Math3 - Statistical inference, Introduction to multivariate statistical models: regression

Mathematics (Indiana Institute of Technology)

UNIT-3

Statistical inference, Introduction to multivariate statistical models: regression and classification problems, principal components analysis, The problem of overfitting model assessment.

Statistical Inference, Model & Estimation

Statistical inference is the process of using data analysis to deduce properties of an underlying probability distribution.[1] Inferential statistical analysis infers properties of a population , for example by testing hypotheses and deriving estimates. It is assumed that the observed data set is sampled from a larger population.

Inferential statistics can be contrasted with descriptive statistics. Descriptive statistics is solely concerned with properties of the observed data, and it does not rest on the assumption that the data come from a larger population.

Recall, a **statistical inference** aims at learning characteristics of the population from a sample; the population characteristics are *parameters*and sample characteristics are *statistics*.

A **statistical model** is a representation of a complex phenomena that generated the data.

- It has mathematical formulations that describe relationships between random variables and parameters.
- It makes assumptions about the random variables, and sometimes parameters.
- A general form: data = model + residuals
- Model should explain most of the variation in the data
- Residuals are a representation of a lack-of-fit, that is of the portion of the data unexplained by the model.

Estimation represents ways or a process of learning and determining the population parameter based on the model fitted to the data.

Point estimation and interval estimation, and hypothesis testing are three main ways of learning about the population parameter from the sample statistic.

An **estimator** is particular example of a statistic, which becomes an **estimate** when the formula is replaced with actual observed sample values.
Point estimation = a single value that estimates the parameter. Point estimates are single values calculated from the sample
Confidence Intervals = gives a range of values for the parameter Interval estimates are intervals within which the parameter is expected to fall, with a certain degree of confidence.
Hypothesis tests = tests for a specific value(s) of the parameter.
In order to perform these inferential tasks, i.e., make inference about the unknown population parameter from the sample statistic, we need to know the likely values of the sample statistic. What would happen if we do sampling many times?

We need the **sampling distribution** of the statistic

- It depends on the model assumptions about the population distribution, and/or on the sample size.
- **Standard error** refers to the standard deviation of a sampling distribution.

Degree of models/assumptions[edit]

Statisticians distinguish between three levels of modeling assumptions;

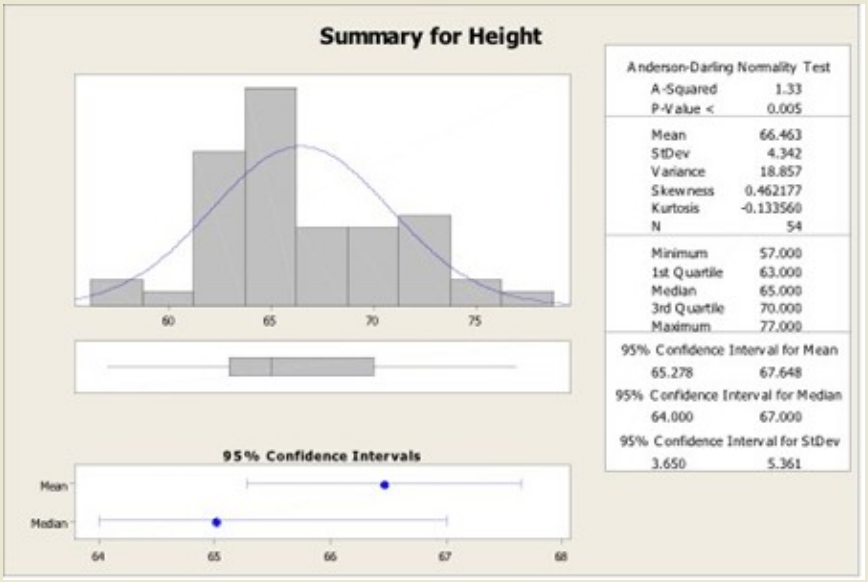
- **Fully parametric**: The probability distributions describing the data-generation process are assumed to be fully described by a family of probability distributions involving only a finite number of unknown parameters.[4] For example, one may assume that the distribution of

population values is truly Normal, with unknown mean and variance, and that datasets are generated by 'simple' random sampling. The family of generalized linear models is a widely used and flexible class of parametric models.

- **Non-parametric**: The assumptions made about the process generating the data are much less than in parametric statistics and may be minimal.[6] For example, every continuous probability distribution has a median, which may be estimated using the sample median or the Hodges–Lehmann–Sen estimator, which has good properties when the data arise from simple random sampling.
- **Semi-parametric**: This term typically implies assumptions 'in between' fully and non-parametric approaches. For example, one may assume that a population distribution has a finite mean. Furthermore, one may assume that the mean response level in the population depends in a truly linear manner on some covariate (a parametric assumption) but not make any parametric assumption describing the variance around that mean (i.e. about the presence or possible form of any heteroscedasticity). More generally, semi-parametric models can often be separated into 'structural' and 'random variation' components. One component is treated parametrically and the other non-parametrically. The well-known Cox model is a set of semi-parametric assumptions.

