**Introduction to Machine Learning** **Fall Semester**

Lecture 10

*Lecturer: Amir Globerson* *Scribe: Amir Globerson, Eran Halperin*

## 10.1 Introduction to Linear Regression

So far, we focused on classification problems. In a classification problem, we get as an input a multidimensional point $\boldsymbol{x}$, which is typically a vector over the real, or another field, and the output is a class $y \in \{0, 1\}$. If there are more than two classes we may want to have $y \in \{0, 1, 2, \ldots, K-1\}$, where $K$ is the number of classes. In the training data we are given examples $(\boldsymbol{x}_i, y_i)$, and in the test set we are given the value of $\boldsymbol{x}$ and we need to find our best 'guess' for the value of $y$. We have seen examples for classifiers in the last few lectures, including the perceptron, support vector machines, and finally boosting.

In some cases, we are interested in giving a more detailed classification of the data. Particularly, in the limit we want to assign every value $\boldsymbol{x}$ a value $y \in R$, thus we can think of it as a classification with an infinite number of classes. Based on the training data we need to learn the relation between $\boldsymbol{x}_i$ and $y_i$, and then in the test data we can assign each point to a value based on an extrapolation of the relation $(\boldsymbol{x}_i, y_i)$ to the entire domain of $\boldsymbol{x}$. In the most general case, we can assume that the output is also a vector, i.e., $\boldsymbol{y} \in R^k$, however in this lecture we will focus on the case where $y \in R$. One of the main techniques used for this extrapolation is termed regression analysis, which will be the topic of this lecture.

There are many natural scenarios in which regression analysis is useful. Consider for example the case in which we want to predict the risk of having high cholesterol based on the weight and height of a person. In this case, physicians usually use a measure called Body Mass Index (BMI), which is defined as $BMI = \frac{weight(kg)}{height^2(m)}$. We can measure the cholesterol level of a number of individuals, and measure their BMI values as well (Figure 10.1). It is clear from the figure that there is a monotonic relation between the BMI and the cholesterol level. In linear regression, we search for a line that in some sense fits this relation as best as we can. We will explain below what criterion we use in order to find this line.

Another example for regression is the prediction of the time a person spends in front of the television based on the person's age and gender. Such information could be useful for marketing purposes. Regression can also be used to predict the preference score that a person gives a book or a movie (e.g., in Netflix) based on some set of features describing the person's past choices (e.g., what is the average score the customer gave to action movies, to movies starring Nicholas Cage, etc.). In another example from the world of computation, we can try to predict the memory usage of a specific job based on some features of the job, for example the user submitting the job, his/her group, which type of machine the user is

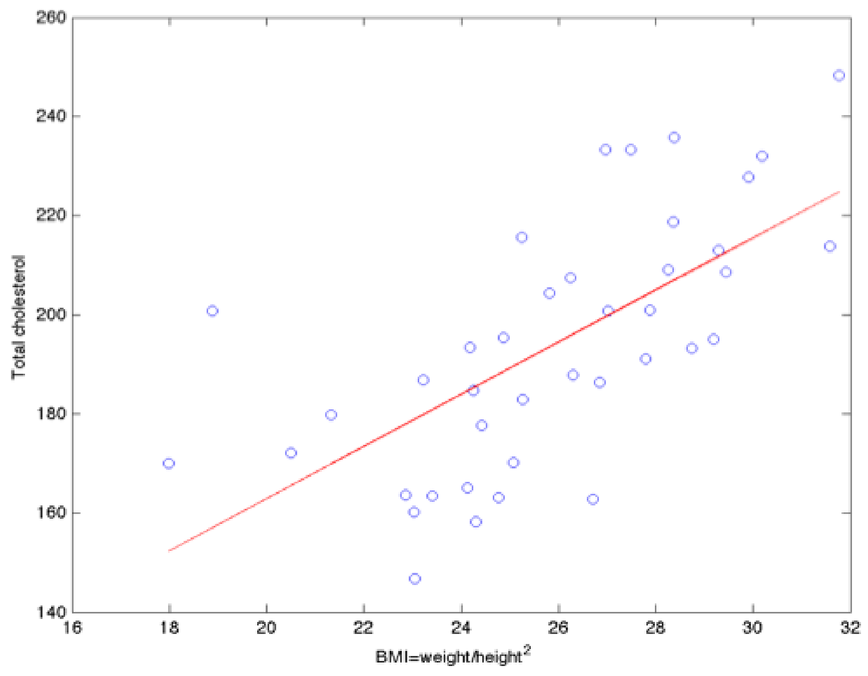

Figure 10.1: Regression line of cholesterol vs. BMI

using (for example what the RAM of the machine is), and the time of day. There are many more natural examples and indeed regression is a widely applied tool that has been studied extensively.

## 10.2 Regression - Formal Definition

The input to a regression problem is a set of points $(\boldsymbol{x}_i, y_i)$. Assume $\boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}$. Assume there is a linear relation between $\boldsymbol{x}$ and $y$, so that

$$y \approx \boldsymbol{a} \cdot \boldsymbol{x}$$

. Note that classical regression would have an additional bias term $\boldsymbol{a} \cdot \boldsymbol{x} + b$. However, we can always add a constant feature $x_{d+1} = 1$, so that $a_{d+1} = b$. We therefore do not explicitly use a bias term.

