CSE 475: Statistical Methods in AI **Monsoon 2019**

SMAI-M-2019 8: Linear Models(II): PCA

Lecturer: C. V. Jawahar *Date: Aug 29, 2019*

8.43 A Related Problem

In the last lecture we looked at the line fitting when we were predicting y_i from $\mathbf{x_i}$. We minimized an error in y_i given $\mathbf{x}_i$. Something like:

$$\sum_{i=1}^{N}(y_i - f(\mathbf{w}, \mathbf{x}_i))^2$$

If you visualize this in a 2D plane (when $\mathbf{x}$ was 1D) the error is parallel to the y axis. Or axis parallel. In general, the error is always defined with respect to y.

Consider a set of points in 2D $i\{(x^1, x^2)_i\}$. A very related problem is how do we fit a line that minimizes the orthogonal distance from the samples to the line? i.e.,

$$\min_{\mathbf{w}} \sum_{i=1}^{N} d_{\perp}^2(\mathbf{x}_i, \mathbf{w})$$

Note that $\mathbf{x_i} = [x^1, x^2]^T$ is a point and $\mathbf{w}$ is a line in 2D.

Another problem that we are interested in is to find a "representative" for the set of samples that we have. Who should represent the set $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}$?.

How do we define the problem? The problem is to find an entity (such as a point, line, plane etc.) that can represent the set, with minimal loss in "information/content".

8.44 Point that minimizes the distance

Let $\mathbf{z}$ be a point that minimizes the sum of distance to all the given N points. How do we find $\mathbf{z}$?. Let us state the problem as:

$$\min_{\mathbf{z}} \sum_{i=1}^{N} [\mathbf{z} - \mathbf{x_i}]^T [\mathbf{z} - \mathbf{x}_i]$$

Expanding

$$\min_{\mathbf{z}} \sum_{i=1}^{N} \mathbf{z^T z} + \mathbf{x_i^T x_i} - \mathbf{2 z^T x_i}$$

Differentiating with respect to $\mathbf{z}$ and equating to zero:

$$2\mathbf{z} = \sum_{i=1}^{N} 2\mathbf{x_i}$$

or $\mathbf{z}$ is nothing but our familiar mean μ.i.e.,

$$\mathbf{z} = \mu = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x_i}$$

.

This is quite intuitive. Here we only argued that mean minimizes the sum of square distances. Also mean is a good representative of the samples. If we want to represent a set of samples with a single (constant) representation, then it has to be mean.

If we want to represent all the samples with a single dimension (like a new feature, or geometrically like a line), then what it should be? Note that this is also equivalent to finding a new feature that can be used to "approximate" all the d features we have.

We want to look at this new feature as projections on to a line $\mathbf{w}$. as $z_i = \mathbf{w}^T\mathbf{x_i}$. Geometrically, if the structure of the data needs to be preserved (or loss of "information" is small), then we want to minimize the orthogonal distance to the line.

8.45 Minimizing Orthogonal Distance

We are interested in finding (i) a fixed point and (ii) a direction that can define our line. Let $\mathbf{w}$ define the direction. We know the fixed point as mean.

Let us first assume that mean is subtracted from all the samples. Now $\mathbf{x_i}^T\mathbf{x_i}$ is nothing but the square of the distance from the origin, and $\mathbf{w}^T\mathbf{x_i}$ is nothing but the projection of the $\mathbf{x}_i$ on $\mathbf{w}$.

Simple geometry (figure missing) tells us that the orthogonal distance we want to minimze is nothing but

$$\mathbf{x_i}^T\mathbf{x_i} - (\mathbf{w}^T\mathbf{x_i})^2$$

(ref: good old Pythagoras theorem)

Our problem now is

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \mathbf{x_i}^T\mathbf{x_i} - (\mathbf{w}^T\mathbf{x_i})^2$$

First term is independent of $\mathbf{z}$ so we could convert this into a maximization problem as:

$$\max_{\mathbf{w}} \sum_{i=1}^{N} (\mathbf{w}^T \mathbf{x_i})^2$$

or more compactly

$$\max_{\mathbf{w}} \mathbf{w}^T (\mathbf{X}^T \mathbf{X}) \mathbf{w}$$

Here $\mathbf{X}$ is a matrix.

