COMP9033
DATA ANALYTICS

4/12

CLEANING AND TRANSFORMING DATA

DR. DONAGH HORGAN

DEPARTMENT OF COMPUTER SCIENCE
CORK INSTITUTE OF TECHNOLOGY

2017.02.22

Credit: Victor Powell

Overview

1. Discovering relationships:
    - Dependence and correlation.
    - Correlation and causation.
    - Quantitative measures.
    - Graphical techniques.
2. Visual cues:
    - What they are, why they're important.
    - Chart types.
    - Effects of misusing visual cues.
3. The data visualisation process:
    - The Four Pillars of Effective Visualisation.
    - Chart formatting.
    - Real world examples.

1. Cleaning data:
    - Duplicate data.
    - Missing data.
    - Outlying data.
    - Erroneous data.
    - How to deal with bad data.
2. Preprocessing data:
    - Converting categories to numbers.
    - Scaling data.
    - Feature generation.
    - Trend estimation.
3. Dimensionality reduction:
    - The curse of dimensionality.
    - Feature selection.
    - Principal component analysis.

Cleaning data

- Typically, when we analyse a data sample, we will want to detect and deal with ”bad” data points, *e.g.*
  - Duplicate data, *e.g.* recording the same event twice.
  - Missing data, *e.g.* user login server went down for an hour.
  - Outlying data, *e.g.* a transient spike in application memory usage due to a bug that is now fixed.
  - Other bad data, *e.g.* typos, measurement errors.
- By removing or replacing these data points with ”better” values, we can ensure that the data remaining for further analysis does not contain irregularities that may lead us to inaccurate conclusions.
- Cleaning data is the third step in SEMMA and CRISP-DM (see Lecture 01).

- Duplicate data can arise in two situations:
  1. Random chance, *e.g.* two rolls of a dice produce the same result.
  2. A data collection error, *e.g.* a survey respondent's answers are accidentally entered into a database twice.
- In the first case, the duplication is natural and reflects the general situation (*e.g.* it is perfectly reasonable to roll two sixes).
- In the second case, the duplication is erroneous and may lead us to incorrectly believe that a certain outcome is more (or less) likely than it is in reality.
- Consequently, whether a given observation is an erroneous duplicate or not depends on the *context* of your data.
- If we determine that our data contains erroneous duplicates, then we *must* remove them to ensure the integrity of our later analysis.

- Missing data can arise in a number of situations, *e.g.*
  - Incomplete or interrupted data collection, *e.g.* a survey respondent choosing not to answer a particular question or a service outage.
  - Deliberate removal of data values, *e.g.* removal of erroneous values.
- In effect, missing data reduces our sample size which, in turn, means that our sample is less likely to reflect the population that it was drawn from.
- Typically, we have three choices when we encounter missing data:
  1. Do nothing.
  2. Remove the missing data.
  3. Replace the missing data with a “reasonable” value.
- There are advantages and disadvantages to each approach!

- Consider the grocery purchase data opposite, where the price of the rice is missing.
- If we decide not to remove or replace the value (*i.e.* do nothing), then we cannot compute the total cost of the groceries or the cost per unit mass of rice.

ITEM	PRICE	QUANTITY
tomatoes	€1.50	500 g
rice	-	400 g
beans	€2.20	240 g

- However, if we choose to remove the missing value, then we must either remove *all* of the data in the rice row or *all* of the data in the price column (*i.e.* we lose even more data).
- If we choose to replace the missing value, then we must infer or estimate some appropriate value to substitute, but to do this requires some other source of information (*e.g.* a previous grocery bill) to draw from, which may not be available.

ITEM	PRICE	QUANTITY
tomatoes	€1.50	500 g
rice	-	400 g
beans	€2.20	240 g

- Typically, the right choice depends on the *context* of your analysis.
- Some analysis algorithms require that there be no missing data in the sample, in which case you must either remove or replace the missing values.
    - This is the case for temporal data in particular, where “gaps” in the data can affect further analysis steps that rely on equally spaced samples.
- Similarly, if your analysis is sensitive to sample size or your sample is small to begin with, then removing samples may not be an option.
- Finally, replacing data is hard because it’s not always clear whether there is an appropriate substitute and, if so, what it might be.