In linear regression, we are searching for such a line, by finding $\boldsymbol{a}, b$ that minimize the following:

$$\hat{\boldsymbol{a}} = \arg\min_{\boldsymbol{a}} \sum_i (y_i - \boldsymbol{a} \cdot \boldsymbol{x}_i)^2 \tag{10.1}$$

Note that this is convex in $\boldsymbol{a}$ and therefore there are no local minima.

In Figure 10.2 we illustrate the optimization. The values $r_i = y_i - \hat{\boldsymbol{a}} \cdot \boldsymbol{x}_i$ are called the residuals of the regression, and the objective of the regression is to find $\boldsymbol{a}$ such that the sum of squares of the residuals is minimized.

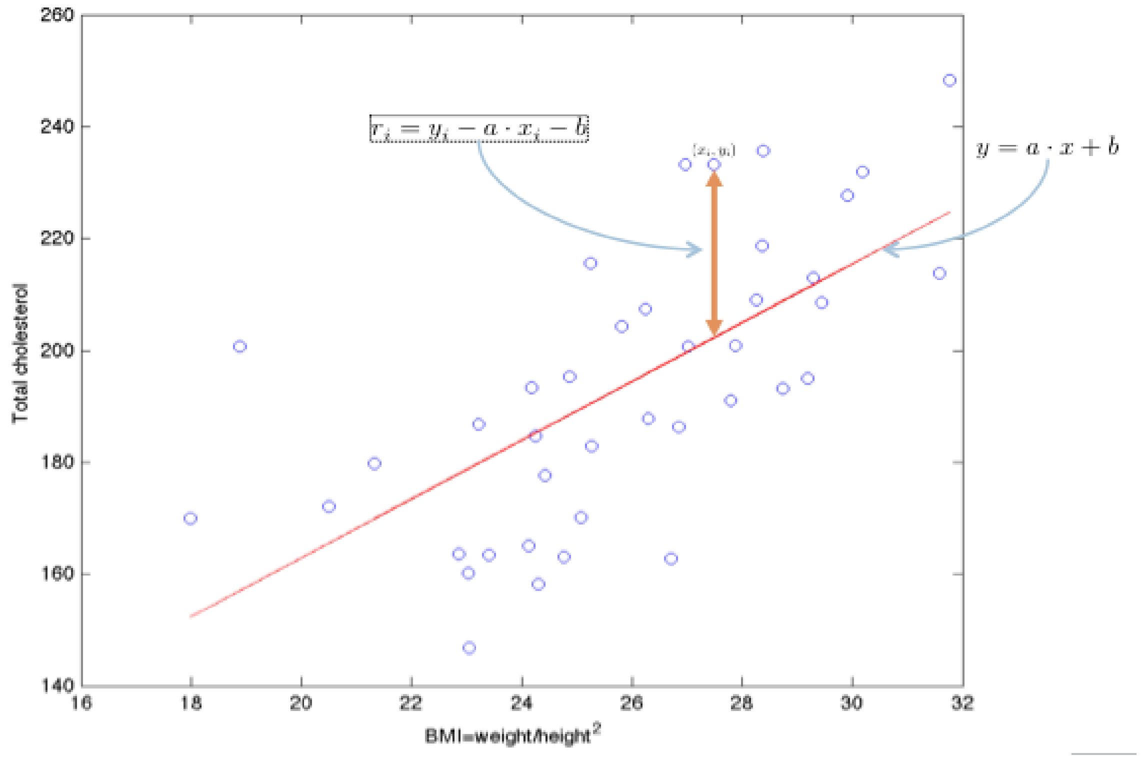

Figure 10.2: Regression line of cholesterol vs. BMI

## 10.3 Solving the regression problem

Our goal in regression is to solve the optimization problem Equation 10.1. We next explain how this can be done using simple algebra.

Begin with some notations. As usual, we assume $\boldsymbol{a} \in \mathbb{R}^d$ is a column vector. Similarly we define $\boldsymbol{y} \in \mathbb{R}^n$ to be the column vector of consisting of all $y_i$ values.

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \tag{10.2}$$

We also use $X$ to denote the matrix in $\mathbb{R}^{n,d}$ whose $i^{th}$ row is $\boldsymbol{x}_i$. We also use $\boldsymbol{v}_j \in \mathbb{R}^n$ to denote the $j^{th}$ column of $X$.

$$X = \begin{pmatrix} -\ \boldsymbol{x}_1\ - \\ \dots \\ \dots \\ \dots \\ -\ \boldsymbol{x}_n\ - \end{pmatrix} = \begin{pmatrix} | & \dots & | \\ \boldsymbol{v}_1 & \dots & \boldsymbol{v}_d \\ | & \dots & | \end{pmatrix}. \tag{10.3}$$

With these definitions the optimization problem in Equation 10.1 can be simply written as:

$$\ell(\boldsymbol{a}) = \|\boldsymbol{y} - X\boldsymbol{a}\|_2^2 = (\boldsymbol{y} - X\boldsymbol{a})^T(\boldsymbol{y} - X\boldsymbol{a}) = \|\boldsymbol{y}\|_2^2 - 2\boldsymbol{y}^T X\boldsymbol{a} + \boldsymbol{a}^T X^T X\boldsymbol{a}$$

Taking the gradient wrt to $\boldsymbol{a}$ we get:

$$\nabla\ell(\boldsymbol{a}) = 2X^T X\boldsymbol{a} - 2X^T\boldsymbol{y}$$

Equating the gradient to zero we get

$$X^T X\boldsymbol{a} = X^T\boldsymbol{y}, \tag{10.4}$$

Any $\boldsymbol{a}$ that satisfies the above equation is optimal.

