Artificial Intelligence

Steve James

Supervised and Unsupervised Learning

Machine learning

- Subfield of AI concerned with learning from data
- Broadly, using:
  - Experience
  - To improve performance
  - On some task

(Tom Mitchell, 1997)

Supervised learning

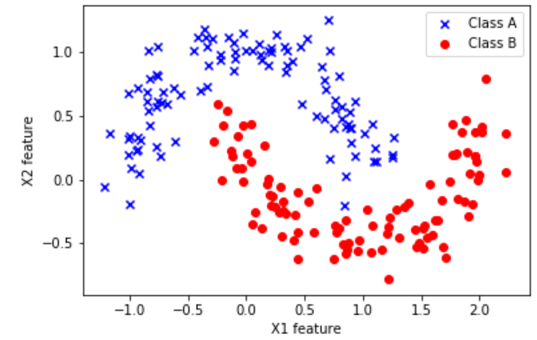
- Input:
  - $X = \{x_1, \dots, x_n\}$
  - $Y = \{y_1, \dots, y_n\}$

Training data

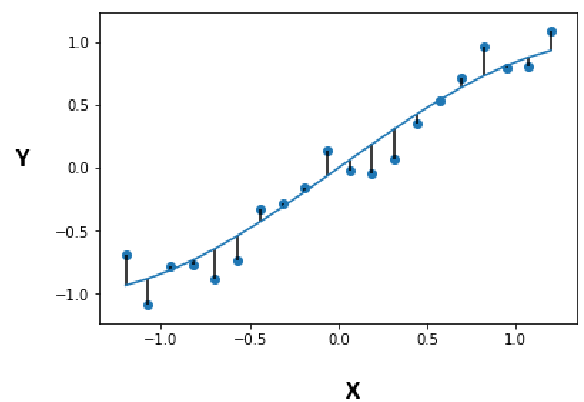
- Learning to predict new labels
  - Given x, y?

Classification vs regression

- If set of labels Y is discrete:
  - Classification
  - Minimise number of errors



- If Y is real-valued
  - Regression
  - Minimise sum squared error



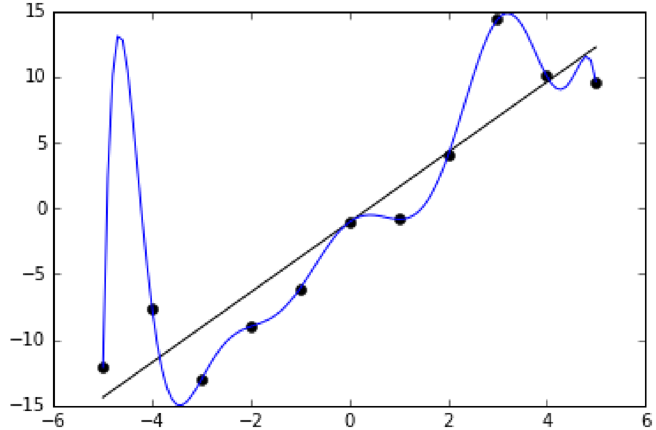
Supervised learning

- Formal definition:
  - Given training data:
    - $X = \{x_1, \dots, x_n\}$
    - $Y = \{y_1, \dots, y_n\}$
  - Produce decision function $f: X \rightarrow Y$
  - That minimises error
    - $\sum_i err(f(x_i), y_i)$

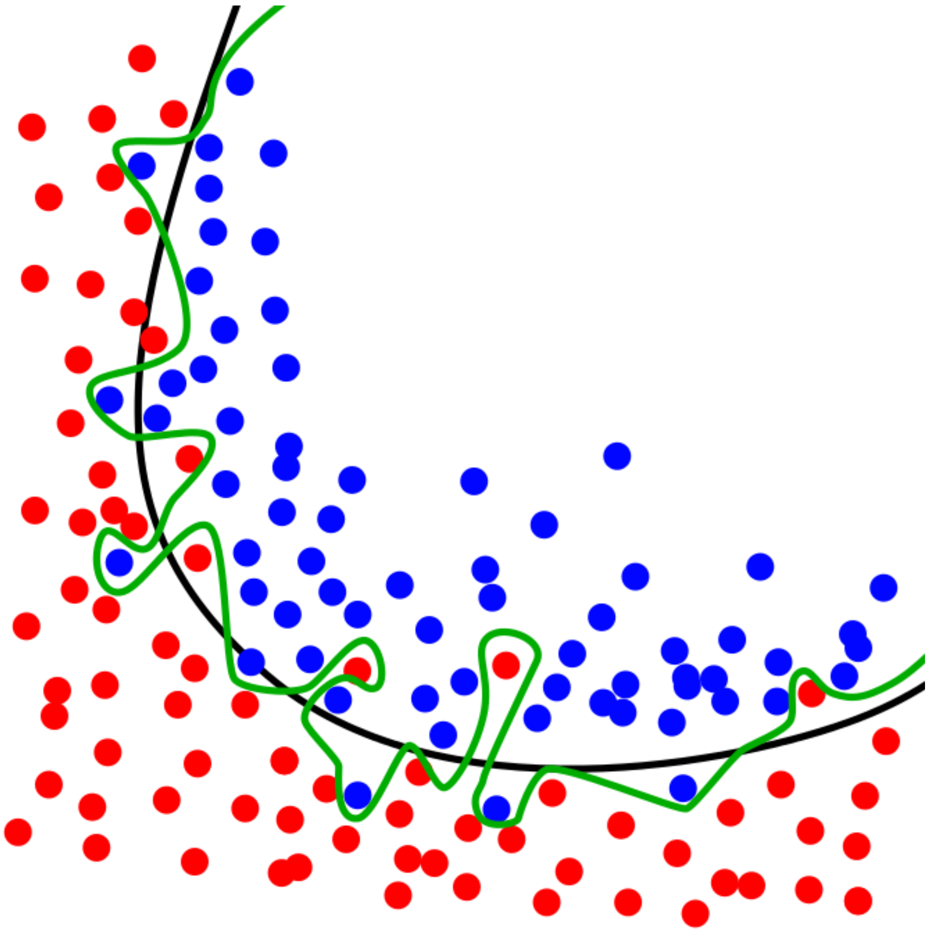
Test/train split

- Minimise error on what?
  - Don’t get to see future data

- General principle: do not measure error on the data you train on



Overfitting



Test/train split

- Methodology:
  - Split data into train and test set
  - Fit f using training set
  - Measure error on test set

- Common alternative: k-fold cross validation
  - Repeat k times:
    - Partition data into train $(n - n/k)$ and test (n/k) data sets
    - Train on training set, test on test set
  - Average results across k choices of test set.

Key idea: hypothesis space

- Typically fixed representation of classifier/regressor
  – Learning algorithm constructed to match

- Representation induces class of functions F from which to find f
  – F is known as the hypothesis space
  – Tradeoff: power vs expressibility vs data efficiency
  – Not every F can represent every function

