COL 865: Deep Learning.
Assignment 0

Due: Monday August 28. 11:50 pm. Weightage: 6%

Notes:

- This assignment should be done individually.
- Some questions have been borrowed from existing online resources.

1 Linear Algebra

1.1 SVD

Suppose A is an $m \times n$ matrix and B is the $n \times m$ matrix obtained by rotating A ninety degrees clockwise on paper. Do A and B have the same singular values? If yes, provide a proof. If no, provide a counterexample.

1.2 Norms

Vector and matrix p-norms are related by various inequalities, often involving the dimensions m or n. For each of the following, verify the inequality and give an example of a non-zero vector or matrix (for general m,n) for which equality is achieved. In this problem x is an m-vector and A is an $m \times n$ matrix.

(a) $||x||_\infty \leq ||x||_2$

(b) $||x||_2 \leq \sqrt{m}||x||_\infty$

(c) $||A||_\infty \leq \sqrt{n}||A||_2$

(d) $||A||_2 \leq \sqrt{m}||A||_\infty$

2 Subgradients

For each of the following convex functions, explain how to calculate a subgradient at a given x :

(a) $f(x) = \max_{i=1\ldots m}(a_i^T x + b_i)$

(b) $f(x) = \max_{i=1\ldots m} |a_i^T x + b_i|$

(c) $f(x) = \sup_{0 \leq t \leq 1} p(t)$, where $p(t) = x_1 + x_2 t + \ldots + x_n t^{n-1}$

(d) $f(x) = x_{[1]} + \ldots + x_{[k]}$, where $x_{[i]}$ denotes the i^{th} largest element of the vector x

(e) $f(x) = \inf_{Ay \preceq b} ||x - y||^2$, i.e., the square of the distance of x to the polyhedron defined by $Ay \preceq b$. You may assume that the inequalities $Ay \preceq b$ are strictly feasible.

3 Probability

Coming Soon..

4 Machine Learning

4.1 PCA

Like the primal and dual problems for SVM, we can have two perspectives of PCA: maximizing the variance or minimizing the reconstruction error. We will find that both these views lead to the same optimization problem.

1. **Maximizing the Variance**

   Consider N data points $X_1 \dots X_N \in R^p$, such that the sample mean of X is zero: $\frac{1}{N}\sum_{i=1}^{N} X_i = 0$. Also consider projection vector $u \in R^p$, where $||u||_2 = 1$. We would like to maximize the sample variance $\tilde{V}[u^T X]$, which is the sample variance of X projected onto u. The sample variance of N samples of a variable z is $\tilde{V}[z] = \frac{1}{N}\sum_{i=1}^{N} z_i^2$, where $\frac{1}{N}\sum_{i=1}^{N} z_i = 0$. We can write an optimization problem to maximize the sample variance:

   $$\max_{u:||u||_2=1} \tilde{V}[u^T X] \tag{1}$$

   Reformulate Eqn. 1 to the following optimization problem, and define the $p \times p$ matrix Σ:

   $$\max_{u:||u||_2=1} u^T \Sigma u \tag{2}$$

2. **Minimizing the Re-construction Error**

   Consider the same variables as those defined in Problem 3.1. Instead of maximizing the projected variance, we now seek to minimize the reconstruction error. In other words, we would like to minimize the difference between X and the reconstructed $uu^T X$. Note that $uu^T X$ first projects X onto u, and then projects this scalar back into the p-dimensional space along the u axis. Minimizing the reconstruction error can be written as the following optimization problem:

   $$\min_{u:||u||_2=1} \frac{1}{N}\sum_{i=1}^{N} ||X_i - uu^T X_i||_2^2 \tag{3}$$

   Reformulate Eqn. 3 to the following optimization problem, with the same Σ as defined in Problem 3.1.

