7: Catchup Session & very short intro to Other Classifiers

Non-examinable

Machine Learning and Real-world Data (MLRD)

Simone Teufel (based on slides by Paula Buttery and Weiwei Sun)

What happens in a catchup session?

- Lecture and practical session as normal
- New material in lecture is non-examinable
- Main purpose: catch up on all ticks in segment
- You can also attempt some starred ticks.
- Demonstrators help as per usual.

Naive Bayes is a probabilistic classifier

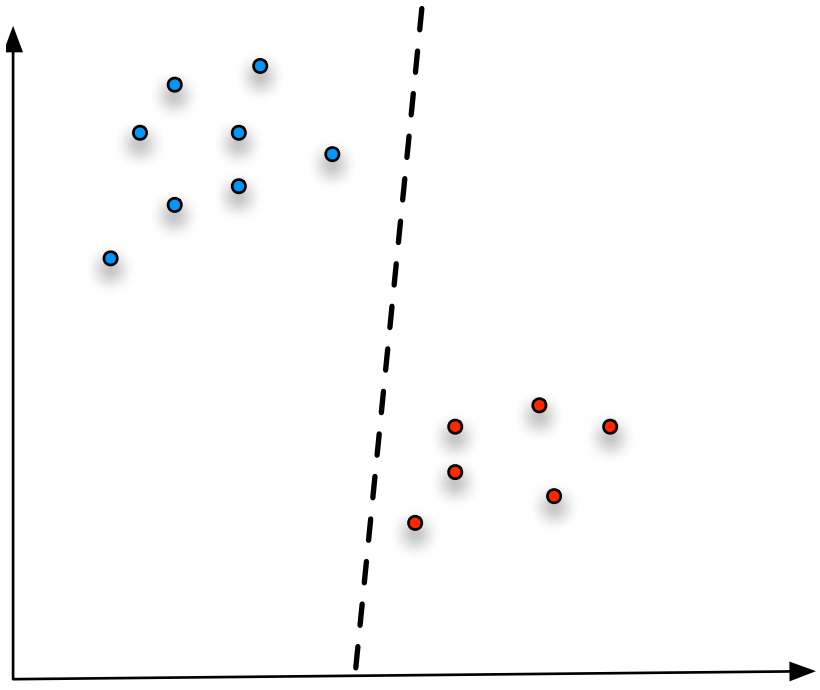
- Given a set of input features a probabilistic classifier provide a distribution over classes.
- That is, for a set of observed features O and classes $c_1 ... c_n \in C$ gives $P(c_i|O)$ for all $c_i \in C$
- For us O was the set all the words in a review $\{w_1, w_2, ..., w_n\}$ where w_i is the ith word in a review, $C = \{\text{POS}, \text{NEG}\}$
- We decided on a single class by choosing the one with the highest probability given the features:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|O)$$

An SVM is a popular discriminative classifier

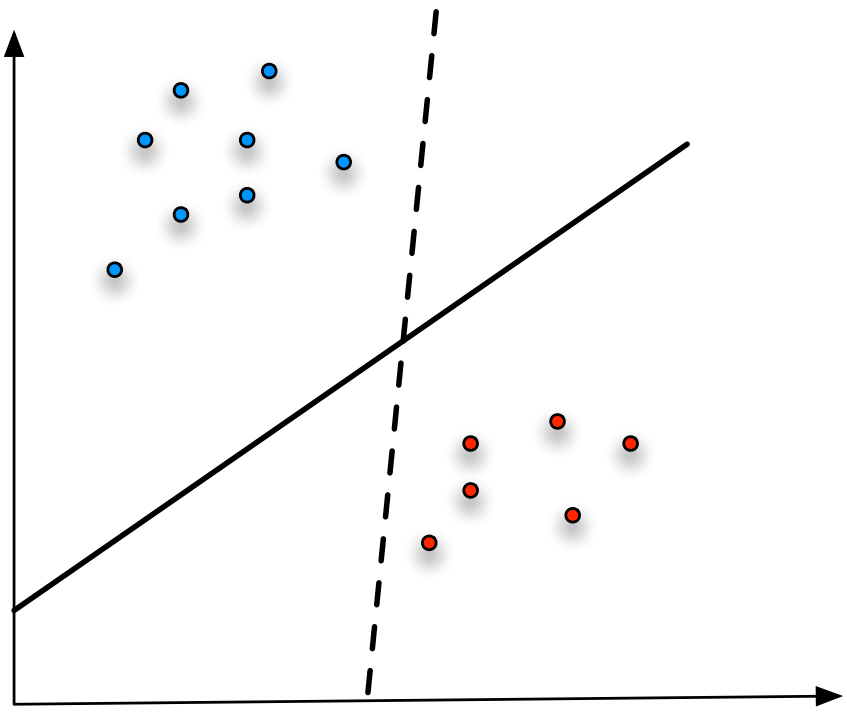
- A Support Vector Machine (SVM) is a non-probabilistic binary linear classifier
- SVMs assign new examples to one category or the other
- SVMs can reduce the amount of labeled data required to gain good accuracy
- SVMs can be efficiently adapted to perform a non-linear classification

SVMs find hyper-planes that separate classes



- Our classes exist in a multidimensional feature space
- A linear classifier will separate the points with a hyper-plane