- Q: What is the dimension of $\mathbf{X}$?
- Q: What is the dimension of $\mathbf{X}^T\mathbf{X}$?
- Q: Please refer to the notations that we used for the MSE in the last lecture.

There is an issue here. Unconstrained optimization of this can lead to $\mathbf{w}$ increasing (to infinity). How do we prevent? We need to add a constraint like $\mathbf{w}^T\mathbf{w} = 1$. This means that $\mathbf{w}$ is only a direction and what we want is a unit norm vector that maximizes our objective.

The popular method for introducing the constraint into the optimization problem is with lagrangians. (read more somewhere else.) Popular notation is λ. This modifies the problem as

$$\max_{\mathbf{w}} \mathbf{w}^T (\mathbf{X}^T \mathbf{X}) \mathbf{w} - \lambda(\mathbf{w}^T\mathbf{w} - 1)$$

Now we use our simple trick of differentiating with respect to $\mathbf{w}$ and equating to zero.

$$(X^T X)\mathbf{w} = \lambda \mathbf{w}$$

Or $\mathbf{w}$ is the eigen vector of the $C = \mathbf{X}^T\mathbf{X}$. (Note that C is nothing but our Σ; we just used a different notation so that we will not get confused with the summation Σ) Since the mean was substracted from all the samples, it is easy to see that C is the covariance matrix.

Therefore the line that minimizes the orthogonal distance passes through the origin and has a direction of the first eigen vector of the covariance matrix.

8.45.1 Why first?

It was obvious from the equation that $\mathbf{w}$ will have to be the eigen vector of C. Why did we pick the first eigen vector? Note that C has many more eigen vectors.

- Q: How many non-zero eigen values C can have?
- Q: Let $\mathbf{x}_i \in R^{100}$ and $N = 30$, how many non-zero eigen values $C = \sum_{i=1}^{N} \mathbf{x_i}\mathbf{x_i}^T$ can have?. If $N = 200$, how many it can have?

What is special for the first (largest) or principal one? (in general, it is assumed that eigen values are sorted in non-increasing order.)

We know that $(X^T X)\mathbf{w} = \lambda\mathbf{w}$ and what we want to maximize is $\mathbf{w}^T(\mathbf{X}^T\mathbf{X})\mathbf{w}$. Substituting, we realize that what we want to maximize is $\lambda\mathbf{w}^T\mathbf{w}$. If $\mathbf{w}$ is a unit vector, it is clear that what we want is the eigen vector corresponding to the largest eigen value.

- Q: Can the eigen value be negative or imaginary for C? What are the properties of C?
- Q: Is C is Positive Semi Definite (PSD) matrix? See the definitions and start.

Note: You may also see the notation of Σ for the covariance matrix, at many places.

8.46 Discussions

We now know how to optimize MSE (mean squared error). We also know how to minimize the orthogonal distance and fit a line/plane. In both cases, we formed an appropriate objective function. We minized the objective and found an expression to compute to the optimal solutions in one step.

- This is not always possible. Not all problems are this simple.
- We may not have all the data when we start. We continue to get data online in a streaming manner.
- we do not want to work with "huge" matrices in the casses of large data sets.

This points to the need of incremental techniques to address this class of problem. We will see some of these issues in the next lectures.

8.47 Linear Dimensionality Reduction

Let us come to PCA from a different angle now.