- If we choose to replace missing data with some ”better” value, then we have a few options available to us, *e.g.*
  - Replace the missing value with a measure of the central tendency of its group or category (*e.g.* mean, median).
  - If the data is ordered (*e.g.* time series), we can interpolate a new value based on the adjacent samples, *e.g.* server load at 1PM is the average of the server load at 12PM and at 2PM.
  - Build a predictive model and use it to impute missing values based on some inferred logical pattern.
- Unfortunately, there is no optimal technique for replacing missing values — again, the best method to use depends on the *context* of the problem you are trying to solve.

- If we have determined that our data contains outliers, then we have four choices:
  1. Do nothing.
  2. Remove the outlying data (*i.e.* replace outlying data with missing data).
  3. Replace the outlying data with a "reasonable" value.
  4. Adjust how we model the data, so that the outliers are accommodated.
- The first three options here are identical to our choices when dealing with missing data, and have the same pros and cons.
- However, the fourth choice is also worth considering, particularly if there are *robust* modelling algorithms available.

- Some modelling techniques are naturally resistant to the effects of outliers, *i.e.* they do not exert an extreme effect on the outcome of the modelling process, *e.g.* decision tree classification.
- Some modelling techniques which are not resistant to outliers can be adapted for them by the use of robust statistics, *e.g.* k-means clustering and k-medoids clustering.
- However, some modelling techniques are not robust to outliers (*e.g.* linear regression) and cannot be adapted, in which case we must either remove or replace the outliers or accept that the outcome of the modelling process will be biased because of them.

- Erroneous data can arise in a number of situations, *e.g.*
  - Human error, *e.g.* typos during manual data entry.
  - Malfunctioning equipment, *e.g.* a faulty sensor.
  - Intentional misreporting, *e.g.* lying, fraud, sabotage.
- Due to the diverse number of causes, it's not always possible to identify erroneous data.
- However, if we can detect it, we *must* address it: drawing conclusions from bad data *will* lead us to invalid solutions.
- If we can detect it, we have two choices for dealing with it:
  1. Remove the bad data (*i.e.* replace bad data with missing data).
  2. Replace the bad data with a "reasonable" value.
- Again, there are advantages and disadvantages to each approach!

Preprocessing data

- When analysing data, we often want to convert it into a different form, *e.g.* to make it more suitable for processing by a particular algorithm.
- This is known as *preprocessing* data and the variables that we process are often referred to as *features*.
- Examples of preprocessing techniques:
  - Encoding category labels (*e.g.* Dog, Cat, Mouse) as algorithm-friendly numerical values.
  - Centering data so that it has zero mean.
  - Scaling data so that it fits in a certain range.
  - Generating new data, based on existing data (*e.g.* polynomial features).
  - Applying other mathematical functions to data.
- Preprocessing is part of the third stage of both SEMMA and CRISP-DM (see Lecture 01).

- In analysing many datasets, we will want to use category labels as inputs to further analysis steps, *e.g.* machine learning algorithms.
- However, mathematical algorithms are not designed to interpret the text we label our categories with.
- As a result, we must encode these labels as numerical values, which can then be processed by the algorithms.
- For example, we might have a dataset containing a gender feature, “M” or “F”:
  - To make these labels more easily understood in later stages of our analysis, we can substitute numerical values for the text labels.
  - For instance, we could assign 0 for every instance of “M” and 1 for every instance of “F”.

- Making simple substitutions for category labels is a quick fix, but is often not enough.
- In many instances, algorithms won't interpret the substituted values in the way we interpret them as humans.
- In the table on the right, the categorical gender labels have been substituted as described on the last slide.
- However, an algorithm that relied on the average value of the column ($\frac{2}{3}$) would not understand what it meant.