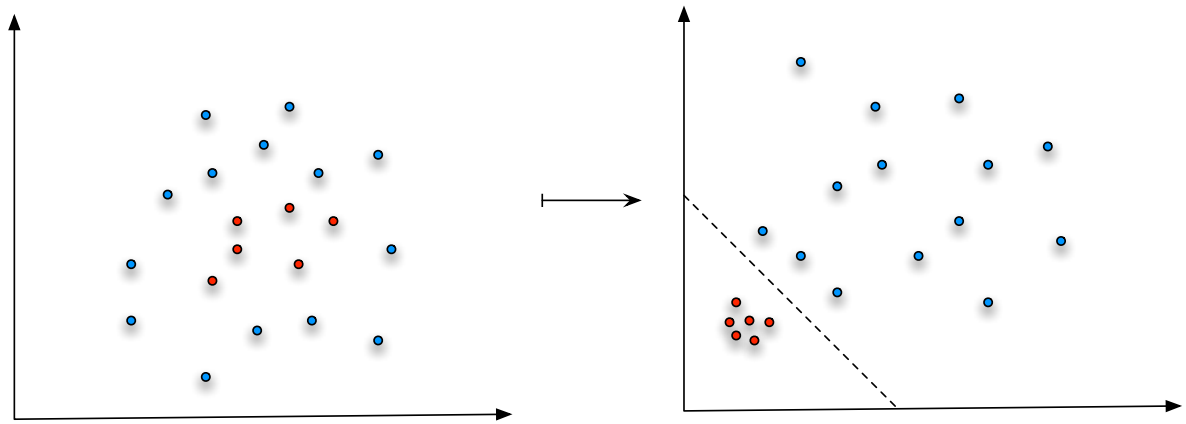
SVMs find a maximum-margin hyper-plane in noisy data



- There are many possible hyper-planes
- SVMs find the best hyper-plane such that the distance from it to the nearest data point from each class is maximised
- i.e. the hyper-plane that passes through the widest possible gap (hopefully helps to avoid over-fitting)

SVMs can be very efficient and effective

- Efficient when learning from a large number of features (good for text)
- Effective even with relatively small amounts of labelled data (we only need points close to the plane to calculate it)
- We can choose how many points to involve (size of margin) when calculating the plane (tuning vs. over-fitting)
- Can separate non-linear boundaries by changing the feature space (using a kernal function)



Choice of classifier will depend on the task

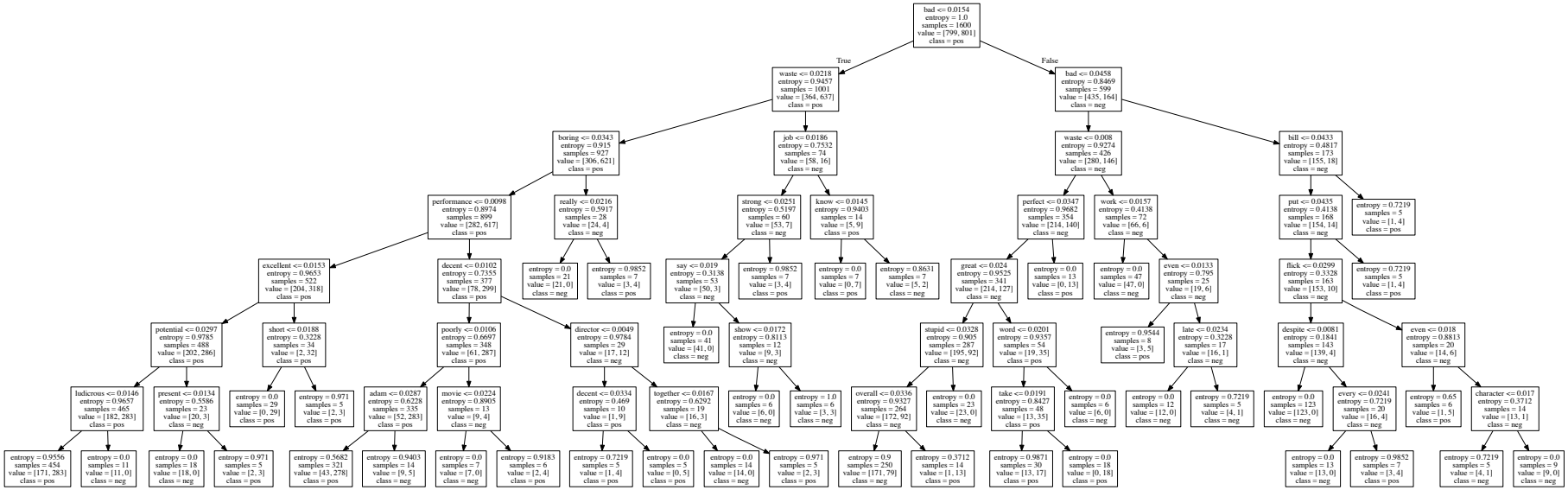
Comparison of a SVM and Naive Bayes on the same task:

- 2000 imdb movie reviews, 1600/400 test/training split
- preprocess with improved tokeniser (lowercased, removed uninformative words, dealt with punctuation, lemmatised words)

	SVM	**Naive Bayes**
Accuracy on train	0.98	0.96
Accuracy on test	0.84	0.80

- But from Naive Bayes I know that *character*, *good*, *story*, *great*, ... are informative features
- SVMs are more difficult to interpret

Decision tree can be used to visually represent classifications



- Simple to interpret
- Can mix numerical and categorical data
- You specify the parameters of the tree (maximum depth, number of items at leaf nodes—both change accuracy)
- But finding the optimal decision tree can be NP-complete