In particular if $X^T X$ is non-singular (i.e., it is invertible) then the optimum is unique and equal to

$$\boldsymbol{a} = (X^T X)^{-1} X^T\boldsymbol{y} \tag{10.5}$$

See Section 10.4 for what happens in other cases.

The linear equations described in Equation 10.4 are called the Normal Equations. These equations have a natural geometric interpretation. Since the equations are equivalent to the identity $X^T(\boldsymbol{y} - X\boldsymbol{a}) = 0$, and since the residuals are defined as $\boldsymbol{r} = \boldsymbol{y} - X\boldsymbol{a}$, the Normal Equations require the residuals to be orthogonal to each of the columns of $X$. The columns of $X$, denoted by $\boldsymbol{v}_1, \dots, \boldsymbol{v}_d$, correspond to the variables used by the regression (e.g., BMI in the examples above) when evaluated on the training data. Therefore, the solution of the linear regression is a projection of $\boldsymbol{y}$ onto the subspace spanned by $\boldsymbol{v}_1, \dots, \boldsymbol{v}_d$ (see Figure 10.3). Algebraically, in the special case where the columns of $X$ are orthogonal ($\boldsymbol{v}_i^T\boldsymbol{v}_j = 0$ for $i \neq j$), we get $a_i = \frac{\boldsymbol{v}_i^T\boldsymbol{y}}{\|\boldsymbol{v}_i\|_2^2}$. In other words, feature $x_i$ will get a large weight $a_i$ if the feature is aligned with the label $y$.

## 10.4 Singularity Issues

As mentioned earlier when $X^T X$ is non-singular we have a unique solution. Otherwise, the equation Equation 10.4 has a continuum of solutions. As the simplest example of this

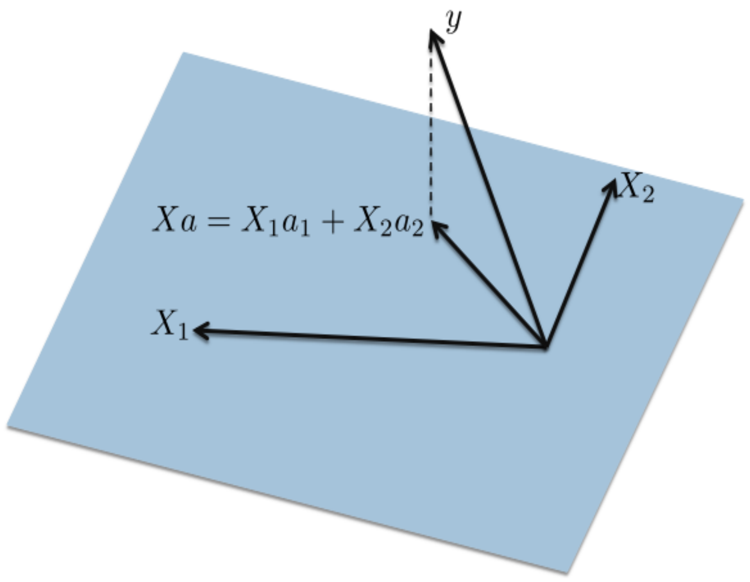

Figure 10.3: Geometric Interpretation of Linear Regression

case, consider the case where input vectors are $\boldsymbol{x}_i = [\alpha_i, 0, \ldots, 0]$. Then this is really a one dimensional regression problem embedded in $\mathbb{R}^d$. The values $a_2, \ldots, a_d$ will have no effect on the solution and we can therefore set them to any value.

Other non-singular cases can be transformed into a form similar to the above. Namely, where some of the $\boldsymbol{x}_i$ coordinates are set to zero. Thus, the singular case corresponds to a well specified regression problem in a lower dimensional space.

Formally, assume $X^TX$ is singular. Denote by $\boldsymbol{v} \in \mathbb{R}^d$ a vector in the null space of $X^TX$. Namely a vector such that

$$X^TX\boldsymbol{v} = \boldsymbol{0} \tag{10.6}$$

Now assume $\boldsymbol{a}$ is some solution to the equations Equation 10.4. Then $\boldsymbol{a}+\boldsymbol{v}$ is also a solution because:

$$X^TX(\boldsymbol{a}+\boldsymbol{v}) = X^TX\boldsymbol{a} + \boldsymbol{0} = X^T\boldsymbol{y} \tag{10.7}$$

In which cases would we expect to have a nonsingular matrix $X^TX$?. Note that the singularity of $X^TX$ depends on its rank. $X^TX$ is a $d \times d$ matrix, and it is singular if its rank is strictly smaller than $d$.

