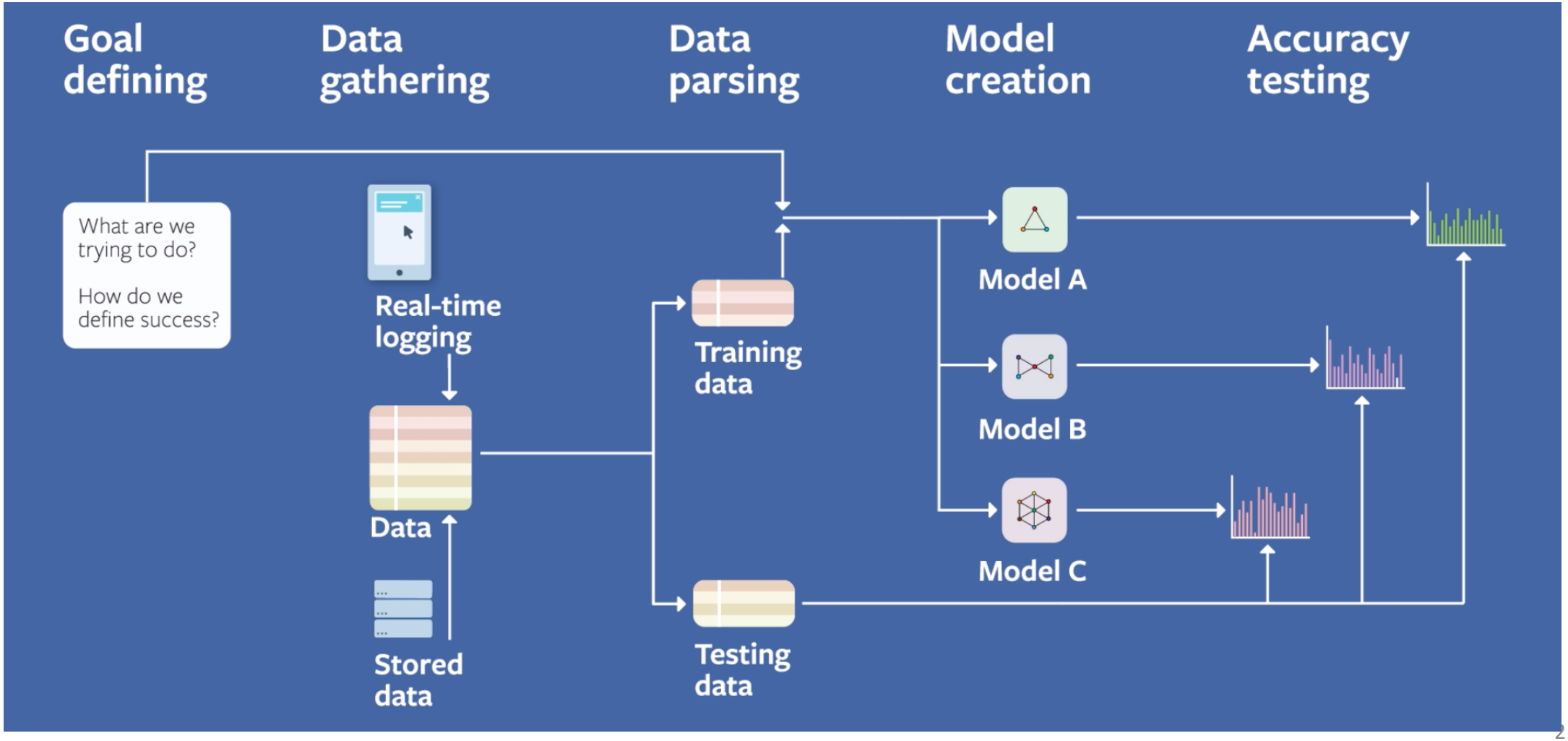
Universitat
de les Illes Balears

Machine Learning

Lesson 5: Feature engineering

Task specification (business problem)




THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Task specification (business problem)

1. **Which Data?**
2. Which Features?
3. Which Model?
4. Which Predictions?
5. How does this Solve Task (Improve Business)?

A simple example…

Size (feet2)	*Number of bedrooms*	*Number of floors*	*Age of home (years)*	*Price ($1000)*
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

… for dataset notation

Dataset = $\{(\mathbf{x}^{(i)}, y^{(i)}); i = 1, \ldots m\}$

$(\mathbf{x}^{(i)}, y^{(i)})$ = *Training example*

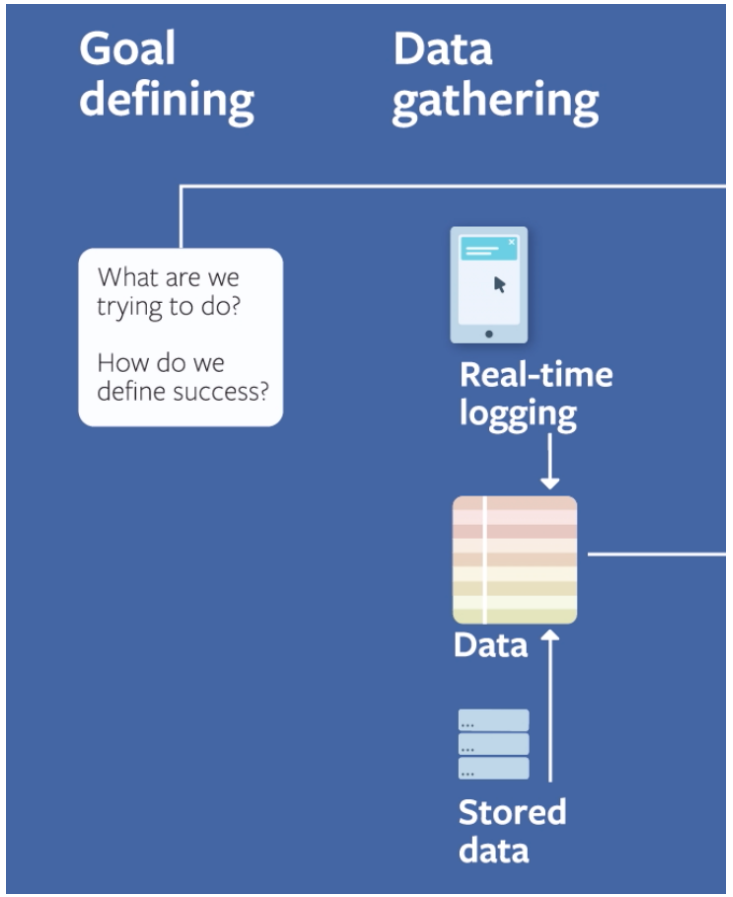
$\mathbf{x}^{(i)}$= *"input" variable (features),* $\mathbf{x}^{(i)} = \left(x_1^{(i)}, x_2^{(i)}, \cdots, x_n^{(i)}\right)$

$y^{(i)}$ = *"output" variable (target),* $y \in \mathcal{Y}$

Building datasets

Preparing the datasets is a **core** part of a machine learning engineer's job. It's an **active** not passive **part** of machine learning research and is one of the most **powerful** variables to create **high-quality** machine learning systems.