Information gain can be used to decide how to split

- Information gain is defined in terms of entropy H

Entropy of tree node:

$$H(n) = -\sum_{p} p_i \log_2 p_i$$

where p_i are the probabilities of each class at node n

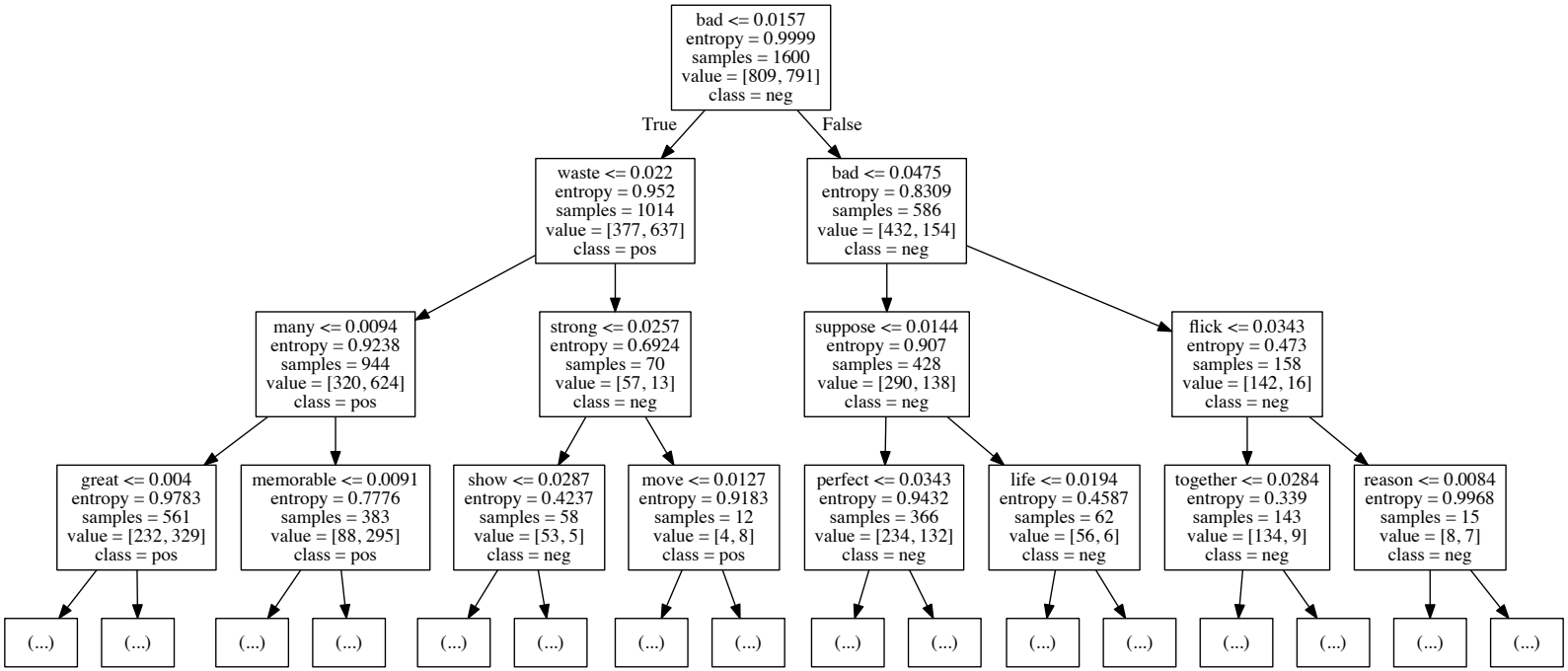
- Information gain I is the reduction in entropy of n achieved by learning the state of the random variable D.

Information gain:

$$I(n, D) = H(n) - H(n|D)$$

where $H(n|D)$ is the weighted entropy of the daughter nodes if we split on D.

Information gain can be used to decide how to split



Results on the movie review dataset:

	SVM	**Naive Bayes**	**DTree** (max depth 7)
Accuracy on train	0.98	0.96	0.80
Accuracy on test	0.84	0.80	0.69

Feedforward Neural Networks

Think about multi-class classificaition:

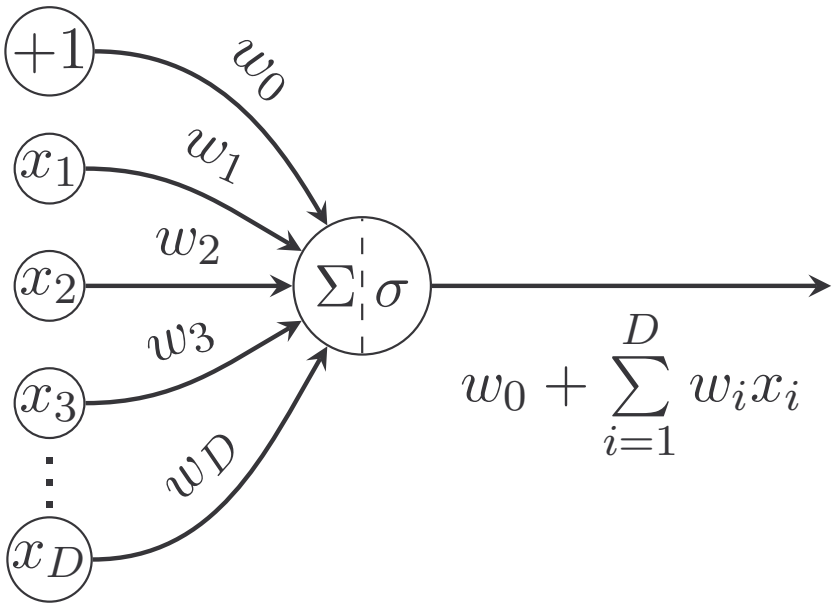
- D – number of features (input)
- K – number of classes (output)
- $\mathbf{x}$ – the input feature vector

