MACHINE LEARNING

AN INTRODUCTION

Christian Treffs

WHAT IS MACHINE LEARNING?

DEFINITION

Machine Learning is ...

...field of study that gives computers the ability to learn without being explicitly programmed.

~ Arthur Samuel (1959)

... a well-posed learning problem: A computer program is said to learn from experience E *with respect to some task* T *and some performance measure* P*, if its performance on* T*, as measured by* P*, improves with experience* E*.*

~ Tom Mitchell (1998)

... is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence.

~ Wikipedia

MACHINE LEARNING FIELDS

SUPERVISED LEARNING (CLASSIFICATION, REGRESSION)

- Linear/Polinomial & Logistic regression
- Neural networks (Forward-/Backpropagation)
- Support-Vector Machines (SVMs)

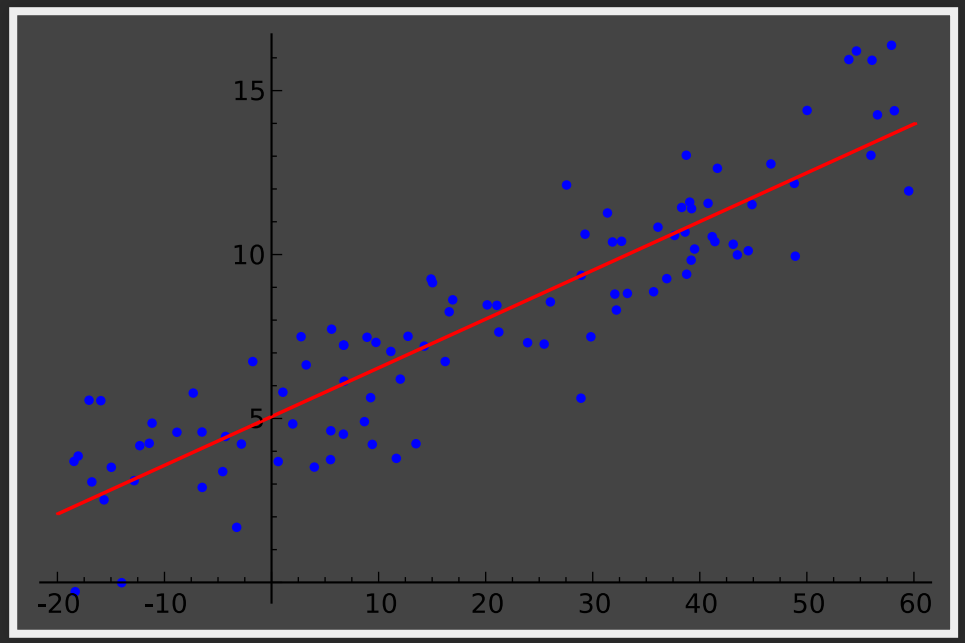
UNSUPERVISED LEARNING

- Clustering (k-Means)
- Dimensionality reduction (Principal Component Analysis (PCA))
- Anomality detection
- Recommender Systems

SUPERVISED LEARNING

... is the ML task of learning a function that maps an input to an output based on example input-output pairs.
It infers a function from labeled training data consisting of a set of training examples.

LINEAR/POLINOMIAL REGRESSION



- Housing price prediction
  - linear: size feet => price $
  - polynomial: (size feet, num rooms, num floors, age of home, ...) => price $

LINEAR REGRESSION

Hypothesis:	$h_\theta(x) = \theta_0 + \theta_1 x$
Parameters:	θ_0, θ_1
Cost Function:	$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
Goal:	$\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$(\text{for } j = 1 \text{ and } j = 0)$$

}

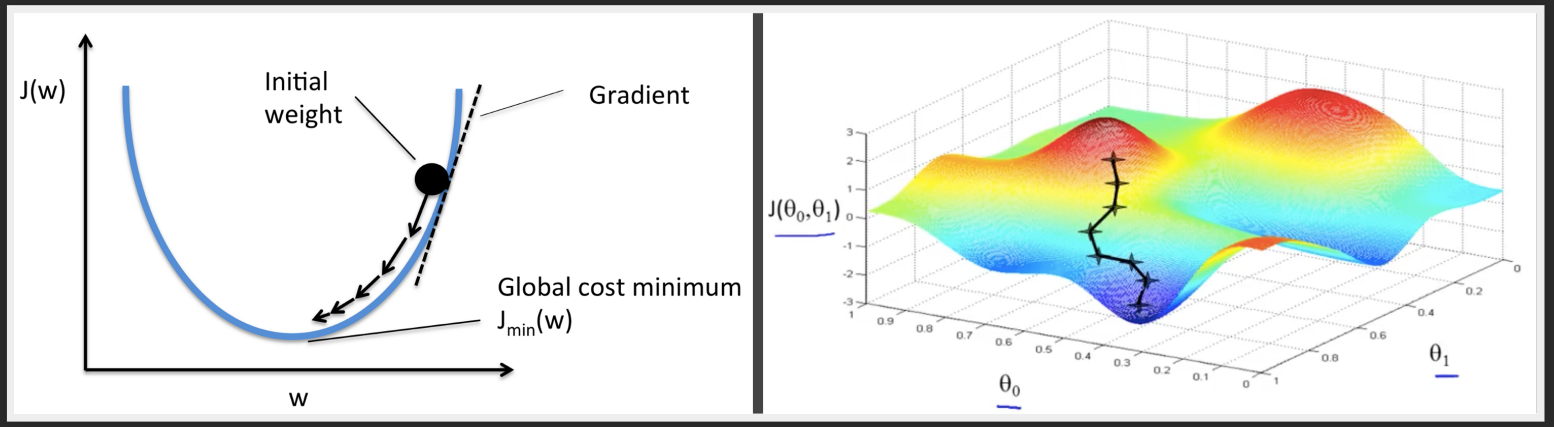
Linear Regression Model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

