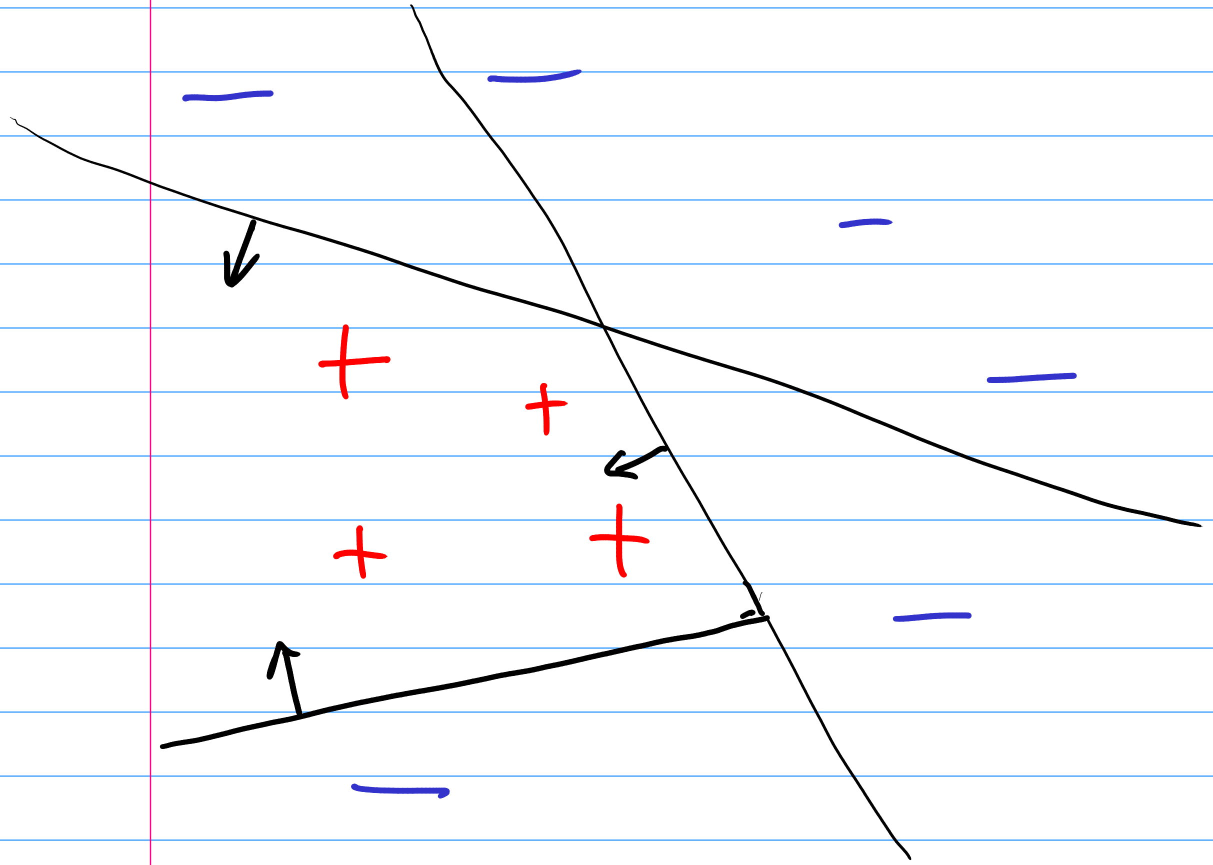
Nonlinear model intro

Marina Meilă-inspired



6 blue $-$s
4 red $+$s

training data

Goals:

K-NN
trees
Kernel methods

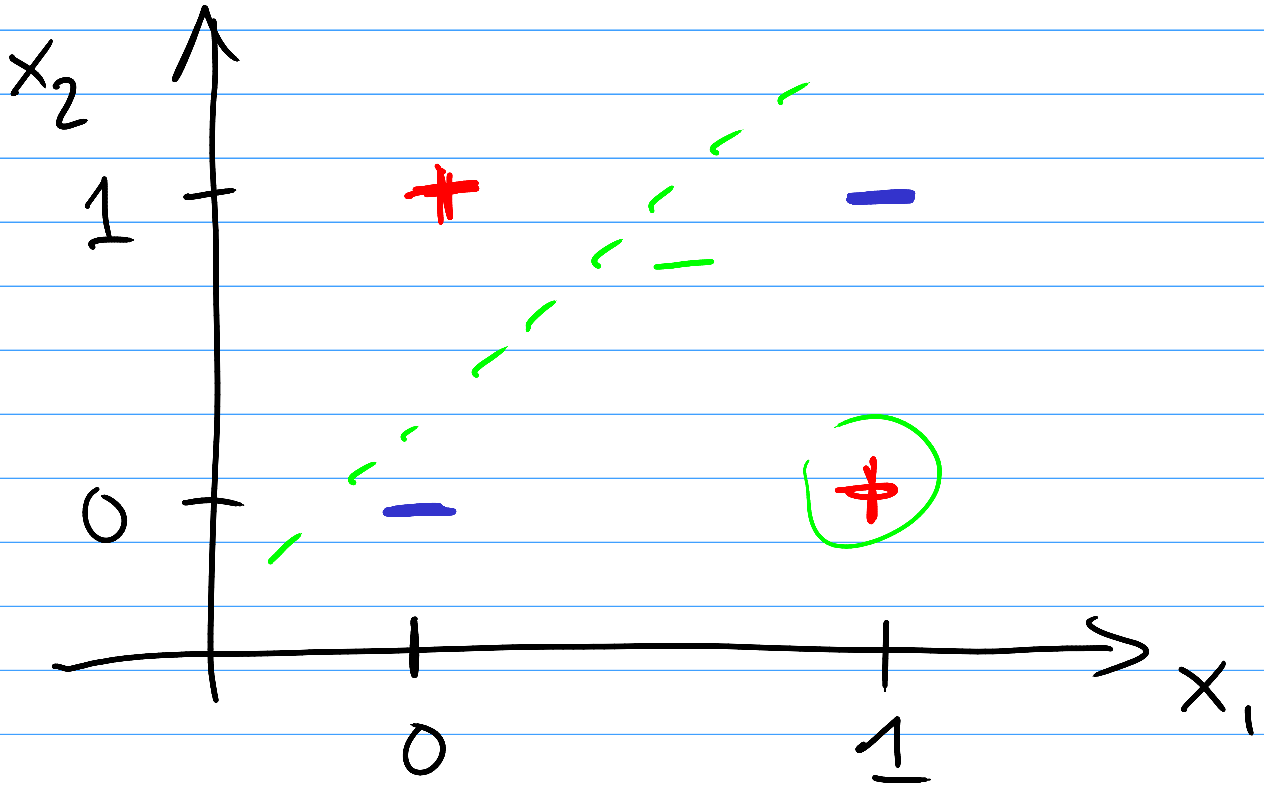
understand form of $f(\vec{x})$, decision regions

What would logistic regression do?

- can only fit one hyperplane

NN: many hyperplanes (1 per hidden neuron)

XOR problem



$$y = x_1 \text{ XOR } x_2$$

$$= \begin{cases} 1 & \text{if } x_1 = 1 \text{ or } x_2 = 1 \text{ but not both} \\ -1 & \text{o.w.} \end{cases}$$