Height Example

We are interested in estimating the true average height of the student population at Penn State. We collect a simple random sample of 54 students. Here is a graphical summary of that sample.



- Parameter of interest is the population mean height, μ.
- Sample statistic, or a point estimator is \(\bar{X}\), and an estimate, which in this example, is 66.432.
- What is the sampling distribution of \(\bar{X}\)?

Principal components analysis (PCA).

Principal components analysis (PCA) is the most popular dimensionality reduction technique to date. It allows us to take an n-dimensional feature-space and reduce it to a k-dimensional feature-space while maintaining as much information from the original dataset as possible in the reduced dataset. Specifically, PCA will create a new feature-space that aims to capture as much variance as possible in the original dataset; I'll elaborate on this later in this post.

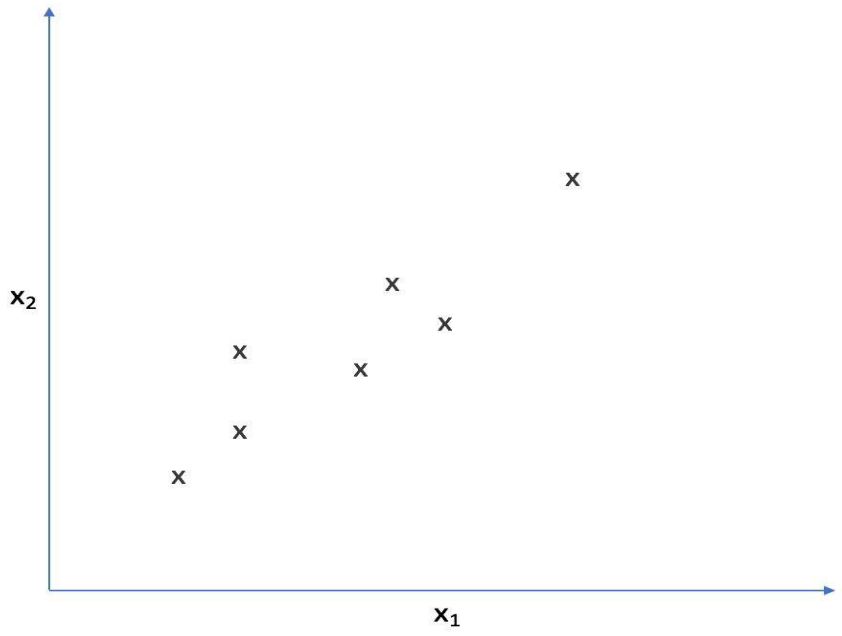
Note: We're only dealing with the feature-space and not any corresponding labels or output. Dimensionality reduction is an ***unsupervised learning*** *technique that is agnostic to the features' labels.*

Let's start by addressing the burning question: how the heck do we reduce the dimensionality of our dataset? A naive approach might be to simply

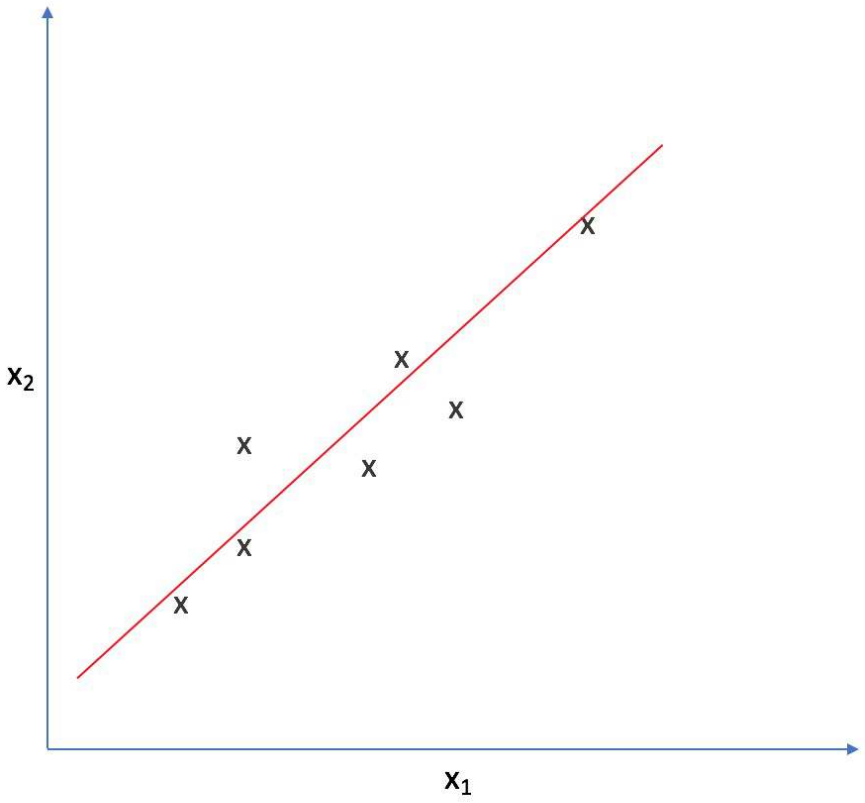
perform feature selection and keep only a subset of the original features. However, although we *are*reducing the dimensionality (we have less features), we're doing so by *deleting*information that might be valuable. A better approach would be to find a way to *compress* the information.
The ultimate goal with dimensionality reduction is to find the most compressed representation possible to accurately describe the data. If all of the features within your dataset were completely independent signals, this would be very hard. However, it's often the case that you will have redundancies present in your feature-space. You may find that many features are simply indicators of some latent signal that we haven't directly observed. As such, we would expect that indicators of the same latent signal would be correlated in some manner. For example, if we were trying to reduce the feature-space of a dataset that contained information about housing prices, features such as number of rooms, number of bedrooms, number of floors, and number of bathrooms all might be indicators of the *size* of the house. One might argue that these latent signals are the ***principal components*** which make up our dataset.

