1^{st} Half Review & Mid-term Guideline

Topics

- Introduction to machine learning
- Linear algebra, calculus and probability theorem
- K nearest neighbors
- Linear regression
- Naïve Bayes
- Logistic regression

Introduction to machine learning

Types of Learning Task

- **Supervised learning**
  - Training data includes desired outputs
- **Unsupervised learning**
  - Training data does not include desired outputs
- **Semi-supervised learning**
  - Training data includes a few desired outputs
- **Self-supervised learning**
  - Training data contains supervised signals extracted from the data itself
- **Reinforcement learning**
  - Rewards from sequence of actions

Supervised Learning

- **Given** examples of a function *(X, Y=F(X))*
- **Predict** function *F(X)* for new examples *X*
  - Discrete *Y*: Classification
  - Continuous *Y*: Regression

Preliminaries

Basic Vector Operations

$$\boldsymbol{x} + \boldsymbol{y} = (x_1, x_2) + (y_1, y_2) = (x_1 + y_1, x_2 + y_2)$$

$$a\mathbf{v} = a(x_1, x_2) = (ax_1, ax_2)$$

$$\boldsymbol{x} \cdot \boldsymbol{y} = x_1 y_1 + x_2 y_2$$

Matrix Times Matrix

$$\mathbf{L} = \mathbf{M} \cdot \mathbf{N}$$

$$\begin{bmatrix} l_{11} & l_{12} & l_{13} \\ l_{21} & l_{22} & l_{23} \\ l_{31} & l_{32} & l_{33} \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \cdot \begin{bmatrix} n_{11} & n_{12} & n_{13} \\ n_{21} & n_{22} & n_{23} \\ n_{31} & n_{32} & n_{33} \end{bmatrix}$$

$$l_{12} = m_{11}n_{12} + m_{12}n_{22} + m_{13}n_{32}$$

Multiplication

- Is AB = BA? Maybe, but maybe not!

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & \ldots \\ \ldots & \ldots \end{bmatrix} \quad \begin{bmatrix} e & f \\ g & h \end{bmatrix}\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} ea+fc & \ldots \\ \ldots & \ldots \end{bmatrix}$$

- Heads up: multiplication is NOT commutative!
- Identity matrix:
  AI = IA = A

- Multiplication is associative:
- (A*B)*C=A*(B*C)

Chain Rule

- Help us to compute derivatives for composite functions.
- Three variables: z, y, x.
  - $z = f(y), y = g(x)$

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} = f'(g(x))g'(x)$$

Bayes' Theorem

- Joint probability $P(A = a, B = b)$: The probability that $A = a$ and $B = b$ happen simultaneously.
- Conditional probability $P(B = b|A = a) = \frac{P(A=a,B=b)}{P(A=a)}$: The probability of $B = b$, provided that $A = a$ has occurred.
- Marginalization: $P(B) = \sum_A P(A, B)$
- Bayes' theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

K nearest neighbors

k-nearest neighbors

- Find the k-nearest neighbors to $x^{(new)}$ in the data
  - i.e., rank the feature vectors according to Euclidean distance
  - select the k vectors which are have smallest distance to $x^{(new)}$
- Regression
  - Usually just average the y-values of the k closest training examples
- Classification
  - ranking yields k feature vectors and a set of k class labels
  - pick the class label which is most common in this set (“vote”)
  - Note: for two-class problems, if k is odd ($k = 1, 3, 5, \ldots$) there will never be any “ties”

Pitfalls: computational cost

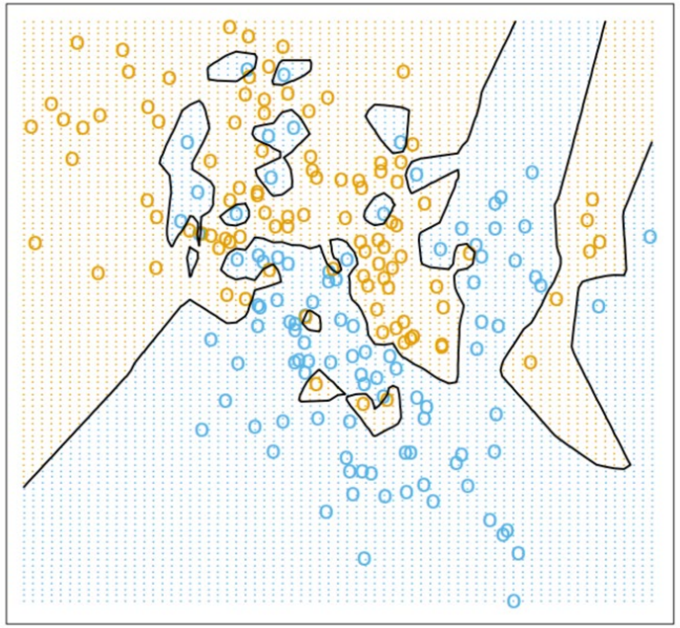
- Number of computations at training time: 0
- Number of computations at test time, per query (naïve algorithm)
  - Calculate m-dimensional Euclidean distances with n data points: $O(mn)$
  - Sort the distances: $O(n \log n)$
- This must be done for each query, which is very expensive by the standards of a learning algorithm!
- Need to store the entire dataset in memory!
- Tons of work has gone into algorithms and data structures for efficient nearest neighbors with high dimensions and/or large datasets.