Here are some cases when this will happen:

- If $n < d$ then rank of $X$ is at most $n$ and is smaller than $d$ and therefore rank of $X^TX$ is less than $d$ and $X^TX$ is singular.

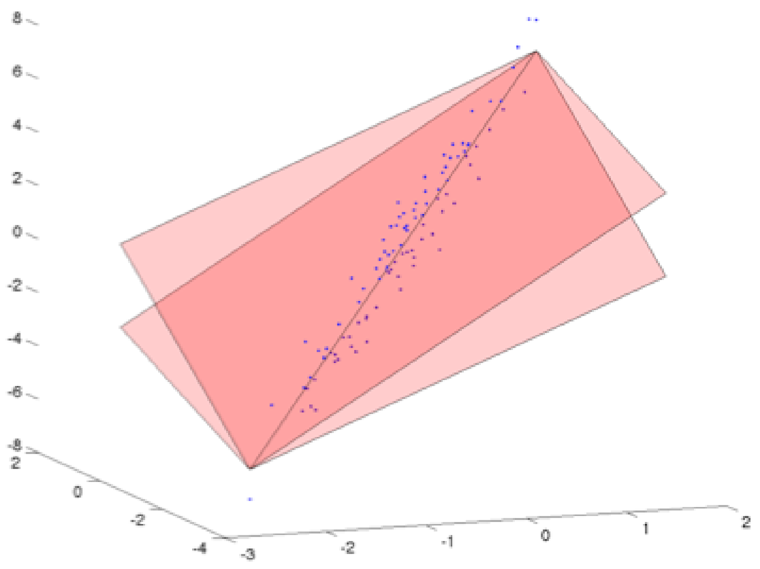

Figure 10.4: An example for a regression in an almost singular scenario

- More generally, rank of $X$ is less than $d$ if $X$ has linearly dependent columns. This means that there is one feature that depends deterministically on the others. To take the simplest case assume feature $i$ is equal to feature $j$. Then putting weight $w$ on feature $i$ is equivalent to putting weight $\alpha w$ on $i$ and $(1-\alpha)w$ on $j$ for any $\alpha \in \mathbb{R}$. This shows that the regression problem will have many solutions.

Typically, if $d < n$ we will not have non-singularity (in the sense of eigenvalues being exactly zero). But it could be that columns of $X$ are "almost" linear dependent, in the sense that it has singular values close to zero. Such a case is depicted in Figure 10.4. In this case, the solution will be unique but there will be other solutions that are almost as good.

## 10.5 Ridge Regression

In the singular case we have many possible solutions for the regression problem. In classification we had a similar situation of having multiple classifiers that separate a set of points. There, we addressed this by adding regularization, which "prefers" low norm solutions.

A similar approach can be used in regression, and is known as ridge regression. The optimization problem is then:

$$\hat{\boldsymbol{a}} = \arg\min_{\boldsymbol{a}} \ell(\boldsymbol{a}) = \arg\min_{\boldsymbol{a}} \{(\boldsymbol{y} - X\boldsymbol{a})^T(\boldsymbol{y} - X\boldsymbol{a}) + \lambda\|\boldsymbol{a}\|_2^2\}. \tag{10.8}$$

where $\lambda$ is a regularization coefficient.

We next solve the above problem. Note that the optimization function can be written as

$$\ell(\boldsymbol{a}) = \|\boldsymbol{y}-X\boldsymbol{a}\|_2^2+\lambda\|\boldsymbol{a}\|_2^2 = (\boldsymbol{y}-X\boldsymbol{a})^T(\boldsymbol{y}-X\boldsymbol{a})+\lambda\|\boldsymbol{a}\|_2^2 = \|\boldsymbol{y}\|_2^2-2\boldsymbol{y}^TX\boldsymbol{a}+\boldsymbol{a}^T(X^TX+\lambda I_d)\boldsymbol{a} \tag{10.9}$$

where $I_d$ is the identity matrix of size $d$.

Continuing as we did in the non-regularized case, we obtain:

$$\hat{\boldsymbol{a}} = (X^TX + \lambda I_d)^{-1}X^T\boldsymbol{y}. \tag{10.10}$$

The solution given by Equation 10.10 has a few interesting properties. First, it is easy to see that when $\lambda = 0$ we get the Normal equations, which is what we should expect. Second, note that the matrix $X^TX + \lambda I$ is positive definite and therefore non-singular for $\lambda > 0$. Therefore Ridge regression can be viewed as a numerical correction to the standard linear regression - by adding a small positive $\lambda$ times the identity to $X^TX$ we implicitly change the set of columns of $X$ so that it has full rank and is far from singular.

### 10.5.1 Regression - Extensions

Regression is a well studied topic. Here we covered the very basics of linear regression. Of course there are many possible extensions. Here we mention a few:

- Sparse regression - The vector of coefficients tells us how much weight to put on each feature. If we have many features it may be hard to interpret all these numbers. Thus, it could make sense to ask for a sparse $\boldsymbol{a}$ with as many zeros as possible, subject to the regression error being low. In practice this is done by adding $\ell_1$ regularization to the regression objective, and it can be shown this will result in sparse solutions.