Garbage in → Garbage out



Building datasets

Example: Predicting the click on an ad (impression)

Impressions

User ID	Impression ID	Post ID
9123KX	vxyen	wpHgY
6725UT	eoysk	mPmIH
9473II	zokcq	mHRwX
1643OK	ucidx	TOfBX
9472KM	xzuxl	zJwBz
1253CC	wmbhi	uczpd

×

Clicks

Impression ID
vxyen
zokcq
ucidx
wmbhi

×

Features

User ID	Country	Category
9123KX	Spain	Shoes
6725UT	Nigeria	Hats
9473II	France	Glasses
1643OK	Canada	Scarves
9472KM	Botswana	Socks
1253CC	Argentina	Pants

▼

Impression ID	User ID	Post ID	Clicks	Country	Category
vxyen	9123KX	wpHgY	1	Spain	Shoes
eoysk	6725UT	mPmIH	0	Nigeria	Hats
zokcq	9473II	mHRwX	1	France	Glasses
ucidx	1643OK	TOfBX	1	Canada	Scarves
xzuxl	9472KM	zJwBz	0	Botswana	Socks
wmbhi	1253CC	uczpd	1	Argentina	Pants

Building datasets: key concepts

1. Data recency / real-time training
2. Training/prediction (off-line/on-line) consistency
3. Record and sampling

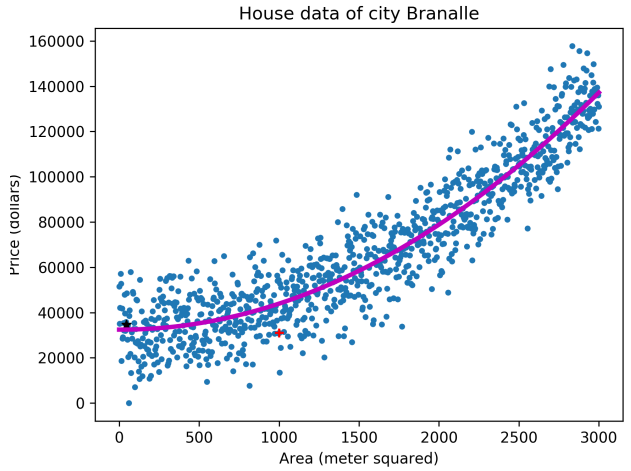
> "More data beats clever algorithms, but better data beats more data."
>
> — Peter Norvig

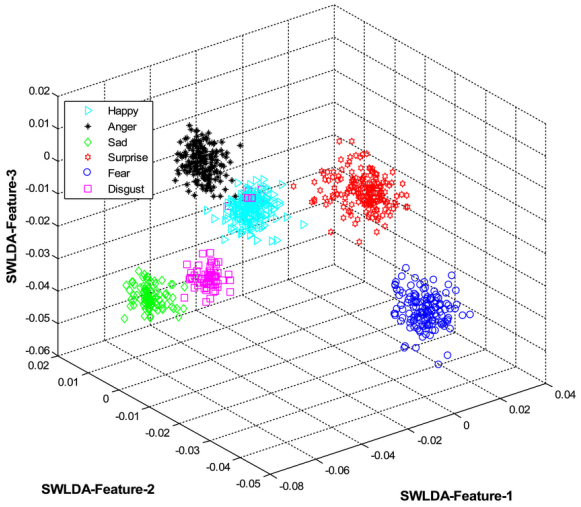
Task specification (business problem)

1. Which Data?
2. **Which Features?**
3. Which Model?
4. Which Predictions?
5. How does this Solve Task (Improve Business)?

Data graphical representation

prediction



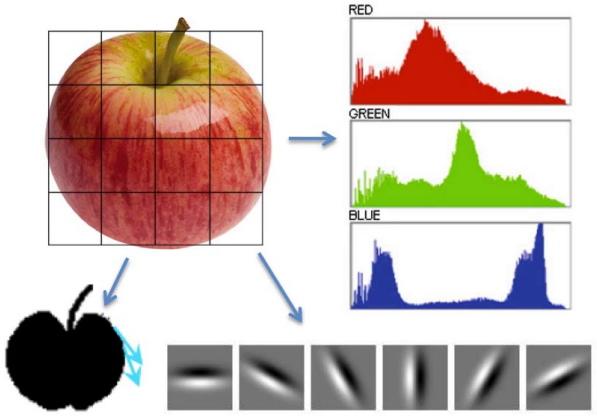


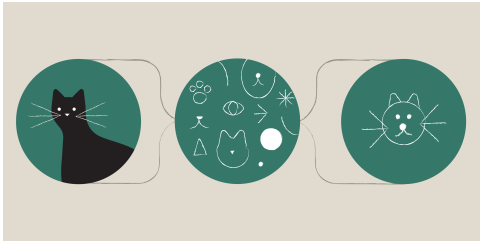
features

Feature engineering

Building better features is the second most important way to **impact machine learning performance** (after data), and **better features** are more important than **model improvements**.

Features: language in which we describe the properties of objects in a domain; You have to choose them, pre-process them, manipulate them ... (data mining).

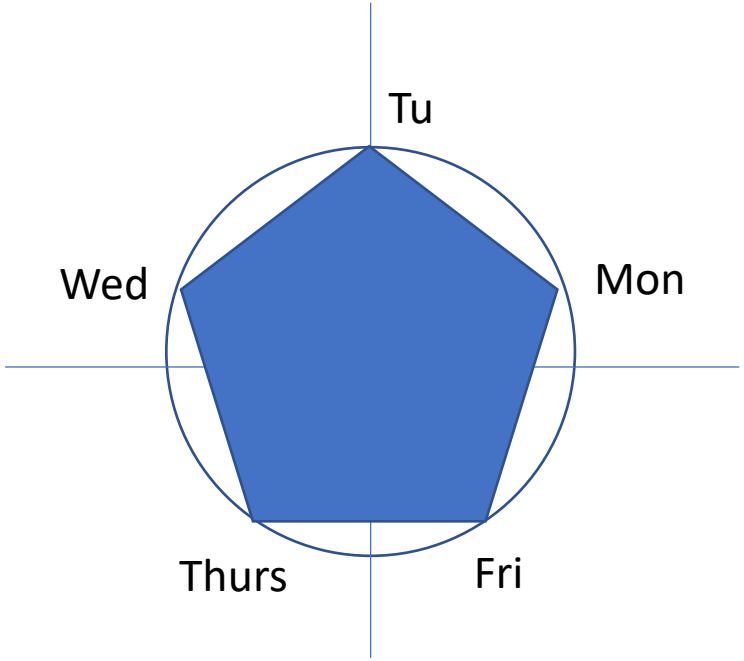




Feature engineering

Most creative aspect of Data Science !!