- training set -> learning algorithm -> cost-function w/ hypothesis -> estimation/prediction
- Idea: find θs to minimize cost-function
- Learning rate α -> adjust to tune gradient descent

GRADIENT DESCENT & LEARNING RATE $\ALPHA$

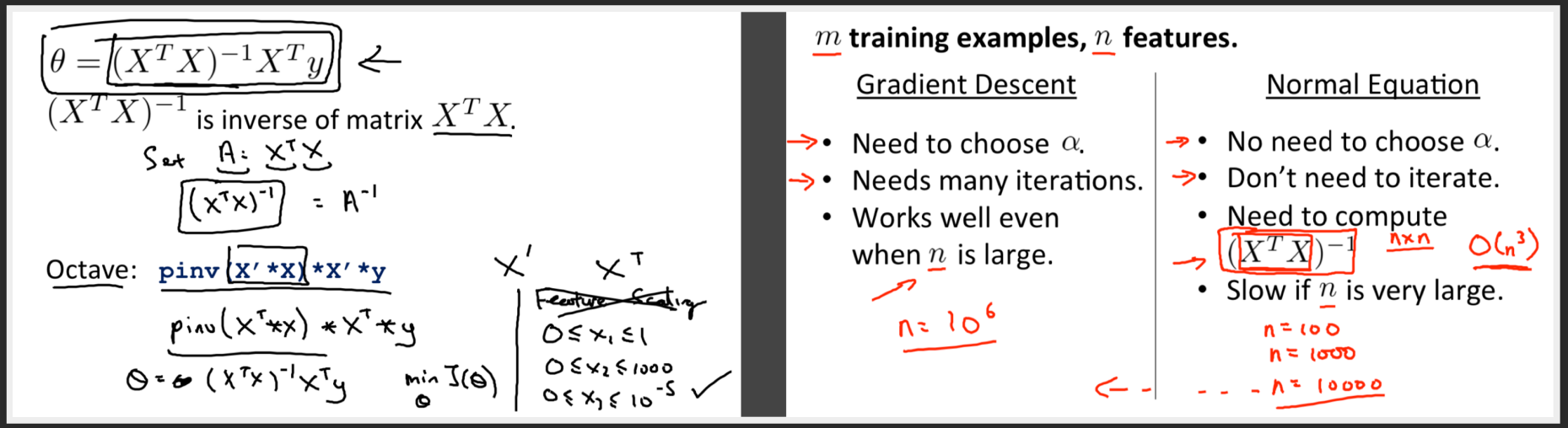


- Learning rate $alpha$
  - too small: slow convergence
  - too large: $J(\theta)$ may not descrease every iteration; may not converge
- Converged: if $J(\theta)$ decreases by less than 10^{-3} in 1 iteration

TECHNIQUE: FEATURE SCALING

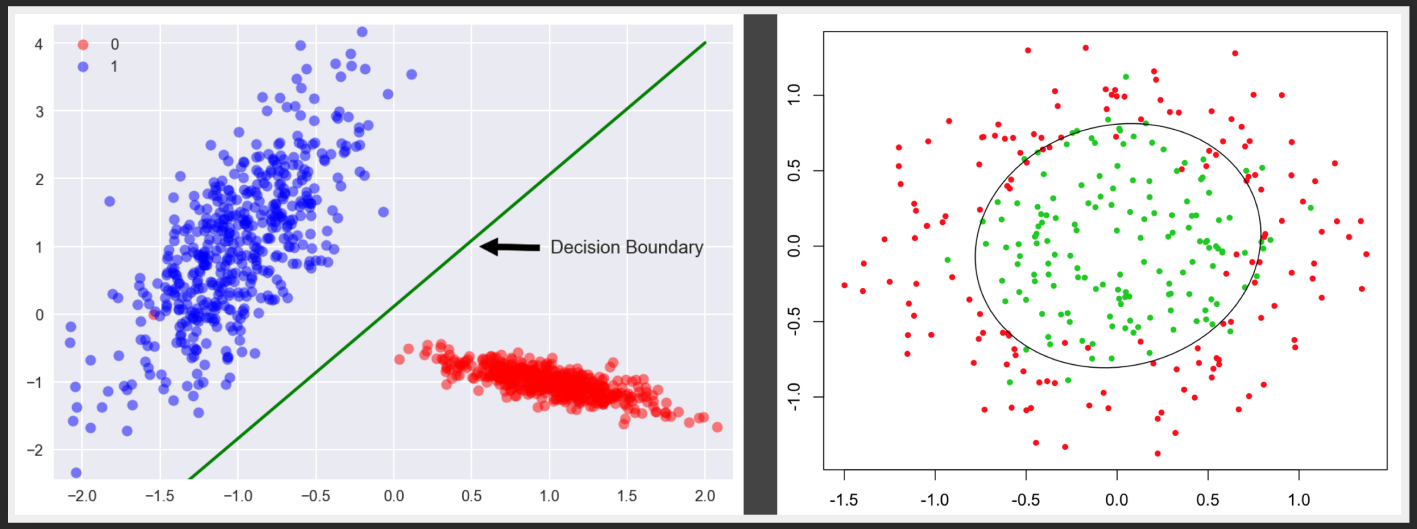
- Make sure features are on a similar scale
- Get every feature into approximately $-1 ≤ x_i ≤ 1$ range.
- Mean normalization:
  - replace x_i with $x_i-μ_i$ to make features have approx. 0 mean