ntuition

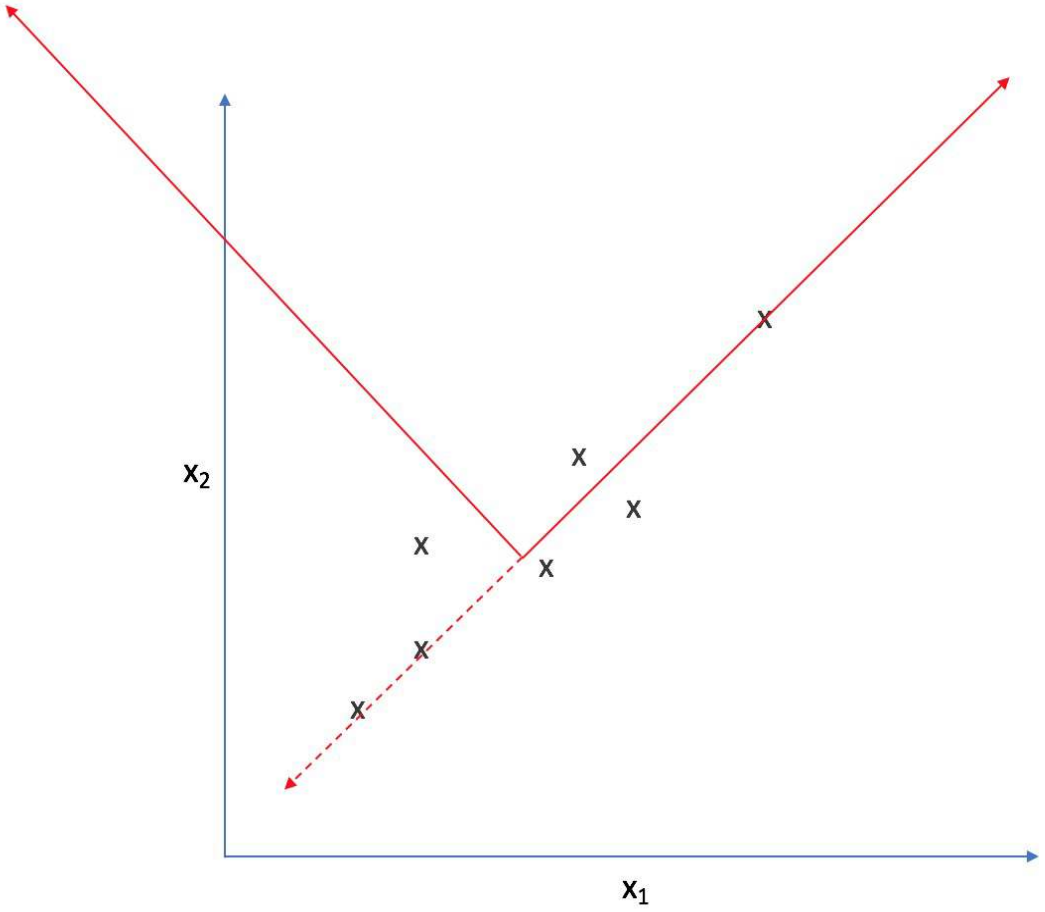
Let's look at the following two-dimensional feature-space.