- Brainstorming
- Business acumen
- Trial and error



Coming up with features is difficult, time-consuming, requires expert knowledge. "Applied machine learning" is basically feature engineering. — Andrew Ng

Feature engineering

Categorical Features

How to numerically represent a category ?

- Almost always need treatment (e.g. CATBOOST)
- Can deal to high cardinality (Curse of dimensionality !)
- How to impute missing values ?

Feature engineering

Categorical Features

- OneHotEncoding → For a few categories
- HashEncoding → For large number of categories
- MeanEncoding → Easy to implement / prone to overfitting

One Hot Encoding

OneHotEncoding

- Use N features
- Represent each category as a flag like (present/not present)

Original Value	Dummy Variables					
	Mon	Tues	Wed	Thurs	Fri	Sat
Sun	0	0	0	0	0	0
Mon	1	0	0	0	0	0
Tues	0	1	0	0	0	0
Wed	0	0	1	0	0	0
Thurs	0	0	0	1	0	0
Fri	0	0	0	0	1	0
Sat	0	0	0	0	0	1

If there are N categories, what happens when N becomes very large?

- ZIP codes, names, ip, …
- OHE: Unfeasible for variables with more than >10k categories
- Hash trick →use hash to map any str to int and use it as feature
- Risk of collisions

	Hash Integer Value
belvedere tiburon	582753783
berkeley	1166288024
martinez	-157684639
mountain view	-1267876914
san leandro	1219729949
san mateo	986716290
south san francisco	-373608504

HashEncoding

	Hash Integer Value
belvedere tiburon	582753783
berkeley	1166288024
martinez	-157684639
mountain view	-1267876914
san leandro	1219729949
san mateo	986716290
south san francisco	-373608504

1166288024 mod 16 + 1 = 9

- Map to hash int I
- Col = I mod n_cols + 1

Original Value	Hash Variables											
	1	2	3	4	5	6	9	12	13	14	15	16
alameda	1	0	0	0	1	0	0	0	0	0	0	0
belmont	1	0	0	0	0	0	0	0	0	0	1	0
benicia	1	0	0	1	0	0	0	0	0	0	0	0
berkeley	1	0	0	0	0	0	1	0	0	0	0	0

TargetEncoding

- Target encoding is the process of replacing a categorical value with the mean of the target variable. Any non-categorical columns are automatically dropped by the target encoder model.

Trend	Target	Trend_Encoded
Up	1	0.66
Up	1	0.66
Down	0	0.33
Flat	0	0.5
Down	1	0.33
Up	0	0.66
Down	0	0.33
Flat	0	0.5
Flat	1	0.5
Flat	1	0.5

	Target		
Trend	0	1	Probability (1)
Up	1	2	0.66
Down	2	1	0.33
Flat	2	2	0.5

The main drawbacks of this method are its dependency to the distribution of the target, and its lower predictability power compare to the binary encoding method.

Feature engineering

Numerical Features

- More readily to fed to algorithms
- Data cleaning! Check for outliers
- Easier to impute missing values
- Different features vastly on different scales
- Skewed distributions (normality assumptions ?)

Feature normalization
1 to 1 transformations

Are typically needed when the model requires the predictors to be in common units.

Feature scaling is used to bring all values into the range [0,1]. This is also called unity-based normalization. This can be generalized.

$$x_j^{(i)} = \frac{x_j^{(i)} - x_{min}}{x_{max} - x_{min}}$$

$$x_j^{(i)} = a + \frac{x_j^{(i)} - x_{min}}{x_{max} - x_{min}}(b - a)$$

Standard normalization: works well for populations that are normally distributed. Ensures that transformed variables have a mean 0 and std 0

$$x_j^{(i)} = \frac{x_j^{(i)} - \mu}{\sigma}$$

the **statistics** required for the transformation (e.g., the mean) **are estimated from the training set** and are applied to all data sets (e.g., the test set or new samples).

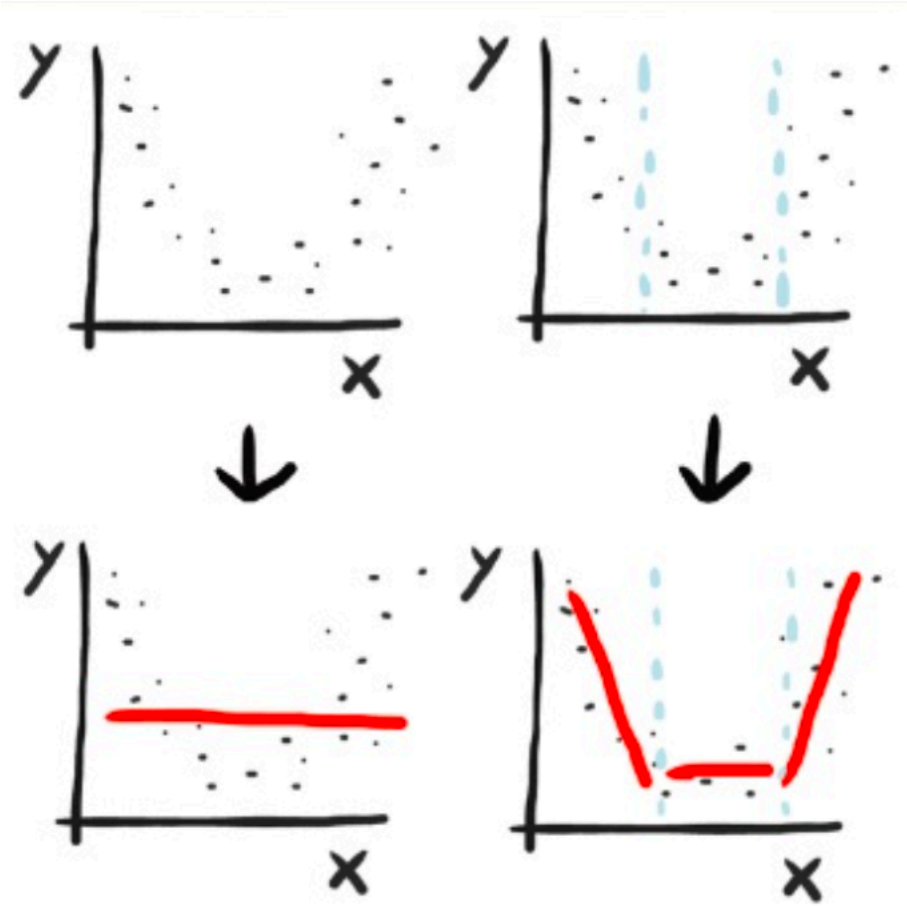
Feature normalization
1 to Many transformations

Binning

Translating a quantitative variable into a set of two or more qualitative buckets

Binning *may* avoid the problem of having to specify the relationship between the predictor and outcome.