GRADIENT DESCENT ALTERNATIVE: NORMAL EQUATION



- = Method to solve for α analytically
- + more efficient
- – consumes more memory

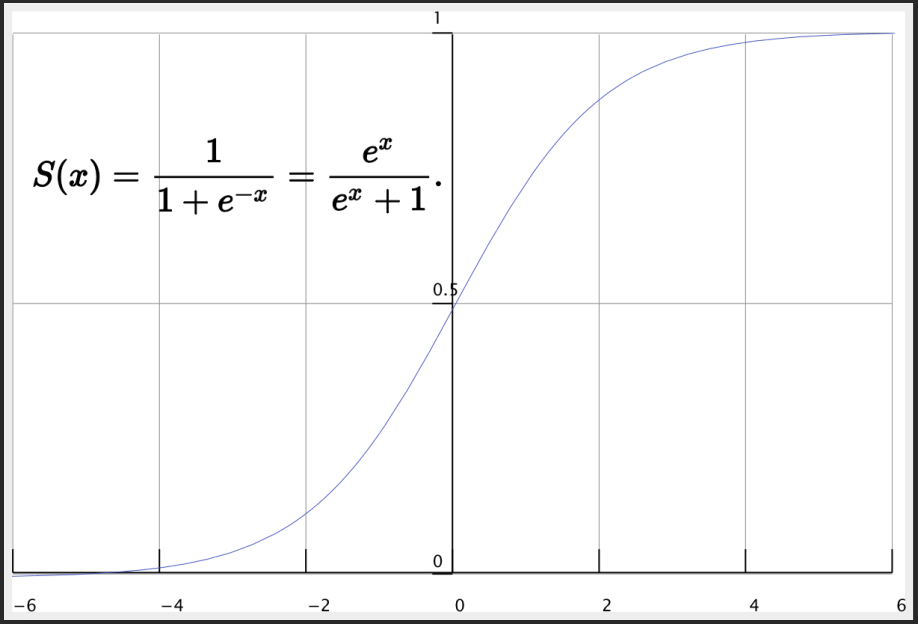
LOGISTIC REGRESSION



- Classification
  - Email: Spam / Not Spam?
  - Online Transactions: Fraudulent Yes/No?
  - Tumor: Malignant/Benign?

SIGMOID FUNCTION

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}.$$



COST FUNCTION & GRADIENT DESCENT

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

(simultaneously update all θ_j)

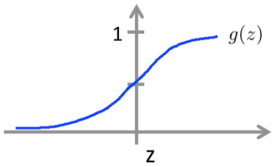
$\}$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Logistic regression

$$h_\theta(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1+e^{-z}}$$



Suppose predict "$y = 1$" if $h_\theta(x) \geq 0.5$

$\theta^T x \geq 0$

$g(z) \geq 0.5$ when $z \geq 0$

$h_\theta(x) = g(\theta^T x)$

predict "$y = 0$" if $h_\theta(x) < 0.5$

$\theta^T x < 0$

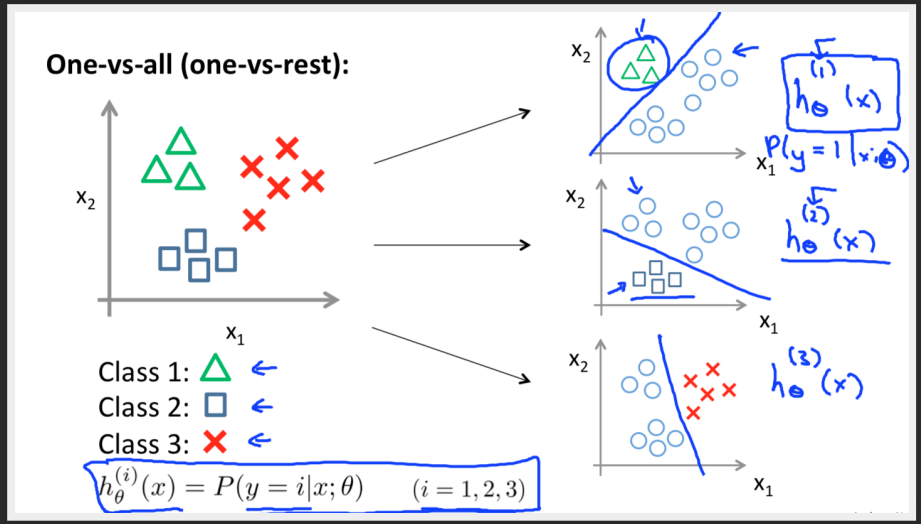
$g(z) < 0.5$ when $z < 0$

$y \in \{0, 1\}$

0: "Negative Class" (e.g., benign tumor)

