- 1. Devise the update rule for minimizing $f(x) = (\max(0, x) - \frac{1}{2})^2$ using gradient descent. Roughly, when will gradient descent succeed or fail (based on where you start and step size)?
- 2. Given a data set A, under what circumstances will its projection onto its principal components equal its projection onto its
    - right singular vectors
    - left singular vectors
- 3. Let S be a set of documents and let T be a set of terms. Suppose that C is a binary term-document incidence matrix (so entry (i,j) is a 1 if term i appears in document j and 0 otherwise). What do the entries of $C^T C$ represent?
- 4. How did we derive the equations for simple linear regression from class?
- 5. Why can you assume without loss of generality that a mistake-bounded learner only updates its state when it makes a mistake?
- 6. You are given a data set with m points and an algorithm that satisfies the weak-learning condition (it always outputs a classifier with accuracy 60%). Each classifier output by the weak-learning algorithm can be encoded using two bits. How can you construct a classifier that can be described by less than m bits and is correct on every data point in the data set (you may assume m is very large). What is the size of your final classifier?
- 7. How can you compare Markov's inequality, Chebyshev's inequality, and the Chernoff bound?

1)

Compute

$$f^{'}(x) = \begin{cases} 0 & x \leq 0 \\ 2x - 1 & x > 0 \end{cases}$$

Update rule is: $x_{i+1} = x_i - \nu(2x_i - 1) = x_i - 2\nu x_i + \nu$. If $x_i = \frac{1}{2}$ then we've found the minimum. Otherwise, we should be moving in that direction. Failure occurs for $x_i \leq 0$ because x_i gets 0 gradient. So x_0 must be initialized > 0. Also, if the step size is too large it's possible that we could jump to $x_{i+1} <= 0$ territory. For example if $x_i = 0.6$ then per the update rule $x_{i+1} = 0.6 - 1.2\nu + \nu$. If $\nu \geq 3$ then we would over shoot. In reality, I've never seen a step size > 0.01, so this is somewhat contrived. ν should be "reasonably small"

2) Answer:

PCA works really on a data set of points, but first requires organizing it into a matrix A. The question emphasizes that there are two ways of doing this:

The points are rows. In this case the principal components are the right singular vectors, the eigenvectors of $A^\top A$.

The points are columns. In this case the principal components are the left singular vectors, the eigenvectors of $AA^\top$. This is actually analogous to (1) except effectively with $A^\top$ and A switched.

In either case, a point equaling its projection onto the first k principal components means that the points in your data set are really k-dimensional – they are all linear combinations of just the k principal components. So that's often something useful to know about a data set.

Brief review of SVD and PCA:

PCA computes the eigen-decomposition of a covariance matrix A^TA, which is by definition symmetric and therefore (spectral theorem) can be written as the product of $Q\Sigma Q^T$ where Q is a orthogonal matrix (square matrix with orthonormal rows and vectors) and Σ is diagonal. The entries of Σ are the eigenvalues and the columns/rows of Q/Q^T are the eigenvectors of the covariance matrix. The covariance matrix is square (n x n), as is Q and Σ.

SVD on the other hand decomposes any matrix (does not have to be square, symmetric, etc.) into USV^T, where U and V^T are square, orthogonal matrices; however, in contrast to the eigen-decomposition above, if working with A, U will be m x m, V^T will be n x n, and S will be m x n.

To start the problem assume A is m x n, where m is the number of data samples and n is the feature dimension. That is, each row is a sample represented by n values.

Compute the SVD of $A = USV^T$ and then perform PCA over A by computing the eigen-decomposition of the covariance matrix:

$A^TA = (USV^T)^T(USV^T) = (VSU^T)(USV^T) = VS^2V^T$, which implies the columns/rows of V/V^T are the eigenvectors of A as well as the right singular vectors of A.

Now consider the case when A is structured as rows of features and columns of samples. That is, A is n by m. Then the covariance matrix of $A = AA^T$ and the SVD of $A = (USV^T)(VSU^T) = (US^2U^T)$. Therefore, in this case, the eigenvectors of the covariance matrix are the left singular vectors of the data matrix.