- Generalization - The approach we presented above is essentially an ERM method for minimizing the expected squared loss $\mathbb{E}_{\boldsymbol{x},y}\left[(y - \boldsymbol{a}\cdot\boldsymbol{x})^2\right]$. Or for a more general regression function $f(\boldsymbol{x})$ the expected value $\mathbb{E}_{\boldsymbol{x},y}\left[(y - f(\boldsymbol{x}))^2\right]$. It is possible to derive generalization bounds that relate the empirical training error to the generalization error, and these would depend on the complexity of the class $f(\boldsymbol{x})$ in a similar way to classification.

- Non linear regression - Here we studied the case $y = f(\boldsymbol{x})$ where $f(\boldsymbol{x})$ is linear. Of course $f$ can be a more complex function. One example is a neural net. Another is kernel regression where $f(\boldsymbol{x}) = \boldsymbol{a}\cdot\boldsymbol{\phi}(\boldsymbol{x})$ and $\boldsymbol{\phi}(\boldsymbol{x})$ is a given non-linear transformation on $\boldsymbol{x}$. If $\boldsymbol{\phi}(\boldsymbol{x})$ corresponds to a kernel, we can use the kernel trick as we did in classification.

## 10.6 Unsupservised Learning

So far, we viewed machine learning as learning a mapping from input $\boldsymbol{x}$ to output $y$. In this case the objective was clear: find a mapping $f(\boldsymbol{x})$ that has low error on training data.

This approach relies on:

- Having a clear defined task of mapping $\boldsymbol{x}$ to a label $y$.
- Having large amounts of labeled data to train our classifier on.

There are many settings where one of these assumptions does not hold. Instead, we are given just $\boldsymbol{x}$ instances $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and there is no prediction goal. This is the *unsupervised learning* setting. Some examples where it comes up are:

- Scientific research - Assume we measure sequence the genome of a million individuals. We would like to be able to characterize what these genomes look like. Is there a set of *typical* genomes, are there parts that are fixed and part that vary in predictable ways. The same applies to any measurements performed on large populations (e.g., psychometric evaluations).
- Lack of labeled data - Assume we want to classify images into categories. But we can only label a small set of examples (e.g., because we need to pay the labelers). On the other hand, we have access to huge amounts of unlabeled data. Can we use it? This is known as semi-supervised learning.

Unsupervised learning is thus not as well defined as its supervised counterpart. A somewhat vague definition would be that its goal is to find underlying structure in data, such that this structure can be used to either explain the data, or for integrating within a supervised learning model.

To get some more intuition consider Figure 10.5. The red dots correspond to data points $\boldsymbol{x}_i$ in $\mathbb{R}^2$. It is clear that these points have some structure that could be useful. In one, we see the data can be separated into two clusters. And in the other the data approximately lies on a line. Of course this is easy to detect in $\mathbb{R}^2$. But how can we do it in $\mathbb{R}^{10^6}$? In the next few classes we will learn to do precisely that.

The practical goal of many unsupervised algorithms is to transform data into a *simpler* form. Two common examples are:

- Dimensionality Reduction - Transform $\boldsymbol{x}$ from $\mathbb{R}^d$ to $\mathbb{R}^r$ where $r < d$. Of course the transformation should not lose important information about $\boldsymbol{x}$.
- Clustering - Transform $\boldsymbol{x}$ into an integer $\{1, \ldots, K\}$ that tells us which cluster $\boldsymbol{x}$ belongs to. Of course this can also be considered a dimensionality reduction method.

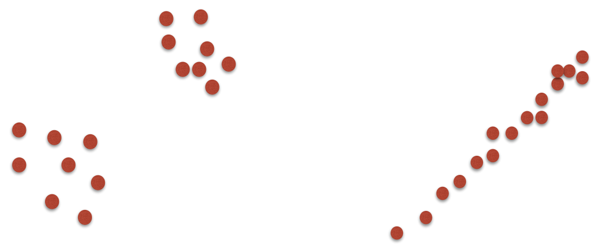

Figure 10.5: Examples of types of underlying structure in unlabeled data.

### 10.6.1 Principal Component Analysis

Consider the points on the right hand side of Figure 10.5. The structure that is apparent is that they lie approximately on a line. This has several important implications:

- It tells us that $x_2$ is roughly determined by $x_1$.
- We can approximately represent each point via a single number rather than two. To clarify, assume that the data lies exactly on a line corresponding to vector $\boldsymbol{v}$. Then each point $\boldsymbol{x}$ can be written as $\boldsymbol{x} = \alpha\boldsymbol{v}$ and can thus be represented by $\alpha$ alone.

Of course our data will usually have dimension higher than two. There we will be interested in cases where $\boldsymbol{x}$ lies on a linear subspace of $\mathbb{R}^d$, where the dimension of the subspace is smaller than $d$. For example for $d = 3, r = 2$ the data is on a plane through the origin.

