Machine Learning

Given a collection of data, a machine learner explains the underlying process that generated the data in a general and simple fashion.

Different learning paradigms:

supervised learning
unsupervised learning
semi-supervised learning
reinforcement learning

Supervised Learning

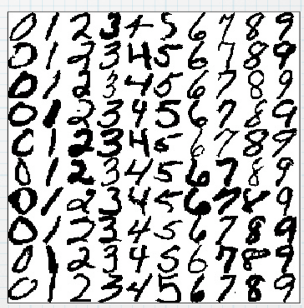
- Sample data comprises input vectors along with the corresponding target values (labeled data)
- Supervised learning uses the given labeled data to find a model (hypothesis) that predicts the target values for previously unseen data

Supervised Learning: **Classification**

- Each element in the sample is labeled as belonging to some *class* (e.g., apple or orange).
- The learner builds a model to predict classes for all input data.
- There is no order among classes.

Classification Example: Handwriting Recognition

- You've been given a set of *N* pictures of digits. For each picture, you're told the digit number
- Discover a set of rules which, when applied to pictures you've never seen, correctly identifies the digits in those pictures



Supervised Learning: **Regression**

- Each element in the sample is associated with one or more continuous variables
- The learner builds a model to predict the value(s) for all input data
- Unlike classes, values have an order among them

Regression Example: Automated Steering

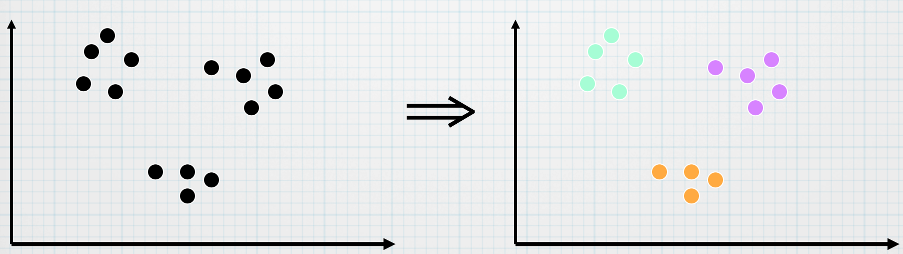
- A CMU team trained a neural network to drive a car by feeding in many pictures of roads, plus the *value* according to which the graduate student was turning the steering wheel at the time.
- Eventually the neural network learned to predict the correct value for previously unseen pictures of roads.
- "No Hands Across America"

Unsupervised Learning

- The given data consists of input vectors *without* any corresponding target values
- The goal is to discover groups of similar examples within the data (**clustering**), or to determine the distribution of data within the input space (**density estimation**)

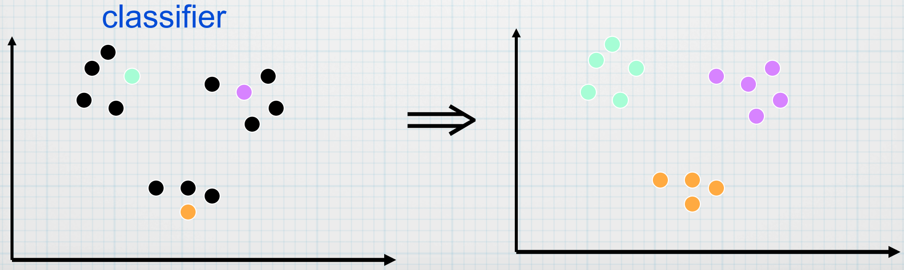
Unsupervised Learning: **Clustering**

- The goal is to discover groups of similar examples within the data



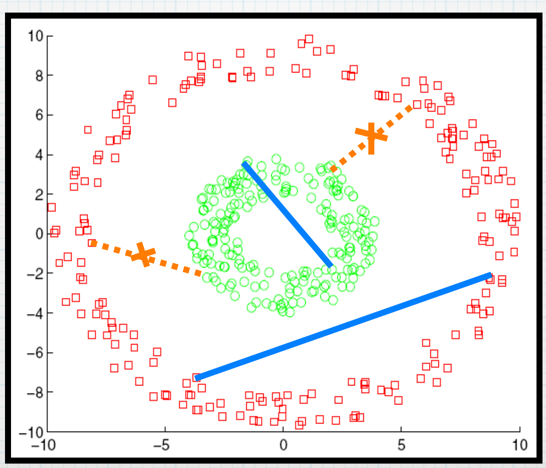
Semi-supervised Learning

- *Unlabeled data* may be easily available, while *labeled* ones may be expensive to obtain because they require human effort.
- *Semi-supervised learning is a recent learning paradigm*: it exploits unlabeled examples, in addition to labeled ones, to improve the generalization ability of the resulting classifier



Semi-supervised Clustering

- Constraints (*must-link*; *cannot-link*) on pairs of points are available



Reinforcement Learning

- The problem here is to find suitable actions to take in a given situation in order to maximize a reward
- Trial and error: no examples of optimal outputs are given
- Trade-off between *exploration* (try new actions to see how effective they are) and *exploitation* (use actions that are known to give high reward)

A Classification Example

(from *Pattern Classification* by Duda & Hart & Stork – Second Edition, 2001)

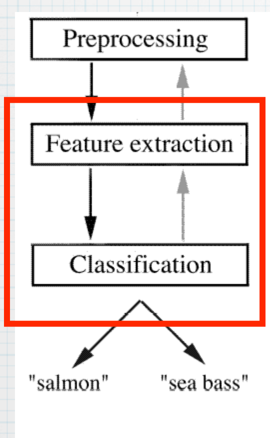
- A fish-packing plant wants to automate the process of sorting incoming fish according to species
- As a pilot project, it is decided to try to separate sea bass from salmon using optical sensing

To solve this problem, we adopt a *machine learning* approach:

We use a *training set* to tune the parameters of an adaptive model

- Length
- Lightness
- Width
- Position of the mouth
- …

Features to explore for use in our classifier



- **Preprocessing**: Images of different fishes are isolated from one another and from background;
- **Feature extraction**: The information of a single fish is then sent to a feature extractor, that measure certain "features" or "properties";
- **Classification**: The values of these features are passed to a classifier that evaluates the evidence presented, and build a model to discriminate between the two species

- **Domain knowledge**:
  a sea bass is generally longer than a salmon

- **Feature**: *Length*

- **Model**:
  Sea bass have some typical length, and this is greater than the length of a salmon