Think about a particular class, say y_k. We describe the "friendship" between $\mathbf{x}$ and y_k in the following way:

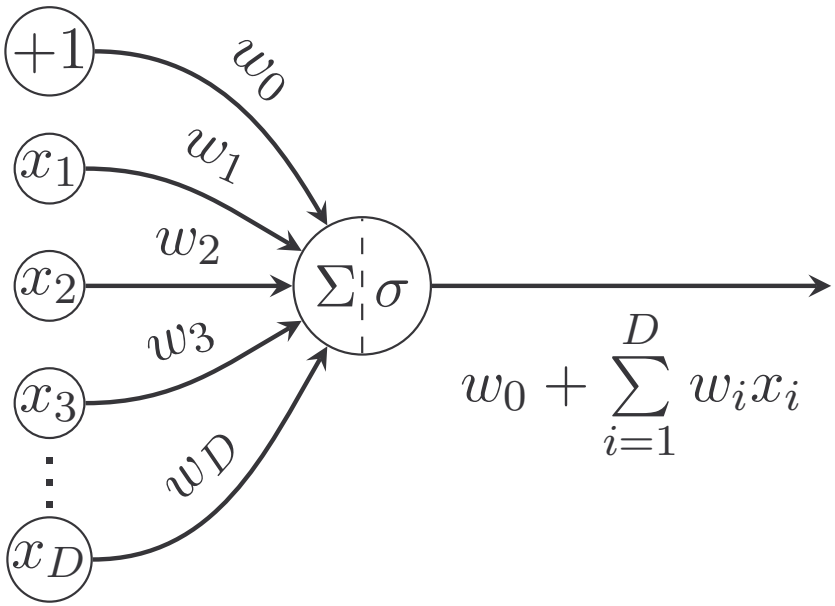
$$\text{score\_function}(x, y_k) = w_0 + \sum_{i=1}^{D} w_i x_i$$

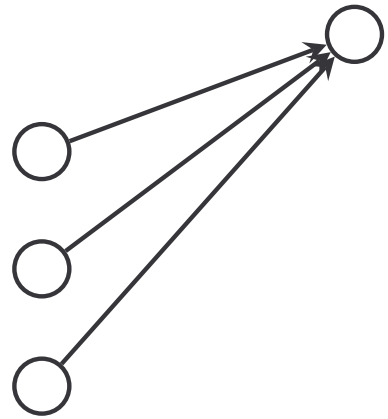
where $\mathbf{w}$ measures how much each feature w_i contributes to y_k.

Feedforward Neural Networks



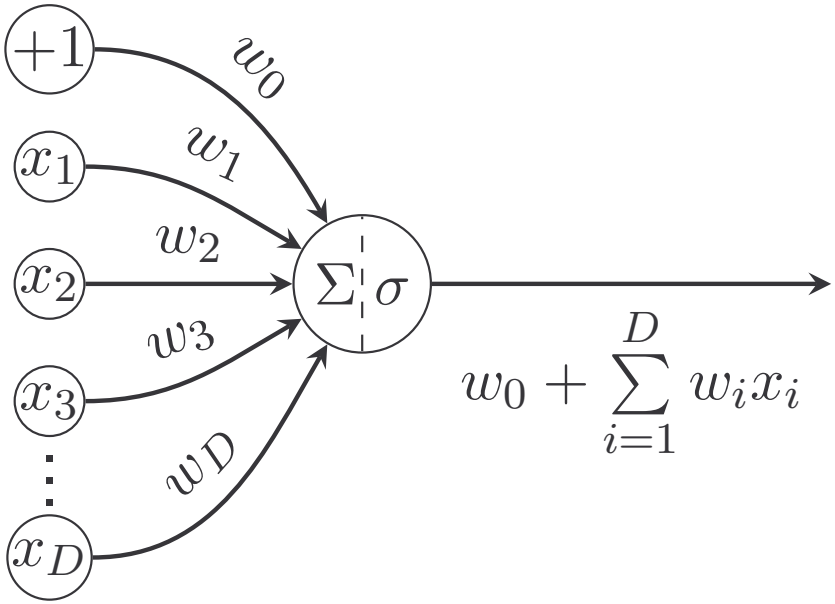
Feedforward Neural Networks

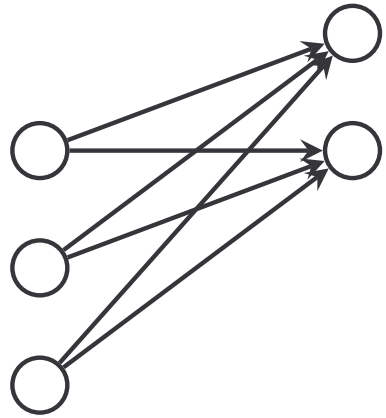




For each class y_k, we do the same thing.