In general, we start with a feature representation $\mathbf{x}$ corresponding to a physical phenomena or an object of interest. (In the case of supervised learning, we also have a corresponding y. i.e., $((\mathbf{x}_i, y_i))$ In practice, this $\mathbf{x}$ is not the result of very careful selection of measurements/features, These features are either what we could think of as relevant or what is feasible in practice. There could be redundancy and correlation within these features too. They may not be the best for our problem

also. A problem of interest to us is to find a new feature representation $\mathbf{z}$, often lower in dimension, as

$$\mathbf{z} = \mathbf{U}\mathbf{x}. \tag{8.11}$$

If $\mathbf{x}$ is a d-dimensional vector and $\mathbf{z}$ is a k-dimensional vector (where $k < d$), $\mathbf{U}$ is a $k \times d$ matrix.

Two variants:

- The above linear transformation (matrix multiplication) leads to a set of "new" features that are "derived" out existing features. New features are linear combination of existing features. We will have a closer look at this today. This can be supervised or unsupervised. i.e., y may be available (supervised) or unavailable (unsupervised).
- Second direction to create a new lower dimensional representation by selecting only the useful/important features from the original set. This is a classical subset selection problem, which is hard to solve. (Similar to your familiar knapsack problem.) This is not in our scope. Such problems are attempted with greedy or backtracking algorithms.

Some people distinguish these two carefully as feature extraction and feature selection. We can also look at them as dimensionality reduction. Dimensionality reduction makes the downstream computations efficient. Some time storage/memory is also made efficient through the dimensionality reduction. If your original $\mathbf{x}$ is "raw-data" (say an image as such represented as a vector), then such techniques are treated as principled ways to define and extract features.

When we do the dimensionality reduction, we may be removing useful information (or noise). Due to this, there is a minor chance that we may reduce the accuracy (read performance) of the down stream task (say classification). In general, dimensionality reduction schemes aim at no major reduction in accuracy (happy with some increase in accuracy), but in a lower dimension.

If "noise" (irrelevant information) is removed or suppressed in the entire process, we may expect some increase in the accuracy. At the same time, if we loose some " relevant information", then we may loose the accuracy. But, alas, we do not know what is noise and what is signal. We would like the machine to figure out this from the examples/data. If both these problems are solved independendly (as is the case in the classical ML schemes), there is noway, we can tell the dimensionality reduction scheme what is the best to do.

The above equation is for linear dimensionality reduction. We also have many equivalent non-linear dimensionality reduction schemes. They are mostly not in our scope.

8.48 PCA

Principal Component Analysis (PCA) is one of the most popular technique for dimensionality reduction. It is unsupervised. It can be seen from two different view points:

- A dimensionality reduction that retains maximum variance along the new dimensions.
- A representation/compression from which one can reconstruct the original data with minimal error.

8.48.1 Minimizing Variance

Let $\mathbf{u}$ be a dimension on which we want to project the data $\mathbf{x}_i$ so that we obtain a new representation z_i that has maximum variance.

It is easy to see that the mean of the original representation gets projected to the mean of the new representation.

$$\bar{z} = \frac{1}{N}\sum_{i=1}^{N} z_i = \frac{1}{N}\sum_{i=1}^{N} \mathbf{u}^T\mathbf{x}_i = \frac{1}{N}\mathbf{u}^T\sum_{i=1}^{N}\mathbf{x}_i = \mathbf{u}^T\mu$$

We are interested in finding a $\mathbf{u}$ that maximizes the variance after the projection

$$\arg\max \frac{1}{N}\sum_{i=1}^{N}(z_i - \bar{z})^2 = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{u}^T\mathbf{x}_i - \mathbf{u}^T\mu)^2$$

$$\arg\max \frac{1}{N}\left(\mathbf{u}^T\sum_{i=1}^{N}[\mathbf{x}_i - \mu][\mathbf{x_i} - \mu]^T\right)\mathbf{u}$$

$$\arg\max \frac{1}{N}\mathbf{u}^T\Sigma\mathbf{u}$$

with the constraint of $||\mathbf{u}|| = 1$, the solution to this problem can be seen as the eigen vector corresponding to the largest eigen value of the covariance matrix Σ. (see the derivation somewhere else in the notes.) When the data is centered or mean is subtracted, one can see that $\Sigma = \mathbf{X}\mathbf{X}^T$. Where $\mathbf{X}$ is a $d \times N$ data matrix. (Note: may be we did use the notation in an earlier lecture for the transpose of this matrix.)