In conclusion, the condition under which the right singular vectors of a data matrix A are also the principal directions is A being structured as rows of samples. On the other hand, when A is columns of samples then the left singular vectors of A and the principal directions.

A couple of other useful notes previous students discovered about PCA and SVD:

The eigen-decomposition of a symmetric expresses how a matrix scales (but does not rotate) a specific set of vectors. First you apply an invertible, linear transformation by multiplying by Q^T, then scales via Σ, finally inverting the original transformation.

SVD is a rotation, followed by scaling, followed by rotation

3) Answer:

The dimension of C^TC is num docs by num docs, so the element at (i,j)(i,j) is the number of shared terms between doc-i and doc-j. Also for the diagonal elements like (i,i)(i,i) it means the total number of terms in a document.

4) Answer:

Basically we want to find w to minimize $|X \cdot w - y|_2^2$, taking the derivative w.r.t. w we will have $w = (X^\top X)^{-1} X^\top y$. (check lecture in gradient descent).

Appendix: Derivation of the vector derivative:

Matrix or vector derivative can be understood by examining the derivative of each individual component. In the derivation below, I will use the numerator layout notation, that is if $y \in R^m$ and $x \in R^n$, then $\frac{\partial y}{\partial x}$ is a m by n matrix.

In the case, let $f(v) = |v(w)|_2^2$ where $v(w) = X \cdot w - y$, therefore,

$$\frac{\partial(|X \cdot w - y|_2^2)}{\partial w} = \frac{\partial f(v)}{\partial v} \frac{\partial v}{\partial w}$$

.

Note that $f(v) = v_1^2 + v_2^2 + \ldots + v_m^2$ and $\frac{\partial f}{\partial v1} = 2v_1$. Therefore, $\frac{\partial f(v)}{\partial v} = 2v^T$ (the transpose T comes from the numerator layout notation defined above).

Next we calculate $\frac{\partial v}{\partial w}$. We first write X as $[x_1, x_2, \ldots, x_n]$ where x_i is the column vector of X. Therefore

$$v = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n - y$$

and

$$\frac{\partial v}{\partial w_i} = x_i$$

We then have

$$\frac{\partial v}{\partial w} = [x_1, x_2, \ldots, x_n] = X$$

.

Putting everything together using the chain rule:

$$\frac{\partial(|X \cdot w - y|_2^2)}{\partial w} = \frac{\partial f(v)}{\partial v} \frac{\partial v}{\partial w} = 2v^T X = 2(X \cdot w - y)^T X$$

.

Let the above derivative be 0, we have

$$(X \cdot w - y)^T X = [0, 0, \ldots, 0]$$

which is equivalent to

$$X^T (X \cdot w - y) = [0, 0, \ldots, 0]^T$$

. This further gives

$$X^T X \cdot w = X^T y$$

. Therefore we have

$$w = (X^T X)^{-1} X^T y$$

5) Answer

Suppose L' is a mistake-bounded learner that would update its state at correct samples, we can always construct another learner L as follows:

- Run L' for each sample
- If sample prediction is correct, undo the state change performed by L'.

It can be shown that L is also a mistake-bounded learner.

The above reasoning shows that for every L', we can always construct a L that only updates its state when making a mistake. Therefore, without losing generality, we can always assume a mistake-bounded learner only updates its state when making a mistake.

[Original explanation]

you can always change the ordering so that it makes mistake at the very beginning of the T (the mistake bound) examples and stop making mistake. In this way, the learner who may change at correct label does not improve the mistake bound T.

Detail: Suppose the mistake bound is T for a learner L that only make change at mistake, we can safely reorder the list of training examples so that the T examples come in the first place. In this situation, the learner L will perform the same thing and make mistakes at the first T training examples. (since before reordering, L does not update in correct training examples). Consider any other learner L' that may change in correct label, in the re-ordering case, it will behave exactly like L so the mistake bound is still at least T, which make no improvement.