	AGE	GENDER
Alice	36	1
Bob	58	0
Carol	22	1

- Instead, we can encode the gender variables using *one hot encoding*:
  - This encodes the two-valued gender category as a pair of binary attributes, each of which represents the state of a particular condition.
  - Now, the features make sense: on average, two thirds of the set are female and one third is male.
- We can apply one hot encoding to arbitrarily large sets of categories, but in doing so we create K features for each category of K labels that we process.

	AGE	IS MALE	IS FEMALE
Alice	36	0	1
Bob	58	1	0
Carol	22	0	1

- Scaling data involves adjusting the ranges of the variables in the dataset to make them comparable in some fashion.
- For instance, the film Taxi Driver has a rating of 99% on Rotten Tomatoes and a rating of 8.3/10 on IMDB:
  - Both of these scores are meaningful, but we need scaling in order to make logical comparisons between them.
  - We would need to scale again if we wanted to see the scores in a stars-out-of-five system.
- Some algorithms require that the input data be scaled, while others run faster on scaled inputs than on unscaled inputs.

- Standardising data is a form of scaling that uses the standard score to adjust the values of individual data points.
- In Lecture 02, the standard score of the data point x_i is denoted by $z(x_i)$ and defined as

$$z(x_i) = \frac{x_i - \bar{x}}{\sigma_x}. \tag{4.1}$$

- If we have a set of data $X = \{x_1, x_2, \ldots, x_n\}$, then we can standardise it by computing the standard score of each element in the set, *i.e.* $X' = \{z(x_1), z(x_2), \ldots, z(x_n)\}$ and using X' instead of X in any further analysis.
- Because it has been standardised, the new data sample X' has zero mean ($\bar{x}' = 0$) and unit standard deviation ($\sigma_{x'} = 1$).
- For this reason, standardisation is also often referred to as *normalisation*.

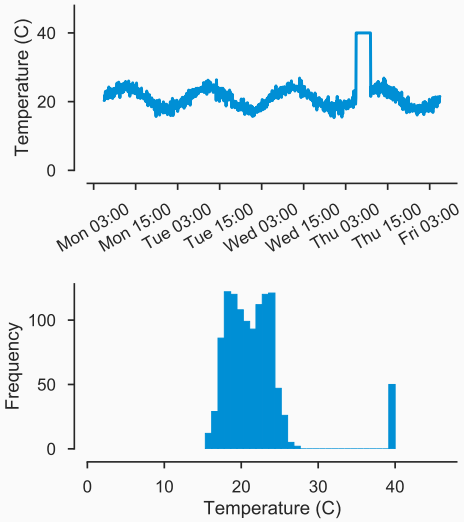
- Min/max scaling is an alternative scaling technique that scales features so that all of their values lie in the range $[0, 1]$.
- Min/max scaling works similarly to standardisation, but transforms each data point according to

$$x_i' = \frac{x_i - \min(X)}{\max(X) - \min(X)}. \tag{4.2}$$

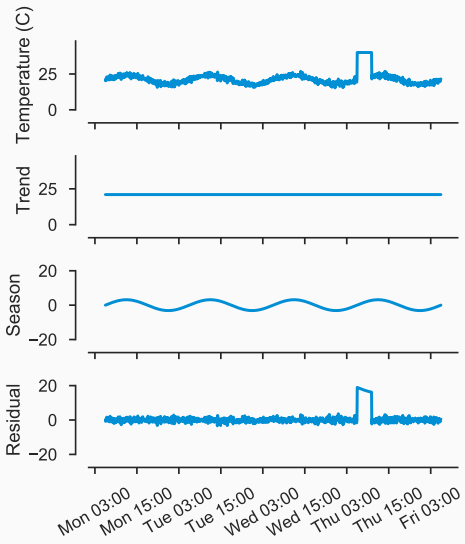
- Some machine learning techniques (*e.g.* neural networks) require that the input data be scaled to $[0, 1]$.