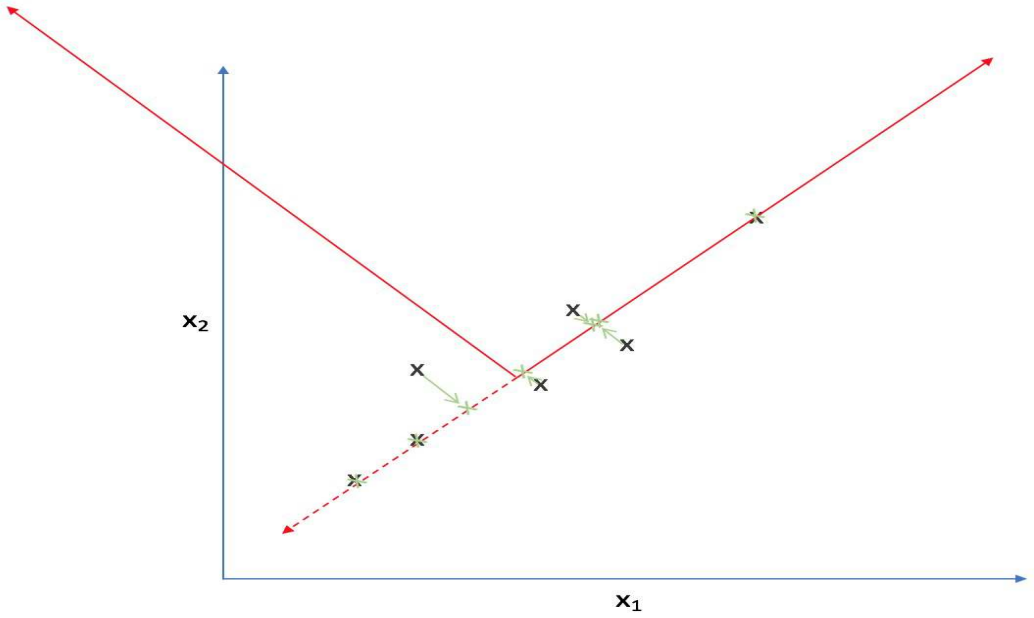
It appears that most of the points on this scatterplot lie along the following line. This suggests some degree of correlation between $x _1$ and $x _2$.



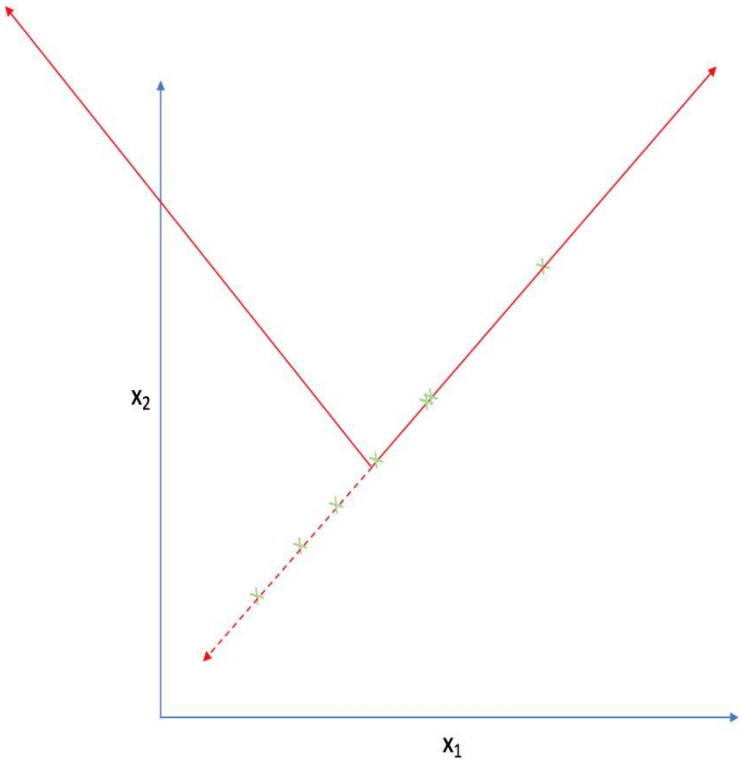
As such, we could reorient the axes to be centered on the data and parallel to the line above.



Strictly speaking, we're still dealing with two dimensions. However, let's project each observation onto the primary axis.