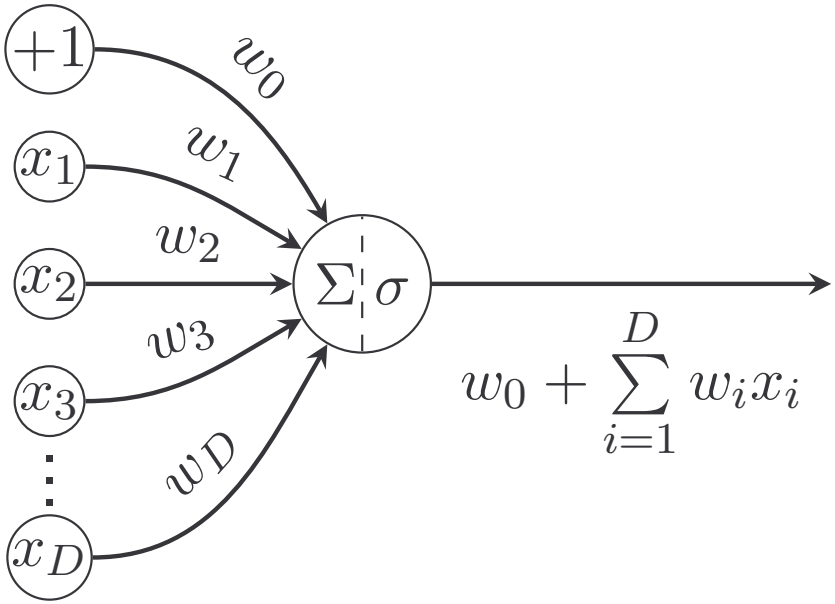
Feedforward Neural Networks

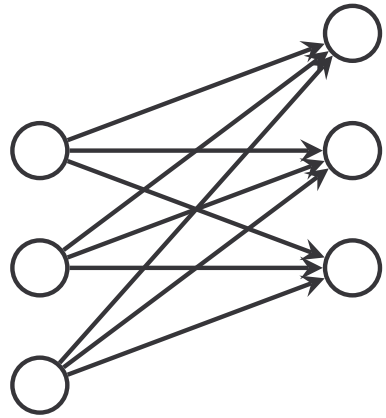




For each class y_k, we do the same thing. **Again**

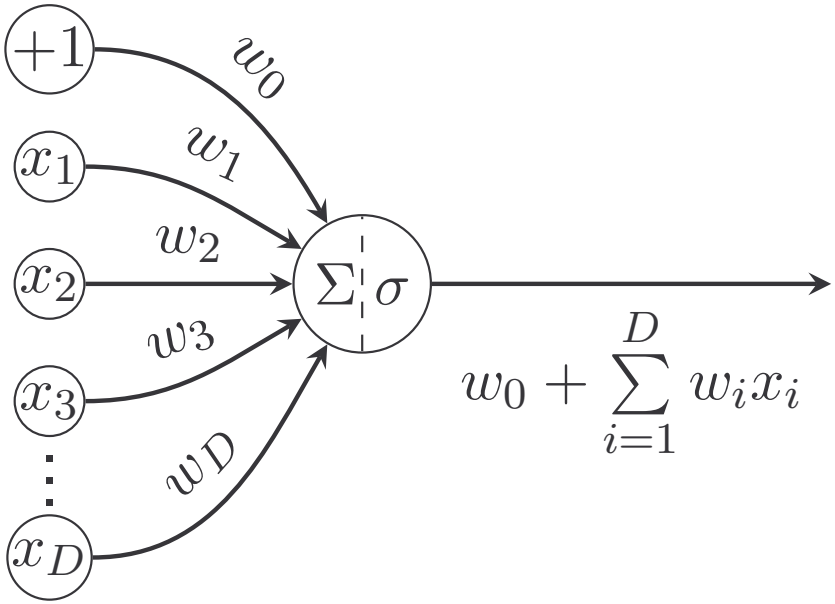
Feedforward Neural Networks

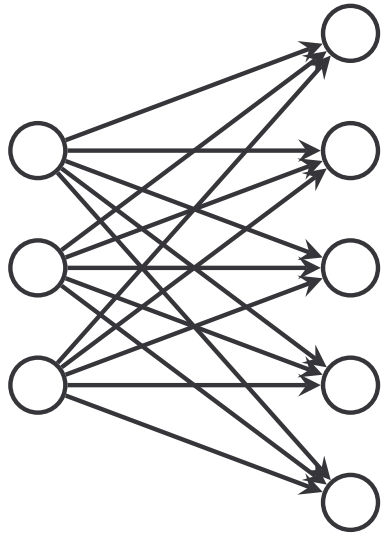




For each class y_k, we do the same thing. Again **and again**

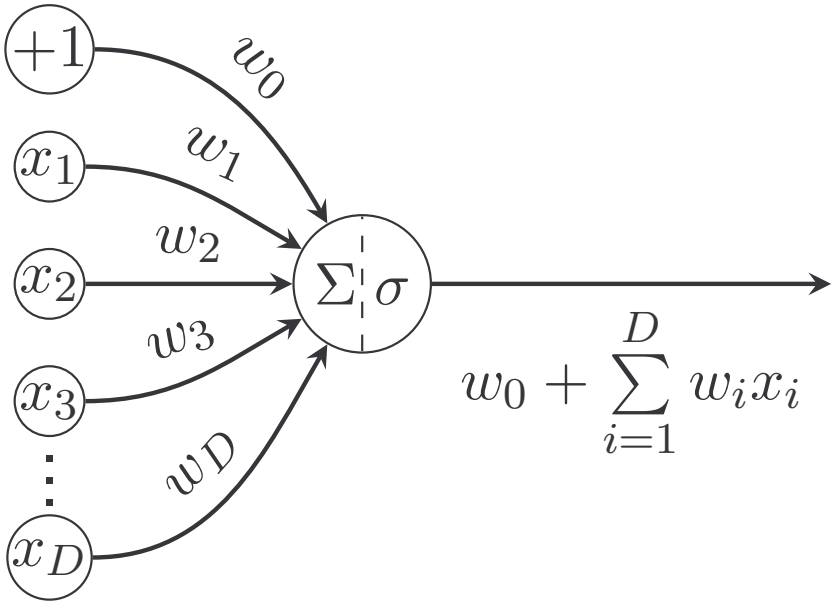
Feedforward Neural Networks

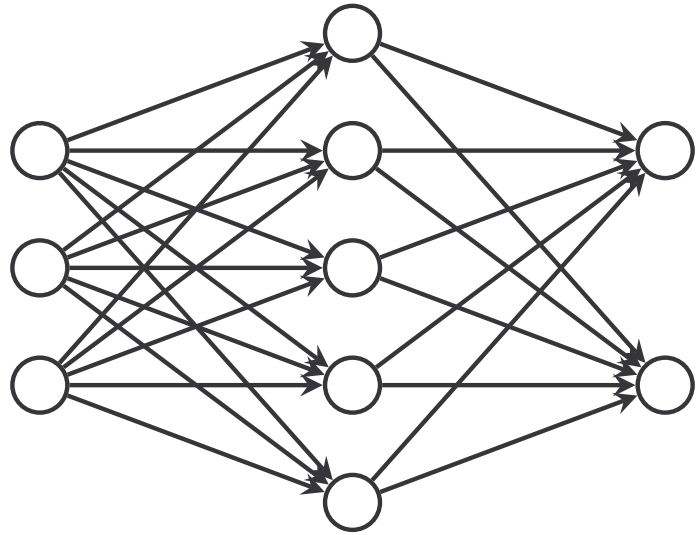