Let us explain how this is related to dimensionality reduction. A linear subspace of $\mathbb{R}^d$ is determined by a set of basis vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r \in \mathbb{R}^d$ that span the subspace. Since $\boldsymbol{v}_i$ are independent we can assume wlog that they are orthonormal so that:

$$\boldsymbol{v}_i \cdot \boldsymbol{v}_j = 0 \tag{10.11}$$

We will also use $V \in \mathbb{R}^{d,r}$ to denote the matrix whose columns are $\boldsymbol{v}_i$. From now on we shall use $V$ to also denote the linear subspace itself (e.g., we will say $\boldsymbol{x}$ is in the subspace $\boldsymbol{v}$). The above means that:

$$V^T V = I_r \tag{10.12}$$

The fact $V$ is a basis means that any vector $\boldsymbol{x}$ in the subspace can be written as:

$$\boldsymbol{x} = \sum_{i=1}^{r} a_i \boldsymbol{v}_i \tag{10.13}$$

Or in matrix form (where $\boldsymbol{a} \in \mathbb{R}^r$

$$\boldsymbol{x} = V\boldsymbol{a} \tag{10.14}$$

where $V \in \mathbb{R}^{d,r}$ is a matrix whose columns are $\boldsymbol{v}_i$. Because $V$ satisfies Equation 10.12 we have:

$$\boldsymbol{a} = V^T \boldsymbol{x} \tag{10.15}$$

Equations 10.14 and 10.15 above demonstrate a simple but useful fact. If we know that $\boldsymbol{x}$ lies in the subspace $V$, then we can "encode" a vector $\boldsymbol{x} \in \mathbb{R}^d$ by a lower dimensional vector $\boldsymbol{a} \in \mathbb{R}^r$, using Equation 10.15. We lose nothing in this encoding since we can always decode back to $\boldsymbol{x}$ via Equation 10.14. This is the simplest version of an autoencoder: namely a procedure for encoding $\boldsymbol{x}$ via a low dimensional $\boldsymbol{a}$ and then mapping $\boldsymbol{a}$ back to $\boldsymbol{x}$.

What happens when $\boldsymbol{x}$ isn't exactly in $V$? We can then seek a point $\hat{\boldsymbol{x}}$ in $V$ that is closest to $\boldsymbol{x}$. This is known as the orthogonal projection of $\boldsymbol{x}$ on $V$, and is given by:

$$\hat{\boldsymbol{x}} = \arg \min_{\boldsymbol{x}'=V\boldsymbol{a}} \|\boldsymbol{x}' - \boldsymbol{x}\|_2^2 \tag{10.16}$$

Solve this via:

$$\min_{\boldsymbol{a}} \|V\boldsymbol{a} - \boldsymbol{x}\|_2^2 \tag{10.17}$$

Where taking gradient wrt $\boldsymbol{a}$ gives (you should verify that this is indeed the gradient):

$$\begin{aligned} V^T(V\boldsymbol{a} - \boldsymbol{x}) &= 0 \\ \boldsymbol{a} = V^T\boldsymbol{x} \end{aligned}$$

where we have used the orthogonality of $V$. This results in:

$$\hat{\boldsymbol{x}} = VV^T\boldsymbol{x} \tag{10.18}$$

So $\hat{\boldsymbol{x}}$ is the best approximation (in a quadratic error sense) of $\boldsymbol{x}$ via the subspace $V$. Note again the encoding-decoding interpretation this has. $\boldsymbol{x}$ is first encoded into a vector in $\boldsymbol{a} \in \mathbb{R}^r$ given by $\boldsymbol{a} = V^T\boldsymbol{x}$ and then decoded into $\hat{\boldsymbol{x}} \in \mathbb{R}^d$ via $\hat{\boldsymbol{x}} = V\boldsymbol{a}$. Note that both the encoding and decoding are linear. But now $\hat{\boldsymbol{x}} \neq \boldsymbol{x}$ since $\boldsymbol{x}$ was not not necessarily in the subspace.

Principal Component Analysis is a very popular and effective dimensionality reduction method that looks for a linear subspace that approximates the data. It was invented by Karl Pearson in 1901 and has been actively studied and extended since. If you have high dimensional data, it's the first thing you should try to understand the structure of the data and reduce its dimensionality!

Assume you are given a set of data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. With $\boldsymbol{x}_i \in \mathbb{R}^d$. Now, you suspect that the data approximately lies on an $r$ dimensional linear subspace, and you would like to find its basis $V \in \mathbb{R}^{d,r}$. Given $\boldsymbol{x}_i$ and a subspace $V$, we known that the best approximation of $\boldsymbol{x}_i$ in $V$ is given by $\hat{\boldsymbol{x}}_i = VV^T\boldsymbol{x}_i$. It thus makes sense to look for the $V$ that minimizes the distance between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}_i$. Namely, we want to solve the optimization problem: (the factor $\frac{1}{n}$ doesn't affect the optimization. It'll be useful later for interpreting the problem):

$$\min_{V:V^TV=I} \sum_i \|\boldsymbol{x}_i - VV^T\boldsymbol{x}_i\|_2^2 \tag{10.19}$$

The vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r$ that solve the above are known as the principal components of the data $X$.

