Introduction to Machine Learning **Fall Semester, 2014/5**

Recitation 11: January 11

Lecturer: Regev Schweiger *Scribe: Regev Schweiger*

11.1 PCA Example - Digits

Downloaded data from `http://www.cse.buffalo.edu/~jcorso/t/555pdf/homework2-data.tar.gz`. Looking at images of the digit 0 only. The following code computes the first 500 PCs, looks at their respective variances, and plots the first PC.

```
X = zeros(28*28, 5000);
for i = 1:5000; a = imread(sprintf('%05d.pgm', i-1)); X(:,i) = a(:); end

[V,D] = eigs(cov(X'), 500);
plot(diag(D));
imshow(reshape(V(:,1), 28, 28), [min(V(:,1)), max(V(:,1))]);
```

11.2 Probabilistic View of PCA

Consider the following model. Let $\mathbf{z}_1, \ldots, \mathbf{z}_n$ be points in $\mathbb{R}^d$, and assume W is a full rank matrix over $\mathbb{R}^{m\times d}$. Assume that we do not observe the points $\mathbf{z}_1, \ldots, \mathbf{z}_n$, and we do not know what is W (but we know d). Instead, assume we observe the points $\mathbf{x}_i = W\mathbf{z}_i + \epsilon_i$, where $\epsilon_i \sim N(0, \sigma^2 I_m)$, i.e., it is a multivariate normal noise. Thus, we obtain points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in m dimensions, however these points are truly points in d dimensions with additional normal noise. We would like to treat the problem as a likelihood inference problem. Let us write down the log likelihood of the above formulation:

$$\ell(Z, W; X) = -mn\log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\|\mathbf{x}_i - W\mathbf{z}_i\|^2.$$

Thus, maximizing the log likelihood is equivalent to solving the following:

$$(\hat{Z}, \hat{W}) = \arg\min_{Z,W}\sum_{i=1}^{n}\|\mathbf{x}_i - W\mathbf{z}_i\|^2.$$

For any fixed W we get a simple linear regression problem, since $\mathbf{x}_i$ is fixed. The solution to the linear regression is given by the Normal equations:

$$\hat{\mathbf{z}}_i = (W^tW)^{-1}W^t\mathbf{x}_i.$$

Note that since W is of full rank, we know that W^tW is nonsingular. Note that this assumption is not limiting since if W is not full rank we can change the problem by assuming that the points $\mathbf{z}_i$ arise from a smaller dimension. Plugging the regression solution into the likelihood we get that we need to minimize the following:

$$\hat{W} = \arg\min_W \sum_{i=1}^{n} \|\mathbf{x}_i - W(W^tW)^{-1}W^t\mathbf{x}_i\|^2.$$

Let $W = USV^t$ be the singular value decomposition of W. It is easy to see that $W(W^tW)^{-1}W^t = US(S^tS)^{-1}S^tU^t = U_dU_d^t$, where U_d is the $m \times d$ matrix consisting of the first d columns of U. Thus, we get that maximizing the likelihood is equivalent to maximizing the following:

$$\hat{U_d} = \arg\min_{U_d} \sum_{i=1}^{n} \|x_i - U_dU_d^tx_i\|^2.$$

Therefore, we obtain the exact formulation of PCA. The probabilistic formulation allows for a few natural extensions to PCA. First, we can deal with PCA with missing data using the Expectation Maximization algorithm. Second, we can now easily model a mixture of PCAs, where the assumption is that every point was sampled from a mixture of lower dimension hyperplanes.

It is worth noting that we $\mathbf{z}_1, \ldots, \mathbf{z}_n$ as parameters, i.e., the assumption is that these are fixed points. It is possible to add the assumption about the distribution of these points as well. Specifically, the case in which $\mathbf{z}_i \sim N(0, \tau^2)$ has been studied in the literature. This variation of PCA is referred to as probabilistic PCA. The optimization is similar to the original PCA in that it is sufficient to find the eigenvalues and eigenvectors of XX^t in order to compute the maximum likelihood estimate for W, however the maximum likelihood estimate of W is not the PCA solution.

It is interesting to note that the probabilistic interpretation of PCA is analogous to the probabilistic interpretation of regression. Consider the case of linear regression of one variable, i.e., under the model $x_2 = ax_1 + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$. Instead of using linear regression, we can use PCA by considering a different model, i.e., $\binom{x_1}{x_2} = \binom{a_1}{a_2} z + \delta$, where a_1, a_2, z are unobserved parameters, and $\delta \sim N(0, \tau^2 I_2)$. As we showed above, the solution to this problem is the PCA, reducing the two dimensions (x_1, x_2) into one dimension. The difference between the two resulting optimizations is shown in Figures 11.1 and 11.2.

11.3 Kernel PCA

11.3.1 The case $m \gg n$

Consider the case that we have much more features than examples $m \gg n$. Performing PCA requires calculating the eigenvectors of XX^T, which is an $m \times m$ matrix. This can

