CS 391L: Machine Learning **Spring 2024**

Homework 3 - Theory - Solutions

Lecture: Prof. Adam Klivans

Keywords: SVD, PCA

1. Form the $d \times d$ matrix A^TA and compute its eigendecomposition. Consider an eigenvector v of A^TA with corresponding eigenvalue λ. By definition, $A^TAv = \lambda v$. We claim that this means Av is an eigenvector of AA^T with eigenvalue λ. Indeed,

$$(AA^T)Av = A(A^TAv) = A(\lambda v) = \lambda(Av).$$

Thus, for each eigenvector v of A^TA, Av is an eigenvector of AA^T with the same corresponding eigenvalue. And the vectors of the nullspace of AA^T constitute the (trivial) eigenvectors with eigenvalue 0.

2. *Note to peer graders:* For the true/false questions, just the answer is enough. But here we provide a short explanation just for clarity.

   (a) True. Linear regression is all about modeling labels as a linear function of the input.

   (b) True. PCA does not make any use of labels, and instead looks for the best k-dimensional subspace capturing the data.

   (c) False. SVD itself is an operation defined semantically on a matrix, whereas PCA is one that defined semantically on a set of data (where each data point is a vector). For instance, it makes perfect sense to run SVD on a single image (as in the programming assignment), but not to run PCA on just one image. The connection, however, is that PCA can be implemented by arranging your data in a certain way and then using SVD on that matrix. This can roughly be thought of as an "implementation detail" of PCA.

   (d) The cleanest way to do this is to run PCA on your dataset, take projections of your data along the principal components, and then run regression on the projected data. (This is technically known as principal component regression.) When there are linear dependencies within the data, the dataset will be low rank and so by taking the appropriate number of principal components, one can actually capture the entirety of the dataset with zero error. This method also happens to be inherently resistant to noise / imperfect linear dependencies.