Best dimension to project so as to maximize the variance is the eigen vector corresponding to the largest eigen value. The second best will be the second largest one and so on.

8.48.2 Minimizing Reconstruction Loss

Let $\mathbf{u_1}, \ldots \mathbf{u_d}$ be d orthonormal vectors. We can represent the vectors $\mathbf{x}$ as $\sum_{i=1}^{d}\alpha_i\mathbf{u}_i$. Where the scalar α_i is $\mathbf{x}^T\mathbf{u_i}$. However, if we use smaller than d basis vectors,

there could be some loss or reconstruction error. Let us consider the loss when we use only one $\mathbf{u}$. i.e.,

$$\mathbf{x} - \mathbf{u}\mathbf{u}^T\mathbf{x}$$

Sum of the reconstruction loss for all the N data samples is now:

$$\sum_{i=1}^{N} ||\mathbf{x_i} - \mathbf{u}\mathbf{u}^T\mathbf{x}_i||^2$$

$$= \sum_{i=1}^{N} \left(\mathbf{x}_i^T\mathbf{x}_i + (\mathbf{u}\mathbf{u}^T\mathbf{x}_i)^T(\mathbf{u}\mathbf{u}^T\mathbf{x}_i) - 2\mathbf{x}_i^T\mathbf{u}\mathbf{u}^T\mathbf{x}_i\right)$$

We would like to minimize this. First term is positive (non negative). It is independnet of $\mathbf{u}$. Therefore, we would like to minimize:

$$\sum_{i=1}^{N} (\mathbf{x_i}^T\mathbf{u}\mathbf{u}^T\mathbf{u}\mathbf{u}^T\mathbf{x_i} - 2\mathbf{x}_i^T\mathbf{u}\mathbf{u}^T\mathbf{x}_i)$$

We also know that $\mathbf{u}^T\mathbf{u} = 1$.

$$= \sum_{i=1}^{N} -\mathbf{x}_i^T\mathbf{u}\mathbf{u}^T\mathbf{x}_i = \sum_{i=1}^{N} -\mathbf{u}^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{u} = -\mathbf{u}^T\Sigma\mathbf{u}$$

Minimizing the reconstruction error now becomes that of Maximizing

$$\mathbf{u}^T\Sigma\mathbf{u}$$

with our familiar constraint of $\mathbf{u}^T\mathbf{u} = 1$. This reduces the solution as the eigen vectors corresponding to the largest eigen values.

8.48.3 PCA: Algorithm

- Center the data by subtracting the mean μ
- Compute the covariance matrix $\Sigma = \frac{1}{N}\mathbf{X}\mathbf{X}^T = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T$
- Compute eigen values and eigen vectors of Σ.
- Take the k eigen vectors corresponding to the largest k eigen values. Keep the eigen vectors as the rows and create a matrix $\mathbf{U}$ of $k \times d$.
- Compute the reduced dimensional vectors as

$$\mathbf{z}_i = \mathbf{U}\mathbf{x}_i$$

8.48.3.1 How many eigen vectors?

In practice what should be the value of k? This is often decided by looking at how much information is lost in doing the dimensionality reduction. An estimate of this is

$$\frac{\sum_{i=k+1}^{d}\lambda_i}{\sum_{i=1}^{d}\lambda_i}$$

Often k is picked such that the above ratio is less than 5% or 10%.

Q: Why this ratio is useful? Can you find an explanation?