For each class y_k, we do the same thing. Again and again **and again. This is called perceptron, which was invented by Frank Rosenblatt in 1958.**

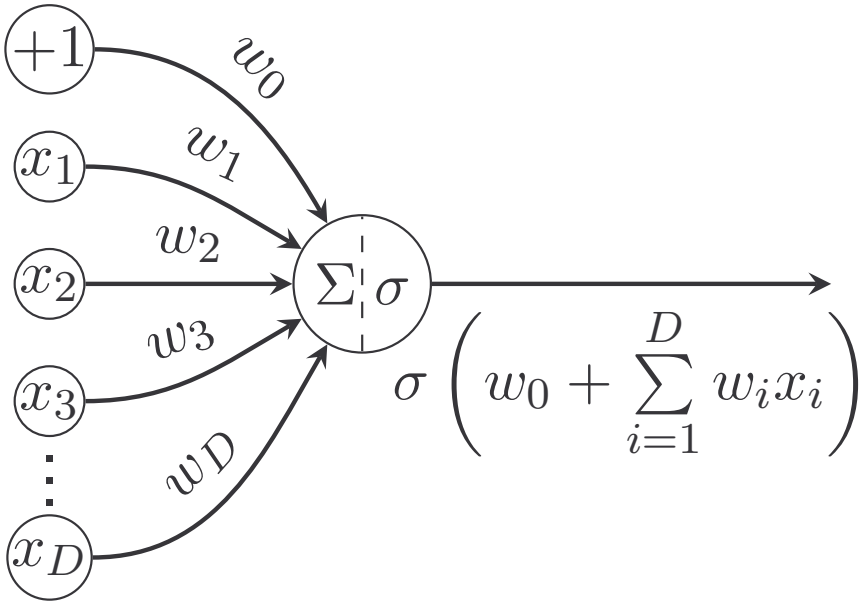
Feedforward Neural Networks

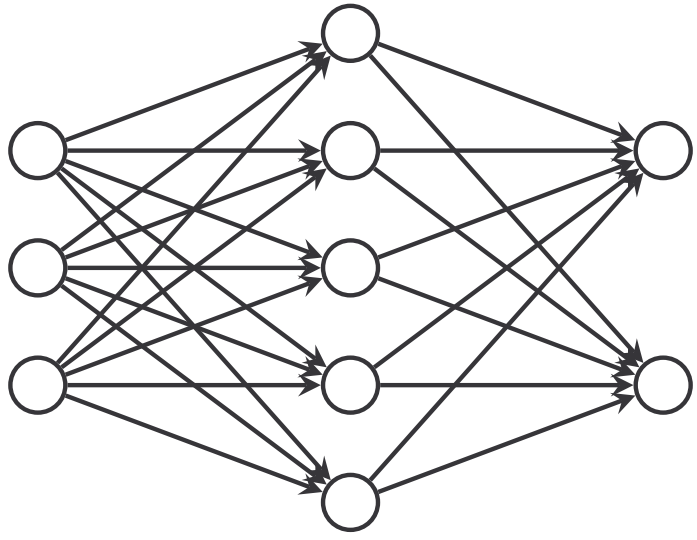




For each class y_k, we do the same thing. Again and again and again. This is called perceptron, which was invented by Frank Rosenblatt in 1958.**Things will be much more fun if we have a stack of perceptrons.**

