Kernels: a way to modify existing algorithms

- Kernelized linear regression
- Kernelized logistic regression

...

Infinite-dimensional Features

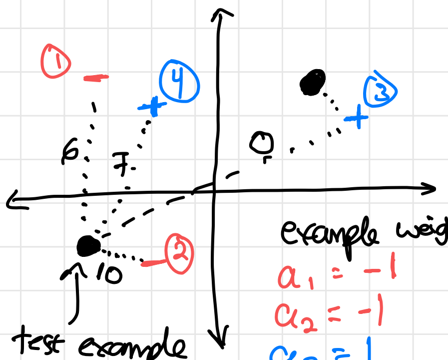
Recall: By preprocessing data, we can add new features

price	Area	#beds	Area^2	#beds=3	...

Kernels systematically add many features (possibly infinite) but also give a way to work with these big feature vectors efficiently

Make predictions by measuring similarity of test example to each training example

(similar to k-NN)



example weights:

$a_1 = -1$

$a_2 = -1$

$a_3 = 1$

$a_4 = 1$

In Kernel logistic regression:

Prediction $f(x) = \underbrace{\sum_{i=1}^{n}}_{\text{sum over training examples}} \underbrace{a_i}_{\text{weights that we learn}} \underbrace{k(x, x^{(i)})}_{\text{Kernel function measures how similar the 2 inputs are}}$

where

If

$f(x) > 0$ predict +1

$f(x) < 0$ predict -1

predict: $-1 \cdot 6 - 1 \cdot 10 + 1 \cdot 0 + 1 \cdot 7$

$= -9 \Rightarrow$ predict -1

Announcements

- HW1 due today
- Proposal due 2/16
- Section: Vote!

Let's consider $K(x,z) = x^T z$ — captures some notion of similarity. If $x=z$, then $K(x,z) \geq 0$. If point in opposite directions then $K(x,z) < 0$

Big claim: Logistic Regression Algorithm can be rewritten in terms of kernels only
↑ Any x's only show up inside of kernel function

Logistic Regression via Gradient Descent

$w^{(0)} \leftarrow \underline{0} \in \mathbb{R}^d$

for $t=1,\dots,T$:

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

$w \leftarrow w^{(T)}$ (final w)

Prediction time: Given x, compute $w^T x$ → If >0 predict $+1$; → If <0 predict -1

Observation: w is linear combination of $x^{(i)}$'s

Idea: "Reparameterize" algorithm to update coefficients of this linear combination

Kernel logistic Regression ← mathematically computes same thing as before

$a^{(0)} \leftarrow \underline{0} \in \mathbb{R}^n$ define $w^{(t)} = \sum_{i=1}^{n} a_i^{(t)} \cdot x^{(i)}$

for $t=1,\dots,T$:

for $i=1,\dots,n$

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

$$= \left(\sum_{j=1}^{n} a_j^{(t-1)} x^{(j)}\right)^T x^{(i)}$$

$$= \sum_{j=1}^{n} a_j^{(t-1)} x^{(j)T} x^{(i)}$$

$$= \sum_{j=1}^{n} a_j^{(t-1)} K(x^{(j)}, x^{(i)})$$

return $a = a^{(T)}$

Prediction: $w^T x$ for test input x

$$= \sum_{j=1}^{n} a_j K(x^{(j)}, x)$$

If we can compute kernel between any 2 x's
we dont have to store x's themselves