- Polynomial feature generation is a technique for generating new data from existing data.
- For instance, given the dataset, $X = \{x_1, x_2\}$, we could construct the features x_1x_2, x_1^2 and x_2^2:
  - These new features can replace or supplement their source features.
  - This can be particularly useful when there's a non-linear relationship between the features and the target.
- Polynomial feature generation is easy to automate, but it can increase the amount of data to be analysed:
  - The newly generated features do not necessarily add value — they could be redundant, in which case we've just made the problem harder to solve.
  - However, we can use feature selection to determine the best subset to use or techniques like principal component analysis to condense the features.

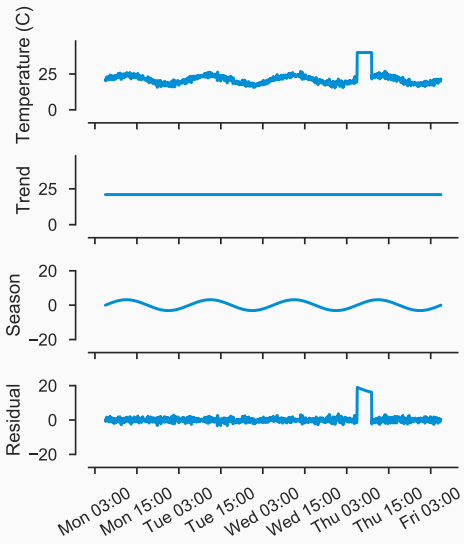
- Seasonal trend removal is a commonly used preprocessing technique in time series analysis.
- The key idea is that, if there is some regular behaviour in the data, this can be estimated and removed.
- For instance, in the sensor data above, there is a clear pattern of behaviour: the temperature oscillates regularly between a minimum and maximum value.



- Estimation techniques like the Holt Winters algorithm can be used to decompose signals into *trend*, *season* and *residual* components:
  - The trend component represents the long term behaviour of the data.
  - The seasonal component represents the regular short term behaviour of the data.
  - The residual component represents the remainder when the trend and seasonal components are removed from the original signal.



- Once the long term trend and seasonally varying components have been estimated, we can subtract them from the original data sample to produce the residual.
- The residual then represents the data that could not be estimated, *i.e.* random variations from moment to moment or other behaviours which do not form part of the long term or seasonal trends.



Dimensionality reduction

- When we analyse data, it often has more than just a single feature, *e.g.*
  - Weather data might describe pressure, temperature and rainfall measurements.
  - A JVM application may be described by key performance indicators, such as heap usage, CPU usage, disk I/O and network throughput.
  - Credit application records might contain information about gender, age, marital status, salary range and whether the applicant is a home owner or not.
- The number of features in a dataset is also known as its *dimensionality*, *e.g.*
  - The weather forecast described above has *three* features/dimensions.
  - The JVM application metrics above represent *four* features/dimensions.
  - The credit application described above has *five* features/dimensions.
- While the inclusion of additional features can provide us with extra information, it can also negatively affect the outcome of our analysis, a phenomenon known as *the curse of dimensionality*.

- If the number of dimensions increases and the sample count remains constant, then the sample density in a given neighbourhood (*i.e.* in a given multidimensional space) will decrease:
  - A sample that was representative of its population with the lower number of features may no longer be representative with the higher number of features.
- Distance metrics (*e.g.* Euclidean, Manhattan) become ineffective:
  - As the number of dimensions grows, the difference in the distances between any two pairs of points decreases.
  - Consequently, using the distance between points as a measure of similarity (*e.g.* k-nearest neighbours, clustering) becomes ineffective when the number of dimensions becomes larger.
- Algorithms take significantly (perhaps prohibitively) longer to run:
  - Algorithms whose run time depends on the number of features (*e.g.* brute force combinatorial clustering) can take significantly longer to run as it grows larger.