A set of bins can be perceived as being able to model more patterns without having to visualize or think about the underlying pattern.



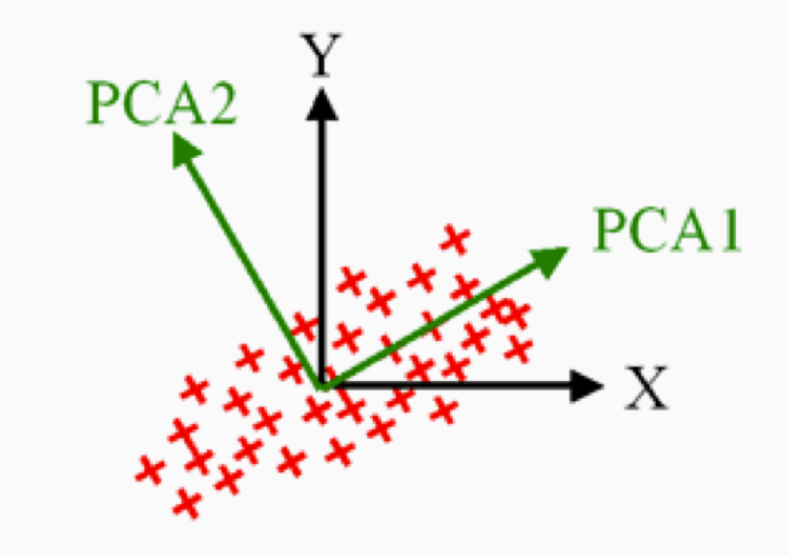
Feature normalization
Many to Many transformations

- correcting collinearity
- reduce the dimensionality of the predictor space

PCA Principal Components Analysis

Linear Projection method in the sense that they take a matrix of numeric predictor values (X) and create new components that are linear combinations of the original data (X*)

The combinations summarize the maximal amount of variation in tl predictor space



Feature selection

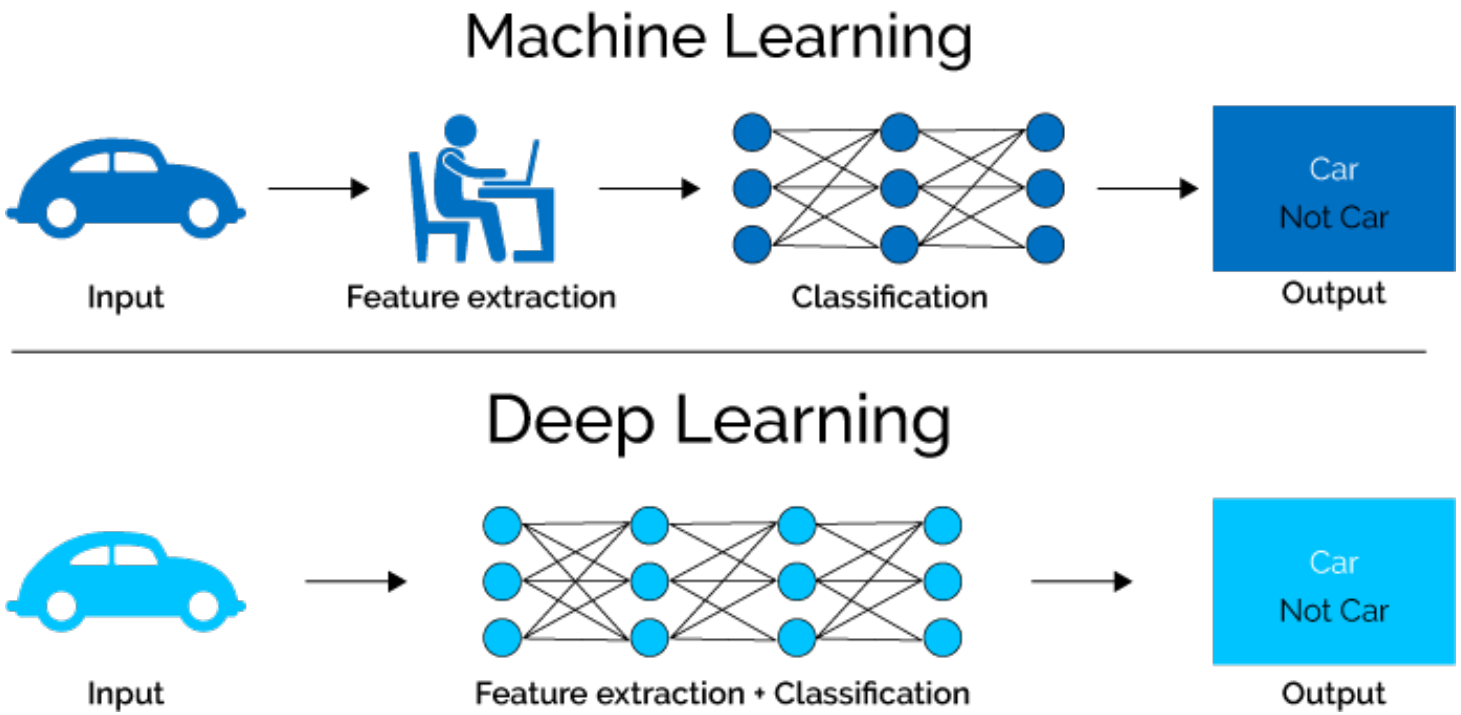
Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in.

Feature selection and **Data cleaning** should be the first and most important step of your model designing.

Irrelevant or partially relevant features can **negatively impact** model performance.

Feature selection techniques should be distinguished from **feature extraction.**

Deep learning (neural networks)



Feature selection heuristics

- **Wrapper methods**: the selection of a set of features as a search problem, where different combinations are prepared, evaluated and compared to other combinations. The search process may be methodical such as a best-first search, it may stochastic such as a random hill-climbing algorithm, or it may use heuristics, like forward and backward passes to add and remove features.

- **Embedded methods**: learn which features best contribute to the accuracy of the model while the model is being created (for example: regularization methods).

