(t)})$ for any other choice of $\mathbf{W}$. In particular:

$$L(\mathbf{M}, \mathbf{W}^{(t)}, \mathbf{Y}^{(t)}) \leq L(\mathbf{M}, \mathbf{W}^{(t-1)}, \mathbf{Y}^{(t)}).$$

Chaining together the above inequalities proves that $L(\mathbf{M}, \mathbf{W}^{(t)}, \mathbf{Y}^{(t)}) \leq L(\mathbf{M}, \mathbf{W}^{(t-1)}, \mathbf{Y}^{(t-1)})$.

(b) (5pts) Prove that for the standard Frobenius norm loss, $L(\mathbf{M}, \mathbf{W}, \mathbf{Y}) = \|\mathbf{M} - \mathbf{W}\mathbf{Y}\|_F^2$, the right matrix update step has the following closed form, which does not require an SVD to compute:

$$\mathbf{Y}^{(t)} = (\mathbf{W}^{(t-1)^T}\mathbf{W}^{(t-1)})^{-1}\mathbf{W}^{(t-1)^T}\mathbf{M}$$

Hint: Rewrite the loss rewrite $L(\mathbf{M}, \mathbf{W}, \mathbf{Y}) = \|\mathbf{M} - \mathbf{W}\mathbf{Y}\|_F^2$ using the fact that the squared Frobenius norm of a matrix is equal to the sum of its squared column norms.

Following the hint, we can write:

$$\|\mathbf{M} - \mathbf{W}\mathbf{Y}\|_F^2 = \sum_{i=1}^{d} \|\mathbf{m}^{(i)} - \mathbf{W}\mathbf{y}^{(i)}\|_2^2,$$

where $\mathbf{m}^{(i)}$ and $y^{(i)}$ are the i^{th} columns of $\mathbf{M}$ and $\mathbf{Y}$.

Our free parameters in the optimization problem are the columns $\mathbf{y}_1, \ldots, \mathbf{y}_d$. Since $\mathbf{y}^{(i)}$ only appears in the i^{th} term of the sum above, we should choose:

$$\mathbf{y}^{(1)} = \arg\min_{\mathbf{y}} \|\mathbf{m}^{(1)} - \mathbf{W}\mathbf{y}^{(1)}\|_2^2 \quad \mathbf{y}^{(2)} = \arg\min_{\mathbf{y}} \|\mathbf{m}^{(2)} - \mathbf{W}\mathbf{y}^{(2)}\|_2^2 \quad \ldots \quad \mathbf{y}^{(d)} = \arg\min_{\mathbf{y}} \|\mathbf{m}^{(d)} - \mathbf{W}\mathbf{y}^{(d)}\|_2^2$$

But this is just a set of d linear regresison problems. So, we can used the closed form for the minimum of a linear regression problems. I.e., to minimize $L(\mathbf{M}, \mathbf{W}, \mathbf{Y})$, we should choose:

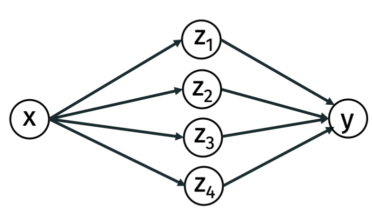
$$\mathbf{y}^{(i)} = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{m}^{(i)}.$$

Stacking everything together horizontally,

$$\arg\min_{\mathbf{Y}} L(\mathbf{M}, \mathbf{W}, \mathbf{Y}) = [(\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{m}^{(1)}, \ldots, (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{m}^{(d)}] = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{M}.$$

Problem 2: Neural Networks for Curve Fitting (15pts)

Consider the following 2-layer, feed forward neural network for single variate regression:



Let $W_{H,1}, W_{H,2}, W_{H,3}, W_{H,4}$ and $b_{H,1}, b_{H,2}, b_{H,3}, b_{H,4}$ be weights and biases for the hidden layer. Let $W_{O,1}, W_{O,2}, W_{O,3}, W_{O,4}$ and b_O be weights and bias for the output layer. The hidden layer uses rectified linear unit (ReLU) non-linearities and the output layer uses no non-linearity.