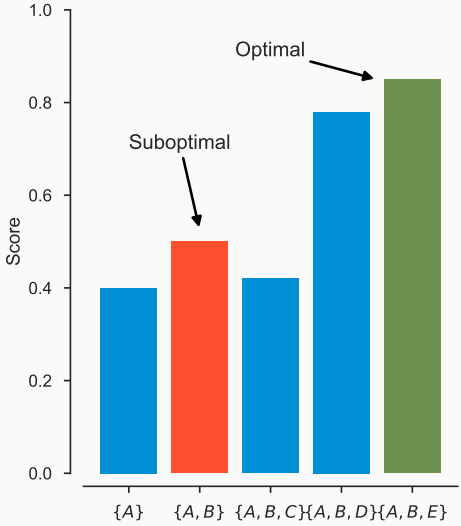
- The curse of dimensionality can be mitigated using *dimensionality reduction*, a term that describes techniques that can summarise data of a given dimension with data of a lower dimension.
- The key idea is to remove as many redundant features from the data as possible, leaving only those which contribute the most value.
- Dimensionality reduction can also be useful when we are dealing data that may contain redundant information, *e.g.* a data set where the underlying relationship we are interested in is actually quite simple with respect to the number of features we've collected.

- Feature selection is a commonly used dimensionality reduction technique.
- The idea is to find a subset of the features of the input data that can produce a model that is as good as, or better than, one that could be generated using all the input data.
- When selecting features, we typically have two options:
  1. Conduct a brute force search: can produce an optimum result, but is often computationally demanding.
  2. Take a heuristic approach: will not produce an optimum result, but can be quite close, depending on the technique.
- If an exhaustive search is not feasible, a heuristic method *must* be used instead.

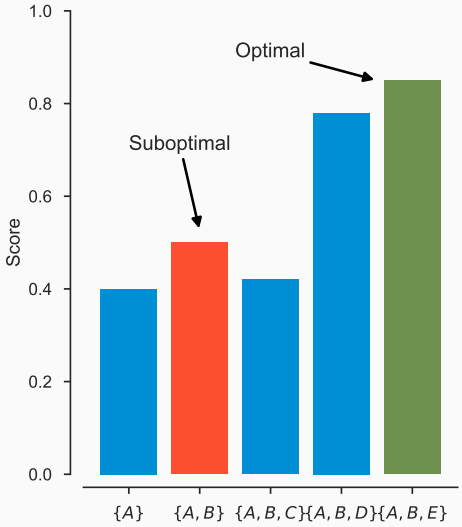
1. Filters:
   - These select inputs according to some criteria, *e.g.* Pearson correlation.
   - Filters are generally faster than wrappers, although they are not “tuned” to the learning algorithm to be used and so can give poor results in some cases.
2. Wrappers:
   - These involve the use of learning algorithms to find the best inputs, *e.g.* using decision trees to rank features by their importance.
   - Wrappers can be computationally intensive, although some support the use of “greedy” search strategies, which can reduce the computational requirements.
3. Embedded methods:
   - These select features during the training process, *e.g.* subset selection in linear regression.
   - Embedded methods are specific to the algorithm in use and not every algorithm supports embedded feature selection.

- *Hill climbing* is a generally applicable, wrapper-type feature selection technique:
  1. Using a single feature or minimal set of features, build a model and measure its accuracy.
  2. Add a new feature, construct a new model and measure its accuracy.
  3. If the accuracy of the new model is better than the previous one, then repeat Step 2; otherwise, stop.
- While hill climbing can be an effective dimensionality reduction technique, it requires us to choose an initial feature or set of features:
  - If we have further information about which feature(s) are most relevant in the data, then we may be able to make a good choice.
  - If we don't have any insights, then we might make a bad choice, which may lead to a bad model being produced.
- Hill climbing is also prone to stopping early, *i.e.* on a set of features that is suboptimal.

- When building a model, we have five features to choose from: A, B, C, D and E.
- From domain knowledge, we know that A is an important feature, while the others may not be.
- We can use hill climbing to decide on which features to include!