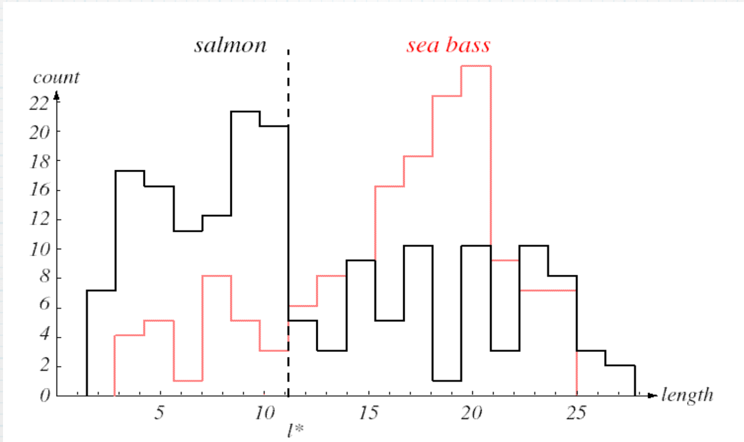
- **Classification rule**:

  If *Length* >= *l** then *sea bass*

  otherwise *salmon*

  - How to choose *l** ?
  - Use length values of sample data (training data)

Histograms of the length feature for the two categories



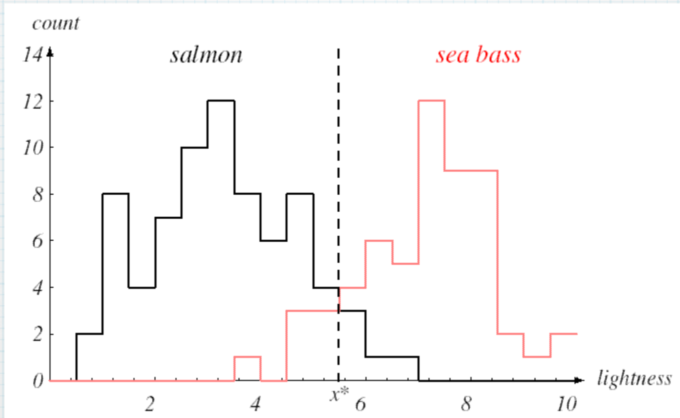
Leads to the smallest number of errors on average

We cannot reliably separate sea bass from salmon by length alone!

- **New Feature:**

  Average lightness of the fish scales

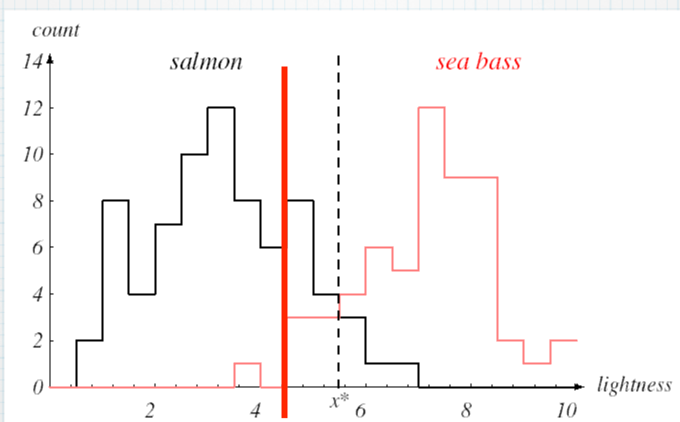
Histograms of the lightness feature for the two categories



Leads to the smallest number of errors on average

The two classes are much better separated!

Histograms of the lightness feature for the two categories



Classifying a sea bass as salmon costs more. Thus we reduce the number of sea bass classified as salmon.

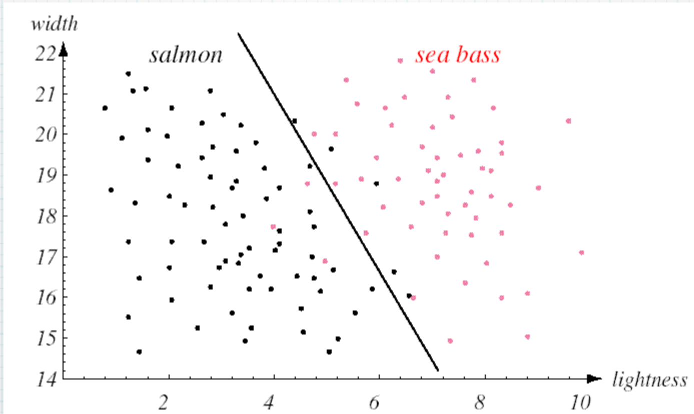
Our actions are equally costly

- **In Summary**:

The overall task is to come up with a decision rule (i.e., a decision boundary) so as to minimize the cost (which is equal to the average number of mistakes for equally costly actions).

- No single feature gives satisfactory results
- We must rely on using more than one feature
- We observe that:

  sea bass usually are wider than salmon
- Two features: *Lightness and Width*
- Resulting fish representation: $\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

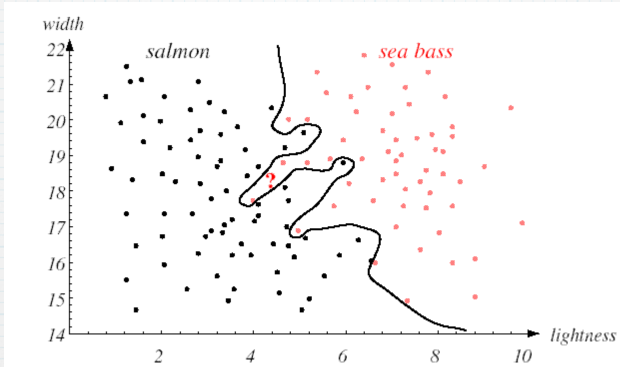


Decision rule: Classify the fish as a sea bass if its feature vector falls above the decision boundary shown, and as salmon otherwise

Should we be satisfied with this result?

Options we have:

- Consider additional features:
  - Which ones?
  - Some features may be redundant (e.g., if *eye color* perfectly correlates with *width*, then we gain no information by adding eye color as feature.)
  - It may be costly to attain more features
  - Too many features may hurt the performance!
- Use a more complex model



All training data are perfectly separated

Should we be satisfied now??

