

Non-linear models

We've spent a lot of time talking about linear models—models which are parameterized by a weight vector of equal dimension to the input vectors. These are nice, but limited. Here, we consider two very different techniques: nearest neighbor models and decision trees.

Nearest-neighbor (kNN)

1NN—simple intuition: to classify a new data point, just return the class of the closest training point.

k NN—instead of single closest point, average over the k nearest.

δ NN—instead of the k nearest, use as many as fit in a ball of radius δ .

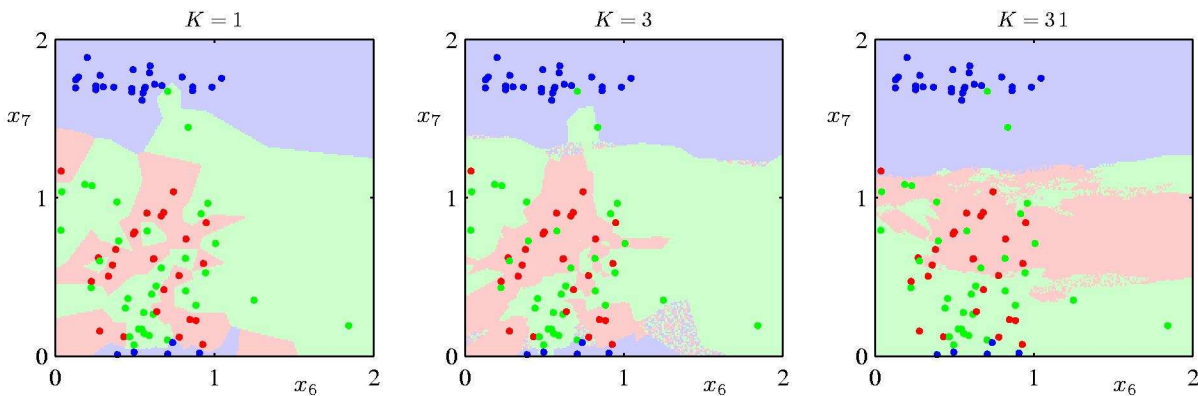
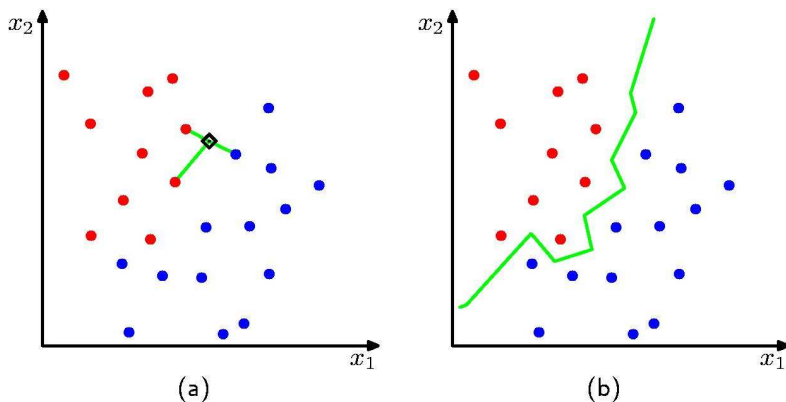
What is the VC dimension of such an algorithm?

How does one train such an algorithm?

Despite its simplicity, k NN is a really really good classifier. (But very sensitive to irrelevant features.)

Can be used also for regression, by averaging responses.

(See figs 2.27a,b and 2.28a,b,c in PRML.)



Decision trees

Idea: suppose we could only use one feature to make a classification decision. Let's choose that feature. Now, look at all example for which this feature is on and choose a single feature to make a classification decision. Then look at all for which the first feature was off. Recurse until no data left.

Two issues: (a) how to choose a single feature, (b) how to choose to stop.

Entropy is a measure of randomness: closeness to uniformity. In particular, how many bits to send a message (on average). If four options A,B,C,D, each with prob $1/4$, then best coding is binary, which gives two bits per character. What if $p(a) = 1/2$, $p(b) = 1/4$, and $p(c) = p(d) = 1/8$. We can code this with 1.75 bits/char on average (how?).

The minimum number of bits is the *entropy*

$$H(X) = - \sum_x p(X = x) \log_2 p(X = x)$$

Zero entropy means deterministic, high entropy means close to uniform.

$H(Y|X)$ is the number of bits needed to send Y , given that both the sender and recipient knew X . (Conditional entropy.)

$$\begin{aligned} H(Y|X) &= \sum_x p(X = x) H(Y|X = x) \\ &= - \sum_x p(X = x) \sum_y p(Y = y|X = x) \log_2 p(Y = y|X = x) \end{aligned}$$

Information gain $IG(Y|X)$ is: i must send Y — how many bits would I save if both ends knew X ?

$$IG(Y|X) = H(Y) - H(Y|X)$$

Idea: choose the feature with the highest information gain.

Stopping: use threshold of either number of elements in leaf or entropy of leaf.

Dealing with real-valued features: if X is real-valued, consider all possible split locations ($X \leq z$ and $X > z$) and find the best z to split. Best = maximum IG. Only need to search over splits that exist in training data.