   $$\max_{u:||u||_2=1} u^T \Sigma u \tag{4}$$

4.2 Bias Variance Trade-off

Consider a regression problem with a set of training data points $\{x^{(i)}, y^{(i)}\}_{i=1}^m$, where $x^{(i)} \in \mathcal{R}^n$ and $y^{(i)} \in \mathcal{R}$. Consider optimizing the squared loss given by:

$$J(\theta) = \frac{1}{m}\sum_i (y^{(i)} - h_\theta(x_i))^2 \tag{5}$$

Let θ denote the optimal parameter obtained by minimizing the above squared loss expression. The loss $J(\theta)$ of prediction over a new data point (x, y) is given by $J(\theta) = (y - h_\theta(x))^2$. Let $h_{\bar{\theta}}(x) = E_D[h_\theta(x)]$ denote the average prediction over x. Then, bias and variance are defined as bias $= [h_{\bar{\theta}}(x) - y]$ and variance $= E_D[(h_{\bar{\theta}}(x) - h_\theta(x))^2]$ where the expectations are taken over the set of training points coming from the underlying distribution. Show that $E_D[J(\theta)]$ can be written as a function of the bias term and the variance term.

Next, we will generalize this setting to also handle noise. Let each target variable be represented as $y = f(x) + \epsilon$ where $f(x)$ is the underlying function that we are trying to learn and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is zero mean i.i.d. Gaussian noise. Let us know re-define the Bias term as $[h_{\bar{\theta}}(x) - f(x)]$. Variance term remains the same as before. In this setting, express the expected error $E_D[J(\theta(x)]$ as a function of the bias, variance and noise terms (i.e., a term dependent on σ^2). You should simplify your expression as much as possible.

4.3 Kernelizing the Perceptron

Let there be a binary classification problem with $y \in \{0, 1\}$. The perceptron uses hypotheses of the form $h_\theta(x) = g(\theta^T x)$, where $g(z) = \mathbf{1}\{z \geq 0\}$. In this problem, we will consider a stochastic gradient descent-like implementation of the perceptron algorithm where each update to the parameters θ is made using only one

training example. However, unlike stochastic gradient descent, the perceptron algorithm will only make one pass through the entire training set. The update rule for this version of the perceptron algorithm is given by

$$\theta^{(i+1)} := \theta^{(i)} + \alpha[y^{(i+1)} - h_{\theta^{(i)}}(x^{(i+1)})]x^{(i+1)}$$

where $\theta^{(i)}$ is the value of the parameters after the algorithm has seen the first i training examples. Prior to seeing any training examples, $\theta^{(0)}$ is initialized to $\vec{0}$. Let K be a Mercer kernel corresponding to some very high-dimensional feature mapping ϕ. Suppose ϕ is so high-dimensional (say, ∞-dimensional) that it's infeasible to ever represent $\phi(x)$ explicitly. Describe how you would apply the "kernel trick" to the perceptron to make it work in the high-dimensional feature space ϕ, but without ever explicitly computing $\phi(x)$.

[Note: You don't have to worry about the intercept term. If you like, think of ϕ as having the property that $\phi_0(x) = 1$ so that this is taken care of.] Your description should specify

1. How you will (implicitly) represent the high-dimensional parameter vector $\theta^{(i)}$, including how the initial value $\theta^{(0)} = \vec{0}$ is represented (note that $\theta^{(i)}$ is now a vector whose dimension is the same as the feature vectors $\phi(x)$);

2. How you will efficiently make a prediction on a new input $x^{(i+1)}$. I.e., how you will compute $h_{\theta^{(i)}}(x^{(i+1)}) = g({\theta^{(i)}}^T \phi(x^{(i+1)}))$, using your representation of $\theta^{(i)}$; and

3. How you will modify the update rule given above to perform an update to θ on a new training example $(x^{(i+1)}, y^{(i+1)})$; i.e., using the update rule corresponding to the feature mapping ϕ:

   $$\theta^{(i+1)} := \theta^{(i)} + \alpha[y^{(i+1)} - h_{\theta(i)}(x^{(i+1)})]\phi(x^{(i+1)})$$

   [Note: If you prefer, you are also welcome to do this problem using the convention of labels $y \in \{-1, 1\}$, and $g(z) = sign(z) = 1$ if $z \geq 0, -1$ otherwise.]