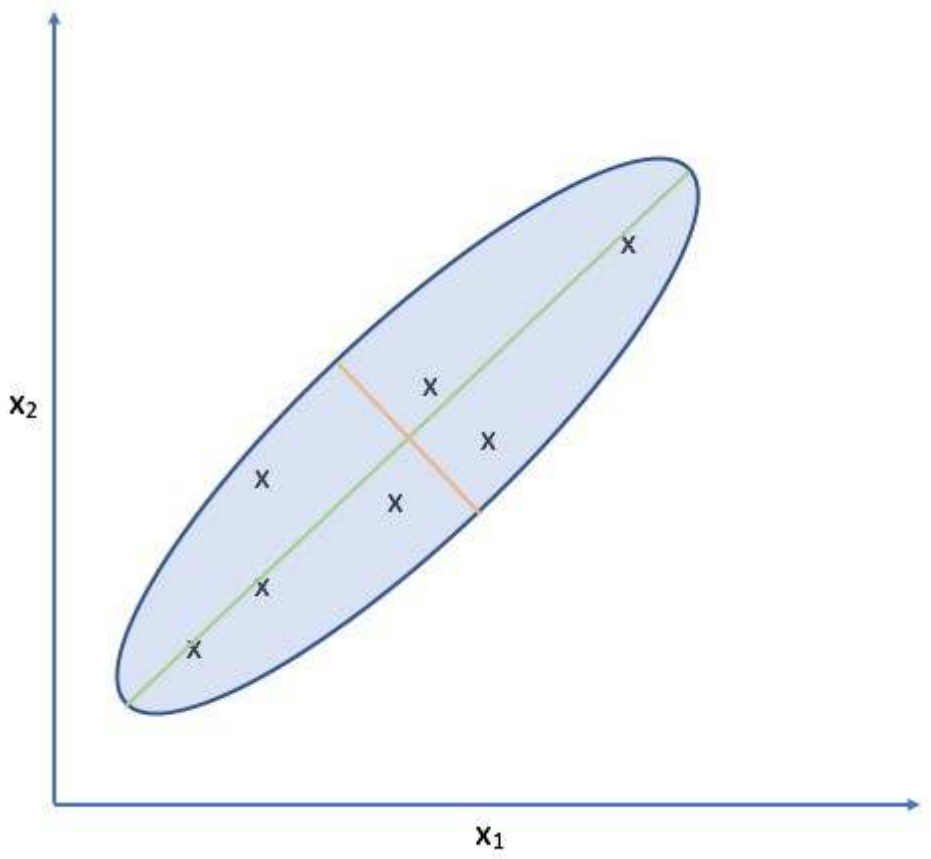


Now, every observation lies on the primary axis.

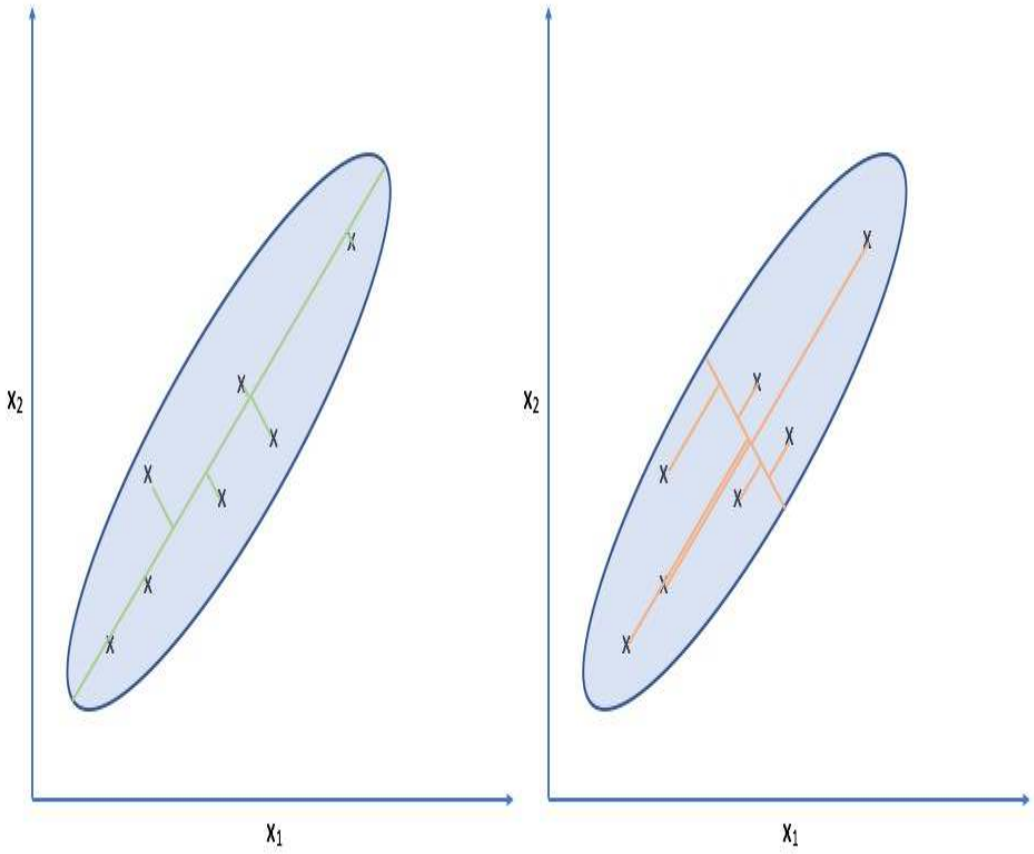


We just compressed a two dimensional dataset into one dimension by translating and rotating our axes! After this transformation, we only really have one relevant dimension and thus we can discard the second axis.

Boom. Dimensions reduced.
We can view the cumulative distance between our observations and projected points as a measure of information loss. Thus we'd like to orient the axes in a manner which minimizes this. How do we do this? That's where **variance** comes into play.
Let's take another look at the data. Specifically, look at the spread of the data along the green and orange directions. Notice that there's a much more deviation along the green direction than there is along the orange direction.