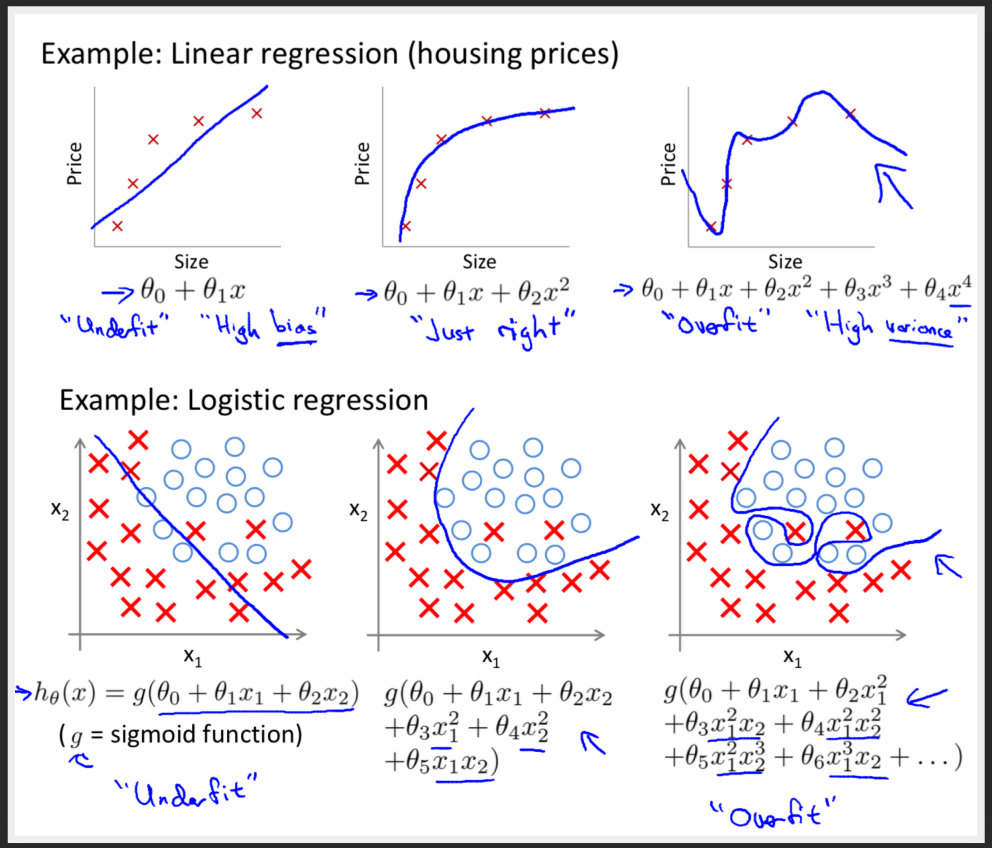
1: "Positive Class" (e.g., malignant tumor)

MULTI-CLASS CLASSIFICATION: ONE-VS-ALL



- Classification with multiple segments
  - Email foldering/tagging: Work, Friends, Family, Hobby
  - Medical diagrams: Not ill, Cold, Flu
  - Weather: Sunny, Cloudy, Rain, Snow

REGULARIZATION



- bias vs. variance
  - high bias - underfit
  - high variance - overfit
- to address overfitting:
  - reduce features
  - regularization

REGULARIZED LINEAR REGRESSION

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n}\theta_j^2\right]$$

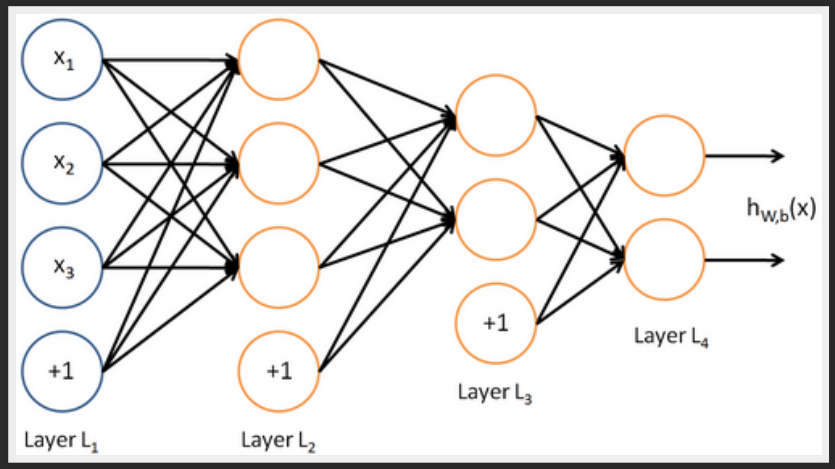
$$\min_\theta J(\theta)$$

REGULARIZED LOGISTIC REGRESSION

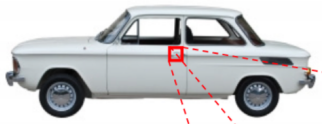
$$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

$\theta_1, \theta_2, \ldots, \theta_n$

NEURAL NETWORKS

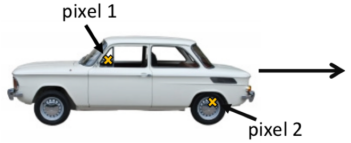


You see this:

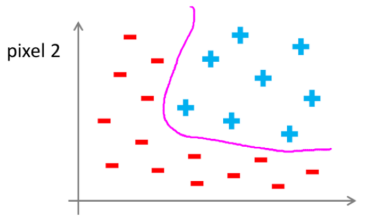


But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50



Learning Algorithm



50 x 50 pixel images→ 2500 pixels

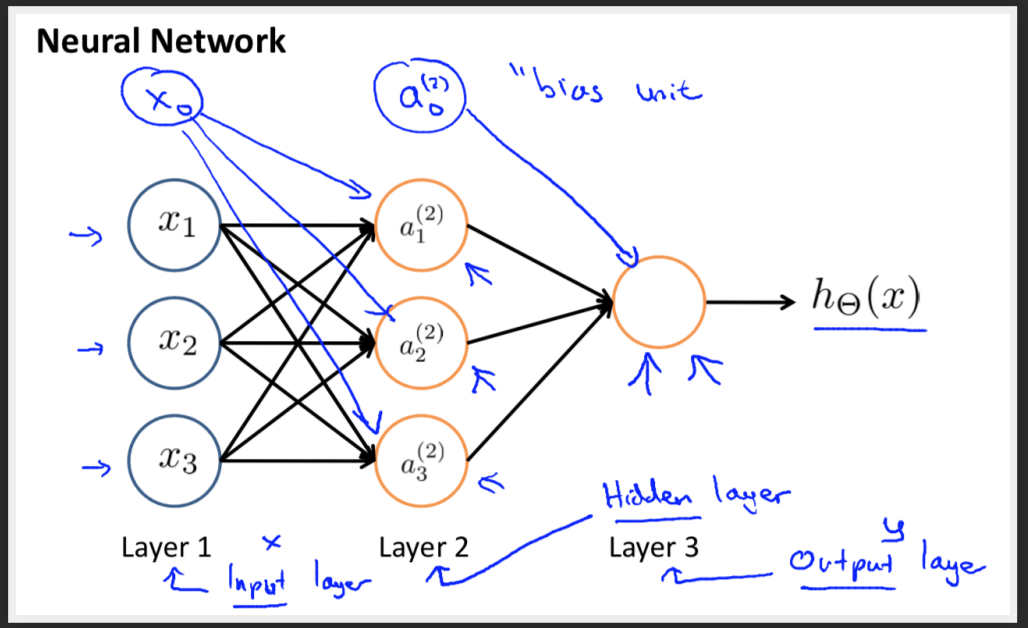
$n = 2500$ (7500 if RGB)

$$x = \begin{bmatrix} \text{pixel 1 intensity} \\ \text{pixel 2 intensity} \\ \vdots \\ \text{pixel 2500 intensity} \end{bmatrix}$$

0-255

Quadratic features ($x_i \times x_j$): ≈3 million features

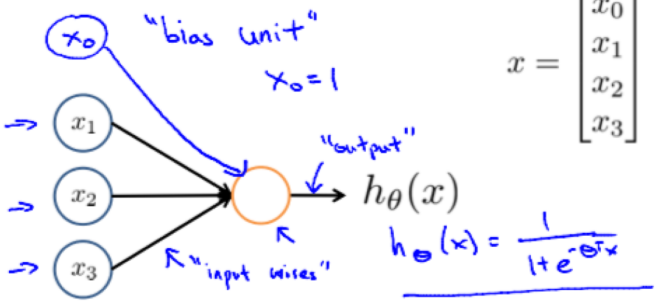
NEURAL NETWORK STRUCTURE



- $\alpha_i^{(j)}$ “activation” of unit i in layer j
- non-liner hypotheses
- computer vision: car detection (car / not a car)
- Autonomous driving

NEURAL NETWORK ACTIVATION

Neuron model: Logistic unit



$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

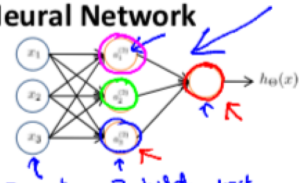
"weights" (parameters)

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

Sigmoid (logistic) activation function.

$$g(z) = \frac{1}{1 + e^{-z}}$$

Neural Network



$a_i^{(j)}$ = "activation" of unit i in layer j

$\Theta^{(j)}$ = matrix of weights controlling function mapping from layer j to layer $j+1$

$\Theta^{(1)} \in \mathbb{R}^{3 \times 4}$

$h_\Theta(x)$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

$\Theta^{(2)}$

If network has s_j units in layer j, s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$.

$s_{j+1} \times (s_j + 1)$

DEMO

- AND / OR

MULTIPLE OUTPUT NEURAL NETWORK

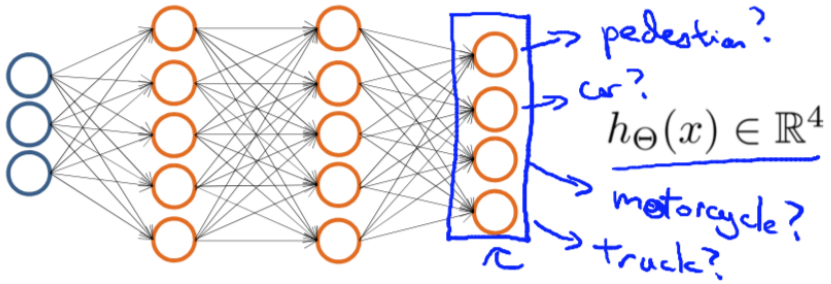
Multiple output units: One-vs-all.


Pedestrian


Car


Motorcycle


Truck



Want $h_\Theta(x) \approx \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, etc.

when pedestrian when car when motorcycle

COST FUNCTION NEURAL NETWORK

Neural network:

$$h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$$

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)\right]$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

TRAINING NEURAL NETWORK: FORWARD & BACK-PROPAGATION

Training a neural network

1. Randomly initialize weights
2. Implement forward propagation to get $h_\Theta(x^{(i)})$ for any $x^{(i)}$
3. Implement code to compute cost function $J(\Theta)$
4. Implement backprop to compute partial derivatives $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$

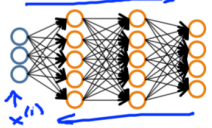
for i = 1:m {

- Perform forward propagation and backpropagation using example $(x^{(i)}, y^{(i)})$
  (Get activations $a^{(l)}$ and delta terms $\delta^{(l)}$ for $l = 2, \ldots, L$).