We now turn to solving the above problem. Begin by expanding the objective a bit, using the orthogonality of $V$:

$$\begin{aligned}
\|\boldsymbol{x}_i - VV^T\boldsymbol{x}_i\|_2^2 &= (\boldsymbol{x}_i - VV^T\boldsymbol{x}_i)^T(\boldsymbol{x}_i - VV^T\boldsymbol{x}_i) \\
&= \|\boldsymbol{x}_i\|_2^2 - 2\boldsymbol{x}_i^TVV^T\boldsymbol{x}_i + \boldsymbol{x}_i^TVV^TVV^T\boldsymbol{x}_i \\
&= \|\boldsymbol{x}_i\|_2^2 - 2\boldsymbol{x}_i^TVV^T\boldsymbol{x}_i + \boldsymbol{x}_i^TVV^T\boldsymbol{x}_i \\
&= \|\boldsymbol{x}_i\|_2^2 - \boldsymbol{x}_i^TVV^T\boldsymbol{x}_i
\end{aligned}$$

To simplify this further, note that $\boldsymbol{x}_i^TVV^T\boldsymbol{x}_i$ is a scalar and we can therefore write $\boldsymbol{x}_i^TVV^T\boldsymbol{x}_i = \mathrm{Tr}\left(\boldsymbol{x}_i^TVV^T\boldsymbol{x}_i\right)$. Now we can use the fact that the trace is invariant under cyclic permutations (namely $\mathrm{Tr}\,(ABC) = \mathrm{Tr}\,(CAB)$) to write: $\boldsymbol{x}_i^TVV^T\boldsymbol{x}_i = \mathrm{Tr}\left(\boldsymbol{x}_i^TVV^T\boldsymbol{x}_i\right) = \mathrm{Tr}\left(V^T\boldsymbol{x}_i\boldsymbol{x}_i^TV\right)$. So we have:

$$\|\boldsymbol{x}_i - VV^T\boldsymbol{x}_i\|_2^2 = \|\boldsymbol{x}_i\|_2^2 - \mathrm{Tr}\left(V^T\boldsymbol{x}_i\boldsymbol{x}_i^TV\right) \tag{10.20}$$

And summing over $i$ (and also using linearity of trace and $ABC + ADC = A(B + D)C$):

$$\sum_i \|\boldsymbol{x}_i - VV^T\boldsymbol{x}_i\|_2^2 = \sum_i \|\boldsymbol{x}_i\|_2^2 - \mathrm{Tr}\left(V^T\left(\sum_i \boldsymbol{x}_i\boldsymbol{x}_i^T\right)V\right) \tag{10.21}$$

Note we can write:

$$\frac{1}{n}\sum_i \boldsymbol{x}_i\boldsymbol{x}_i^T = \frac{1}{n}X^TX \equiv \Sigma \tag{10.22}$$

The matrix $\Sigma$ is known as the correlation matrix of the data $X$. If the data has zero mean, namely $\sum_i \boldsymbol{x}_i = 0$ then $\Sigma$ is the covariance matrix. To see what $\Sigma$ means, note that:

$$\Sigma_{j,k} = \frac{1}{n}\sum_i x_{ij}x_{ik} \tag{10.23}$$

Namely, it calculates the average of the product between the $j$ and $k$ dimensions of the vector $\boldsymbol{x}$.

Putting all of this together we have an optimization problem equivalent to Equation 10.19 (we ignored additive constants that don't depend on $V$):

$$\max_{V:V^TV=I} \operatorname{Tr}\left(V^T\Sigma V\right) \tag{10.24}$$

To solve this, we write $\Sigma$ in terms of its eigen-decomposition. We first note that $\Sigma$ is PSD (since it can be written as $X^TX$) and therefore has non-negative eigenvalues. We can therefore write it as:

$$\Sigma = UDU^T \tag{10.25}$$

where $D$ is diagonal with eigenvalues $\lambda_i > 0$ on the diagonal (assume they are sorted in decreasing order), and $U \in \mathbb{R}^{d,d}$ is a matrix whose columns $\boldsymbol{u}_i$ are orthogonal eigenvectors of $\Sigma$. So $UU^T = I$.

**Theorem 10.1** *The solution to the PCA problem in Equation 10.24 is $\boldsymbol{v}_1 = \boldsymbol{u}_1, \ldots, \boldsymbol{v}_r = \boldsymbol{u}_r$. Namely, the principal components are the $r$ eigenvectors of $\Sigma$ with the largest eigenvalues.*

We prove this in two parts (proof follows Shalev Shwartz and Ben David) . First we show which value the proposed solution obtains, and then we show that this is the best value possible. For the proposed solution we have (using the definition of trace, and the fact that $\boldsymbol{u}_i$ are eigenvectors with eigenvalues $\lambda_i$):

$$\operatorname{Tr}\left(V^T\Sigma V\right) = \sum_{i=1}^{r} \boldsymbol{u}_i^T\Sigma\boldsymbol{u}_i = \sum_{i=1}^{r} \lambda_i\boldsymbol{u}_i^T\boldsymbol{u}_i = \sum_{i=1}^{r}\lambda_i \tag{10.26}$$

For any other $V$, we will show the objective can't be greater than the above. Given any $V$ such that $V^TV = I$ we have:

$$\operatorname{Tr}\left(V^T\Sigma V\right) = \operatorname{Tr}\left(V^TUDU^TV\right) \tag{10.27}$$