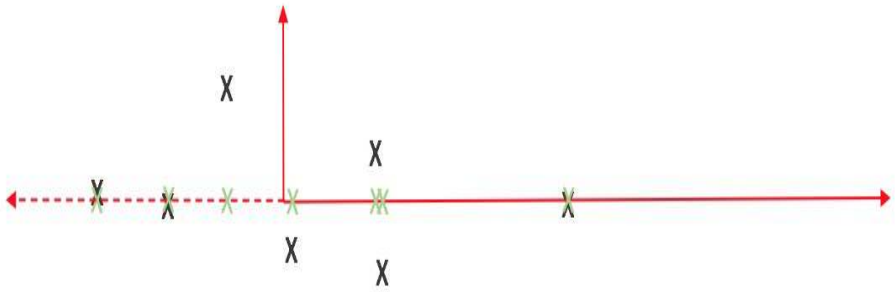


Projecting our observations onto the orange vector requires us to move much further than we would need to for projecting onto the green vector. It turns out, the vector which is capable of capturing the maximum variance of the data minimizes the distance required to move our observations as projection onto the vector.



Comparing the original observations with our new projections, we can see that it's not an exact representation of our data. However, one could argue

it does capture the essence of our data - not exact, but enough to be meaningful.



Note: While this might look like linear regression, it is distinctly different. For one, linear regression measures error as the difference between the true output and the predicted output for a given input. This error is parallel to the y-axis. For PCA, there is no y-axis; we're doing unsupervised learning. Further, we measure error as the distance between the original observation and the projected observation along the direction of projection (orthogonal to the new axis).

More generally, we'd like to find a set of k vectors on which we can project our data to reduce the feature-space from n-dimensions to k-dimensions. Further, we'd like to find these k vectors such that we minimize the projection error or information loss; we earlier established that we can accomplish this by finding the direction of maximum variance within the data.

Algorithm

So far, we've established a method for reducing the dimensionality by finding the set of orthogonal vectors (through translation and rotation) which captures the maximum amount of variance within the data. Now,

we'll discuss a way to achieve this algorithmically. This algorithm must contain a way to 1) find a new set of vectors to describe the data and 2) develop a way to project our original observations onto these vectors.

Before we transform our feature-space into such a representation, it's important that we normalize and scale the features. This allows variance to be a homogenous measure across all features.

The steps to perform PCA are as follows.

1. Compute the covariance matrix.

$$\Sigma=[\ \mathrm{cov}(x1,x1)\ \mathrm{cov}(x1,x2)\cdots\mathrm{cov}(x1,xn)\ \mathrm{cov}(x2,x1)\ \mathrm{cov}(x2,x2)\cdots\mathrm{cov}(x2,xn)\vdots\vdots\ddots\vdots\mathrm{cov}(xn,x1)\ \mathrm{cov}(xn,x2)\cdots\mathrm{cov}(xn,xn)\ ]$$

2. Find eigenvectors (U) and eigenvalues (S) of the covariance matrix using singular value decomposition.

$$[\ {\left[\ {U,S,V}\ \right]}\ =\ \mathrm{svd}(\Sigma\ )]$$

3. Select k first columns from eigenvector matrix.

$$U=[\ |\ |\ |\ |\ |\ |\ u1\ u2\cdots uk\cdots un\ |\ |\ |\ |\ |\ |\ ]$$
$$U\ \mathrm{reduce}=[\ |\ |\ |\ |\ u1\ u2\cdots uk\ |\ |\ |\ |\ ]$$

4. Compute projections of original observation onto new vector form.

[z = U _{reduce}^Tx]

It's also possible to decompress z and restore an approximation of the original feature-space.

[{x _{approx}} = {U _{reduce}}z]

We can compare this approximation to the original value in order to calculate the variance retained in our compressed feature-space. A common metric to evaluate our PCA feature compression is **explained variance**, which is another way of saying the amount of original variance by which our compressed representation is still able to retain. (Recall that we lost some variance of the data when projecting the observations onto the new vector in the 2D example.)

For example, if we were to compress our feature-space in a manner that retains 95% variance of the data, we could write:

[1 - \frac{{\frac{1}{m}\sum\limits _{i = 1}^m {{{\left| {{x _i} - {x _{i,approx}}} \right|}^2}} }}{{\frac{1}{m}\sum\limits _{i = 1}^m {{{\left| {{x _i}} \right|}^2}} }} \ge 0.95]

hoosing k

One method for choosing k would be to start at $k=1$, and continue increasing k until we drop below some set threshold of explained variance. This would involve calculating the explained variance for each iteration and is computationally inefficient.

A better approach to calculating k is to leverage the eigenvalues (S) we found in singular value decomposition. While the eigenvectors represent the directions of maximum variance, the eigenvalues represent the magnitudes of variance in each corresponding direction.

S=[s 11 0 0 0 0 s 22 0 0 0 0 ⋱ 0 0 0 0 s nn]

This is quite convenient and it is much easier to use these magnitudes in calculating the explained variance for a given k.

1 - \frac{{\frac{1}{m}\sum\limits _{i = 1}^m {{{\left| {{x _i} - {x _{i,approx}}} \right|}^2}} }}{{\frac{1}{m}\sum\limits _{i = 1}^m {{{\left| {{x _i}} \right|}^2}} }} = \frac{{\sum\limits _{i = 1}^k {{s _{ii}}} }}{{\sum\limits _{i = 1}^n {{s _{ii}}} }}]

The problem of overfitting model assessment.