not linearly separable

k-NN, k-nearest neighbors

pro: one parameter k (int)
no optimization, fitting

cons: all data have to be stored and looked at to make a prediction

$$f(\vec{x}) = \sum_{i \in N_k(\vec{x})} y_i / k$$

$f(\vec{x})$: Algorithm

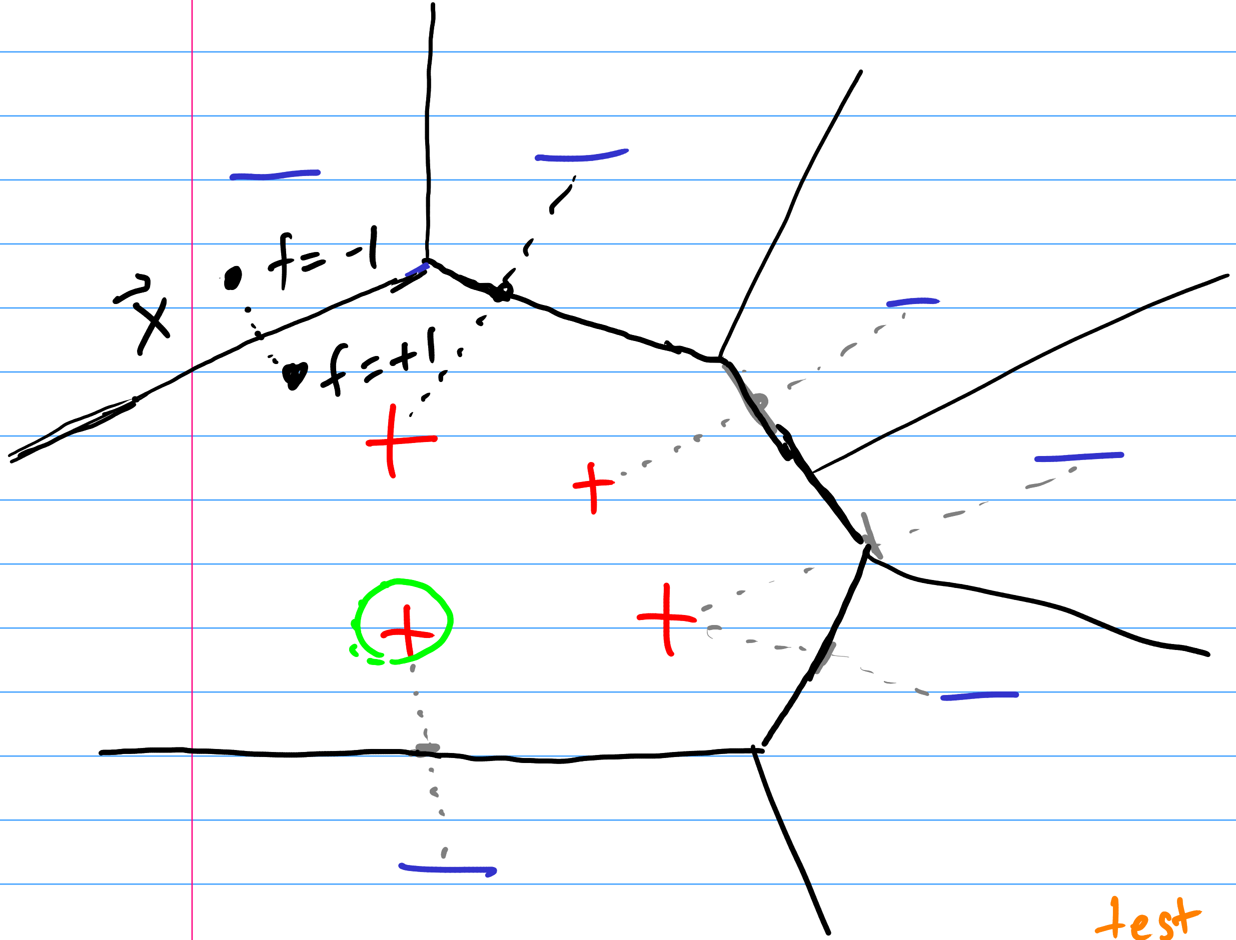
1) Find k-NNs of $\vec{x}$ in training set

$$N_k(\vec{x}) = \{ i_1, \ldots, i_k : \text{distance to } \vec{x}_{i_1}, \ldots, \vec{x}_{i_k} \text{ less than any other point} \}$$

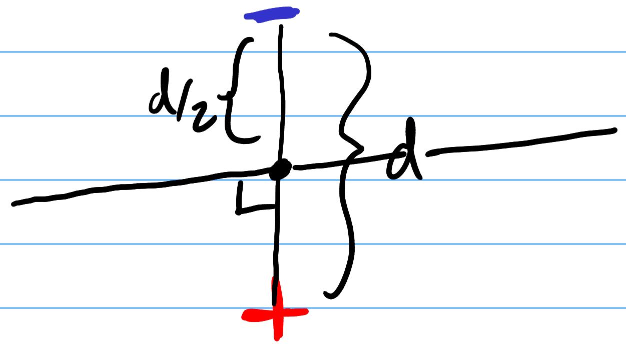
2) Output label of majority (classification) of neighbors
avg. label of neighbors (regression)