$\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)} (a^{(l)})^T$

}

Compute $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$.



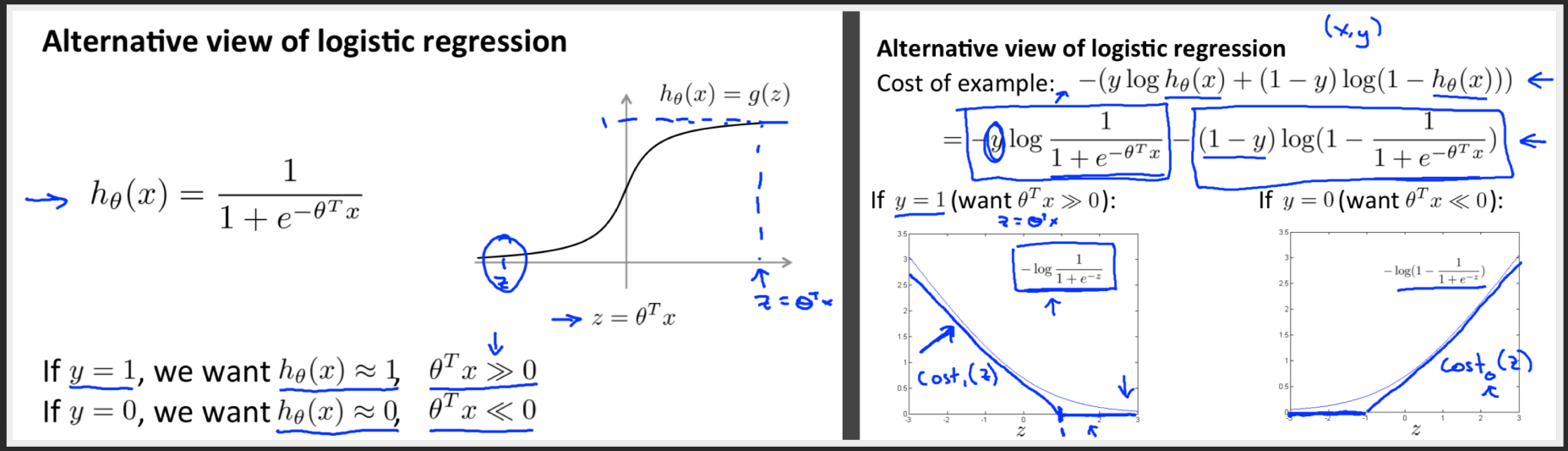
Training a neural network

5. Use gradient checking to compare $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$ computed using backpropagation vs. using numerical estimate of gradient of $J(\Theta)$.
   - Then disable gradient checking code.
6. Use gradient descent or advanced optimization method with backpropagation to try to minimize $J(\Theta)$ as a function of parameters Θ

$\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$

$J(\Theta)$ — non-convex.

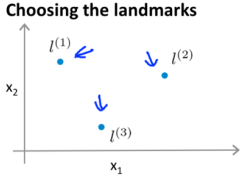
SUPPORT VECTOR MACHINES (SVM)



- Alternative view of logisc regression
- large margin classificaion

KERNELS

Choosing the landmarks



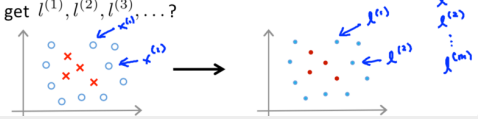
Given x:

$f_i = \text{similarity}(x, l^{(i)})$

$= \exp\left(-\frac{||x - l^{(i)}||^2}{2\sigma^2}\right)$

Predict $y = 1$ if $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$

Where to get $l^{(1)}, l^{(2)}, l^{(3)}, \ldots$?



SVM with Kernels

Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$.

choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \ldots, l^{(m)} = x^{(m)}$.

Given example x:

$f_1 = \text{similarity}(x, l^{(1)})$

$f_2 = \text{similarity}(x, l^{(2)})$

$f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix}$ $f_0 = 1$

For training example $(x^{(i)}, y^{(i)})$:

$f_1^{(i)} = \text{sim}(x^{(i)}, l^{(1)})$

$f_2^{(i)} = \text{sim}(x^{(i)}, l^{(2)})$

$f_i^{(i)} = \text{sim}(x^{(i)}, l^{(i)}) = \exp(-\frac{0}{\sigma^2}) = 1$

$f_m^{(i)} = \text{sim}(x^{(i)}, l^{(m)})$

$x^{(i)} \in \mathbb{R}^{n+1}$ $(\text{or } \mathbb{R}^n)$

$f^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ f_2^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix}$ $f_0^{(i)} = 1$

SVM with Kernels

Hypothesis: Given x, compute features $f \in \mathbb{R}^{m+1}$ $\theta \in \mathbb{R}^{m+1}$

Predict "y=1" if $\theta^T f \geq 0$ $\theta_0 f_0 + \theta_1 f_1 + \cdots + \theta_m f_m$

Training:

$$\min_\theta C \sum_{i=1}^{m} y^{(i)} cost_1(\theta^T f^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^{n=m} \theta_j^2$$