- **We must consider: Which decisions will the classifier take on novel patterns, i.e. fish not yet seen? Will the classifier suggest the correct actions?**

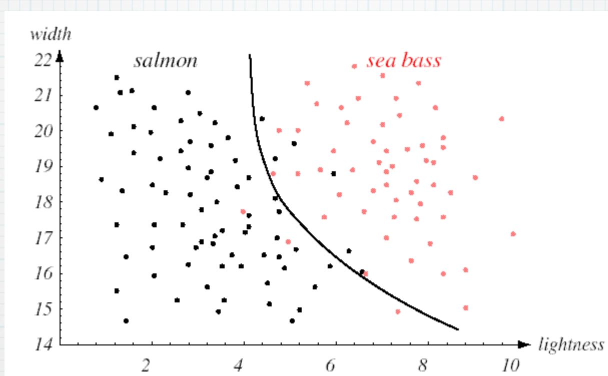
This is the issue of GENERALIZATION

Generalization

- A good classifier should be able to generalize, i.e. perform well on unseen data
- The classifier should capture the underlying characteristics of the categories
- The classifier should NOT be tuned to the specific (accidental) characteristics of the training data
- Training data in practice contain some noise

- **As a consequence:**

We are better off with a slightly poorer performance on the training examples if this means that our classifier will have better performance on novel patterns.



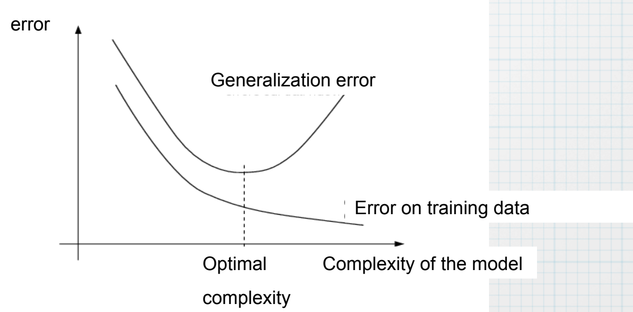
The decision boundary shown may represent the optimal tradeoff between accuracy on the training set and on new patterns

How can we determine automatically when the optimal tradeoff has been reached?

Generalization

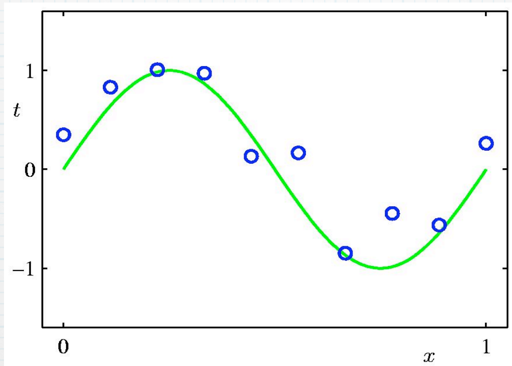
- The idea is to use a model with an intermediate complexity, which gets most of the points right, without putting too much trust in any individual point.
- The goal of ***statistical learning theory*** is to formalize these arguments by studying mathematical properties of learning machines, i.e. properties of the class of functions that the learning machine can implement (formalization of the complexity of the model).

Tradeoff between performance on training and novel examples



Evaluation of the classifier on novel data is important to avoid *over-fitting*

Example: Regression Estimation



$t = \sin(2\pi x)$

- **Data**: input-output pairs $(x_i, t_i) \in \Re \times \Re$
- **Regularity**: $(x_1, t_1), \cdots, (x_N, t_N)$ drawn from $P(x, t)$
- **Learning**: choose a function $f : \Re \rightarrow \Re$ such that a given error function is minimized

Polynomial Curve Fitting

- We fit the data using a polynomial function of the form:

$$y(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

- It is linear in the *unknown parameters*: **linear model**

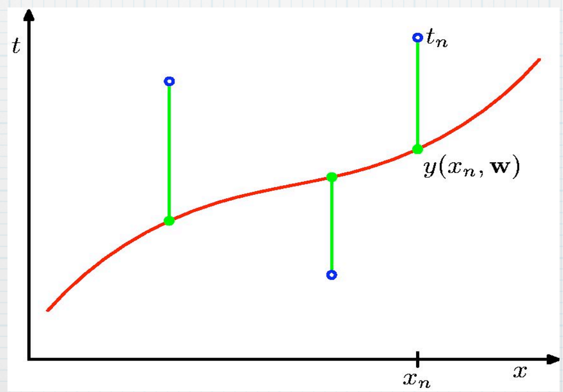
- Fit the polynomial to the training data so that a given error function is minimized:

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \left(y(x_n, \boldsymbol{w}) - t_n \right)^2$$

Polynomial Curve Fitting

Geometrical interpretation of the sum-of-squares error function

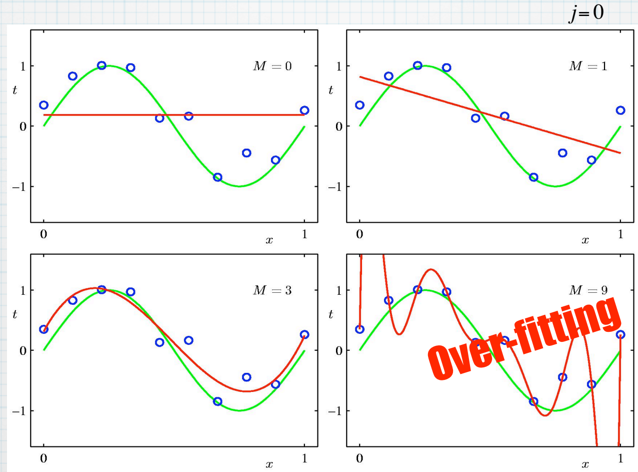
$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(y(x_n,\boldsymbol{w}) - t_n\right)^2$$



Polynomial Curve Fitting

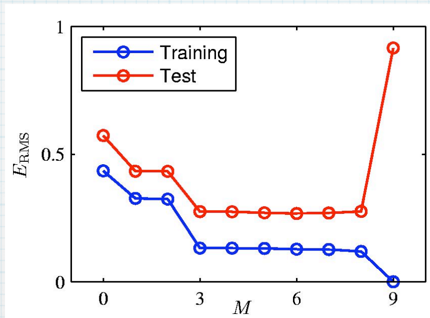
Model selection: We need to choose the order M of the polynomial:

$$y(x,\boldsymbol{w}) = w_0 + w_1x + w_2x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$



Polynomial Curve Fitting

Model selection: we can investigate the dependence of the generalization performance on the order M



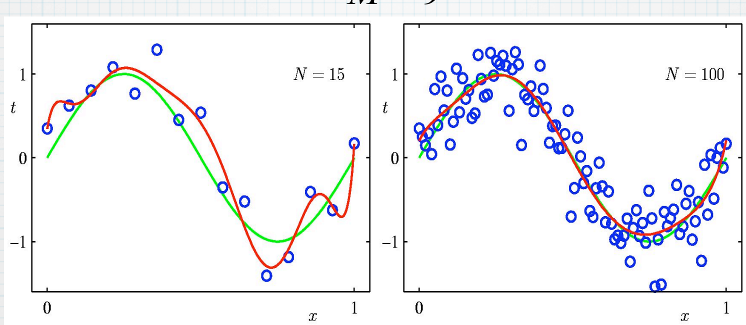
$$E_{RMS} = \sqrt{2E(\mathbf{w}^*)/N}$$

Root-mean-square error

Polynomial Curve Fitting

We can also examine the behavior of a **given model** as the size of the data set changes

$$M = 9$$



For a given model, the over-fitting problem becomes less severe as the number of training data increases

How to dodge the over-fitting problem?