1-NN

$$d(\vec{x},\vec{x}') = \|\vec{x}-\vec{x}'\|_2$$



Picture w/ 2 pts

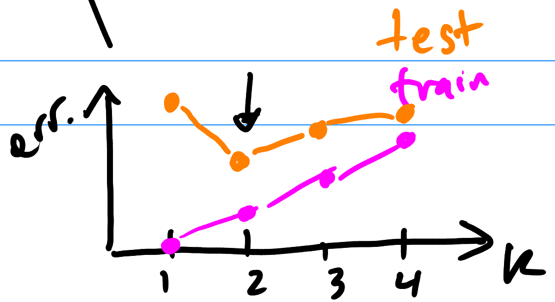


$k>1$: still polygons but don't perfectly fit data

$\rightarrow$ bias for $k>1$

$\downarrow$ variance

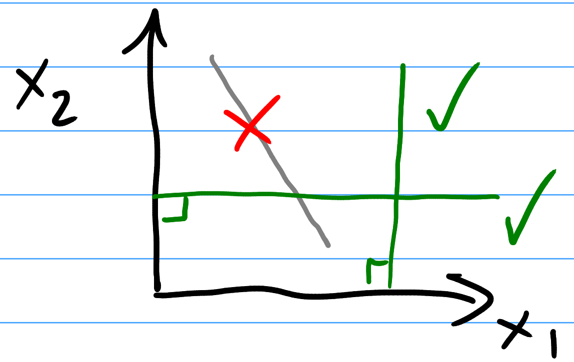
Pick k with cross-validation



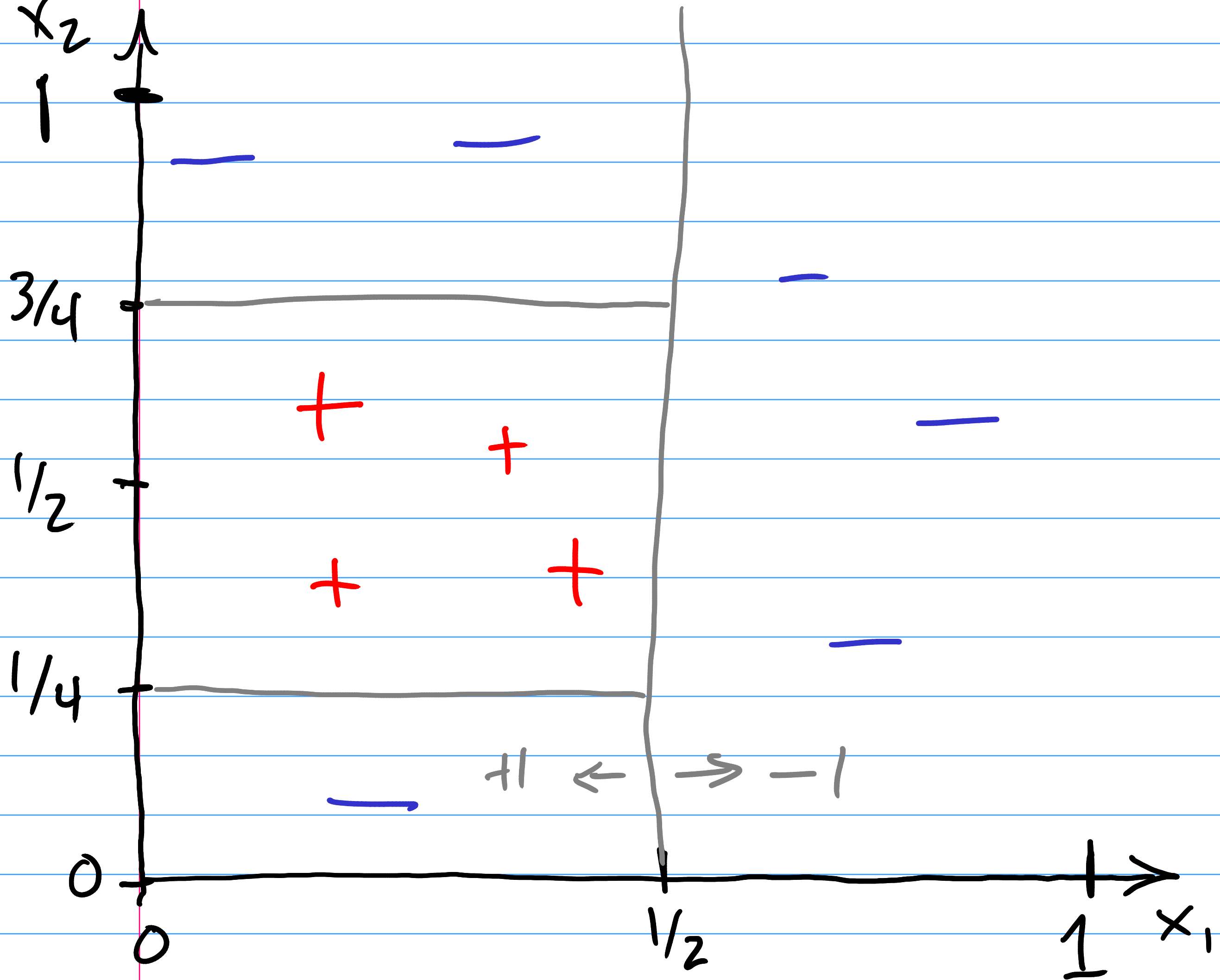
Trees

Idea: split your domain on a single dimension at a time