8.49 Example: Eigen Face

A classical example/application of PCA is in face recognition and face representation. Let us assume that we are given N face images each of $\sqrt{d} \times \sqrt{d}$. We can visualize this as a d dimensional vector. Assume that our input is all the faces from a passport office. All the faces are approximately of the same size. They are all frontal. And also expression neutral.

- Let us assume that mean μ is subtracted from each of the face. If all of the inputs were faces, then the mean is also looking very similar to a face.
- Let the input be $\mathbf{x}_1 \ldots \mathbf{x_N}$ be the mean subtracted faces and $\mathbf{X}$ be the $d \times N$ matrix of all these faces. (Q: Practically which would be larger here? N or d?)
- We are interested in finding the eigen vectors of the matrix $\mathbf{X}\mathbf{X}^T$. Say they are $\mathbf{u_1} \ldots \mathbf{u_k}$. Note that $k \leq \min(N, d)$.
- We then project each of the inputs $\mathbf{x}$ to these new eigen vectors and obtain a new feature.

$$z_i = \mathbf{u_i}^T\mathbf{x} \qquad i = 1, \ldots, k$$

- The new representation $\mathbf{z}$ is obtained like this. You can also look at this as forming a $k \times d$ matrix $\mathbf{U}$ which has its rows eigen vectors.
- We obtain now a set of compact (k dimensional) representation $\mathbf{z_i}$ for each of the input face images $\mathbf{x_i}$, where $i = 1, \ldots, N$

8.49.1 Eigen Vectors of $\mathbf{X}\mathbf{X}^\mathbf{T}$ from $\mathbf{X}^\mathbf{T}\mathbf{X}$

It is worth to see whether N of d is large more closely. In many cases N is larger than d. The reverse is also possible. In the case of eigen faces, it is quite possible that the image size is say 100×100 (or $d = 10000$) and N is only 1000. Let us see how the eigen vectors of $\mathbf{X}^\mathbf{T}\mathbf{X}$ and $\mathbf{X}\mathbf{X}^\mathbf{T}$ are related? The first one is a $N \times N$ matrix while the second is a $d \times d$ matrix. (note that eigen vector computation is a costly numerical computation and a smaller matrix is clearly preferred. Q: What is the computational complexity? $O(?)$)

Let us assume that $\mathbf{u}_1, \ldots \mathbf{u}_m$ are the m eigen vectors of $\mathbf{X}^T\mathbf{X}$ and $\mathbf{v}_1, \ldots, \mathbf{v}_m$ are the eigen vectors of $\mathbf{X}\mathbf{X}^T$. Note that $m \leq \min(d, N)$.

$$\mathbf{X}^T\mathbf{X}\mathbf{u} = \lambda\mathbf{u} \tag{8.12}$$

$$\mathbf{X}\mathbf{X}^T\mathbf{v} = \lambda\mathbf{v} \tag{8.13}$$

Note that $\mathbf{u}$ is $N \times 1$ and $\mathbf{v}$ is $d \times 1$. Assume $d >> N$. We are interested in computing $\mathbf{v}$ from $\mathbf{u}$.

Let us premultiply both side by $\mathbf{X}^T$.

$$\mathbf{X}^T\mathbf{X}\mathbf{X}^T\mathbf{v} = \lambda\mathbf{X}\mathbf{v}$$

Let us replace $\mathbf{X}^T\mathbf{v}$ by $\mathbf{u}$.

$$\mathbf{X}\mathbf{X}^T\mathbf{u} = \lambda\mathbf{u}$$

The computational procedure can be now:

- Compute the m principal eigen vectors for $\mathbf{X}\mathbf{X}^T$, say $\mathbf{v}_1, \ldots \mathbf{v}_m$.
- Compute the eigen vectors of our interest as

$$\mathbf{u}_i = \mathbf{X}^T\mathbf{v}_i$$

Q: Write the steps of the Eigen face based face representation learning.