k-nearest neighbors

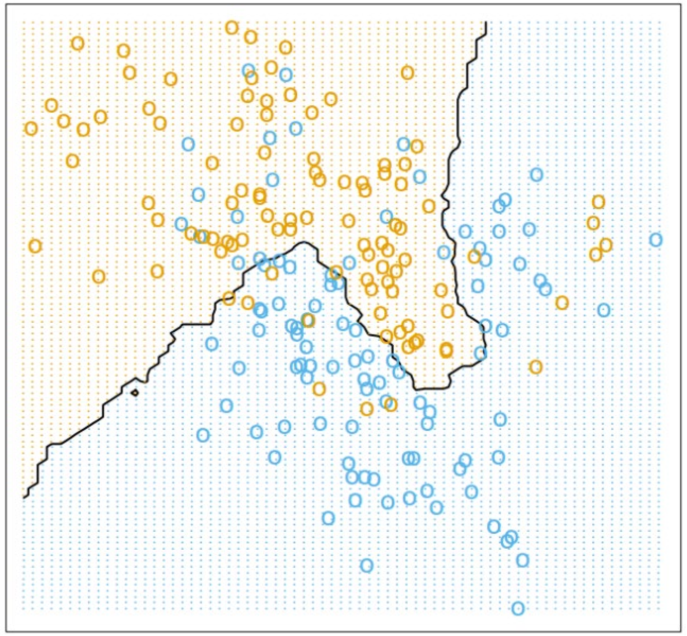
- Tradeoffs in choosing k
  - Small k
    - Good at capturing fine-grained patterns
    - May overfit, i.e. be sensitive to random idiosyncrasies in the training data
  - Large k
    - Makes stable predictions by averaging over lots of examples
    - May underfit, i.e. fail to capture important regularities
  - Rule of thumb: $k < \sqrt{n}$, where n is the number of training examples

k-nearest neighbors

$k = 1$



$k = 15$



Linear Regression

Linear regression

n features

- $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

- Define "feature" $x_0 = 1$, then
- $h_\theta(x) = \theta^T x$

$$\theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \cdots \\ \theta_n \end{pmatrix} \qquad x = \begin{pmatrix} 1 \\ x_1 \\ \cdots \\ x_n \end{pmatrix}$$

Linear regression

- Normal equation

$$\theta = (X^T X)^{-1} X^T y$$

$$y = \begin{pmatrix} y^{(1)} \\ \cdots \\ y^{(m)} \end{pmatrix} \qquad \theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \cdots \\ \theta_n \end{pmatrix} \qquad X = \begin{pmatrix} x_0^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_0^{(m)} & \cdots & x_n^{(m)} \end{pmatrix}$$

- Example

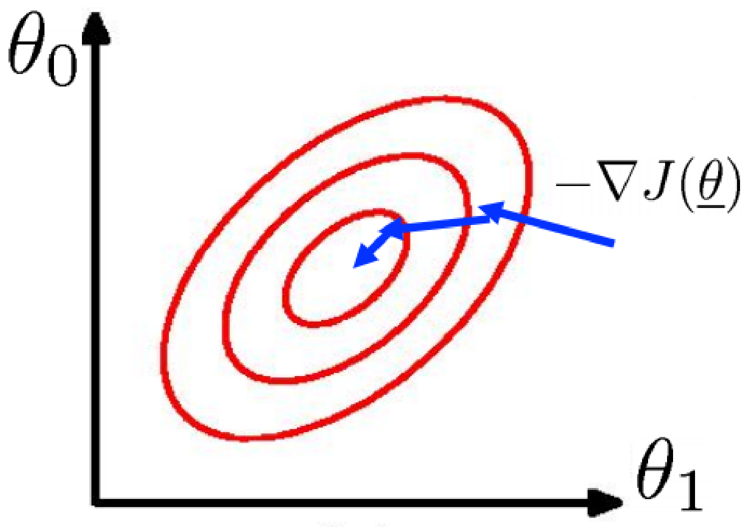
Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178

