CS 4780 HW2 Solutions

Problem 1

Let:

$$D = ||\vec{x} - \vec{z}||^2 = \sum_{\alpha=1}^{d} (x_\alpha - z_\alpha)^2$$

where d is the dimensionality of the data, x_α and z_α are i.i.d. for all α, and $x_\alpha, z_\alpha \sim N(0,1)$ for all α.

(a) First, calculate the expectation and variance of $(x_\alpha - z_\alpha)$ for arbitrary α, given that $x_\alpha, z_\alpha \sim N(0,1)$. Is this distribution a Gaussian normal distribution?

Because the normal distribution $N(0,1)$ is symmetric about 0, $(x_\alpha - z_\alpha)$ and $(x_\alpha + z_\alpha)$ are identically distributed. Thus $(x_\alpha - z_\alpha)$ is a normal distribution with expectation 0 and variance 2.

(b) Now, using (a), calculate the expectation of D and simplify it to a function of d. Let β_α $N(0,2)$ represent the distribution $(x_\alpha - z_\alpha)$:

$$\begin{aligned}
\mu_D = \mathbb{E}[D] &= \mathbb{E}\left[\sum_{\alpha=1}^{d} (x_\alpha - z_\alpha)^2\right] && (1)\\
&= \sum_{\alpha=1}^{d} \mathbb{E}\left[\beta_\alpha^2\right] && (2)\\
&= \sum_{\alpha=1}^{d} (\mathbb{E}\left[\beta_\alpha^2\right] - \mathbb{E}\left[\beta_\alpha\right]^2) + \mathbb{E}\left[\beta_\alpha\right]^2 && (3)\\
&= \sum_{\alpha=1}^{d} \mathbb{V}\left[\beta_\alpha\right] + \mathbb{E}\left[\beta_\alpha\right]^2 && (4)\\
&= \sum_{\alpha=1}^{d} 2 && (5)\\
&= 2d && (6)
\end{aligned}$$

Where line 2 follows from the linearity of expectation, and line 4 from the definition of variance.

(c) Using (a) and (b), calculate the variance of D and simplify it to a function of d. You may use the fact that $\mathbb{E}[X^4] = 3\sigma^4$ if $X \sim N(0, \sigma^2)$:

$$\sigma_D^2 = \mathbb{V}[D] = \mathbb{V}\left[\sum_{\alpha=1}^{d}(x_\alpha - z_\alpha)^2\right] \tag{1}$$

$$= \sum_{\alpha=1}^{d} \mathbb{V}\left[\beta_\alpha^2\right] \tag{2}$$

$$= \sum_{\alpha=1}^{d} \mathbb{E}\left[(\beta_\alpha^2)^2\right] - \mathbb{E}\left[\beta_\alpha^2\right]^2 \tag{3}$$

$$= \sum_{\alpha=1}^{d} \mathbb{E}\left[\beta_\alpha^4\right] - 4 \tag{4}$$

$$= \sum_{\alpha=1}^{d} 3 * 2^2 - 4 \tag{5}$$

$$= 8d \tag{6}$$

Where line 2 follows from the linearity of variance over IID random variables, line 3 follows from the definition of variance, line 4 follows from part (a), and line 5 uses the fact given in the prompt.

(d) Using (b) and (c), show that $\frac{4\sigma_D}{\mu_D} \to 0$ as the dimensionality $d \to \infty$:

$$\frac{4\sigma_D}{\mu_D} = \frac{4\sqrt{\sigma_D^2}}{\mu_D}$$

$$= \frac{4\sqrt{8d}}{2d}$$

$$= 4\sqrt{\frac{2}{d}}$$

Which goes to 0 as d becomes large.

Problem 2

(a) Construct a data set of labeled vectors such that presenting the vectors to the perceptron in one order causes it to converge more slowly than it does for another order. Show that this is the case.

Consider the dataset consisting of the three points $(0, 1), (10, 1), (11, -1)$. Let the x-axis be a separating hyperplane, so in this dataset the point labels correspond to the y-coordinate values.

Consider a run of the perceptron algorithm which examines the points in the order presented above. The first point is automatically incorrectly classified, so the weight

vector w becomes $[0, 1]$. This vector correctly classifies the other two points, so the algorithm has converged after a single pass through the data set.

Now consider a run of the perceptron algorithm which examines the points in reverse from the order presented above. The first point is automatically incorrectly classified, so the weight vector w becomes $[-11, 1]$. The second point yields a prediction of

$$sign(-11 * 10 + 1 * 1) = sign(-109) = -1$$

which is incorrect, so the weight vector then becomes $[-1, 2]$. The third point is then correctly classified, and the weight vector does not change. Note that the point $(10, 1)$ is still incorrectly classified using this weight vector, showing that the perceptron has failed to converge after a single pass through the data set. Thus, the algorithm takes longer to converge for this particular ordering of the data.

(b) Let there exist a set of vectors $x_1...x_n$ such that the perceptron converges after no more than a single pass through the data set, regardless of order of presentation. What is the greatest possible difference between the largest and smallest convergence times, where time is measured in number of updates to the weight vector? If we assume instead that the perceptron might take more than one pass over the data set to converge, does this maximum difference increase?

If we assume that the perceptron will always converge within the first pass, then the shortest possible time it could take for any data set is 1 update, while the longest for any dataset of size n is n updates. However, it is impossible to have a single dataset with both times. The greatest difference is actually **n-2**.