- **Filter methods**: select variables regardless of the model. They are based only on general features like the correlation with the variable to predict (for example: PCA)

Principal Component Analysis

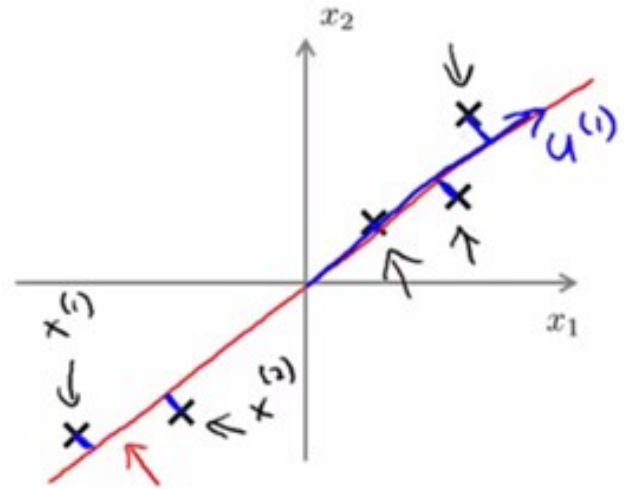
The main goal of principal component analysis (PCA) is to reduce the dimensionality of a data set.

If the features are correlated with each other, either heavily or lightly, it is possible to reduce the data dimensions while retaining the variation present in the dataset.

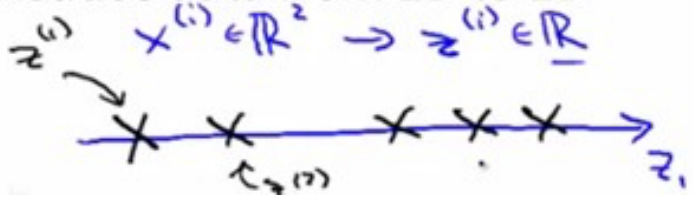
The PCA is done by transforming the features to a new set of features, which are known as the principal components (or simply, the PCs), which are orthogonal and ordered according to the retention of variation present (importance).

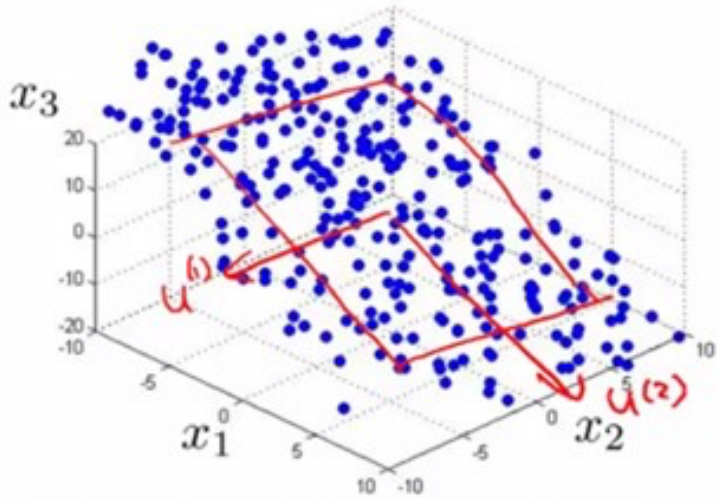
Principal Component Analysis

Principal Component Analysis (PCA) algorithm



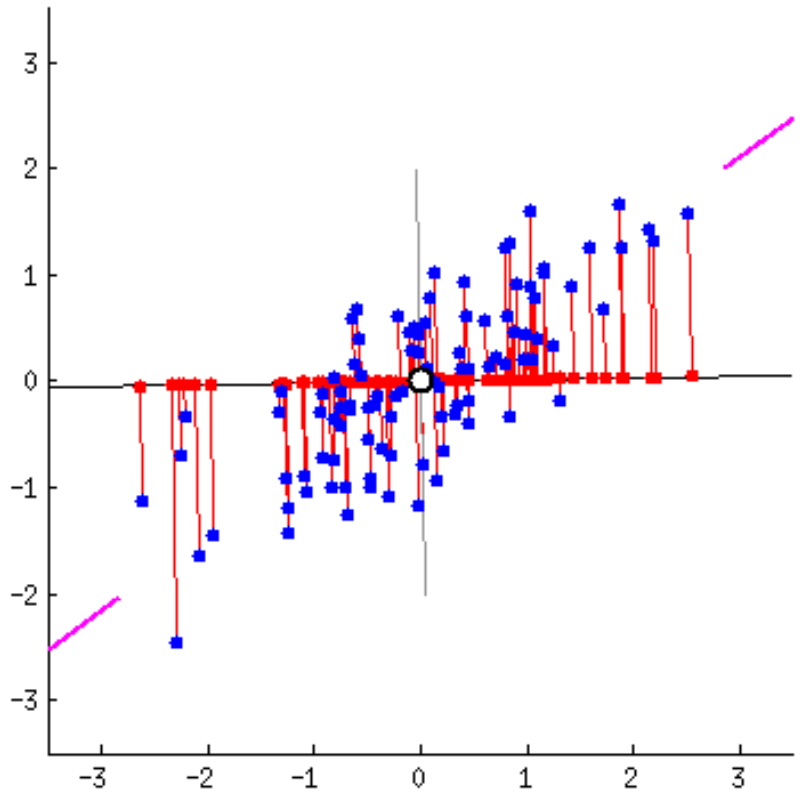
Reduce data from 2D to 1D





Reduce data from 3D to 2D

Image Source: Machine Learning Lectures by Prof. Andrew NG at Stanford University


3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

PCA terminology

- Dimensionality: number of features.
- Correlation: shows how strongly two features are related to each other. The value of the same ranges for -1 to +1. Positive indicates that when one feature increases, the other increases as well, while negative indicates the other decreases on increasing the former. And the modulus value of indicates the strength of relation.
- Orthogonal: Uncorrelated to each other, i.e., correlation between any pair of variables is 0.
- Eigenvectors/Eigenvalues: consider a non-zero vector **v**. It is an eigenvector of a square matrix **A**, if $\mathbf{Av}=\lambda\mathbf{v}$, where **v** is the eigenvector and λ is the eigenvalue associated with it.
- Covariance Matrix: consists of the covariances between the pairs of features. The (i,j)th element is the covariance between i-th and j-th feature.

PCA algorithm

1. Standardize the features

$$z = \frac{value - mean}{standard\ deviation}$$

2. Calculate the features' covariance matrix

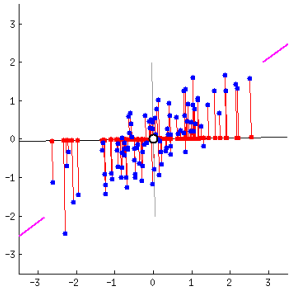
$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

1. Calculate the eigenvectors and eigenvalues
    1. Directions where data changes and changes the most

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix} X = \lambda X$$

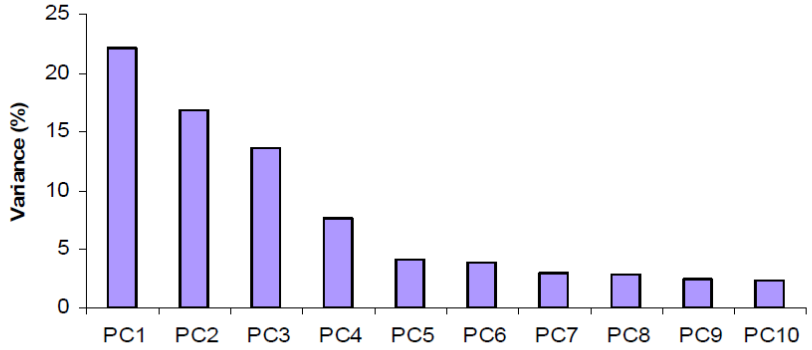
1. Select the num of components ranked by eigenvalue

2. Project the standardized features to the PCA space using to get the new features in orthogonal space



PCA: dimensionality reduction

Can *ignore* the components of less significance.