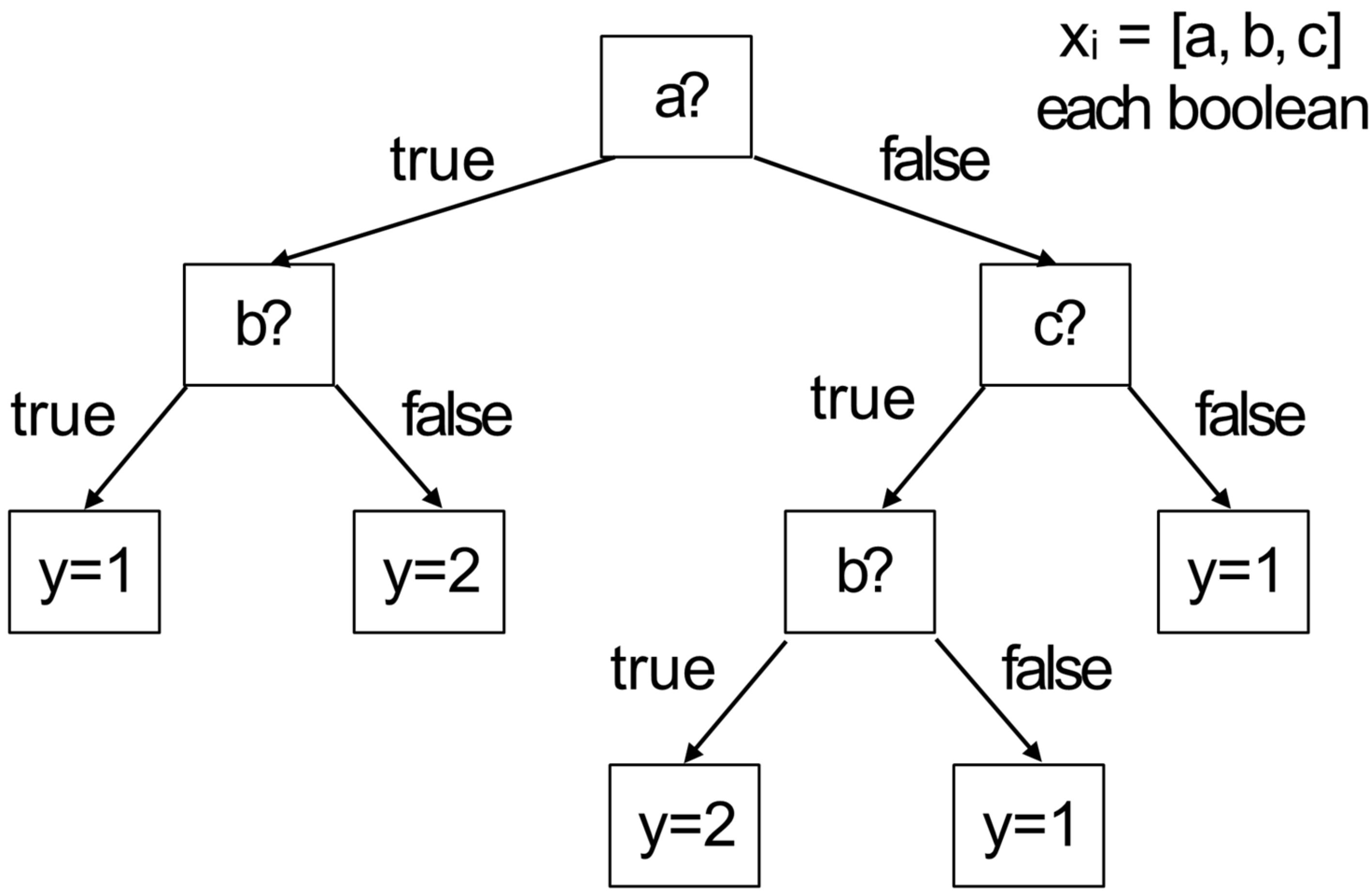
Decision trees

- Assume:
  - Two classes (true / false)
  - Input is vector of discrete values

- Simplest thing we could do?
  - How about if/else rules?

- Relatively simple classifier
  - Tree of tests
  - Evaluate test for each x_i and follow branch
  - Leaves are class labels

Decision trees



Decision trees

- How to make one?

- Given $X = \{x_1, \ldots, x_n\}, \mathrm{Y} = \{\mathrm{y}_1, \ldots, \mathrm{y}_\mathrm{n}\}$

- Repeat:
  - If all labels are the same, then we have a leaf node
  - Pick an attribute and split data based on its value
  - Recurse on each half

- If we run out of splits and data not perfectly in one class, then take a max

Attribute picking

- But which attribute to split over?
- Information contained in a data set:

$$I(D) = -f_1 \lg(f_1) - f_2 \lg(f_2)$$

- How “bits” of information do we need to determine the label in a dataset?
- Pick attribute with max information gain:

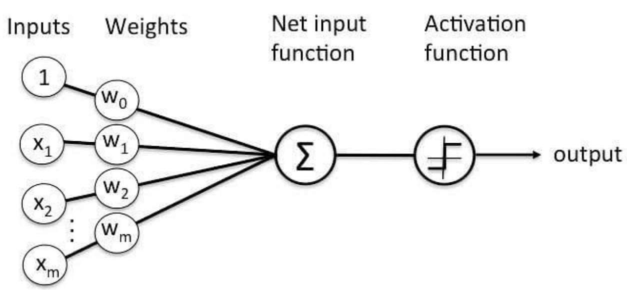
$$Gain(E) = I(D) - \sum_i f_i I(E_i)$$

Neural networks

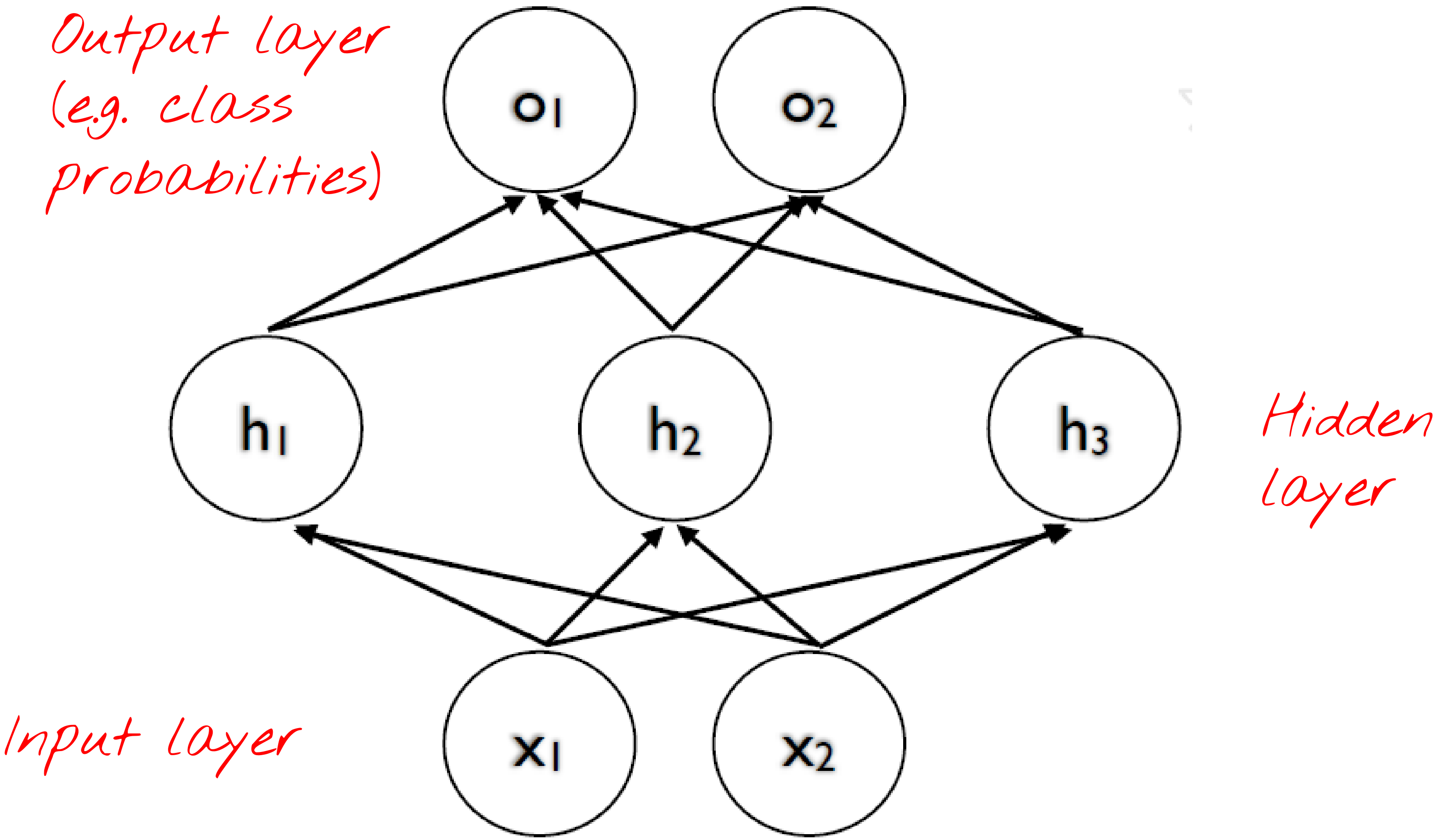
- Single neuron:
  - Takes as input x, produces output of the form

$$y = a(w \cdot x + b)$$

- w (weights) and b (bias) are learnable parameters
- a is activation function. Many choices
  - $a(x) = \operatorname{sgn} x$
  - $a(x) = \sigma(x)$
  - $a(x) = ReLU(x)$

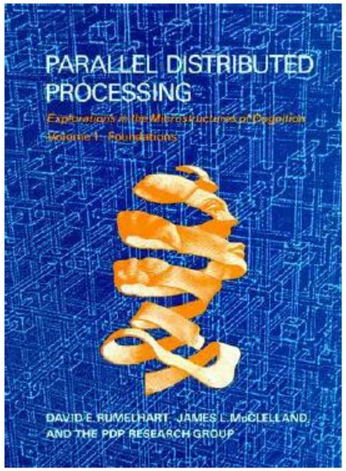


Neural networks



Neural network learning

- Feed input to NN
  - Compute forward pass and get output
  - Compare output to label and compute error
  - Update weights/biases based on error.