splits (diff. sides of hyperplane) need to be aligned w/ coordinate axes



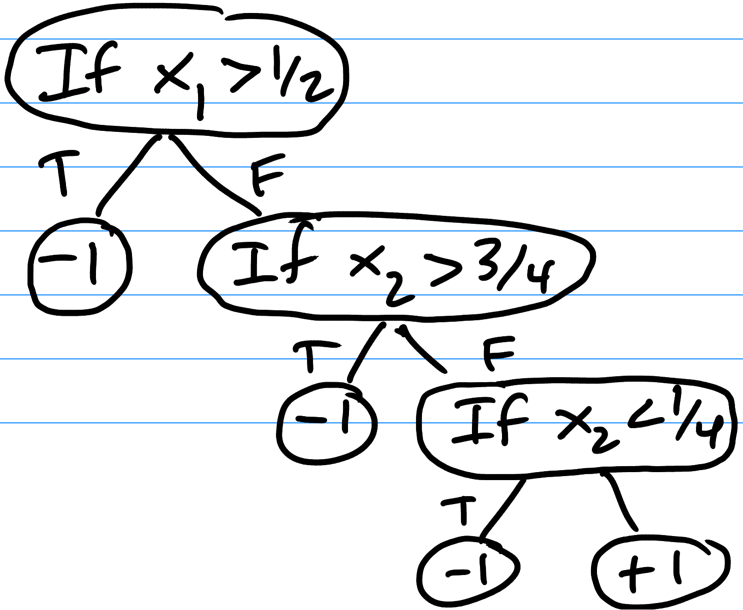
depth 3 fits perfectly



1) Find "best" split at current
- makes least mistakes

2) Repeat but subdivide regions instead

Practically:
- best to use random forests
- or boosted trees



Kernel methods / smoothers

$$f(\vec{x}) = \sum_{i=1}^{n} w_i \underbrace{k(\vec{x}, \vec{x}_i)}_{\text{kernel function}} y_i$$