You do lose some information, but if the eigenvalues are small, you don't lose much

Explained variance

The explained variance tells you how much information (variance) can be attributed to each of the principal components. This is important as while you can convert from a n dimensional space to a k dimensional space, you lose some of the variance (information) when you do this.

How to chose *k (number of PCs)*?

- To choose K, use the following criterion

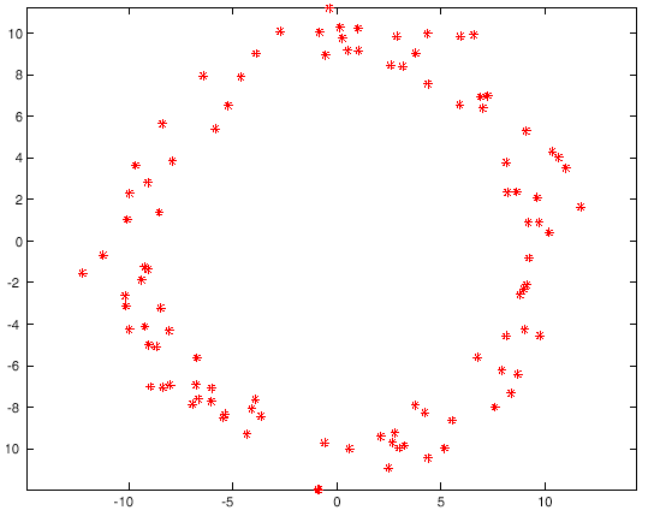
$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n} \lambda_i} > Thr$$

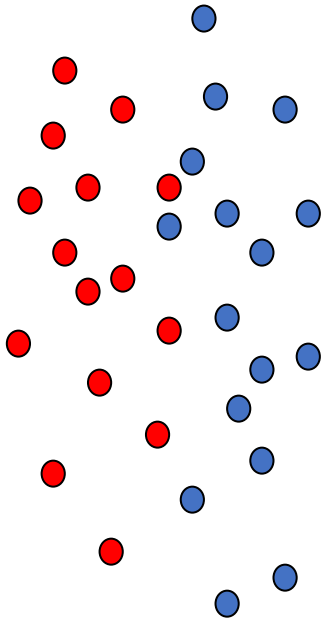
e.g., if $Thr = 0.9$ we say that we "preserve" 90% of the information in our dataset.

- If *k=n*, then we "preserve" 100% of the information in our dataset.

PCA limitations

PCA assumes that the data has a Gaussian distribution

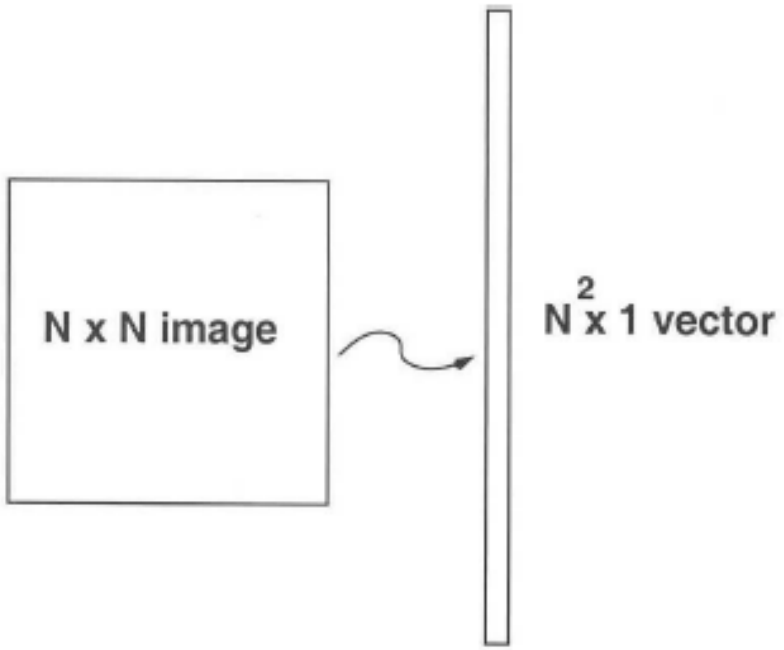




The direction of maximum variance is not always good for classification

Eigenfaces (Turk & Pentland)

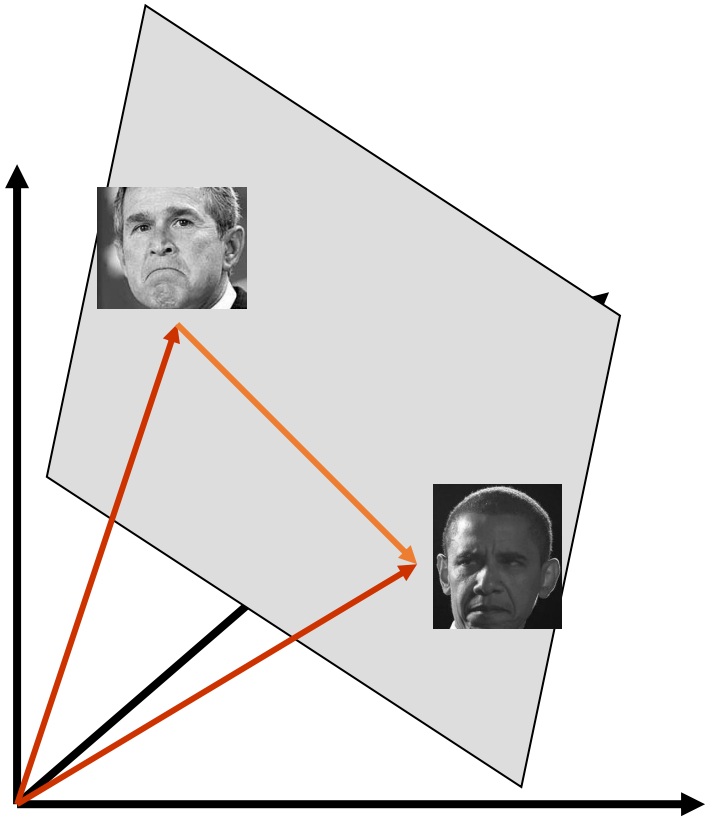
In the **Eigenfaces algorithm**, each point $x^{(i)} \in \mathbb{R}^{100\times100}$ was a 10000 dimensional vector, with each co- ordinate corresponding to a pixel intensity value in a 100x100 image of a face.