- But how?
  - Compute derivative of weights with respect to error (chain rule)
  - Can be efficiently done using dynamic programming (backpropagation algorithm)

Regression

- Broadly similar to classification
  - But predicting real valued numbers
  - Can modify decision trees, neural nets for regression (e.g. output layer is real valued, use MSE)

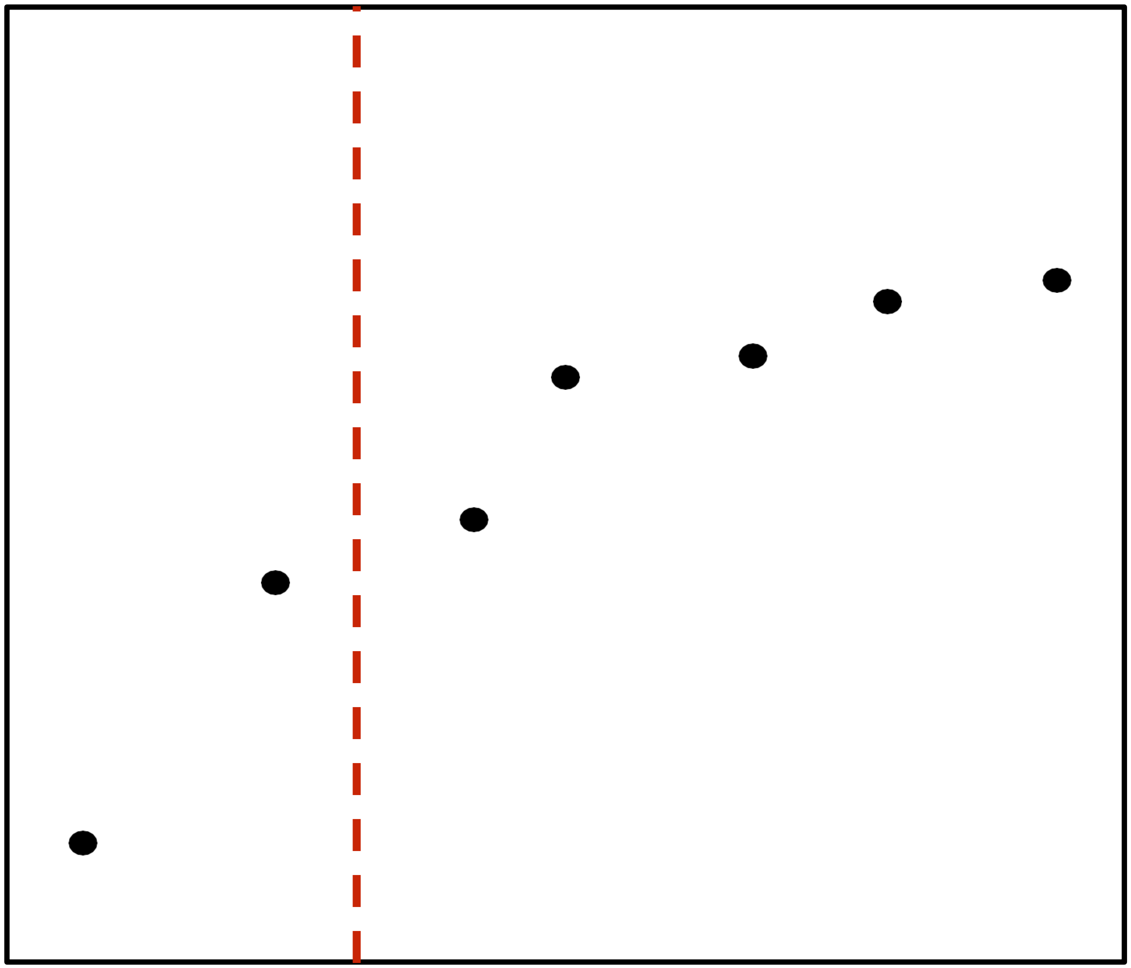
- Most ML methods are parametric:
  - Characterised by setting a few parameters
  - $y = f(x, w)$

Nonparametric regression

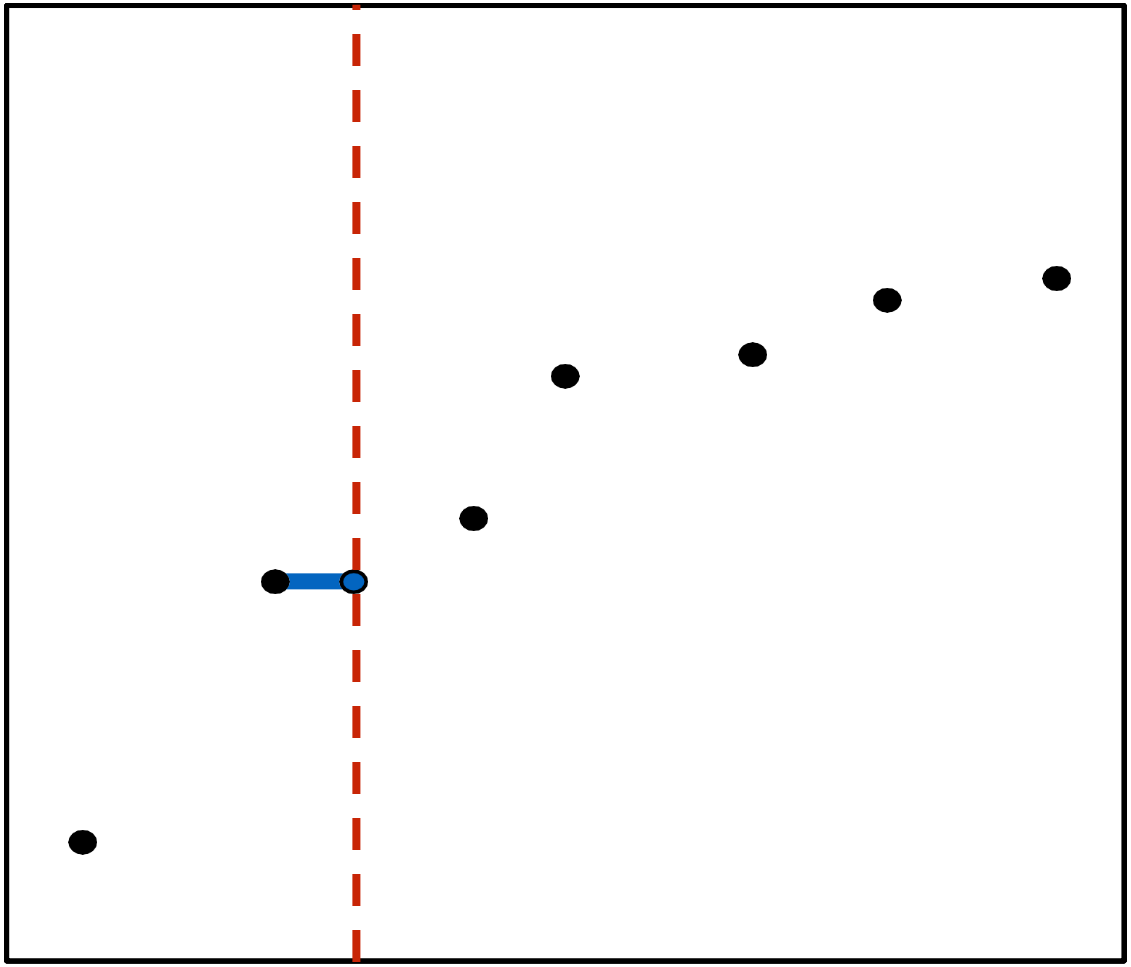
- Alternative approach:
  - Let the data speak for itself
  - No finite-sized parameter vector
  - Usually more interesting decision boundaries