Assumption: we have a limited number of training data available, and we wish to use relatively complex and flexible models.

One possible solution: add a penalty term to the error function to discourage the coefficients from reaching large values, e.g.:

$$\widetilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(y(x_n, \boldsymbol{w}) - t_n\right)^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

Sum of all squared coefficients

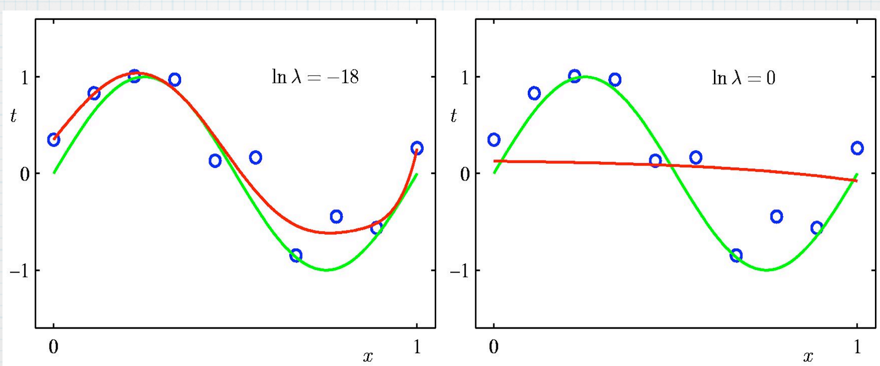
This is an example of a **regularization** technique.

[*shrinkage methods*, *ridge regression*, *weight decay*]

The impact of Regularization

$$\widetilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(y(x_n, \boldsymbol{w}) - t_n\right)^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

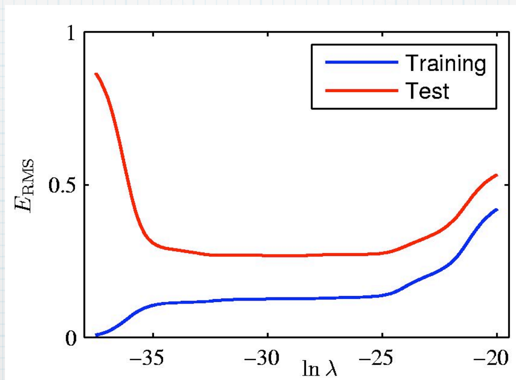
$$M = 9$$



The impact of Regularization

$$\widetilde{E}(\mathbf{w})=\frac{1}{2}\sum_{n=1}^{N}\left(y(x_n,\mathbf{w})-t_n\right)^2+\frac{\lambda}{2}\|\mathbf{w}\|^2$$

$$M=9$$



λ controls the effective complexity of the model and hence determines the degree of over-fitting

Model selection

Partition available data into a ***training set*** used to determine the coefficients $\mathbf{w}$, and into a separate ***validation set*** (also called ***hold-out***) used to optimize the model complexity (M ***or*** λ)

If the model design is iterated several times using a limited number of data, some over-fitting to the validation data can occur.

<u>Solution</u>: Set aside a third ***test set*** on which performance of the selected model is finally evaluated.

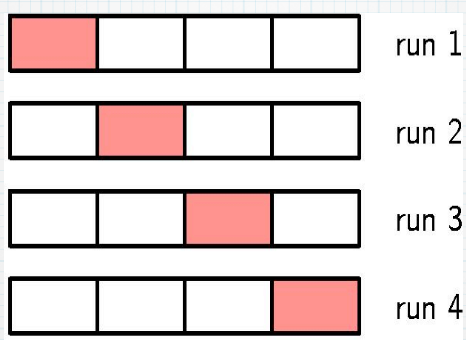
Model selection

Often we have limited data available, and we wish to use as much of the available data as possible for training to build good models.

However, if we use a small set for validation, we obtain noisy estimates of predictive performance.

One solution to this problem is ***cross-validation***.

Cross-validation: Example



4-fold cross-validation

It allows to use ¾ of the available data for training, while making use of all of the data to assess performance

k-fold Cross-validation

- In general: we perform k runs. Each run uses (k-1)/k of the available data for training.
- If the number of data is very limited, we can set k=N (total number of data points). This gives the ***leave-one-out*** cross-validation technique.

k-fold Cross-validation: Drawbacks

- Computationally expensive: number of training runs is increased by a factor of k.
- A single models may have multiple complexity parameters: exploring combinations of settings could require a number of training runs that is *exponential* in the number of parameters.

Bayesian Probabilities

- A key issue in pattern recognition is ***uncertainty***. It is due to *incomplete* and/or *ambiguous* information, i.e. *finite* and *noisy* data.
- **Probability theory** and **decision theory** provide the tools to make optimal predictions given the limited available information.
- In particular, the ***Bayesian interpretation of probability*** allows to quantify uncertainty, and make precise revisions of uncertainty in light of new evidence.