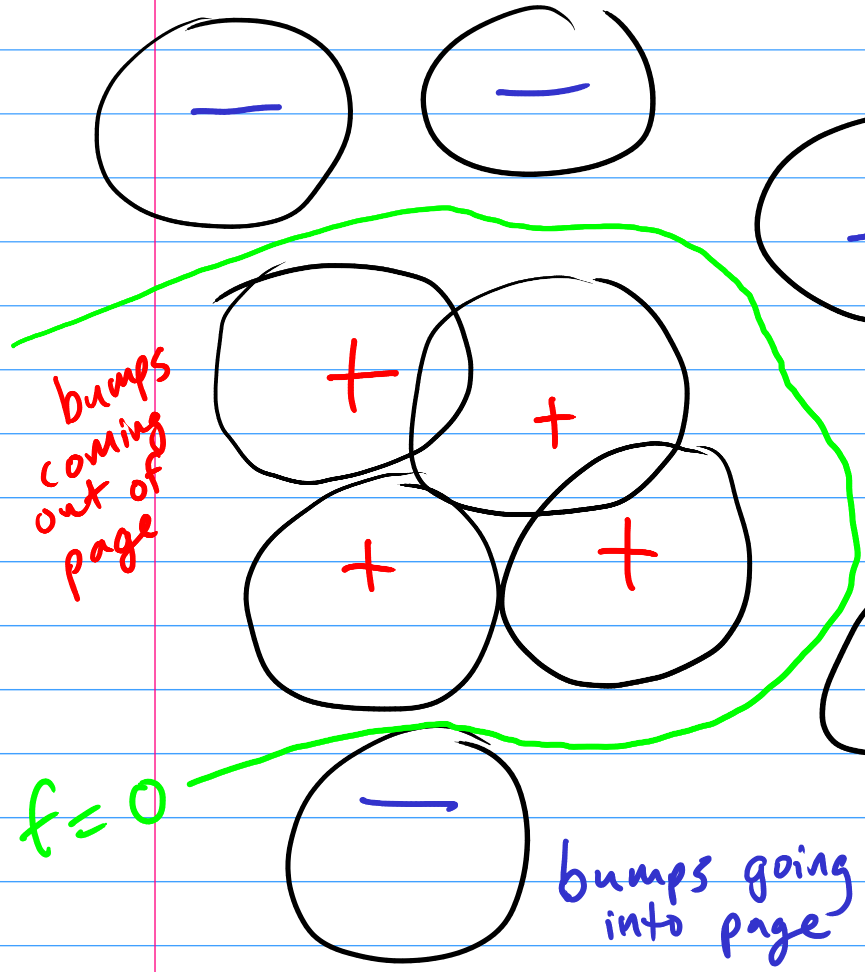
similarity of $\vec{x}, \vec{x}_i$

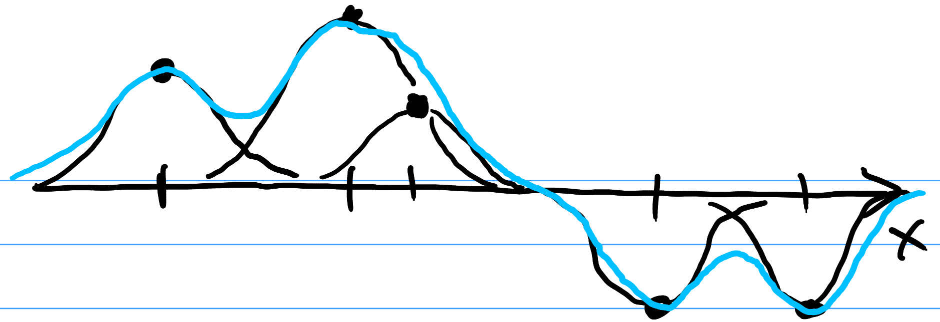
ex/ $k(\vec{x}, \vec{x}') = \exp\left(-\dfrac{\|\vec{x} - \vec{x}'\|^2}{2h^2}\right)$ Gaussian, radial basis function

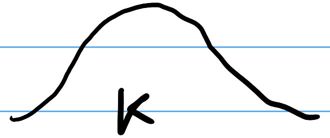
h ← bandwidth parameter

simplest: Nadaraya-Watson smoother

$$f(\vec{x}) = \frac{\sum_{i=1}^{n} k(\vec{x}, \vec{x}_i)\, y_i}{\sum_{i=1}^{n} k(\vec{x}, \vec{x}_i)}$$







smoother

biased by density of pts

- ways to debias w/ linear regression

Beware:

many things called kernels

- not convolutional kernel
- not nullspace of a matrix
- not popcorn

$$\frac{\partial C}{\partial w_i} = 0 \quad \leftarrow \quad i^{\text{th}} \text{ element of } \nabla_{\vec{w}} C$$