- Given $X = \{x_1, \dots, x_n\}$, $\mathrm{Y} = \{y_1, \dots, y_n\}$, distance metric $D(x_i, x_j)$
  - For a new data point x_{new}:
    - Find k nearest points in x (measured by D)
    - Set y_{new} to the (weighted by D) average y_i labels

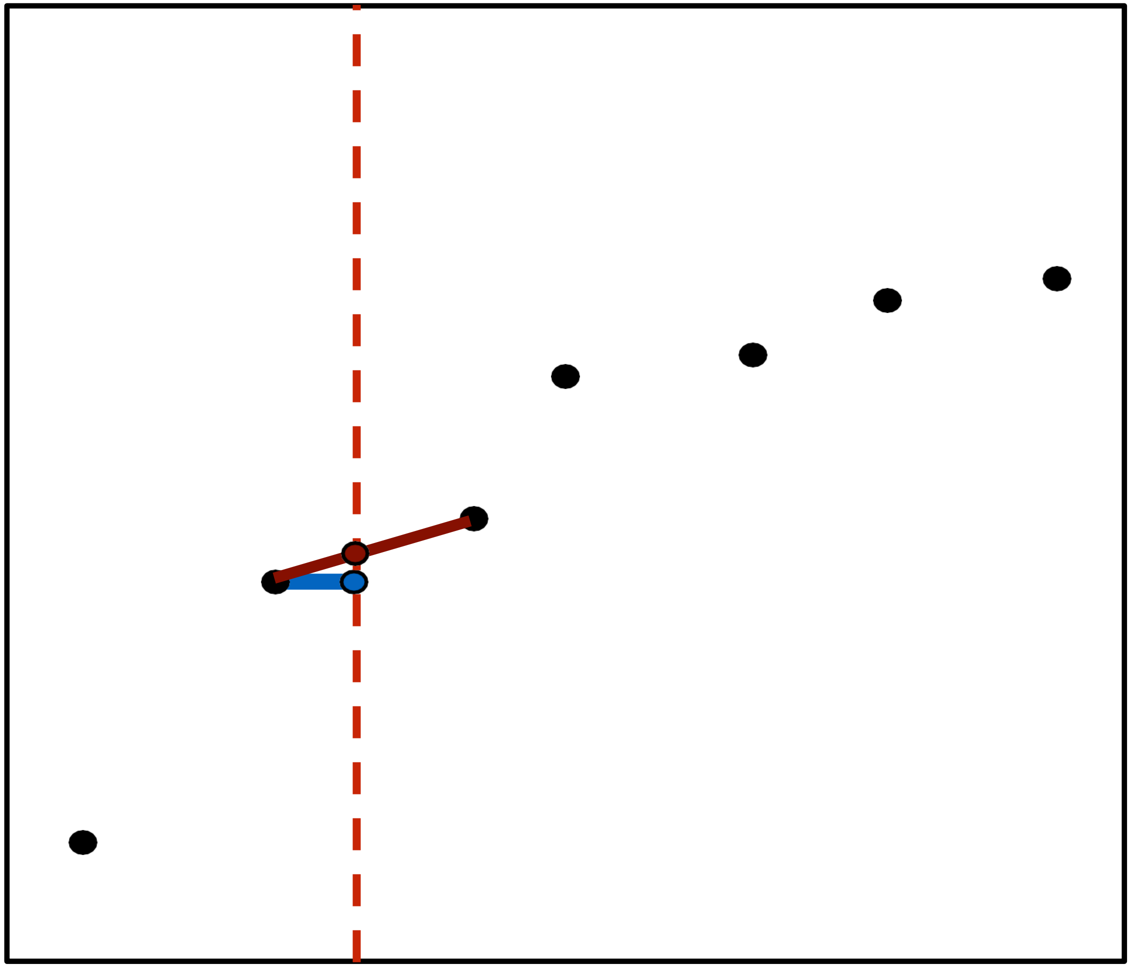
Nonparametric regression



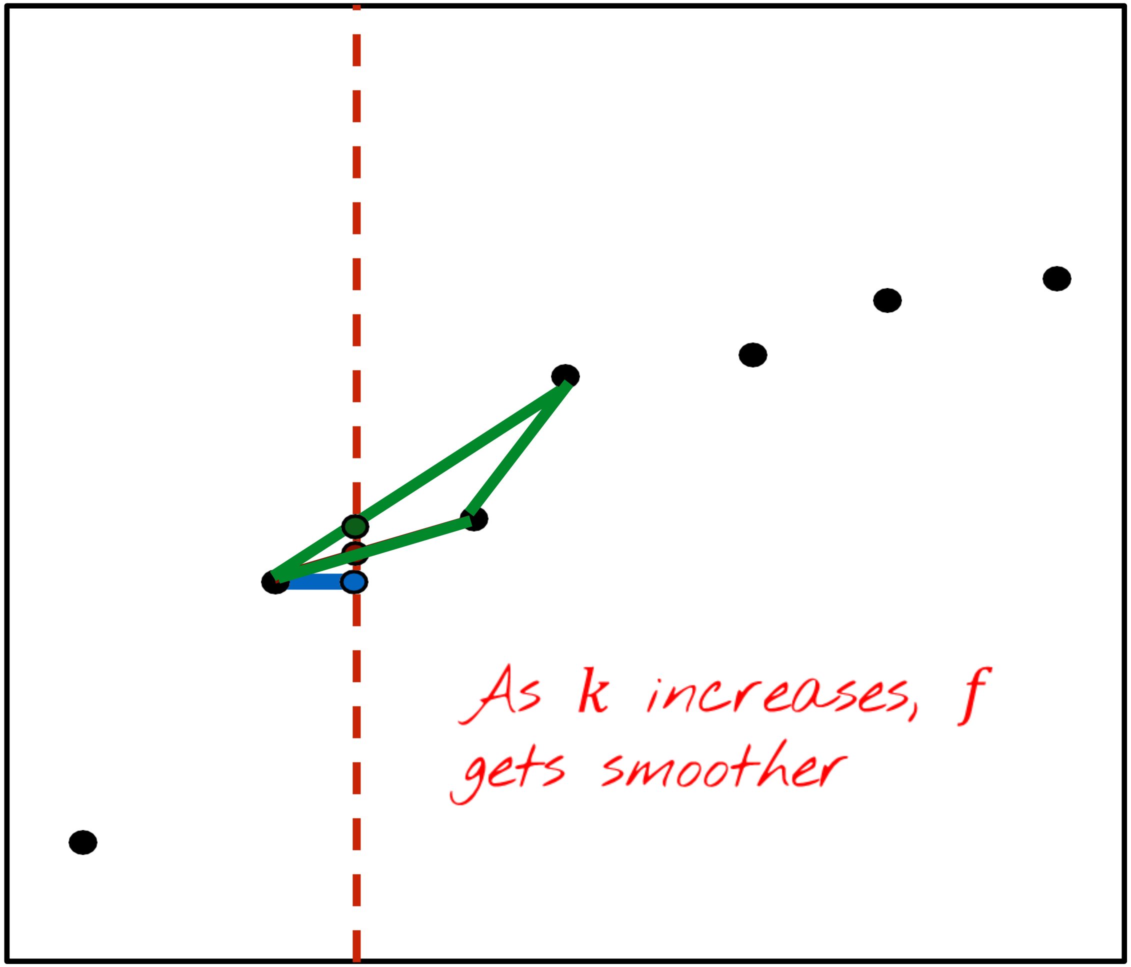
Nonparametric regression



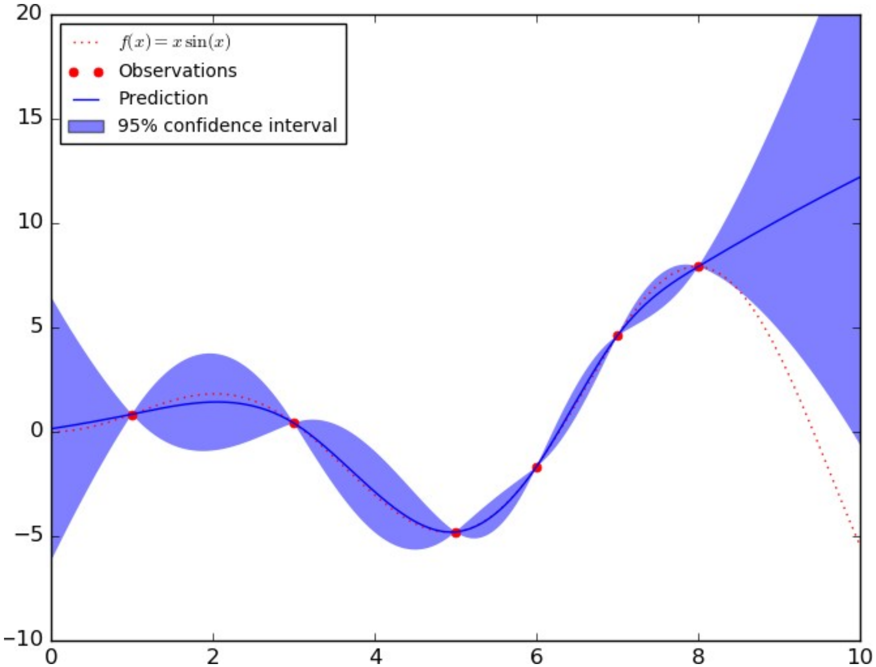
Nonparametric regression

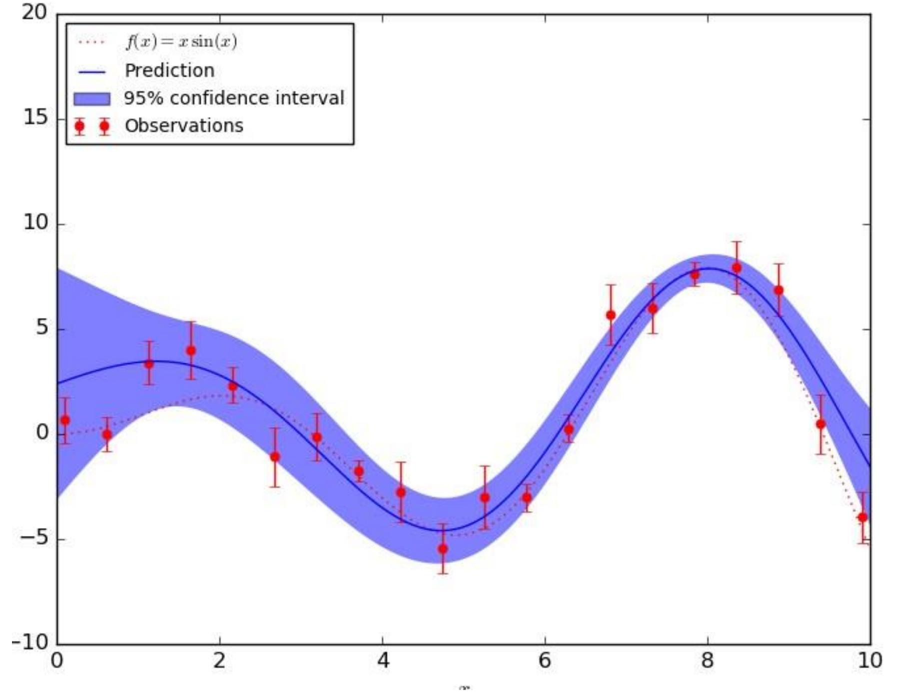


Nonparametric regression



Gaussian processes





Unsupervised learning

- Input:
  - $X = \{x_1, \ldots, x_n\}$
- Try understand structure of the data

- E.g. How many types of cars? How can they vary?

Clustering

- One particular type of unsupervised learning:
  - Split the data into discrete clusters
  - Assign new data points to each cluster
  - Clusters can be thought of as types

- Formally:
  - Given data points $X = \{x_1, \dots, x_n\}$
  - Find number of clusters k
    - Assignment function $f(x) = \{1, \dots k\}$

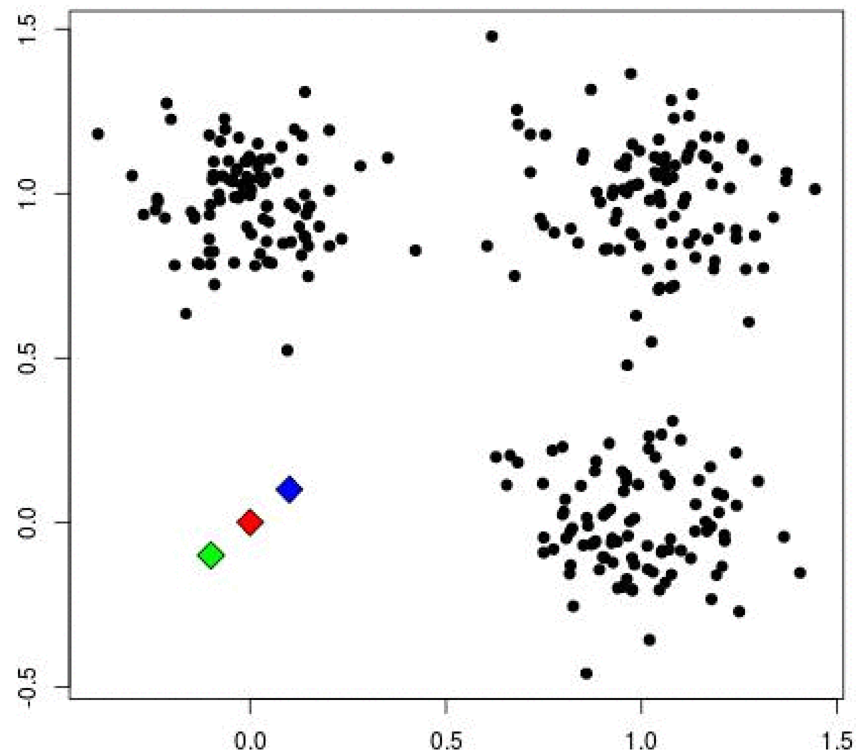
K-means

- One approach
  - Pick k
  - Place k ponts ("means") in the data
  - Assign new point to ith cluster if nearest to ith mean

- Place k means at random
- Assign all points in the data to each "mean"
  - $f(x_j) = i$ such that $d(x_j, \mu_i) \le d(x_j, \mu_l) \forall l \neq i$
  - Move each "mean" to mean of assigned data and repeat

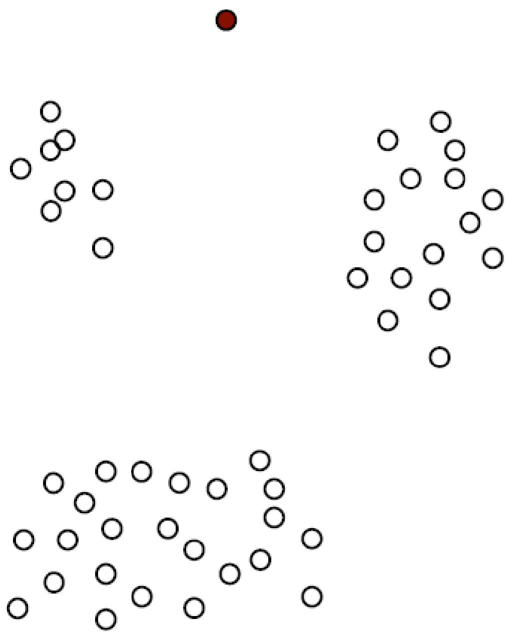
$$\mu_i = \sum_{v \in C_i} \frac{x_v}{|C_i|}$$

Start!