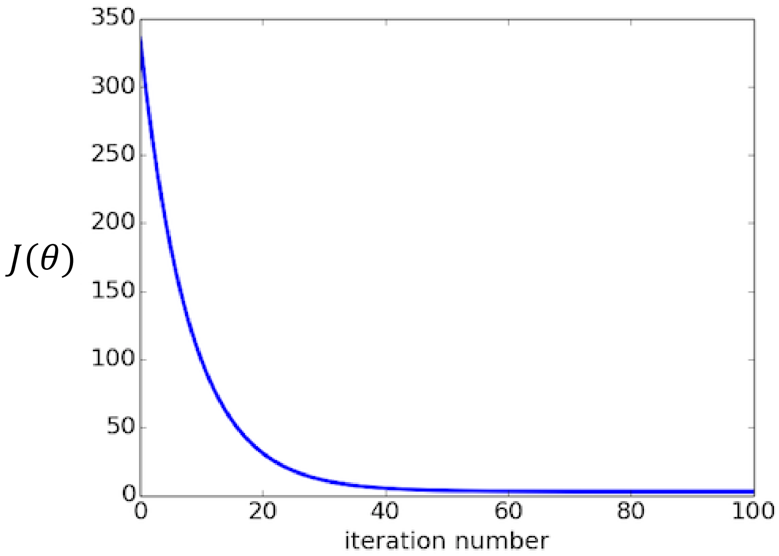
$$X = \begin{pmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{pmatrix} \qquad y = \begin{pmatrix} 460 \\ 232 \\ 315 \\ 178 \end{pmatrix}$$

Linear regression

- Gradient descent
- $\nabla J(\theta) = \left(\frac{\partial J(\theta)}{\partial \theta_0}, \frac{\partial J(\theta)}{\partial \theta_1}, \cdots\right)$

Initialize θ
Do {
  $\theta \leftarrow \theta - \alpha \nabla J(\theta)$
} while (stop condition)



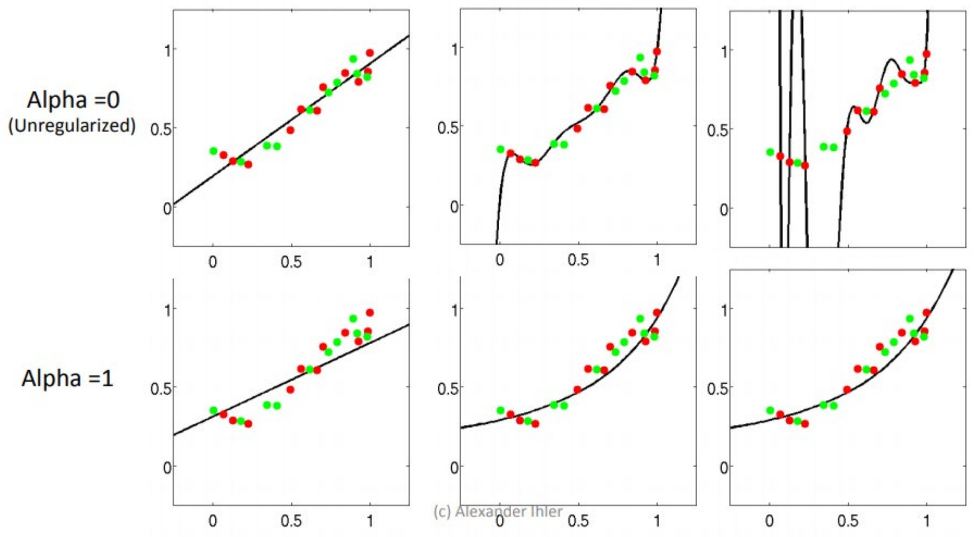


Regularization for preventing overfitting

- Modify loss function to add "preference" for small parameter values

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{m} \left(y^{(i)} - \boldsymbol{\theta}^T \boldsymbol{x}^{(i)} \right)^2 + \frac{\alpha}{2} \|\boldsymbol{\theta}\|$$