- Initially, we build a model using A only and measure its accuracy.
- Next, we include B, build a new model and measure its accuracy.
- As the accuracy has improved, we include C and repeat.
- However, the accuracy now decreases and so hill climbing stops, *i.e.* it finds the local optimum $\{A, B\}$, not the global optimum $\{A, B, E\}$.



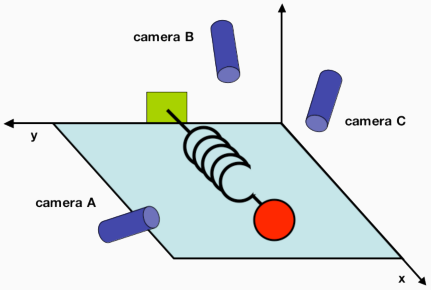
- *Principal component analysis* (PCA) is a dimensionality reduction technique that reduces data to a set of principal components, which represent the most important aspects of the data.
- If a large proportion of the data is redundant (*e.g.* there is significant correlation between one or more input features), then the number of principal components can be significantly smaller than the number of features.
- The principal components can then be input to machine learning algorithms in place of the original features of the data.

- One advantage of PCA is that it produces the *optimal* reduction of features (in a mean squared error sense).
- Additionally, because it is a non-parametric technique, there is no tweaking/tuning involved.
- However, PCA *only* works when the following conditions hold:
  1. The principal components are a linear function of the inputs.
  2. The principal components with the largest variance are the most important.
  3. Principal components are orthogonal to one another (*i.e.* they are perpendicular).

- Suppose we want to understand spring displacement by conducting an experiment, collecting data and analysing the results.
- The equation we want to discover is known as Hooke's Law, which states that the force experienced by the spring, F, is proportional to its displacement along the x axis, *i.e.*

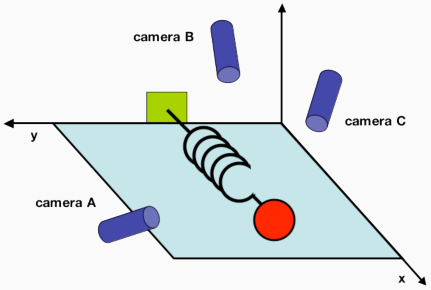
$$F = kx.$$

- Let's assume that we don't have any knowledge of this and try to use PCA to figure it out.

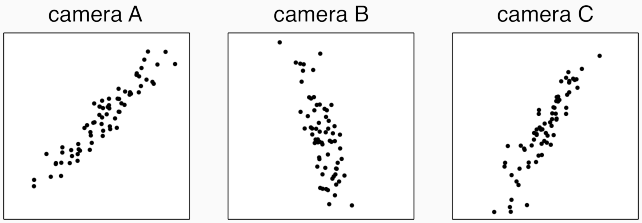


Credit: Jonathon Shlens

- We place three cameras at arbitrary positions to record the movement of the spring during the experiment.
- The positions of the cameras are not necessarily aligned to what we might consider to be the *x*, *y* and *z* axes.
- Also, the data we collect may be affected by noise, *e.g.*
  - Air currents.
  - Friction.
  - Imperfections in the camera lenses and recording technology.



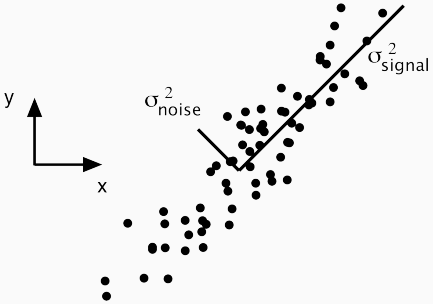
Credit: Jonathon Shlens



- Each camera collects data in two dimensions.
- Therefore, we have six different input features to work with: two axis measurements from each of the three cameras.
- However, the problem can be solved using just one feature: the data in the true x direction.
- We can use PCA to remove the redundancy in our inputs!