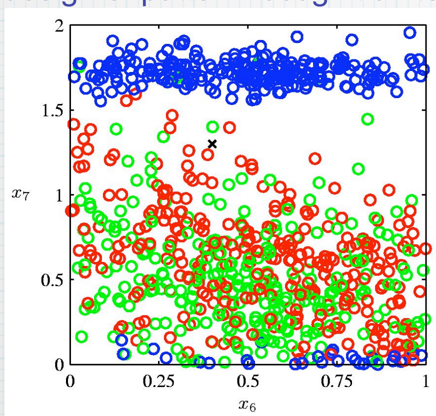
Bayes' Theorem

$$p(Y = y \mid X = x) = \frac{p(X = x \mid Y = y)\,p(Y = y)}{p(X = x)}$$

- $p(Y = y)$ is the ***prior probability***: it expresses the probability ***before*** we observe any data
- $p(Y = y \mid X = x)$ is the ***posterior probability***: it expresses the probability ***after*** we observed the data
- The effect of the observed data is captured through the conditional probability $p(X = x \mid Y = y)$

Curse of Dimensionality

- Real world applications deal with spaces with high dimensionality
- High dimensionality poses serious challenges for the design of pattern recognition techniques



Oil flaw data in two dimensions

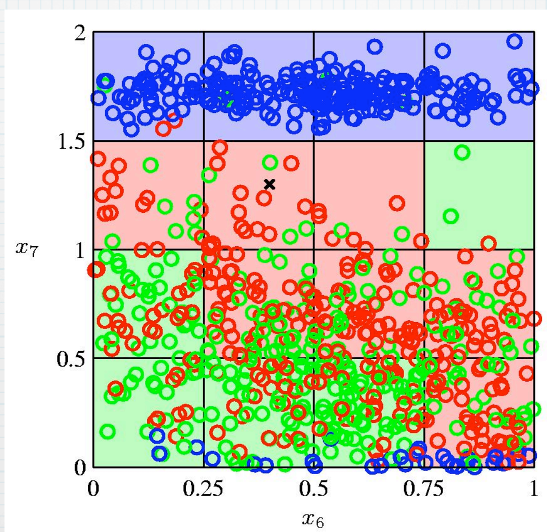
Red = homogeneous

Green = annular

Blue = laminar

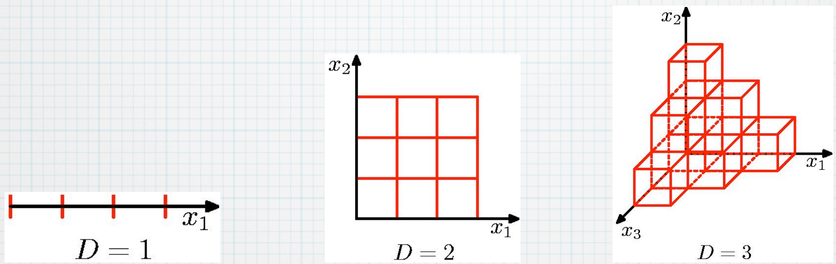
Curse of Dimensionality

- A simple classification approach



Curse of Dimensionality

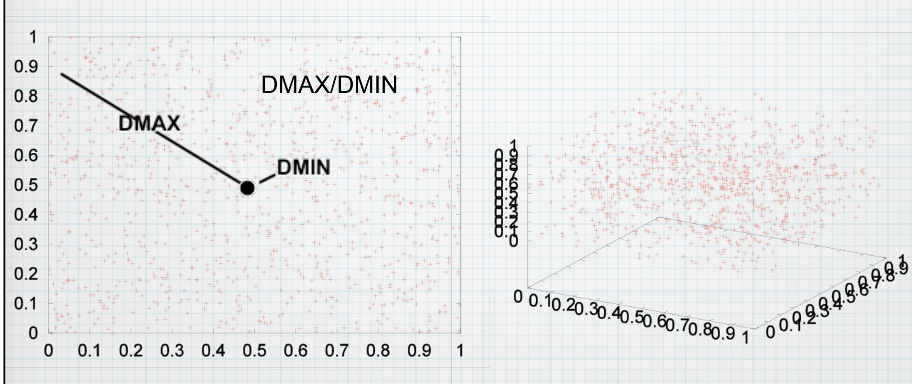
- Going higher in dimensionality…



3^1 3^2 3^3

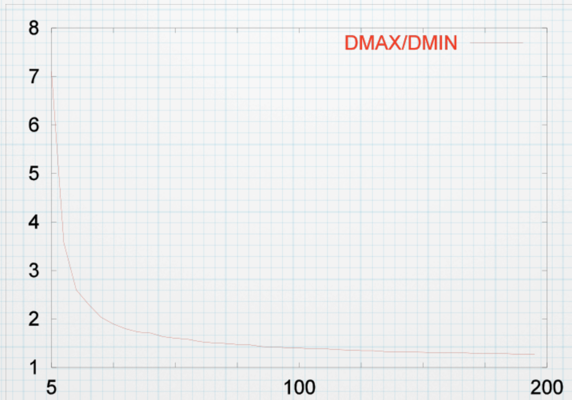
The number of regions of a regular grid grows exponentially with the dimensionality D of the space

Curse of Dimensionality



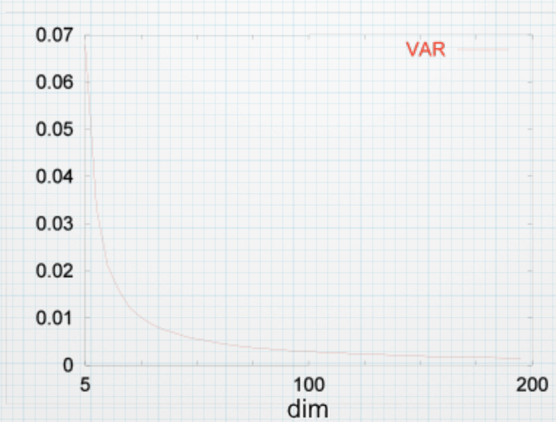
Sample of size *N=500* uniformly distributed in $[0,1]^q$

Curse of Dimensionality



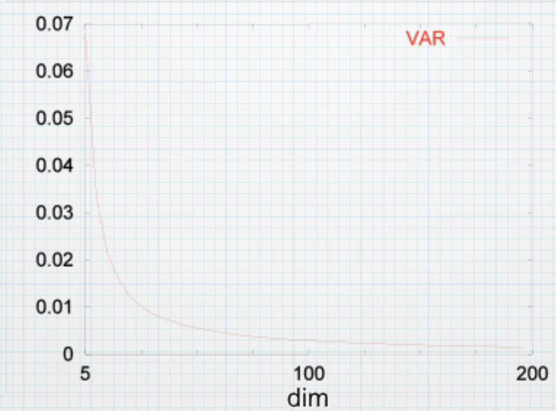
The distribution of the ratio **DMAX/DMIN** converges to **1** as the dimensionality increases