Eigenfaces (Turk & Pentland)

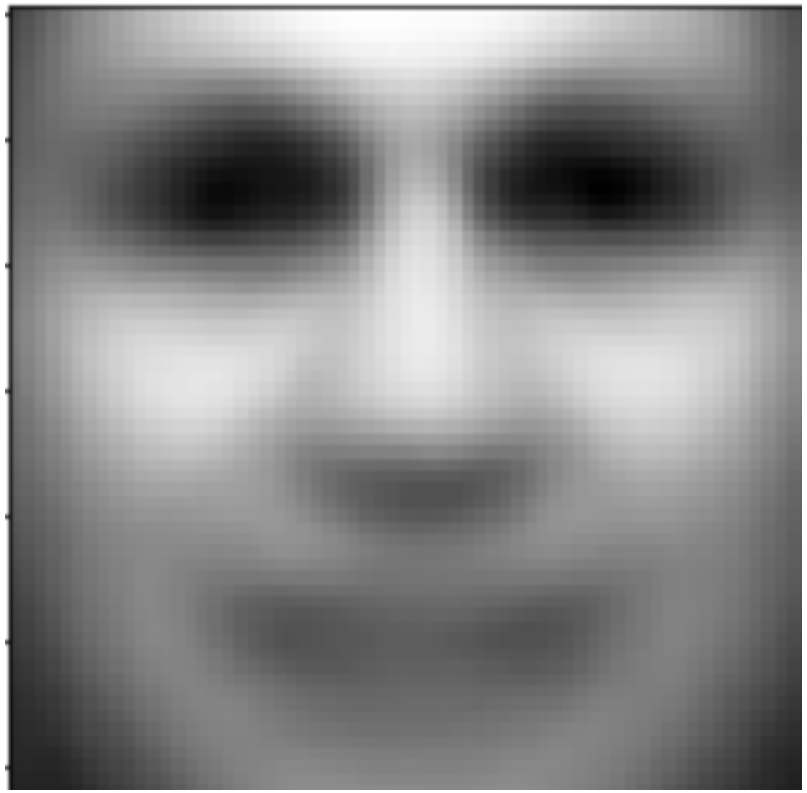
Using PCA, we represent each image $x^{(i)}$ with a much lower-dimensional $y^{(i)}$. In doing so, we hope that the principal components we found retain the interesting, systematic variations between faces that capture what a person really looks like, but not the "noise" in the images introduced by minor lighting variations, slightly different imaging conditions, and so on.

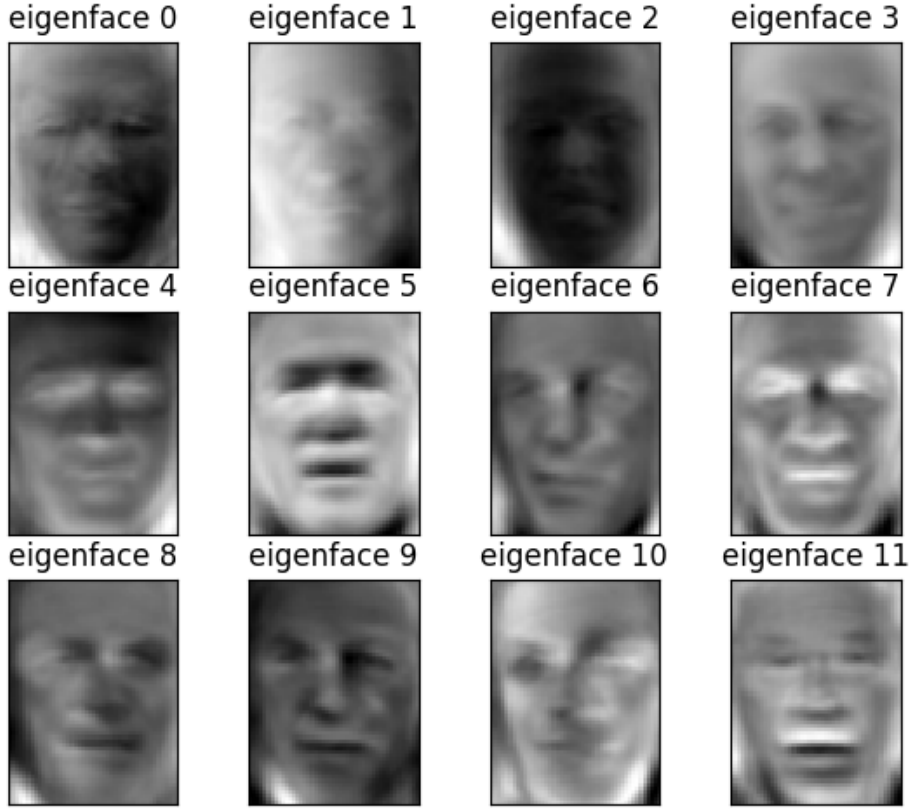
We then measure distances between faces i and j by working in the reduced dimension, and computing distances in the face-space. This resulted in a surprisingly good face-matching and retrieval algorithm.



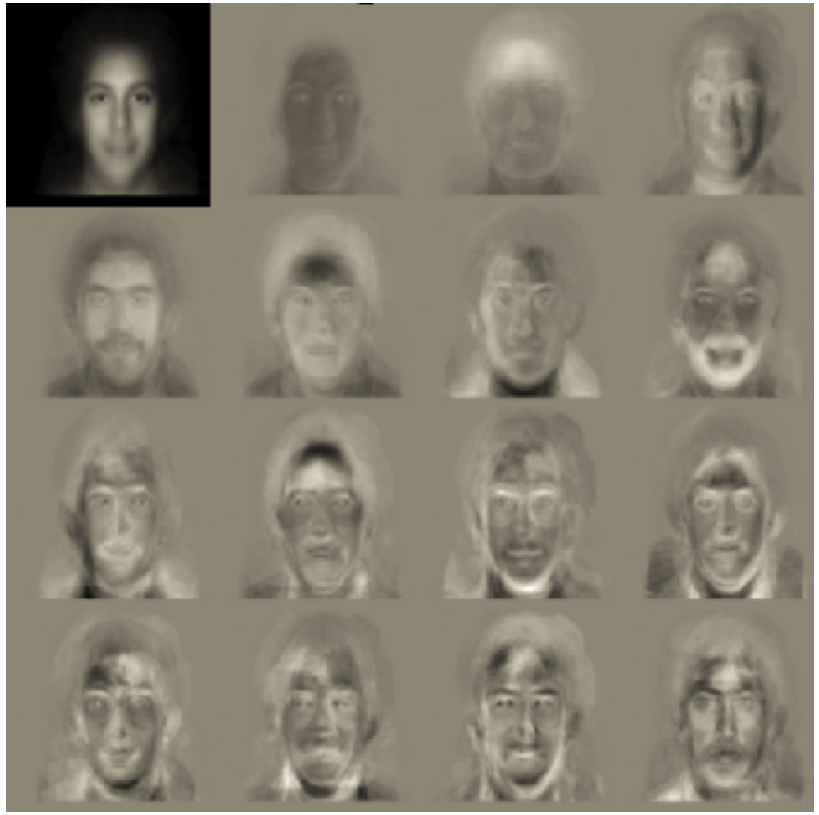
Eigenfaces (Turk & Pentland, 1991)