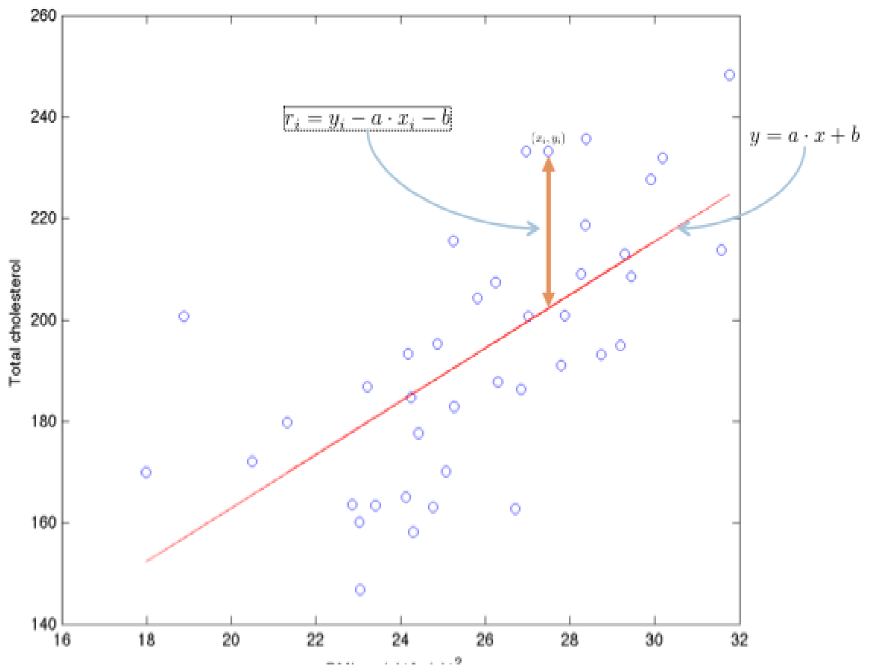

Figure 11.1: Linear regression: minimize the square of the residuals

be computationally very expensive in both time and memory. We show that we can instead calculate the eigenvectors and eigenvalues of X^TX and transform to those of XX^T them more efficiently.

Let $X = U\Sigma V^T$ be the SVD of X, and mark by U_d, Σ_d, V_d the submatrices that correspond to the d largest singular values of X.

Claim 11.1

$$U_d = XV_d\Sigma_d^{-1} \tag{11.1}$$

Proof: For $i = 1, \ldots, n$, we have $X\mathbf{v}_i = \sigma_i\mathbf{u}_i$. Divide by σ_i and concatenate. □

This allows us to do PCA by first computing V, Σ^2 - the eigenvectors and eigenvalues of X^TX (a matrix $n \times n$), an operation of complexity $O(n^3)$, and recover U using (11.1). The alternative, finding U first, is significantly less efficient in this case.

11.3.2 The kernel trick

We can now discuss the *Kernel PCA*. Consider a mapping $\mathbf{x} \to \phi(\mathbf{x})$ (where $\mathbf{x}$ is $m \times 1$ and $\phi(\mathbf{x})$ is $M \times 1$) induced by a proper kernel $K(\cdot, \cdot)$. Recal the definition of the kernel data matrix $\overline{\overline{K}}$ for which $\overline{\overline{K}}_{i,j} = (\phi(\mathbf{x}_i))^t\phi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$.

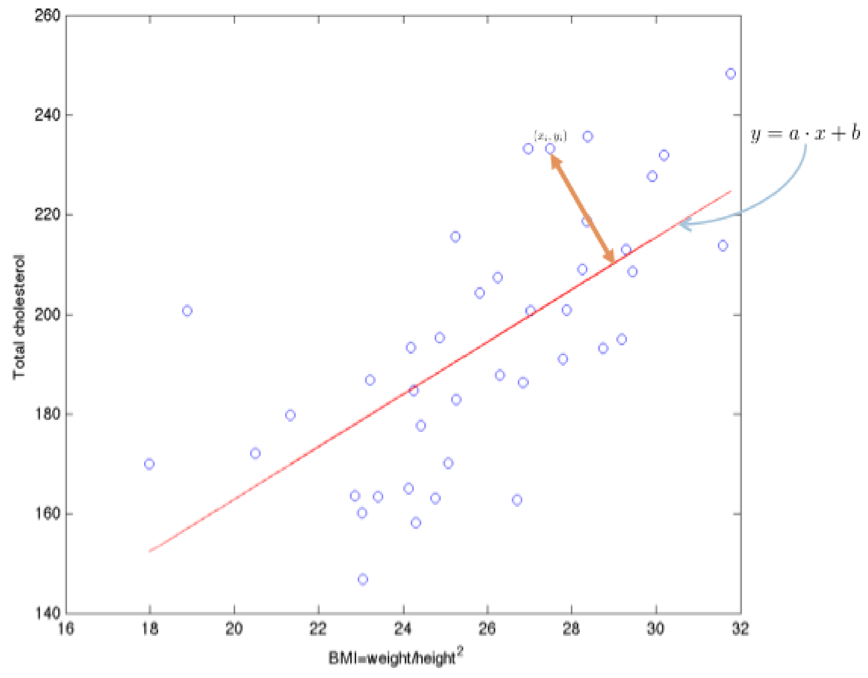

Figure 11.2: PCA: minimize the square of the distances from the line

If we try to do PCA in a straightforward way, we need to compute the matrix U whose dimension depend on M which might be huge or infinite. The main observation is that we do no need to compute U, we are rather interested only in Y - the mapping to the d dimensional subspace. Using (11.1) above, we have that

$$Y = U_d^t X = \Sigma_d^{-1} V_d^t X^t X = \Sigma_d^{-1} V_d^t \overline{\overline{K}}$$

Therefore, in order to compute Y we do not have to compute U, but rather we can compute Y using $\overline{\overline{K}}$ and V_d. Furthermore, the mapping of an arbitrary $\mathbf{x}$ (in the original feature space, before the mapping induced by the kernel K) to the d dimensional space spanned by the columns of U_d can be similarly computed:

$$y = U_d^t \phi(\mathbf{x}) = \Sigma_d^{-1} V_d^t X^t \phi(\mathbf{x}) = \Sigma_d^{-1} V_d^t K(X, \mathbf{x})$$

Where $K(X, \mathbf{x})$ is a column vector with $K(\mathbf{x}_i, \mathbf{x})$ at the i^{th} position.