Overfitting a model is a condition where a statistical model begins to describe the random error in the data rather than the relationships between variables. This problem occurs when the model is too complex.
In regression analysis, overfitting can produce misleading R-squared values, regression coefficients, and p-values. In this post, I explain how overfitting models is a problem and how you can identify and avoid it.

Overfit regression models have too many terms for the number of observations. When this occurs, the regression coefficients represent the noise rather than the genuine relationships in the population.

That's problematic by itself. However, there is another problem.
Each sample has its own unique quirks. Consequently, a regression model that becomes tailor-made to fit the random quirks of one sample is unlikely to fit the random quirks of another sample. Thus, overfitting a regression model reduces its generalizability outside the original dataset.

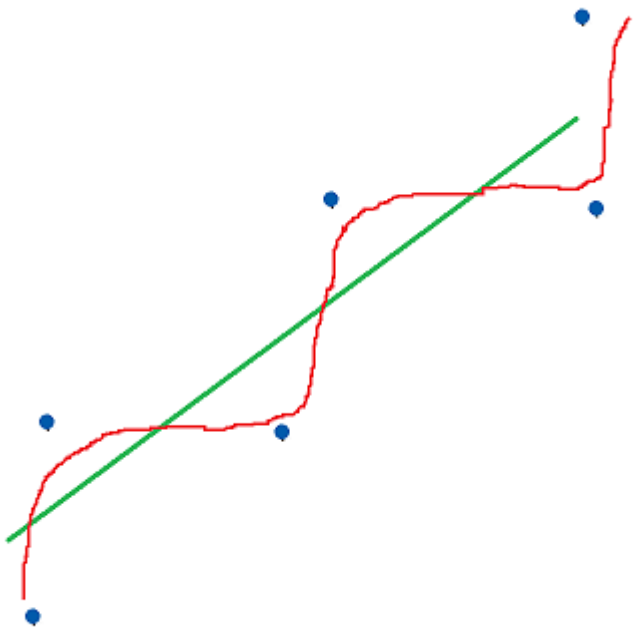
Taking the above in combination, an overfit regression model describes the noise, and it's not applicable outside the sample. That's not very helpful, right? I'd really like these problems to sink in because overfitting often occurs when analysts chase a high R-squared. In fact, inflated R-squared values are a *symptom* of overfit models! Despite the misleading

results, it can be difficult for analysts to give up that nice high R-squared value.

When choosing a regression model, our goal is to approximate the true model for the whole population. If we accomplish this goal, our model should fit most random samples drawn from that population. In other words, our results are more generalizable—we can expect that the model will fit other samples.

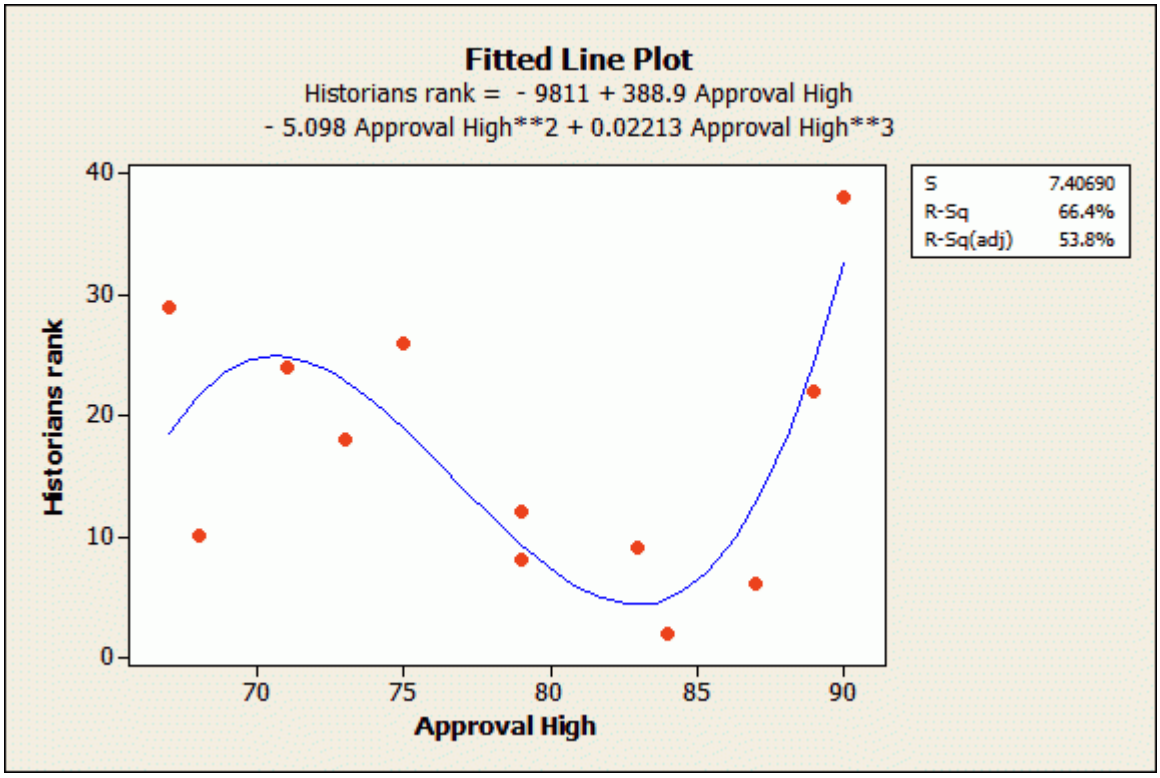
Graphical Illustration of Overfitting Regression Models