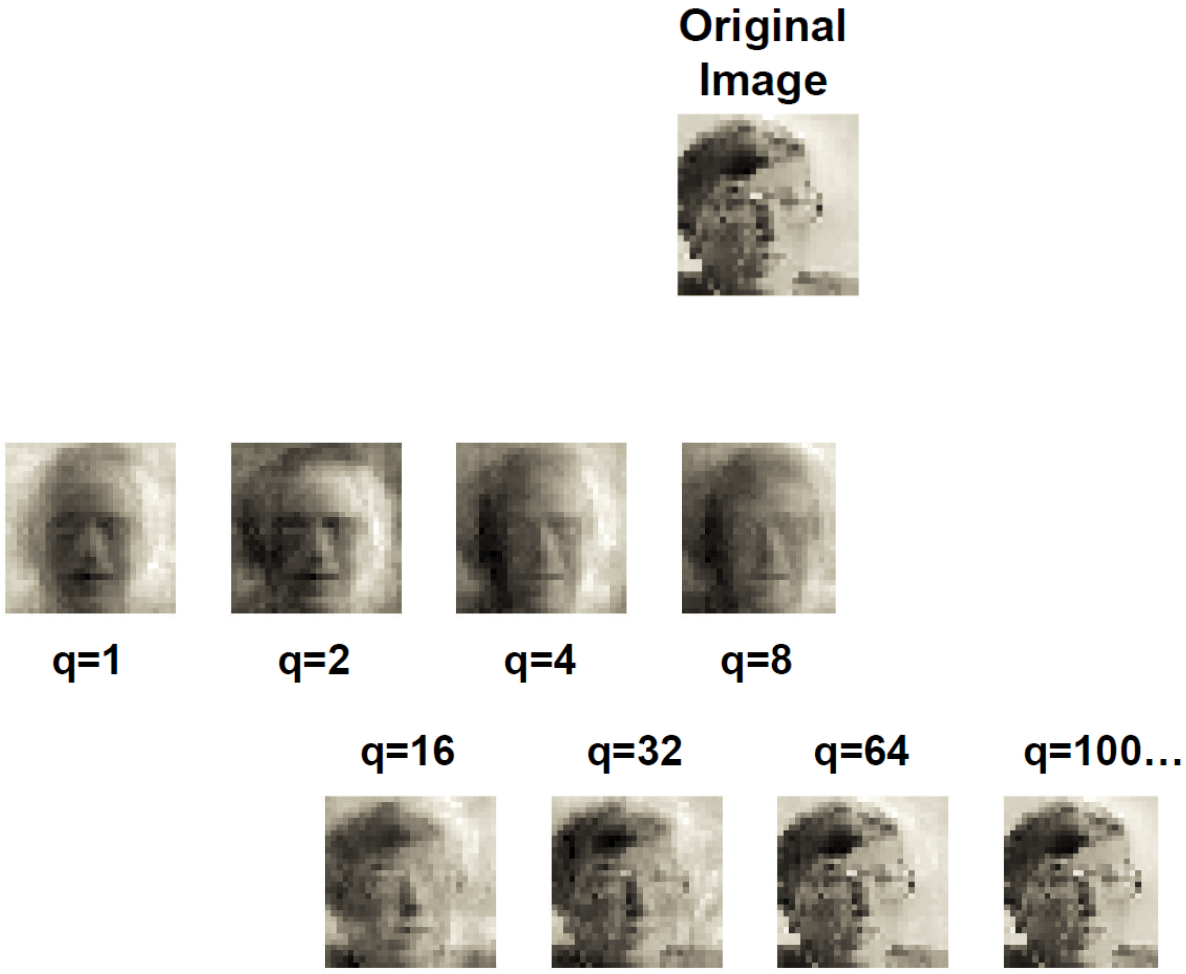
Mean face





Eigenfaces: reconstruction





Eigenfaces: limitations



GaNs…

Feature selection heuristics

- **Wrapper methods**: the selection of a set of features as a search problem, where different combinations are prepared, evaluated and compared to other combinations. The search process may be methodical such as a best-first search, it may stochastic such as a random hill-climbing algorithm, or it may use heuristics, like forward and backward passes to add and remove features.

- **Embedded methods**: learn which features best contribute to the accuracy of the model while the model is being created (for example: regularization methods).

- **Filter methods**: select variables regardless of the model. They are based only on general features like the correlation with the variable to predict (for example: PCA)

Learning performance

- Training dataset
- Supplied test set
- Percentage split
- Cross validation

Learning performance

- **Training dataset**
- Supplied test set
- Percentage split
- Cross validation

Prepare your model on the entire training dataset, then evaluate the model on the same dataset. This is generally problematic not least because a perfect algorithm could game this evaluation technique by simply memorizing (storing) all training patterns and achieve a perfect score, which would be misleading.

Learning performance

- Training dataset
- **Supplied test set**
- Percentage split
- Cross validation

Split your dataset manually using another program. Prepare your model on the entire training dataset and use the separate test set to evaluate the performance of the model. This is a good approach if you have a large dataset (many tens of thousands of instances).

Learning performance

- Training dataset
- Supplied test set
- **Percentage split**
- Cross validation

Randomly split your dataset into a training and a testing partitions each time you evaluate a model. This can give you a very quick estimate of performance and like using a supplied test set, is preferable only when you have a large dataset.

Learning performance

- Training dataset
- Supplied test set
- Percentage split
- **Cross validation**

Split the dataset into k-partitions or folds. Train a model on all of the partitions except one that is held out as the test set, then repeat this process creating k-different models and give each fold a chance of being held out as the test set. Then calculate the average performance of all k models. This is the gold standard for evaluating model performance, but has the cost of creating many more models.

Task specification (business problem)

1. Which Data?
2. Which Features?
3. **Which Model?**
4. Which Predictions?
5. How does this Solve Task (Improve Business)?

Task specification (business problem)

1. Which Data?
2. Which Features?
3. Which Model?
4. **Which Predictions?**
5. **How does this Solve Task (Improve Business)?**

Task specification (business problem)