Density estimation

- Clustering can answer *which cluster?*, but not *does this belong?*

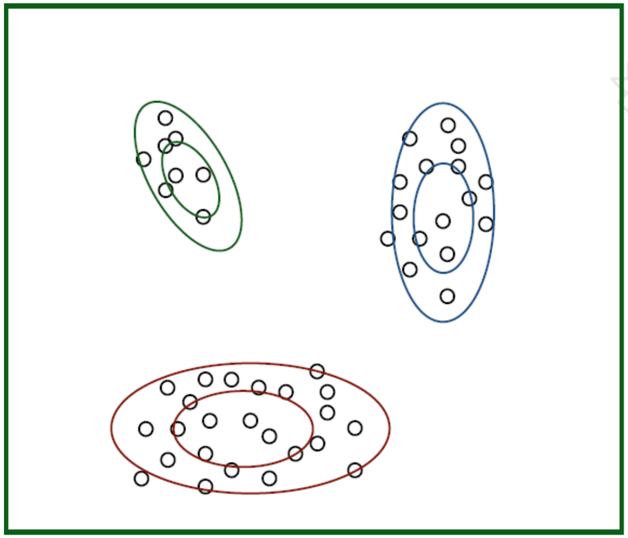


Density estimation

- Estimate the distribution the data is drawn from
- This allows us to evaluate the probability that a new point is drawn from the same distribution as the old data
- Formally:
  - Given data points $X = \{x_1, \dots, x_n\}$
  - Find PDF $P(X)$

GMM

- Model data as mixture of Gaussians
- Each Gaussian has its own mean and variance
- Each has its own weight (sums to 1)
  - Weighted sum of Gaussians is still a PDF



GMM

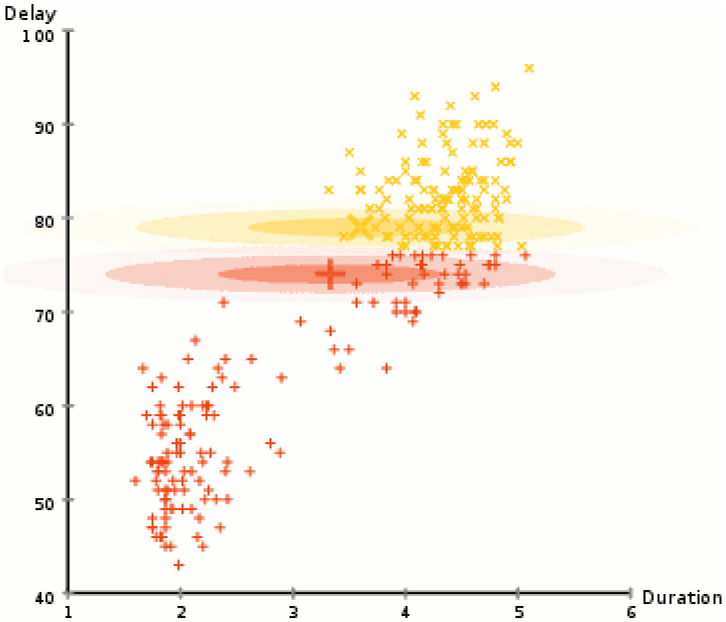
- Almost the same as k-means
- Place means at random, set variances high
- Assign all points to highest probability distribution

$$C_i = \{x_v | N(x_v | \mu_i, \sigma_i^2) > N(x_v | \mu_j, \sigma_j^2) \forall j\}$$

- Set mean, variance and weights to match assigned data and repeat

$$\mu_i = \sum_{v \in C_i} \frac{x_v}{|C_i|}; \sigma_i^2 = variance(C_i); w_i = \frac{|C_i|}{\sum_j |C_j|}$$

GMM



Nonparametric density estimation

- Parametric:
  - Define parameterised model (e.g. Gaussian)
  - Fit parameters
  - Done!
- Key assumptions
  - Data is distributed according to parameterised form
  - We know which parameterised form in advance

- What is shape of the distribution over human faces?

Nonparametric density estimation

- Alternative:
  - Avoid fixed parameterised form
  - Compute density estimate directly from data

- Kernel density estimator

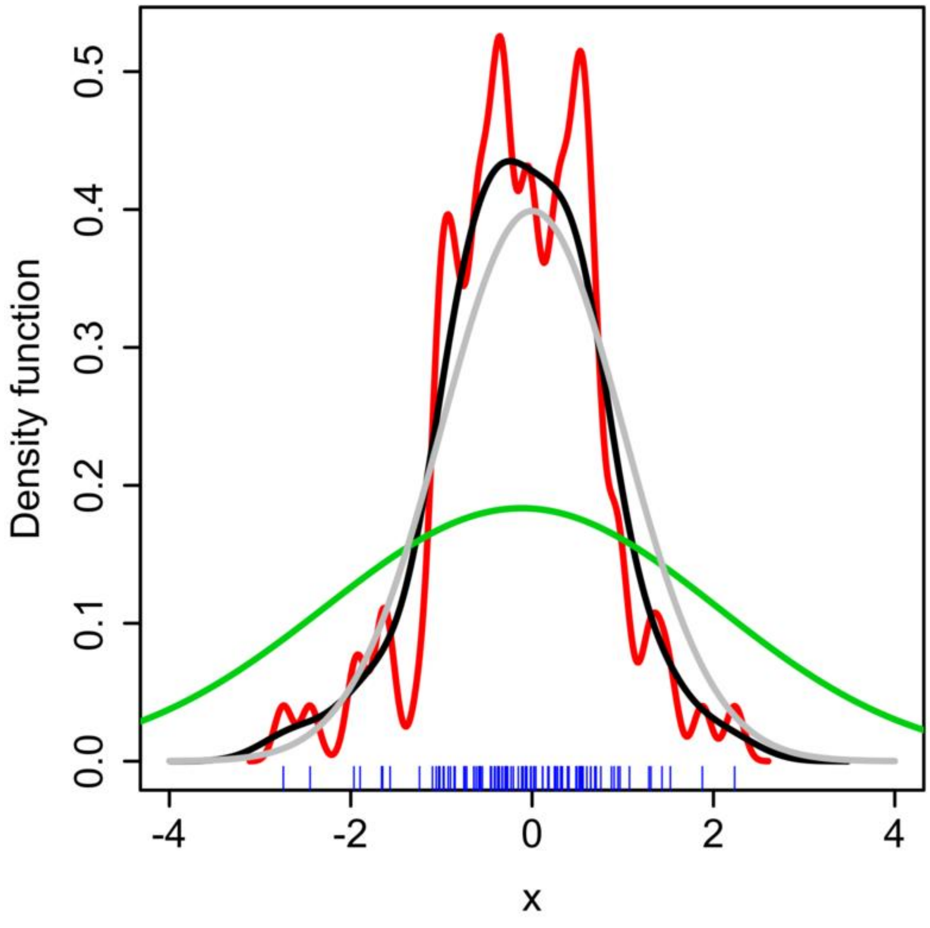
$$PDF(x) = \frac{1}{nb}\sum_{i=1}^{n} D\left(\frac{x_i - x}{b}\right)$$

- Where
  - D is a special kind of distance metric called a kernel
    - Falls away from 0, integrates to 1
  - b is bandwidth. Controls how fast kernel falls away

Nonparametric density estimation

- $$PDF(x) = \frac{1}{nb}\sum_{i=1}^{n} D(\frac{x_i - x}{b})$$
- Kernel: lots of choices, Gaussian often works in practice
- Bandwidth:
  - High: distant points have higher "contribution" to sum
  - Low: distant points have lower

Nonparametric density estimation



Dimensionality reduction

- $X = \{x^1, \dots, x^n\}$, each x^i has m dimensions: $x^i = [x_1, \dots, x_m]$

- If m is high, data can be hard to deal with
  - High-dimensional decision boundary
  - Need more data
  - But data is often not really high-dimensional

- Dimensionality reduction:
  - Reduce or compress the data
  - Try not to lose too much!
  - Find intrinsic dimensionality

Dimensionality reduction

- Often can be phrased as a projection:

$$f : X \rightarrow X'$$

- Where
  - $|X'| \ll |X|$
  - Our goal: retain as much sample variance as possible
- Variance captures what varies within the data

Principal component analysis

- Gather data $x^1, \ldots, x^n$
  - Adjust data to be zero-mean
  - Compute covariance matrix C
  - Compute unit eigenvectors V_i and eigenvalues v_i of C
- Each V_i is a direction, and each v_i is its importance - the amount of the data's variance it accounts for
- New data points:

Original data point

$$\hat{x}^i = [V_1, \ldots, V_p] x^i$$

Compressed data point

Compression matrix V

PCA

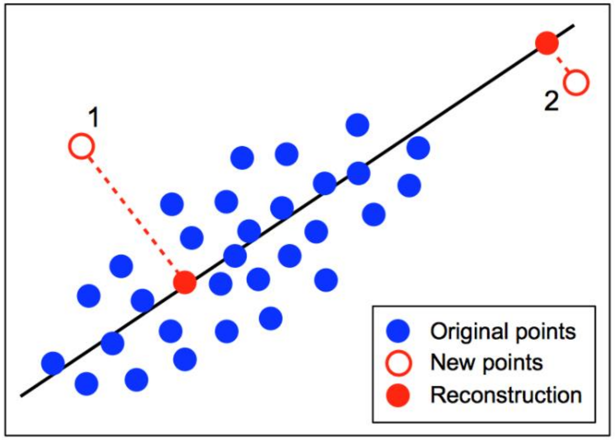
- To recover original data point:

$$\bar{x}^i = V^{-1}\hat{x}^i$$

$$\bar{x}^i = V^T\hat{x}^i$$

V is orthonormal so $V^{-1} = V^T$

- Every data point is expressed as a linear combination of basis (eigenvenctor) functions

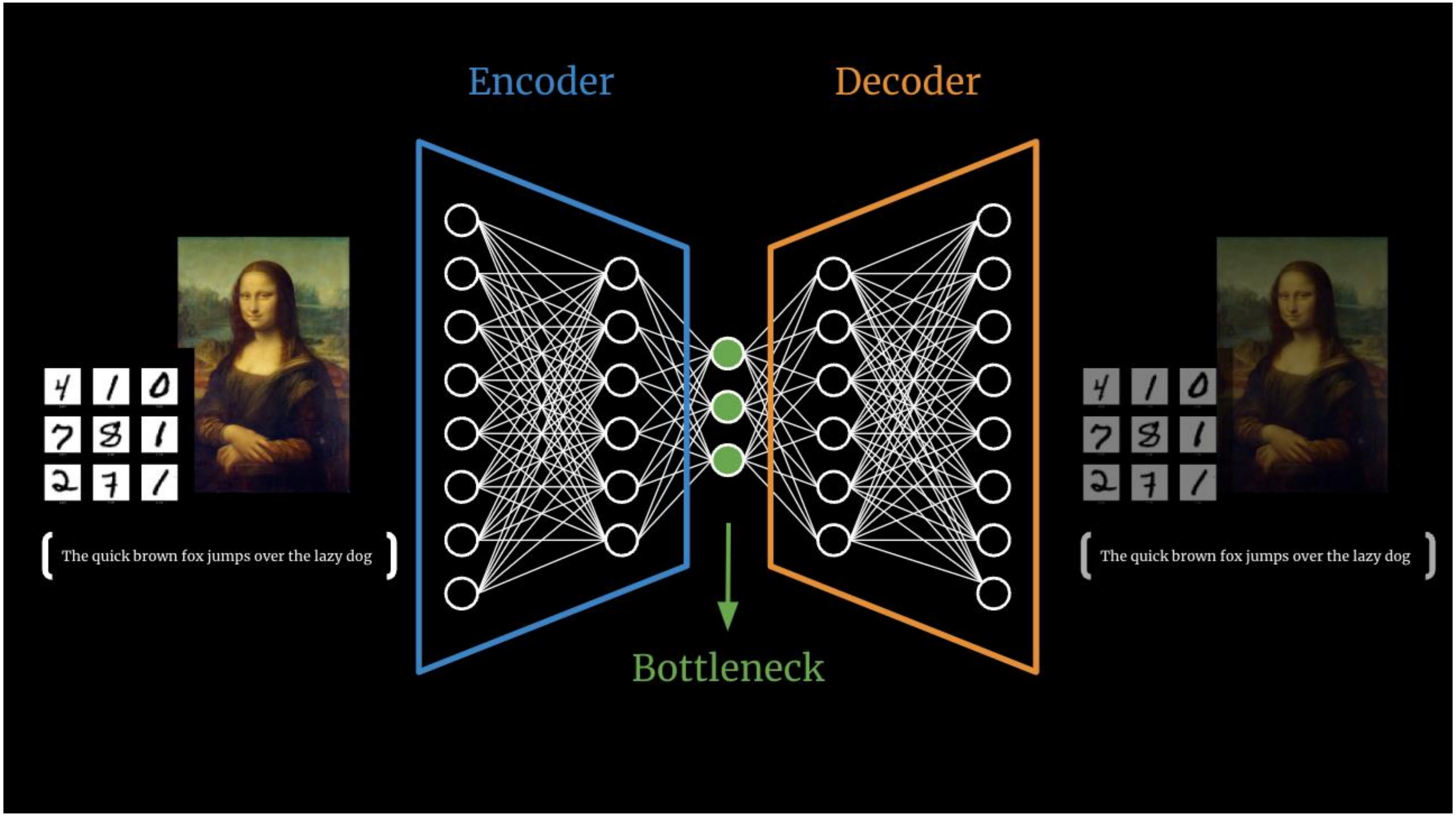


Autoencoders

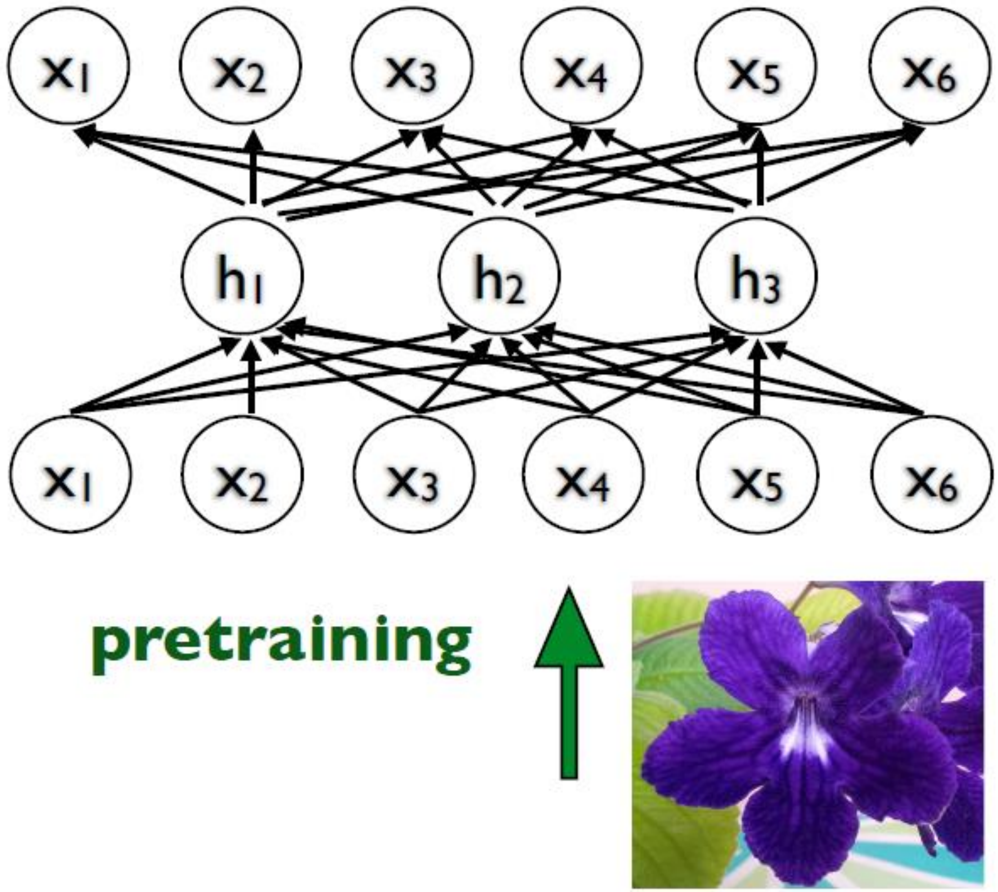
- Fundamental issue with PCA
  - Linear reconstruction
- Can we use a nonlinear method for construction?
  - Extract more complex relationships within the data.
  - Remove “linear reconstruction” property.