Now define the matrix $Z \in \mathbb{R}^{r,d}$ as $Z = V^TU$. Note that $ZZ^T = V^TUU^TV = I_r$. We therefore have $\operatorname{Tr}\left(ZZ^T\right) = \operatorname{Tr}\left(Z^TZ\right) = r$. Denote the $i^{th}$ column of $Z$ by $\boldsymbol{z}_i$ and its squared norm by $c_i = \|\boldsymbol{z}_i\|_2^2$. Then $\operatorname{Tr}\left(Z^TZ\right) = r$ is equivalent to:

$$\sum_i c_i = r \tag{10.28}$$

Note also that $c_i \le 1$ since $Z$ can be completed to a $\mathbb{R}^{d,d}$ orthonormal matrix whose columns have norm exactly one.

We can now write:

$$\mathrm{Tr}\left(V^T\Sigma V\right) = \mathrm{Tr}\left(ZDZ^T\right) = \mathrm{Tr}\left(DZ^TZ\right) = \sum_{i=1}^{d}\lambda_i c_i \tag{10.29}$$

Because $0 \le c_i \le 1$ and $\sum_i c_i = r$ the above is at most $\sum_{i=1}^{r}\lambda_i$, which is what is achieved by the proposed solution in Equation 10.26.

### 10.6.2 PCA for Dimensionality Reduction

PCA models the data as a linear subspace. It is thus natural to transform the original data $\boldsymbol{x}$ to its coordinates in the subspace. As we've seen before this can be done via $\boldsymbol{a} = V^T\boldsymbol{x}$. This can be viewed as projection on the principal components.

So now we have for each $\boldsymbol{x}_i$ a new data point $\boldsymbol{a}_i$. If we stack all of these in a matrix $A$ (each row is $\boldsymbol{a}_i$) it can be easily shown that:

$$A^TA = D \tag{10.30}$$

where $D$ is the diagonal matrix of size $r$ with eigenvalues on its diagonal. Namely, in the new representation the features are uncorrelated.

We can also use the new representation $\boldsymbol{a}_i$to try and recover the original $\boldsymbol{x}_i$ by mapping $\boldsymbol{a}_i$ to the corresponding point in the subspace $\boldsymbol{x}'_i = V\boldsymbol{a}_i$. Some examples of this recovery process can be seen in the slides.

### 10.6.3 PCA as an Autoencoder

Dimensionality reduction involves encoding (or mapping), the input $\boldsymbol{x}$ to a lower dimensional vector $\boldsymbol{a}$. Denote this encoding function by $f : \mathbb{R}^d \to \mathbb{R}^r$. What is a good mapping? Intuitively, it's one that preserves information about $\boldsymbol{x}$. How do we check that no information is lost? We can try to recover $\boldsymbol{x}$ from $\boldsymbol{a}$. Namely, build a decoder $g : \mathbb{R}^r \to \mathbb{R}^d$ such that $g(f(\boldsymbol{x}))$ is close to $\boldsymbol{x}$. Such an encoder-decoder pair is known as an autoencoder. When used with neural nets for encoding decoding it often results in effective dimensionality reduction.

Note that if we don't constrain the encoder or decoder we can always recover with arbitrarily low error even when $r = 1$, assuming $x$ has a finite range of values (why?). Thus it makes sense to put some restrictions on the encoder and/or decoder.

The simplest restriction is to restrict both encoder and decoder to be linear. Thus assume $f(\boldsymbol{x}) = W\boldsymbol{x}$ and $g(\boldsymbol{a}) = V\boldsymbol{a}$. Here $V \in \mathbb{R}^{d,r}$ and $W \in \mathbb{R}^{r,d}$. Also assume that we measure reconstruction via $\ell_2$ norm. Then our optimization problem is:

$$\min_{W,V}\sum_i \|\boldsymbol{x}_i - VW\boldsymbol{x}_i\|_2^2 \tag{10.31}$$

We wish to show that this is in fact equivalent to our PCA formulation in Equation 10.19.

Let us assume for simplicity that $V$ is restricted to have rank $r$ (that is, we are also assuming $r \leq d$).

First, note that wlog we can assume that $V$ satisfies $V^T V = I_r$. Because if $V$ is not orthogonal, we can apply Gram-Schmidt to its columns such that $\bar{V} = VQ$ satisfies $\bar{V}^T \bar{V} = I_r$, and $Q \in \mathbb{R}^{r,r}$ is invertible. We can then set $\bar{W} = Q^{-1}W$ and $\bar{V}\bar{W} = VW$.

Now we note that $VW\boldsymbol{x}_i$ is a vector in the subspace spanned by the columns of $V$. We know that the closet point to $\boldsymbol{x}_i$ in this subspace is $VV^T\boldsymbol{x}_i$. Thus the best we can do is set $W = V^T$. Therefore the problem in Equation 10.31 is equivalent to:

$$\min_{V \in \mathbb{R}^{d,r}} \sum_i \|\boldsymbol{x}_i - VV^T\boldsymbol{x}_i\|_2^2 \tag{10.32}$$

and this is precisely the PCA objective.

Exercise: how can we avoid assuming that $V$ is restricted to rank $r$?