Curse of Dimensionality



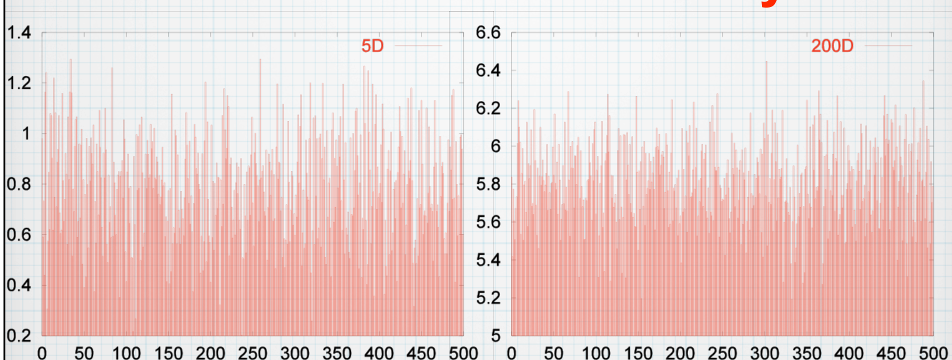
Variance of distances from a given point

Curse of Dimensionality



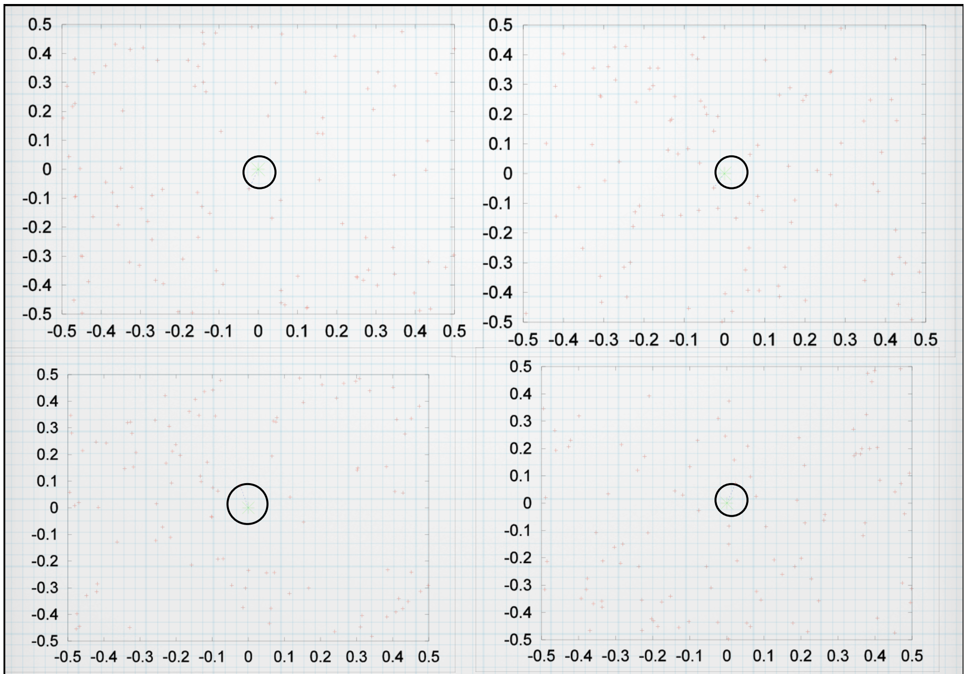
The variance of distances from a given point converges to **0** as the dimensionality increases

Curse of Dimensionality

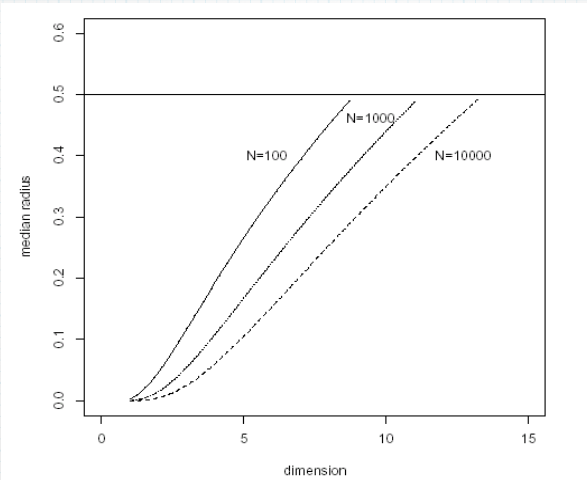


Distance values from a given point

Values flatten out as dimensionality increases



Computing radii of nearest neighborhoods



median radius of a nearest neighborhood

uniform distribution in the unit cube $[-.5,.5]^q$

Curse-of-Dimensionality

As dimensionality increases, the distance from the closest point increases faster

- Random sample of size N ~ uniform distribution in the q-dimensional unit hypercube
- Diameter of a $K = 1$ neighborhood using Euclidean distance: $d(q,N) = O(N^{-1/q})$

q	4	4	6	6	10	10	20	20	20
N	100	1000	100	1000	1000	10000	10000	10^6	10^{10}
d(q,N)	0.42	0.23	0.71	0.48	0.91	0.72	1.51	1.20	0.76

Large $d(q,N) \Rightarrow$ Highly biased estimations

Curse-of-Dimensionality

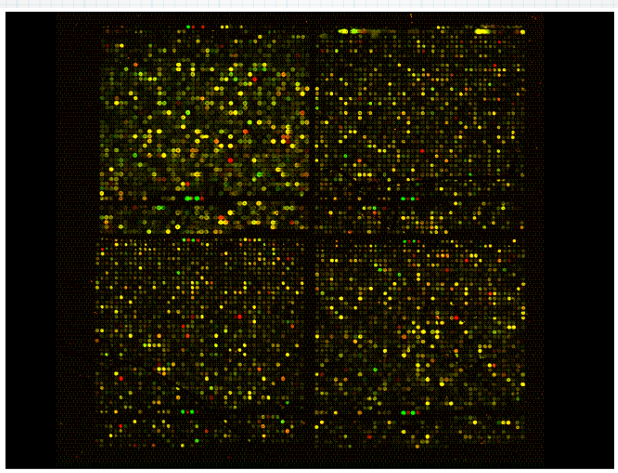
- In high dimensional spaces data become extremely sparse and are far apart from each other

- **The curse of dimensionality affects *any* estimation problem with high dimensionality**