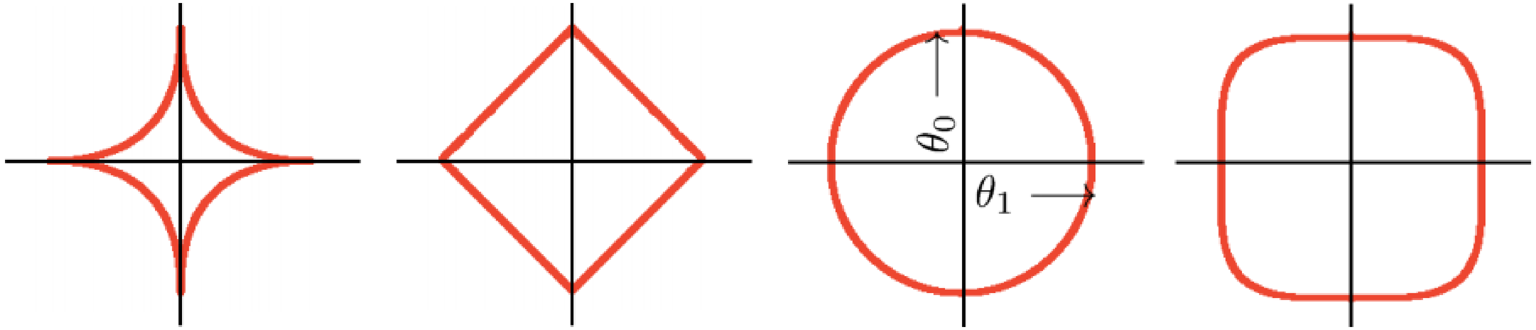
- Compare between unreg. & reg. results



Different regularization functions

- More general, L_p regularizer: $\left(\sum_j |\theta_j|^p\right)^{\frac{1}{p}}$

Isosurfaces: $||\theta||_p$ = constant



p=0.5

p=1
Lasso
L_1

p=2
Quadratic
L_2
ridge

p=4

Naïve Bayes Classifier

Bayes classifier: Naïve Bayes

- Estimate p(y) = [p(y=0) , p(y=1) …]
- Estimate p(x | y=c) for each class c
- Calculate p(y=c | x) using Bayes rule

$$\Rightarrow \quad p(y|x) = p(x|y)p(y)/p(x)$$

- Choose the most likely class c

$$= \frac{p(x|y)p(y)}{\sum_c p(x|y=c)p(y=c)}$$

- Naïve Bayes: p(x | y=c) = $\prod_\iota$ p(x_i | y=c)

Example: Naïve Bayes

Observed Data:

x_1	x_2	y
1	1	0
1	0	0
1	0	1
0	0	0
0	1	1
1	1	0
0	0	1
1	0	1

$$\hat{p}(y=1) = \frac{4}{8} \qquad = (1 - \hat{p}(y=0))$$

$$\hat{p}(x_1, x_2 | y=0) = \hat{p}(x_1|y=0)\,\hat{p}(x_2|y=0)$$

$$\hat{p}(x_1=1|y=0) = \frac{3}{4} \qquad \hat{p}(x_1=1|y=1) = \frac{2}{4}$$

$$\hat{p}(x_2=1|y=0) = \frac{2}{4} \qquad \hat{p}(x_2=1|y=1) = \frac{1}{4}$$

$$\hat{p}(y=1|x_1=1, x_2=1) = \frac{\frac{4}{8} \times \frac{2}{4} \times \frac{1}{4}}{\frac{3}{4} \times \frac{2}{4} \times \frac{4}{8} + \frac{2}{4} \times \frac{1}{4} \times \frac{4}{8}}$$