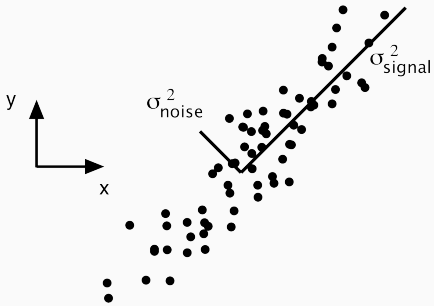
Credit: Jonathon Shlens

- The basic procedure works as follows:
  1. Rotate the input features until you have find the direction along which their variance is maximised.
  2. Set this direction as the first principal component.
  3. For each orthogonal (*i.e.* perpendicular) direction to the first component, repeat Step 1 and add this direction as the next principal component.
  4. Repeat Step 3 until we reach a desired number of principal components or we have run out of inputs.



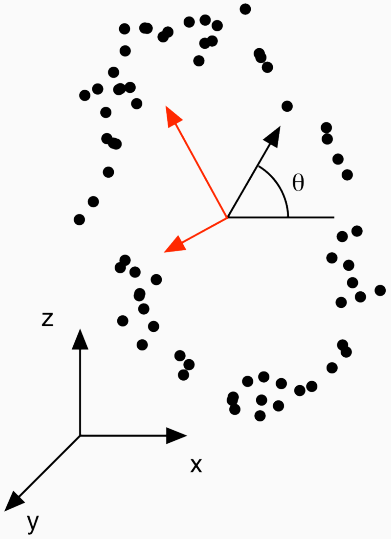
Credit: Jonathon Shlens

- In the image to the left, the first principal component is given by σ^2_{signal}, as this is the direction along which the variance of the data is maximised.
- As we only have two features in this example, we just rotate 90° from the angle of the first principal component to reach the second principal component, σ^2_{noise}.
- We can now discard the original components (x and y) and use the principal components in their place.



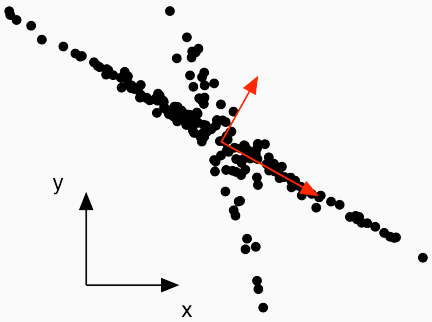
Credit: Jonathon Shlens

- PCA assumes that the relationship between the principal components and the target variable is linear, *e.g.* $y = mx + c$.
- The figure to the right shows measurements of a person's position on a Ferris wheel:
  - The true location of the person is given by the angle, θ, which is a non-linear combination of the x, y and z variables.
  - If we apply PCA to reduce measurements in x, y and z to a single dimension, it will fail.
- Kernel PCA can be used in situations where non-linear relationships exist.



Credit: Jonathon Shlens

- PCA also assumes that the principal components are orthogonal.
- The figure to the right shows measurements along two non-orthogonal axes:
  - While the relationships between the principal components and the inputs are linear, the components themselves are not orthogonal.
  - If we apply PCA to transform the *x* and *y* measurements into principal components, it will fail.
- Independent component analysis can be used to find components that are not orthogonal.



Credit: Jonathon Shlens

Summary

- Lots covered again this week:
  - Cleaning data: duplicates, outliers and errors — remove or replace?
  - Preprocessing: one hot encoding, scaling, feature generation, trend estimation.
  - Dimensionality reduction: feature selection and principal component analysis.
- This week's lab covers how to:
  1. Customise histograms.
  2. Create boxplots.
  3. Remove data.
  4. Replace data.
- Next week: data modelling and assessment!

1. Powell, Victor. *CSV Fingerprints.* (bit.ly/2kydRkf)
2. Hastie et al. *The elements of statistical learning: data mining, inference and prediction.* 2nd edition, February 2009. (stanford.io/1dLkiAv)
3. Shlens, Jonathon. *A tutorial on principal component analysis.* 2014. (bit.ly/2kGjw7R)