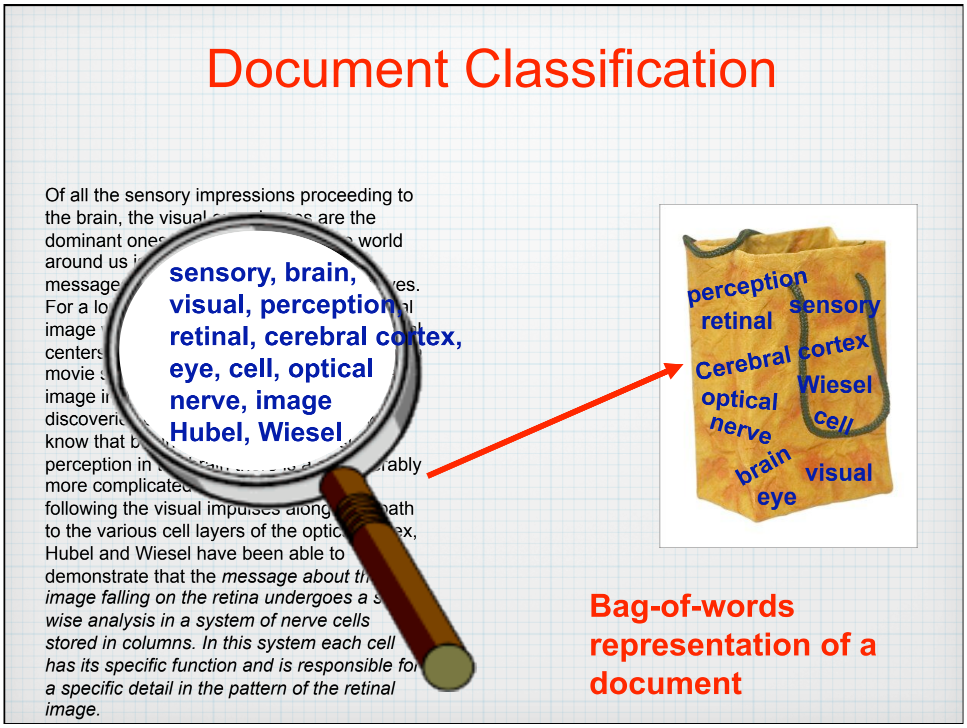
Curse-of-Dimensionality

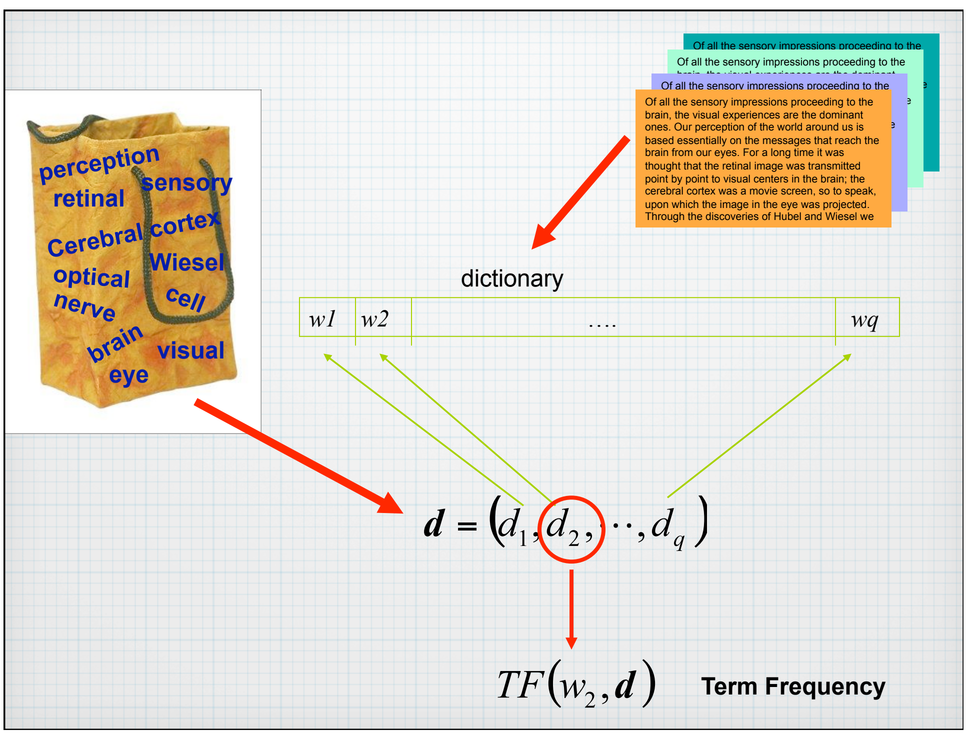
- It is a serious problem in many real-world applications
  - Microarray data: 3,000-4,000 genes;
  - Documents: 10,000-20,000 words in dictionary;
  - Images, face recognition, etc.

Microarray Data Analysis

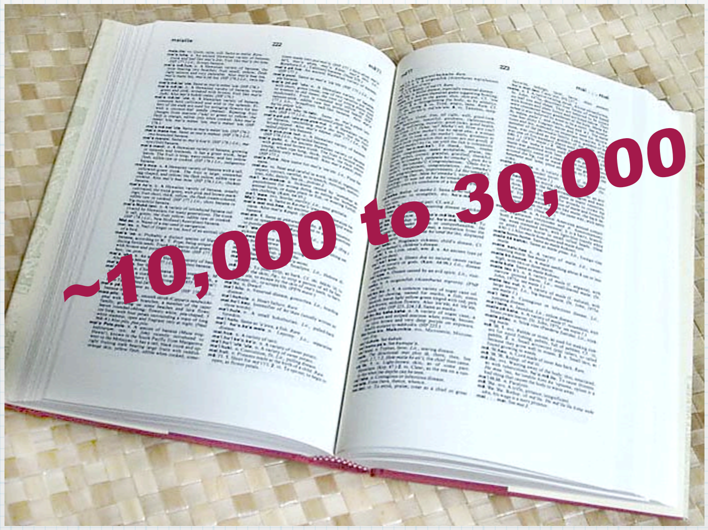


Problem:Which samples are most similar to each other, in terms of their expression profiles across genes


Document Classification
Of all the sensory impressions proceeding to
the brain, the visual ... are the
dominant ones ... world
around us ...
message ... es.
For a lo...
image ...
centers ...
movie ...
image i...
discoveri...
know that ...
perception in ... rably
more complicated
following the visual impulses along ... path
to the various cell layers of the optic ... ex,
Hubel and Wiesel have been able to
demonstrate that the message about th...
image falling on the retina undergoes a s...
wise analysis in a system of nerve cells
stored in columns. In this system each cell
has its specific function and is responsible fo...
a specific detail in the pattern of the retinal
image.
sensory, brain,
visual, perception,
retinal, cerebral cortex,
eye, cell, optical
nerve, image
Hubel, Wiesel
perception
sensory
retinal
Cerebral cortex
Wiesel
optical
nerve
cell
brain
visual
eye
Bag-of-words representation of a document