The image below illustrates an overfit model. The green line represents the true relationship between the variables. The random error inherent in the data causes the data points to fall randomly around the green fit line. The red line represents an overfit model. This model is too complex, and it attempts to explain the random error present in the data.



The example above is very clear. However, it's not always that obvious. Below, the fitted line plot shows an overfit model. In the graph, it appears that the model explains a good proportion of the dependent

variable variance. Unfortunately, this is an overfit model, and I'll show you how to detect it shortly.



If you have more than two independent variables, it's not possible to graph them in this manner, which makes it harder to detect.

How Overfitting a Model Causes these Problems

Let's go back to the basics of inferential statistics to understand how overfitting models causes problems. You use inferential statistics to draw conclusions about a population from a random sample. An important consideration is that the sample size limits the quantity and quality of the conclusions you can draw about a population. The more you need to learn, the larger the sample must be.

This concept is fairly intuitive. Suppose we have a total sample size of 20 and we need to estimate one population mean using a 1-sample t-test. We'll probably obtain a good estimate. However, if we want to use a 2-sample t-test to estimate the means of two populations, it's not as good

because we have only ten observations to estimate each mean. If we want to estimate three or more means using one-way ANOVA, it becomes pretty bad.

As the number of observations per estimate decreases (20, 10, 6.7, etc.), the estimatesbecome more erratic. Furthermore, a new sample is unlikely to replicate the inconsistent estimates produced by the smaller sample sizes.

In short, the quality of the estimates deteriorates as you draw more conclusions from a sample. This idea is directly related to the degrees of freedom in the analysis. To learn more about this concept, read my post: Degrees of Freedom in Statistics.

Applying These Concepts to Overfitting Regression Models

Overfitting a regression model is similar to the example above. The problems occur when you try to estimate too many parameters from the sample. Each term in the model forces the regression analysis to estimate a parameter using a fixed sample size. Therefore, the size of your sample restricts the number of terms that you can safely add to the model before you obtain erratic estimates.

Similar to the example with the means, you need a sufficient number of observations for each term in the regression model to help ensure trustworthy results. Statisticianshave conducted simulation studies* which indicate you should have at least 10-15 observations for each term in a linear model. The number of terms in a model is the sum of all the independent variables, their interactions, and polynomial terms to model curvature.

For instance, if the regression model has two independent variables and their interaction term, you have three terms and need 30-45 observations. Although, if the model has multicollinearity or if the effect size is small, you might need more observations.

To obtain reliable results, you need a sample size that is large enough to handle the model complexity that your study requires. If your study calls for a complex model, you must collect a relatively large sample size. If the sample is too small, you can't dependably fit a model that approaches the true model for your independent variable. In that case, the results can be misleading.

How to Detect Overfit Models

As I discussed earlier, generalizability suffers in an overfit model. Consequently, you can detect overfitting by determining whether your model fits new data as well as it fits the data used to estimate the model. In statistics, we call this cross-validation, and it often involves partitioning your data.

However, for linear regression, there is an excellent accelerated cross-validation method called predicted R-squared. This method doesn't require you to collect a separate sample or partition your data, and you can obtain the cross-validated results as you fit the model. Statistical software calculates predicted R-squared using the following automated procedure:

- It removes a data point from the dataset.
- Calculates the regression equation.
- Evaluates how well the model predicts the missing observation.
- And, repeats this for all data points in the dataset.

Predicted R-squared has several cool features. First, you can just include it in the output as you fit the model without any extra steps on your part. Second, it's easy to interpret. You simply compare predicted R-squared to the regular R-squared and see if there is a big difference.

If there is a large discrepancy between the two values, your model doesn't predict new observations as well as it fits the original dataset. The results are not generalizable, and there's a good chance you're overfitting the model.

For the fitted line plot above, the model produces a predicted R-squared (not shown) of 0%, which reveals the overfitting. For more information, read my post about how to interpret predicted R-squared, which also covers the model in the fitted line plot in more detail.

How to Avoid Overfitting Models

To avoid overfitting a regression model, you should draw a random sample that is large enough to handle all of the terms that you expect to include in your model. This process requires that you investigate similar studies before you collect data. The goal is to identify relevant variables and terms that you are likely to include in your own model. After you get a sense of the typical complexity of models in your study area, you'll be able to estimate a good sample size.