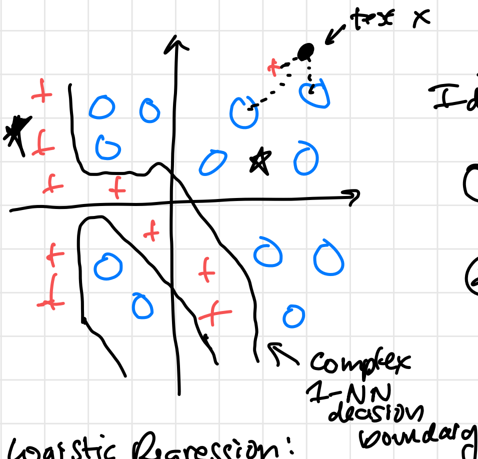
2/1/2024: Non-parametric Methods

	Discriminative	Generative
Parametric Methods - Fixed # of parameters to learn - After learning, training data no longer needed	Logistic Regression Softmax Regression Directly model $p(y\|x)$ • Log. Reg. learn $w \in \mathbb{R}^d$ • Sof. Reg. learn $w^{(1)}, \ldots, w^{(c)} \in \mathbb{R}^d$	Naive Bayes model $p(y)$ and $p(x\|y)$ $\downarrow$ π $\downarrow$ τ extracted from training data by counting
Non-parametric Methods - Size of model proportional to size of training dataset - Usually because we store training dataset & use it to make prediction	K-Nearest Neighbors Kernel Methods	



Logistic Regression:
can't fit data well
(without adding more features)

1-NN: can fit training data always

1-Nearest Neighbor (1-NN)

Idea: Similar points usually have the same label

① Training step: Store training data in memory

② Test time: Given x
find most similar training example:

$$i^* = \underset{i=1,\ldots,n}{\operatorname{argmin}} \ \text{distance}(x, x^{(i)})$$

return $y^{(i^*)}$ (label of the most similar point)

Common distance is Euclidean distance
i.e. $\|x - x^{(i)}\|$

K-Nearest Neighbors:

- Find K closest training examples to test input x
- Return most common label among those K

Why? Reduces effect of anomalous training examples

Pitfalls of K-NN

- Bias vs. Variance

Bias: Error b/c assumptions of model are wrong

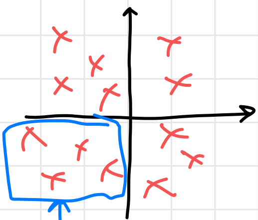
Very low b/c can represent any function

Variance: Error in estimating best possible model in model family caused by overfitting

Can be very large

- Curse of Dimensionality

In high dimensions, you rarely have close neighbors



In $\mathbb{R}^2$, ~1/4 of points are in same quadrant as you

If in $\mathbb{R}^{1000}$ then only $\frac{1}{2^{1000}}$ points are in same quadrant

Closest neighbor is still not that similar, so they might not have same label

---

K-NN

- Idea: Similar points have similar labels
- No good way to "regularize"
- No parameters we could tweak

Logistic Regression

- Only learns a linear decision boundary
- Learn parameters from data
- Regularization (L_2)

$\boxed{\text{Kernel Methods}}$ Combine ideas from K-NN and Logistic Regression

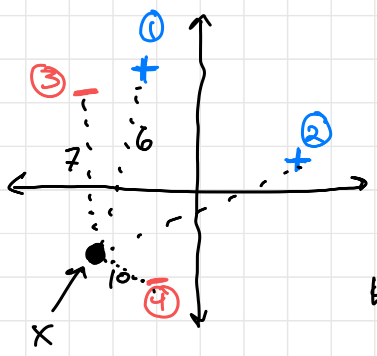
Make a prediction on test example x based on:

$$\sum_{i=1}^{n} \underbrace{\alpha_i}_{\in\mathbb{R},\ \text{parameter to learn}} \underbrace{K(x, x^{(i)})}_{\text{"Kernel function" measures similarity between 2 points}}$$

total of n α_i's to learn

For binary classification:

- Logistic Regression: If $w^T x > 0$, predict $y = +1$
  If $w^T x < 0$, $y = -1$
- Kernel-based classifier: If $\sum_{i=1}^{n} \alpha_i K(x, x^{(i)}) > 0$, predict $+1$
  < 0, predict -1



suppose: $K(x, x^{(1)}) = 6$
$x^{(2)}) = 1$
$x^{(3)}) = 7$
$x^{(4)}) = 10$

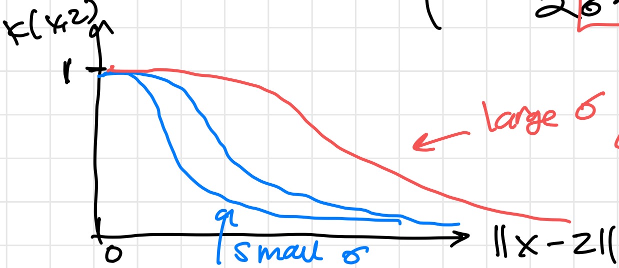
AND $\alpha_1 = 1$, $\alpha_2 = 1$, $\alpha_3 = -1$, $\alpha_4 = -1$

Here: Score $= 6 \cdot 1 + 1 \cdot 1 + \boxed{7 \cdot -1} + \boxed{10 \cdot -1}$
$= -11$ predict -1

One popular option for Kernel:
Radial Basis Function (RBF) Kernel

$$K(x, z) = \exp\left(\frac{-\|x - z\|^2}{2\boxed{\sigma}^2}\right)$$

hyperparameter called "bandwidth"



large σ = width of curve is larger
$\iff$ points further away still considered somewhat similar

How to learn α_i's?

Caveat: This is not recommended practice, but shows connection to logistic regression

Logistic Regression is a kernel method using the kernel $K(x,z) = x^T z$

Logistic Regression

To prediction input x:

Compute $\boxed{w^T x}$

$$= \left(\sum_{i=1}^n \alpha_i x^{(i)}\right)^T x = \sum_{i=1}^n \alpha_i x^{(i)T} x$$

$$= \boxed{\sum_{i=1}^n \alpha_i K(x^{(i)}, x)}$$

Log. Reg. is a kernel method where $K(x,z) = x^T z$

Training: G.D.

$\boxed{w^{(0)} \leftarrow \underline{0}}$

$$\boxed{w^{(t)} \leftarrow w^{(t-1)} + \eta \cdot \frac{1}{n} \cdot \sum_{i=1}^n \underbrace{\sigma(-y^{(i)} w^{(t-1)T} x^{(i)}) \cdot y^{(i)}}_{\text{scalar}} \cdot x^{(i)}}$$

Key observation: update to w is always $c_1 x^{(1)} + c_2 x^{(2)} + \cdots + c_n x^{(n)}$

So: Final w can be written as

$$\boxed{w = \sum_{i=1}^n \alpha_i x^{(i)}}$$

α_i's are the weights of linear combination of the $x^{(i)}$'s

Kernelized Log. Reg.

Goal: A new algorithm where if using $K(x,z) = x^T z$, we get same result as original Log. Reg, but can also use other kernels

Idea: Learn α_i's, not w

Testing: Directly use α_i's, Compute $\boxed{\sum_{i=1}^n \alpha_i K(x, x^{(i)})}$

Training: G.D.

$\boxed{\alpha_i^{(0)} \leftarrow 0 \text{ for each } i}$

$$\alpha_i^{(t)} \leftarrow \alpha_i^{(t-1)} + \eta \cdot \frac{1}{n} \cdot \boxed{\sigma(-y^{(i)} \underline{w}^{(t-1)T} x^{(i)})} \cdot y^{(i)}$$

$$= \sigma\left(-y^{(i)} \sum_{j=1}^n \alpha_j^{(t-1)} K(x^{(j)}, x^{(i)})\right)$$

for every i

Double loop over train data

- 2 different algorithms
- Get same final results when $K(x,z) = x^T z$
- Kernelized L.R.: Can replace $K(x,z)$ with any other kernel