Feedforward Neural Networks



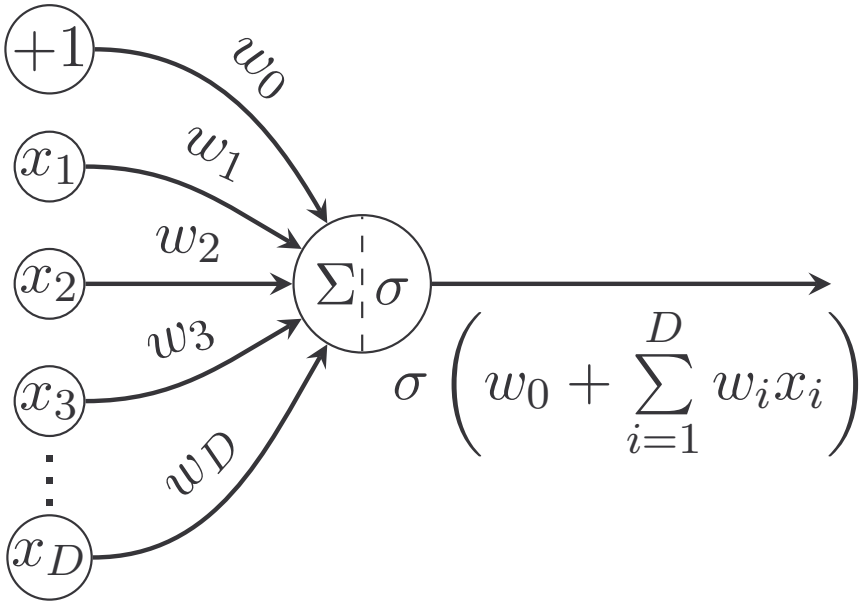


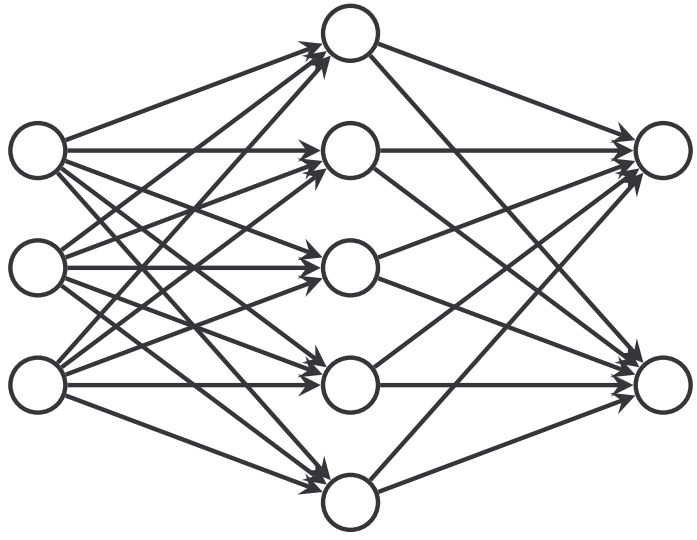
For each class y_k, we do the same thing. Again and again and again. This is called perceptron, which was invented by Frank Rosenblatt in 1958.Things will be much more fun if we have a stack of perceptrons.**Oops, must add something...**

Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$

Otherwise, simple matrix multiplication.