Specifically, for $i = 1, \ldots, 4$, $z_i = \max(0, \bar{z}_i)$ where $\bar{z}_i = W_{H,i}x + b_{H,i}$. And

$$y = b_O + \sum_{i=1}^{4} W_{O,i} z_i.$$

(a) For input parameters $\vec{\theta}$ let $f(x, \vec{\theta})$ denote the output of the neural network for a given input x. We want to train the network under the squared loss. Specifically, given a training dataset $(x_1, y_1), \ldots, (x_n, y_n)$, we want to choose $\vec{\theta}$ to minimize the loss:

$$\mathcal{L}(\vec{\theta}) = \sum_{i=1}^{n} (y_i - f(x_i, \vec{\theta}))^2.$$

Write down an expression for the gradient $\nabla \mathcal{L}(\vec{\theta})$ in terms of $\nabla f(x, \vec{\theta})$. **Hint:** Use chain rule.

$$\begin{aligned} \nabla \mathcal{L}(\vec{\theta}) &= \sum_{i=1}^{n} \nabla (y_i - f(x_i, \vec{\theta}))^2 \\ &= \sum_{i=1}^{n} -2(y_i - f(x_i, \vec{\theta})) \cdot \nabla f(x_i, \vec{\theta}) \end{aligned}$$

(b) Suppose we randomly initialize the network with ± 1 random numbers:

$$\begin{aligned} W_{H,1} = -1, W_{H,2} = 1, W_{H,3} = 1, W_{H,4} &= -1 \\ b_{H,1} = 1, b_{H,2} = 1, b_{H,3} = -1, b_{H,4} &= 1 \\ W_{O,1} = -1, W_{O,2} = -1, W_{O,3} = -1, W_{O,4} &= 1 \\ b_O &= 1 \end{aligned}$$

Call this initial set of parameter $\vec{\theta}_0$. Use forward-propagation to compute $f(x, \vec{\theta}_0)$ for $x = 2$.

First we compute:

$$\begin{aligned} \bar{z}_1 &= -1 & z_1 &= 0 \\ \bar{z}_2 &= 3 & z_2 &= 3 \\ \bar{z}_3 &= 1 & z_3 &= 1 \\ \bar{z}_4 &= -1 & z_4 &= 0 \end{aligned}$$

And then we see that $y = f(x, \vec{\theta}_0) = -3$.

(c) Use back-propagation to compute $\nabla f(x, \vec{\theta}_0)$ for $x = 2$. To do the computation you will need to use the derivative of the ReLU function, $\max(0, z)$. You can simply use:

$$\frac{\partial}{\partial z} \max(0, z) = \begin{cases} 0 & \text{if } z \leq 0 \\ 1 & \text{if } z > 0 \end{cases}$$

This derivative is discontinuous, but it turns out that is fine for use in gradient descent.

First we compute derivatives for the last layer of weights:

$$\frac{\partial f}{b_O} = 1$$
$$\frac{\partial f}{W_{O,1}} = z_1 = 0$$
$$\frac{\partial f}{W_{O,2}} = z_2 = 3$$
$$\frac{\partial f}{W_{O,3}} = z_3 = 1$$
$$\frac{\partial f}{W_{O,4}} = z_4 = 0$$

Then for the hidden layer of nodes:

$$\frac{\partial f}{\partial z_1} = W_{O,1} = -1 \qquad \frac{\partial f}{\partial \bar{z}_1} = -1 \cdot \frac{\partial z_1}{\partial \bar{z}_1} = 0$$
$$\frac{\partial f}{\partial z_2} = W_{O,2} = -1 \qquad \frac{\partial f}{\partial \bar{z}_2} = -1 \cdot \frac{\partial z_2}{\partial \bar{z}_2} = -1$$
$$\frac{\partial f}{\partial z_3} = W_{O,3} = -1 \qquad \frac{\partial f}{\partial \bar{z}_3} = -1 \cdot \frac{\partial z_3}{\partial \bar{z}_3} = -1$$
$$\frac{\partial f}{\partial z_4} = W_{O,4} = 1 \qquad \frac{\partial f}{\partial \bar{z}_4} = 1 \cdot \frac{\partial z_4}{\partial \bar{z}_4} = 0$$

Then for the first layer of weights:

$$\frac{\partial f}{b_{H,1}} = \frac{\partial f}{\partial \bar{z}_1} = 0 \qquad \frac{\partial f}{W_{H,1}} = x \cdot \frac{\partial f}{\partial \bar{z}_1} = 0$$
$$\frac{\partial f}{b_{H,2}} = \frac{\partial f}{\partial \bar{z}_2} = -1 \qquad \frac{\partial f}{W_{H,2}} = x \cdot \frac{\partial f}{\partial \bar{z}_2} = -2$$
$$\frac{\partial f}{b_{H,3}} = \frac{\partial f}{\partial \bar{z}_3} = -1 \qquad \frac{\partial f}{W_{H,3}} = x \cdot \frac{\partial f}{\partial \bar{z}_3} = -2$$
$$\frac{\partial f}{b_{H,4}} = \frac{\partial f}{\partial \bar{z}_4} = 0 \qquad \frac{\partial f}{W_{H,4}} = x \cdot \frac{\partial f}{\partial \bar{z}_4} = 0$$