$$= \frac{1}{4}$$

Logistic Regression

Linear classifier

- Representation

$$\hat{y} = T(\theta \cdot x)$$

- Decision boundary

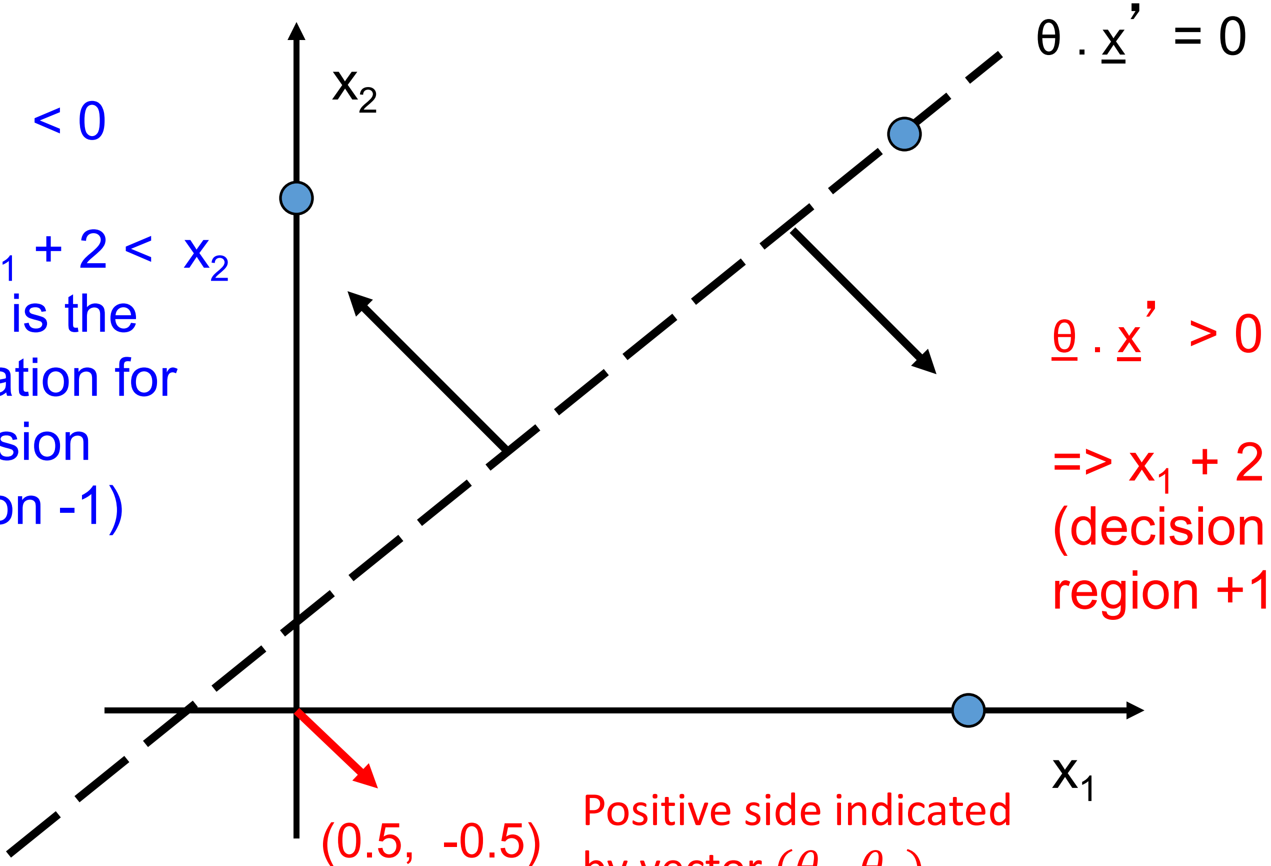
$$\theta \cdot x = 0$$

Linear Decision Boundary

$$\theta = (\theta_0, \theta_1, \theta_2) = (1, .5, -.5)$$

$\underline{\theta} \cdot \underline{x}' < 0$

$\Rightarrow x_1 + 2 < x_2$
(this is the equation for decision region -1)



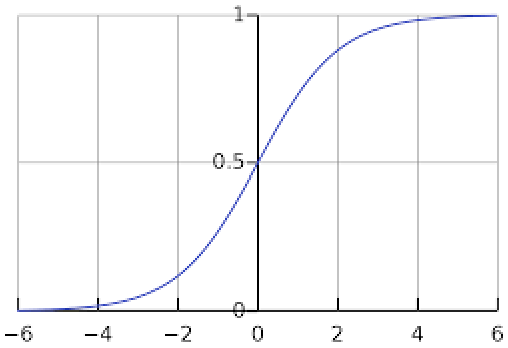
$\underline{\theta} \cdot \underline{x}' > 0$

$\Rightarrow x_1 + 2 > x_2$
(decision region +1)

Positive side indicated by vector (θ_1, θ_2)

From P. Smyth

Logistic regression classifier



- Use logistic/sigmoid function

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

- Interpret $\sigma(\theta x)$ as a probability that $y = 1$
- Use a negative log-likelihood loss function
  - If y = 1, cost is - log Pr[y=1] = - log $\sigma(\theta x)$
  - If y = 0, cost is - log Pr[y=0] = - log (1 - $\sigma(\theta x)$)
- Can write this succinctly:

$$J(\underline{\theta}) = -\frac{1}{m}\Big(\sum_i \underbrace{y^{(i)} \log \sigma(\theta\cdot x^{(i)})}_{\text{Nonzero only if y=1}} + \underbrace{(1-y^{(i)}) \log(1-\sigma(\theta\cdot x^{(i)}))}_{\text{Nonzero only if y=0}}\Big)$$