Feedforward Neural Networks



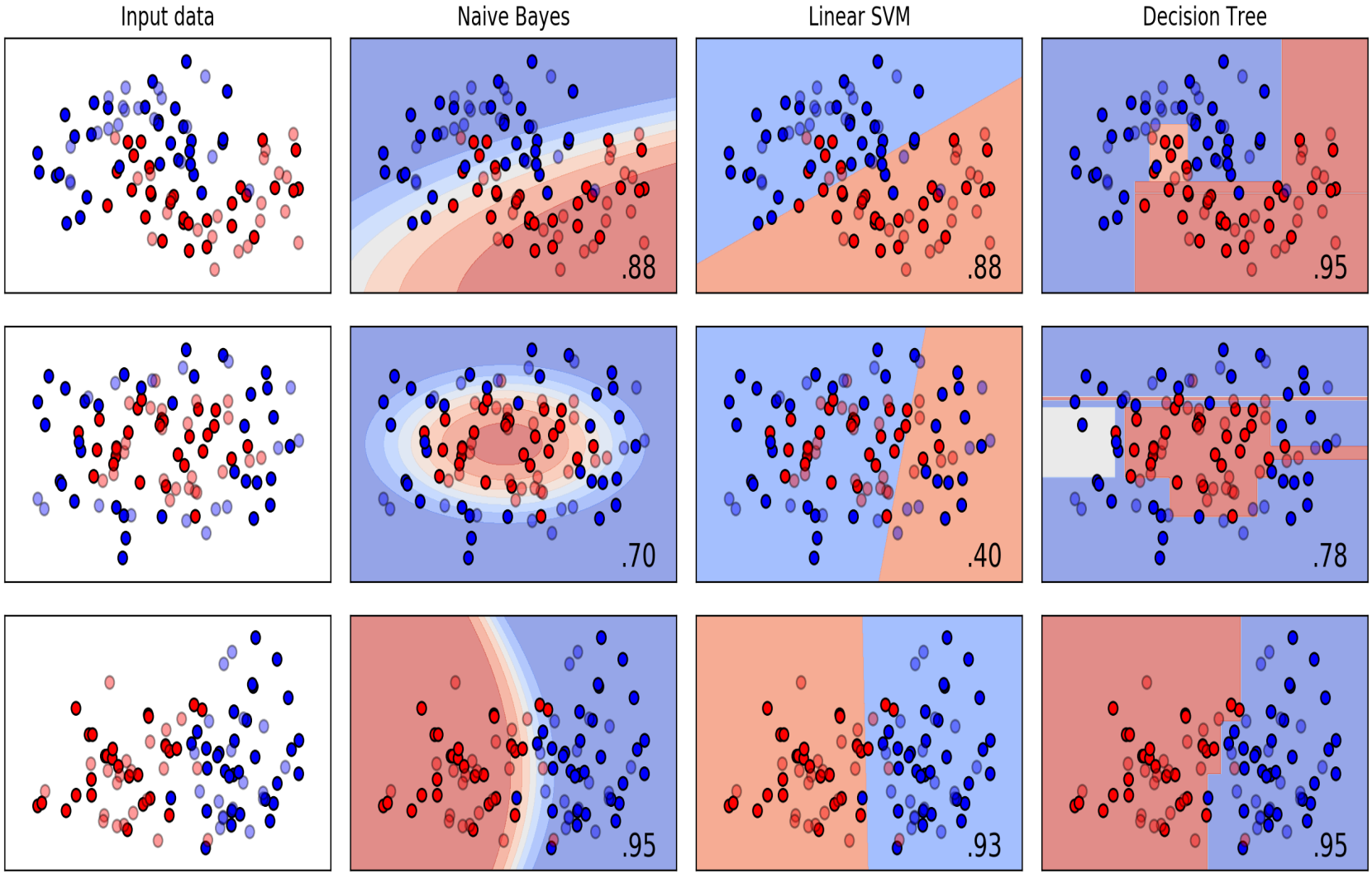


For each class y_k, we do the same thing. Again and again and again. This is called perceptron, which was invented by Frank Rosenblatt in 1958.Things will be much more fun if we have a stack of perceptrons.Oops, must add something...

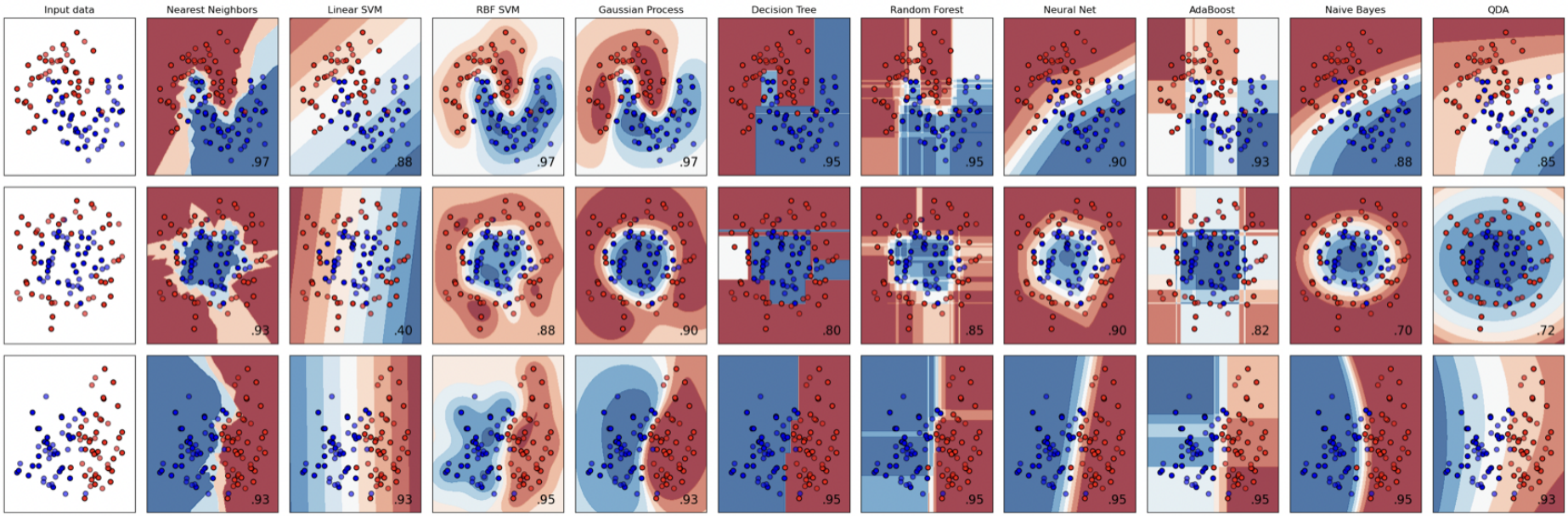
Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$

Otherwise, simple matrix multiplication. **Now you can do non-linear classification.**

Nature of decision Boundaries: artificial data



Modified from SciKit Learn Classifier Comparison


Input data
Nearest Neighbors
Linear SVM
RBF SVM
Gaussian Process
Decision Tree
Random Forest
Neural Net
AdaBoost
Naive Bayes
QDA
.97
.88
.97
.97
.95
.95
.90
.93
.88
.85
.93
.40
.88
.90
.80
.85
.90
.82
.70
.72
.93
.93
.95
.93
.95
.95
.95
.95
.95
.93