Of all the sensory impressions proceeding to the brain, the visual experiences are the dominant ones. Our perception of the world around us is based essentially on the messages that reach the brain from our eyes. For a long time it was thought that the retinal image was transmitted point by point to visual centers in the brain; the cerebral cortex was a movie screen, so to speak, upon which the image in the eye was projected. Through the discoveries of Hubel and Wiesel we
perception
retinal
sensory
Cerebral cortex
Wiesel
optical
nerve
cell
brain
visual
eye
dictionary
w1
w2
....
wq
$\boldsymbol{d} = (d_1, d_2, \cdots, d_q)$
$TF(w_2, \boldsymbol{d})$
Term Frequency

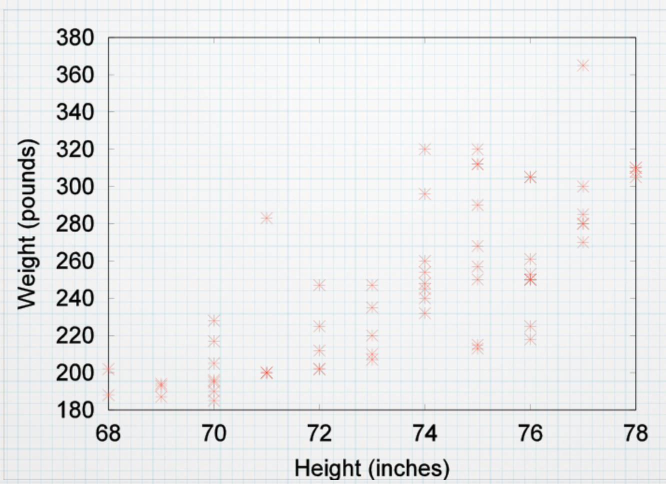
What's the size of the dictionary?



How can we deal with the curse of dimensionality?

Curse-of-Dimensionality

- ➢ Effective techniques applicable to high dimensional spaces exist.
- ➢The reasons are twofold:
  - ✓ Real data are often confined to regions of *lower dimensionality*
  - ✓ Real data typically exhibit *smoothness properties* (at least locally). Local interpolation techniques can be used to make predictions



$$\begin{bmatrix} 7.68 & 92.2 \\ 92.2 & 1912.5 \end{bmatrix}$$

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i$$

2×2 covariance matrix:

$$E\left[(\boldsymbol{x}-\mu)(\boldsymbol{x}-\mu)^T\right] =$$

$$E\left[\begin{pmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{pmatrix}(x_1-\mu_1, x_2-\mu_2)\right] =$$

$$E\begin{bmatrix} (x_1-\mu_1)^2 & (x_1-\mu_1)(x_2-\mu_2) \\ (x_1-\mu_1)(x_2-\mu_2) & (x_2-\mu_2)^2 \end{bmatrix} =$$

$$\frac{1}{N}\sum_{i=1}^{N}\begin{bmatrix} (x_1^i-\mu_1)^2 & (x_1^i-\mu_1)(x_2^i-\mu_2) \\ (x_1^i-\mu_1)(x_2^i-\mu_2) & (x_2^i-\mu_2)^2 \end{bmatrix}$$

$$\frac{1}{N}\sum_{i=1}^{N}\begin{bmatrix} (x_1^i-\mu_1)^2 & (x_1^i-\mu_1)(x_2^i-\mu_2) \\ (x_1^i-\mu_1)(x_2^i-\mu_2) & (x_2^i-\mu_2)^2 \end{bmatrix} =$$

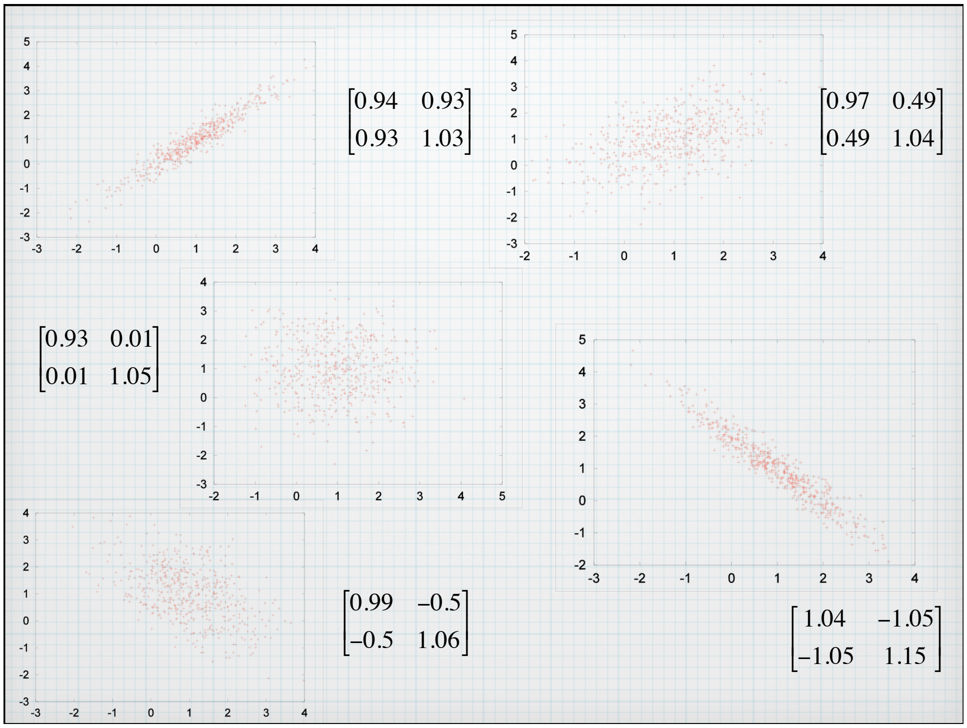
variance

covariance

$$\begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N}(x_1^i-\mu_1)^2 & \frac{1}{N}\sum_{i=1}^{N}\left[(x_1^i-\mu_1)(x_2^i-\mu_2)\right] \\ \frac{1}{N}\sum_{i=1}^{N}\left[(x_1^i-\mu_1)(x_2^i-\mu_2)\right] & \frac{1}{N}\sum_{i=1}^{N}(x_2^i-\mu_2)^2 \end{bmatrix}$$

covariance

variance



Dimensionality Reduction

- Many dimensions are often interdependent (correlated);

We can:

- Reduce the dimensionality of problems;
- Transform interdependent coordinates into significant and independent ones;