- One idea: train neural network to reproduce output

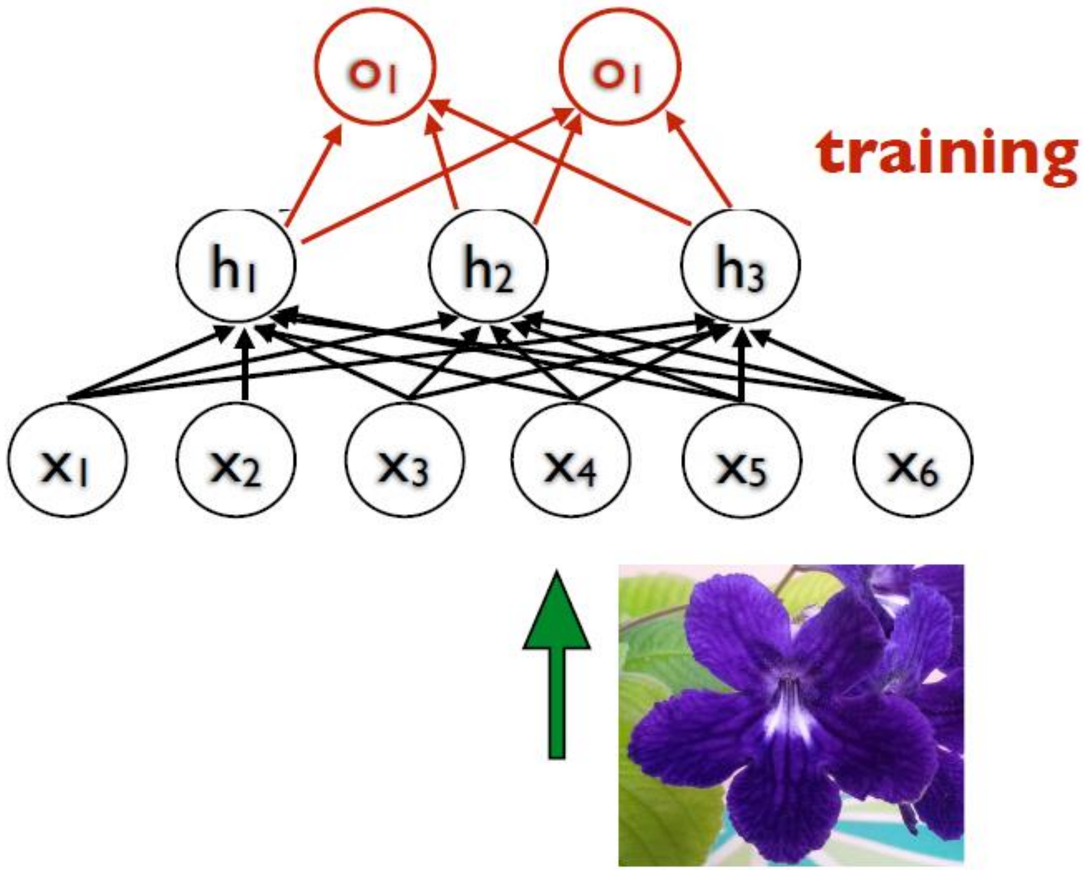
Autoencoders



Autoencoders for classification



Autoencoders for classification



Data mining

- Most common application of unsupervised learning
  - Given large corpus of data, what can be learned?

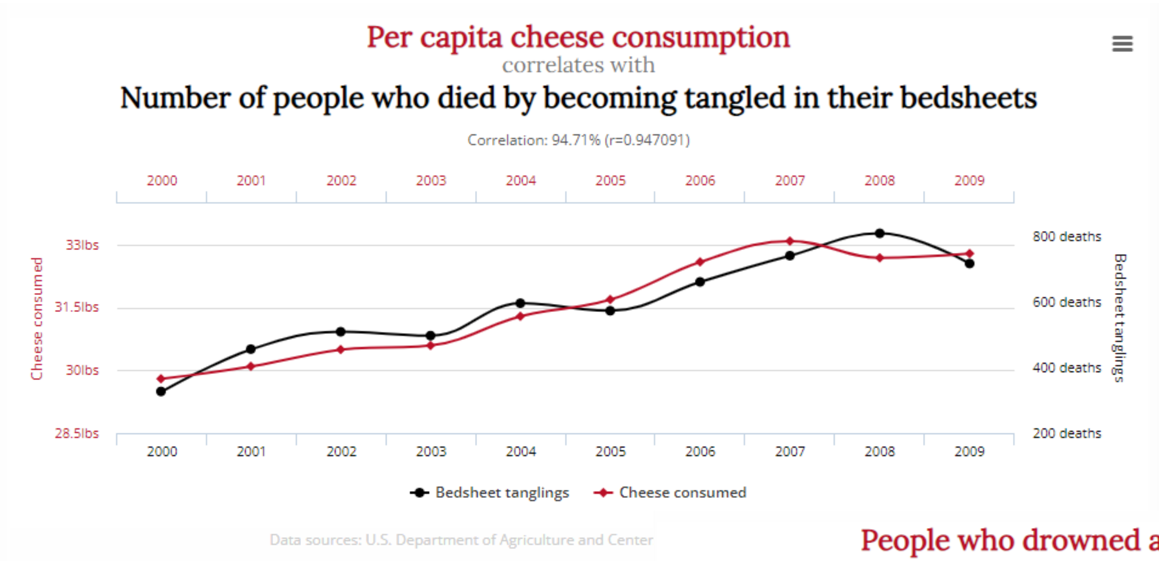
Data mining

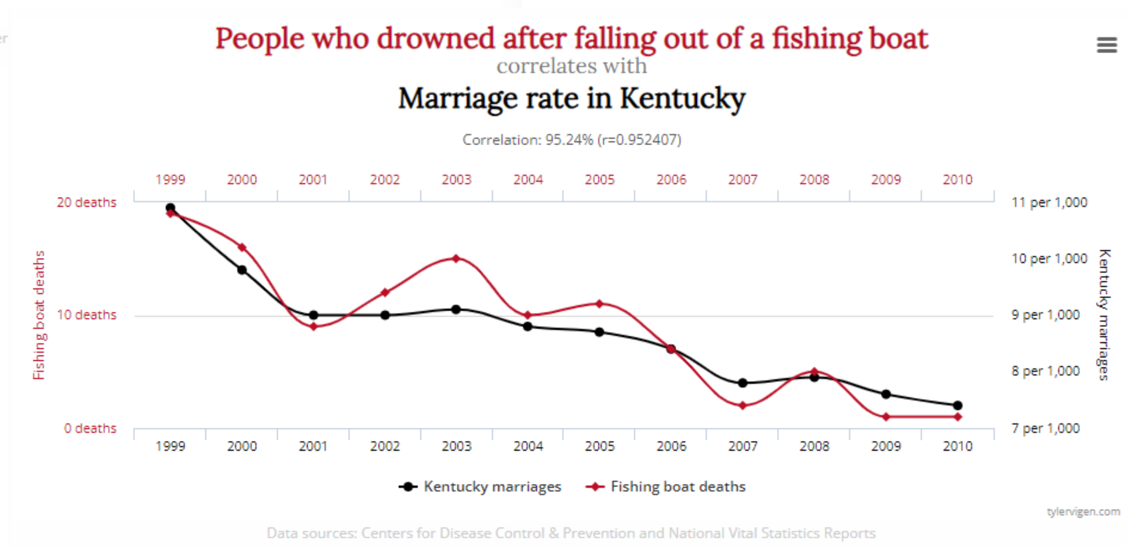
“As Pole’s computers crawled through the data, he was able to identify about 25 products that, when analyzed together, allowed him to assign each shopper a “pregnancy prediction” score. More important, he could also estimate her due date to within a small window, so Target could send coupons timed to very specific stages of her pregnancy.

One Target employee I spoke to provided a hypothetical example. Take a fictional Target shopper named Jenny Ward, who is 23, lives in Atlanta and in March bought cocoa-butter lotion, a purse large enough to double as a diaper bag, zinc and magnesium supplements and a bright blue rug. There’s, say, an 87 percent chance that she’s pregnant and that her delivery date is sometime in late August.”

https://www.nytimes.com/2012/02/19/magazine/shopping-habits.html

Spurious correlations





https://www.tylervigen.com/spurious-correlations