6) Answer

The idea is to use boosting as follows. In this case we want a classifier with training error 0. But because the only possible values for training error are $1, (m-1)/m, ..., 1/m, 0$ (since it is 0-1 loss and there are only m points), this is equivalent to achieving training error $< 1/m$. Now recall that the training error of the AdaBoost hypothesis after T rounds is at most $\exp(-2\gamma^2 T)$. Here $\gamma = 0.1$ since we have a 0.6-accurate weak learner. Thus T only needs to be $\Theta(\log m)$ for the training error to be less than $1/m$, and hence to be 0. After this many rounds, the final classifier classifies the entire training set correctly. And recall again that this final classifier is just the majority of T different classifiers, which each take two bits to describe. Thus we just need $\Theta(\log m)$ bits to describe this classifier.

7) Answer

Roughly speaking, Markov is the weakest bound here, Chebyshev's a little stronger, and the Chernoff bound is the strongest. This question is a little broad, but it's important to familiarize yourself with exactly when each inequality is applicable. For instance, sometimes Chernoff might not be applicable and Chebyshev might be the best you can do. In particular:

Markov applies to nonnegative random variables

Chebyshev applies to random variables for which you have a variance bound

Chernoff applies to sums of i.i.d. random variables (in fact generally indicator variables)

At a high level, all three bounds describe the probability that a random variable falls far from its expectation. They differ in the assumptions made about the underlying distribution.

Markov

A non-negative random variable cannot be much larger than its mean very often. This bound depends only on the expected value of the random variable and that X is non-negative. No other assumptions made.

$$\forall X \geq 0,\ \mathbb{P}(X \geq \lambda) \leq \frac{E(X)}{\lambda}$$

Proof: if you define random variable Z to be 1 if $X \geq \lambda$ and 0 otherwise then it's easy to see:

$$\lambda Z \leq X \implies E(\lambda Z) \leq E(X)$$

$$E(\lambda Z) = \lambda E(Z) = \lambda(1\mathbb{P}(X \geq \lambda) + 0\mathbb{P}(X < \lambda)) = \lambda\mathbb{P}(X \geq \lambda) \leq E(X) \implies \mathbb{P}(X \geq \lambda) \leq \frac{E(X}{\lambda}$$

Chebyshev

The probability a random variable is more than k standard deviations from the mean is no more than $\frac{1}{k^2}$. This bound assumes X has non-zero, finite variance.

$$\mathbb{P}(|X - E(X)| \geq k\sigma) < \frac{1}{k^2}$$

Proof: Let X be a random variable over real numbers and $Y = (X - E(X))^2$. Note, Y is now a non-negative random variable. By Markov's rule we know:

$$\mathbb{P}(Y \geq \lambda) \leq \frac{E(Y)}{\lambda} \implies \mathbb{P}((X - E(X))^2 \geq \lambda) \leq \frac{E(X - E(X))^2)}{\lambda} = \frac{\sigma^2}{\lambda}$$

Set $\lambda = \sigma^2 k^2$ and you have:

$$\mathbb{P}((X - E(X))^2 \geq \sigma^2 k^2) = \mathbb{P}(|X - E(X)| \geq k\sigma) \leq \frac{\sigma^2}{\sigma^2 k^2} = \frac{1}{k^2}$$

Chernoff Bounds

I found it hard to state this succinctly, but similar to the above bounds the probability of a random variable being far from the mean.

Follows from the application of the Markov bound to random variable $Y = e^{tX}$.

Chernoff Bounds

I found it hard to state this succinctly, but similar to the above bounds the probability of a random variable being far from the mean.

Follows from the application of the Markov bound to random variable $Y = e^{tX}$. Specifically, it says:

$$\mathbb{P}(X \geq \lambda) = \mathbb{P}(Y \geq e^{t\lambda}) = \mathbb{P}(e^{tX} \geq e^{t\lambda}) \leq \frac{E(Y)}{e^{t\lambda}} = \frac{E(e^{tX})}{e^{t\lambda}}$$

This works because $Y \geq 0$.

There is a special case I've seen mentioned which is when X is the sum of m independent Bernouli RV's, $X_1, .., X_m$ where $\forall i$, $\mathbb{P}(X_i = 1) = p_i$. Defining $p = mp_0$ and $h(\delta) = (1+\delta)log(1+\delta) - \delta$ then

$$\mathbb{P}(X > (1+\delta)p) \leq e^{-h(\delta)p}$$

More on this in textbook appendix B.3 including the case for $X < (1+\delta)p$.