To show this, imagine that we have a data set of points $x_1...x_n$ and labels $y_1...y_n$. Let it be the case that the perceptron converges after examining each vector once, for any order. Let it also be the case that after examining x_1, the perceptron converges immediately - it is capable of correctly classifying the entire rest of the data set, using hidden weight vector y_1x_1. More formally, $sign(y_1x_1 \cdot x_i) = y_i$ for all i, which implies that $y_1x_1 \cdot y_ix_i > 0$. This in turn implies that for any subset S of $2...n$,

$$y_1x_1 \cdot \sum_{i \in S} y_ix_i = \sum_{i \in S} y_1x_1 \cdot y_ix_i > 0$$

This means that after the perceptron examines ANY arbitrary sequence of vectors in $x_2...x_n$, it will then always correctly classify x_1. Thus the greatest number of updates performed is $n - 1$, since x_1 is guaranteed to be correctly classified on any training run that does not start with x_1. This upper bound is tight, because a set of mutually orthogonal vectors $x_2...x_n$ will take exactly $n - 1$ updates to generate a separating hyperplane (as the prediction for each vector is exactly 0).

Thus, if the perceptron is capable of converging after one update, and also capable of converging after viewing each data point once in any order, then the longest it can possibly take is n-1 updates. If, on the other hand, the perceptron is capable of taking

n updates to converge, then the contrapositive states that the shortest it can possibly take is strictly greater than 1 update. Either way, the greatest difference in convergence time is **n-2**.

If the perceptron is capable of taking longer than n vectors to converge, then the above bound no longer holds. The data set given for part (a) is an example.

(c) The convergence proof given in class guarantees that the perceptron will find a separating hyperplane as long as one exists, but there could exist a wide range of such hyperplanes. Can some separating hyperplanes be considered "better" than others? What would be one way to measure the "goodness" of a hyperplane (only using the training set)?

The size of the smallest margin is a good indicator of how good your hyperplane is. The larger it is, the better. Hyperplanes with a small smallest margin are likely to misclassify additional data points from the class that the smallest margin belongs to. This is especially true for those points that are close to the true hyperplane used to generate the data.

(d) Describe how you could utilize the effects of data ordering to improve upon the perceptron algorithm as presented in class - i.e. to find a better hyperplane, as defined by your criteria defined in question c).

We've demonstrated in parts (a) and (b) that changing the data ordering will also change the update schedule to the weight vector. Thus, we can expect different data orderings to generate different weight vectors, with different smallest margins. Two simple ways to leverage this fact is to randomize the data, in order to prevent pathological cases from yielding artificially small margins, and to train multiple perceptrons on different orderings, picking the one with the best smallest margin.

Problem 3

(a) Estimate λ for the Exponential distribution with PDF $f(x;\lambda) = \lambda e^{-\lambda x}$ where x is non-negative.

Let L be the log-likelihood function.

$$\begin{aligned} L(\lambda) &= log(\prod_{i=1}^{n} \lambda e^{-\lambda x_i}) \\ &= \sum_{i=1}^{n} log(\lambda) - \lambda x_i \\ &= nlog(\lambda) - \lambda \sum_{i=1}^{n} x_i \\ \frac{dL}{d\lambda} = 0 &= \frac{n}{\lambda} - \sum_{i=1}^{n} x_i \\ \lambda &= \frac{n}{\sum_{i=1}^{n} x_i} \end{aligned}$$

(b) Estimate p for the Geometric distribution with PMF $f(x;p) = p(1-p)^x$ where x is non-negative.

Let L be the log-likelihood function.

$$\begin{aligned} L(p) &= log(\prod_{i=1}^{n} p(1-p)_i^x) \\ &= \sum_{i=1}^{n} log(p) + x_i log(1-p) \\ &= nlog(p) + log(1-p) \sum_{i=1}^{n} x_i \\ \frac{dL}{dp} = 0 &= \frac{n}{p} - \frac{\sum_{i=1}^{n} x_i}{1-p} \\ &= (1-p)n - p \sum_{i=1}^{n} x_i \\ &= n - p(n + \sum_{i=1}^{n} x_i) \\ p &= \frac{n}{n + \sum_{i=1}^{n} x_i} \\ &= \frac{n}{\sum_{i=1}^{n} x_i + 1} \end{aligned}$$

The two MLE estimators look very similar, which makes sense as the Exponential distribution is the continuous analog of the Geometric distribution (if you imagine splitting each time step of the Geometric distribution into k parts, your PMF becomes $p(1-p/k)^{kx}$ which approaches pe^{-px} as k becomes large). Both parameters correspond to rate of occurrence, so it makes sense to calculate them both the same way: as the reciprocal of how long it takes, on average, for an event to occur.

The +1 offset in the Geometric distribution estimator reflects the fact that there is a non-trivial, and actually quite large, chance of drawing a 0 from the Geometric distribution as defined here, but little to no chance of drawing a small value from the Exponential distribution. We could define the Geometric distribution on $x \geq 1$ with PMF $p(1-p)^{x-1}$ and the estimators would then be exactly the same.

(c) Estimate μ and σ^2 for the Gaussian distribution with PDF $f(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$.

Let L be the log-likelihood function.

$$\begin{aligned} L(\mu, \sigma) &= log(\prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}) \\ &= \sum_{i=1}^{n} -log(\sigma\sqrt{2\pi}) - \frac{(x_i-\mu)^2}{2\sigma^2} \\ &= -nlog(\sigma\sqrt{2\pi}) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2 \end{aligned}$$

$$\begin{aligned} \frac{dL}{d\mu} = 0 &= \sigma^{-2}\sum_{i=1}^{n}(x_i-\mu) \\ &= (\sum_{i=1}^{n} x_i) - n\mu \\ \mu &= \frac{\sum_{i=1}^{n} x_i}{n} \end{aligned}$$

$$\begin{aligned} \frac{dL}{d\sigma} = 0 &= -\frac{n}{\sigma} + \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{\sigma^3} \\ &= n - \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{\sigma^2} \\ \sigma^2 &= \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n} \end{aligned}$$