θ_0

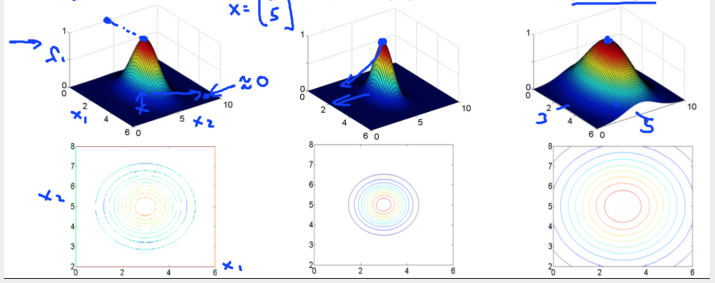
$\sum_j \theta_j^2 = \theta^T \theta$ $\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix}$ (ignore θ_0)

$\theta^T M \theta$ $\|\theta\|^2$ $m = 10{,}000$

Example:

$l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$ $f_1 = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$

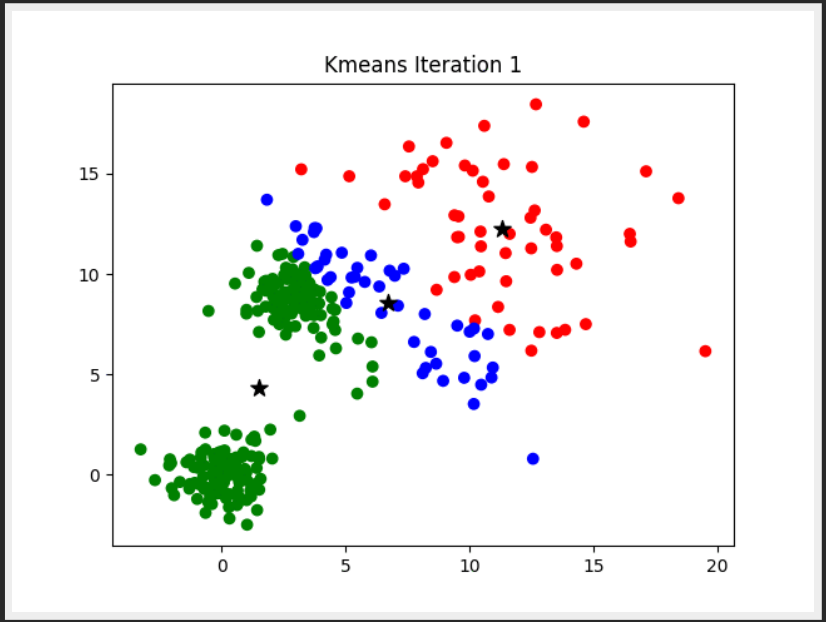
$\sigma^2 = 1$ $\sigma^2 = 0.5$ $\sigma^2 = 3$

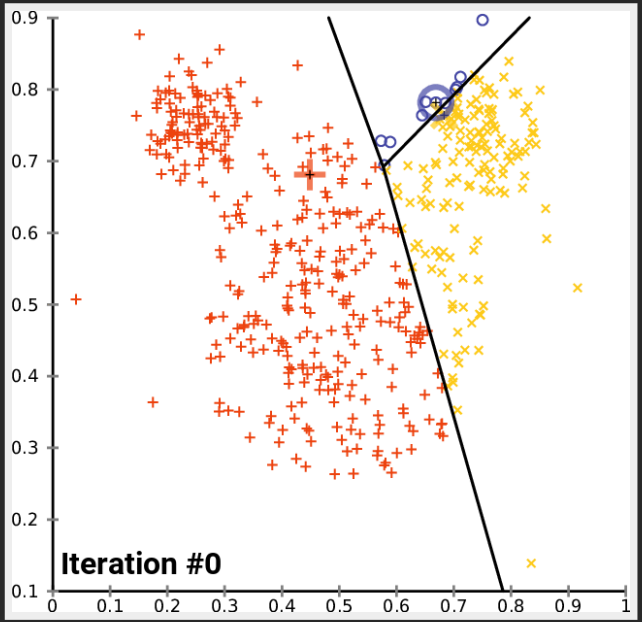


UNSUPERVISED LEARNING

- Organize computing clusters
- Social network analysis
- Market segmentation
- Astronomical data anlysis
- aka Clustering

K-MEANS CLUSTERING





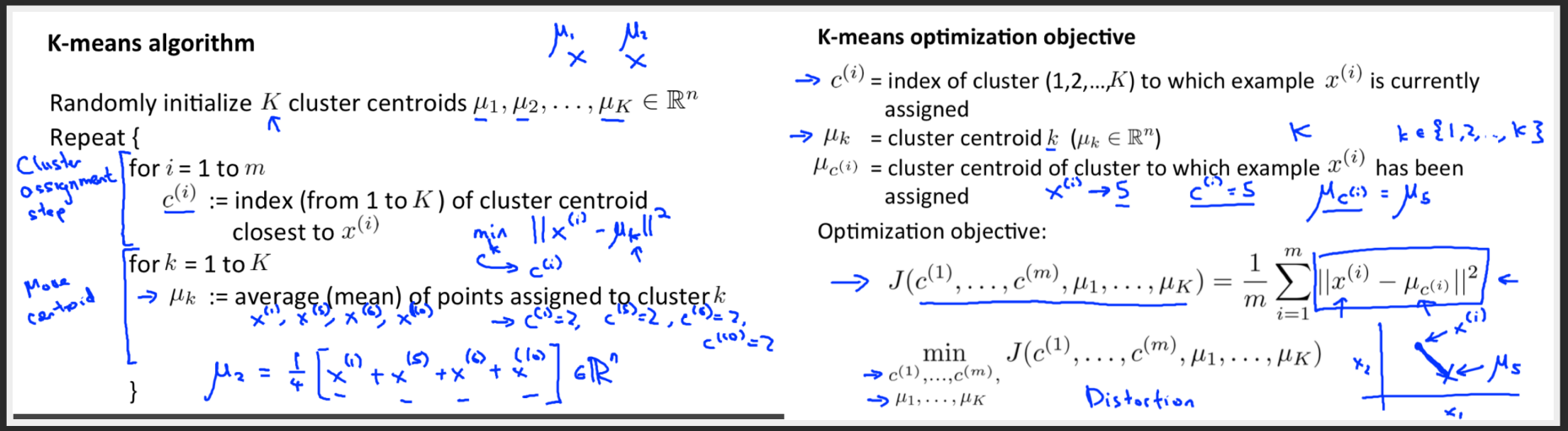
Examples:

- T-shirt sizes (S, M, L, XL)

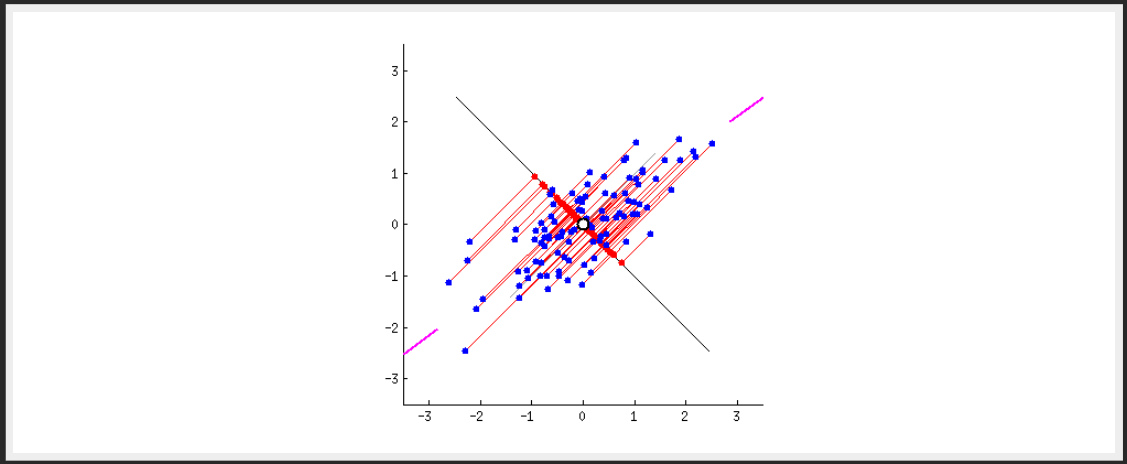
K-MEANS CLUSTERING ALGORITHM

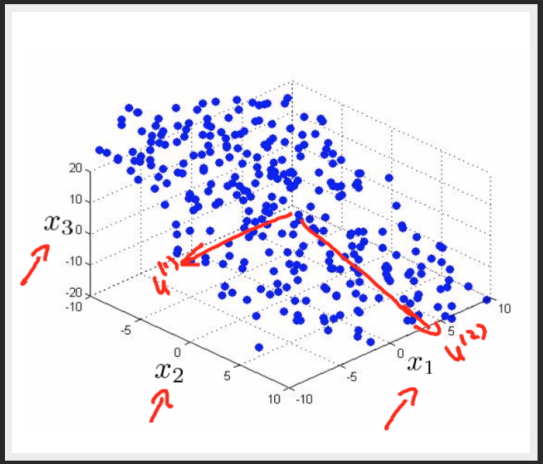
Input:

- K (number of clusters)
- Training set ${x^{(1)},x^{(2)}...x^{(m)}}$



DIMENSIONALITY REDUCTION: PRINCIPAL COMPONENT ANALYSIS (PCA)





- Data compression (Reduce memory needed to store data)
- Speed up other learning algorithms
- Data visualization (reduce to 2D or 3D)

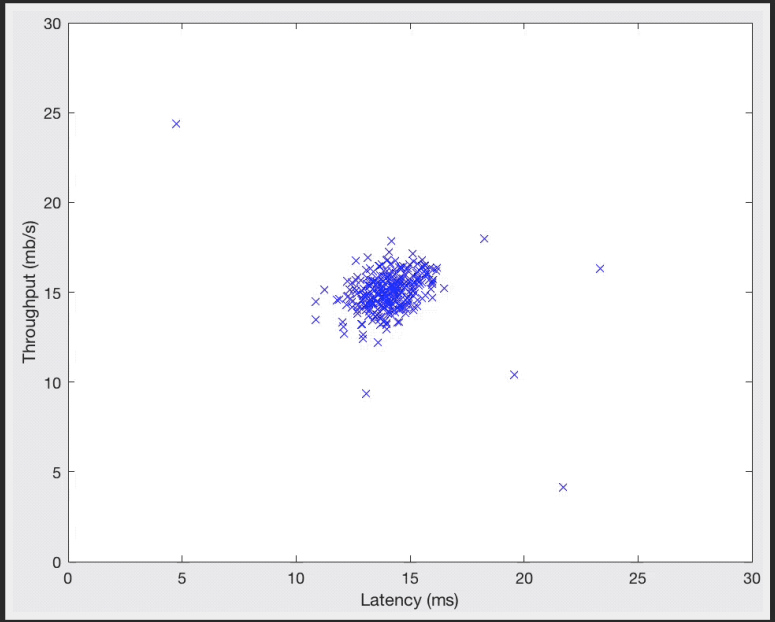
-> Reduce data from n-dimensions to k-dimensions

PCA - STEPS

1. Data preprocessing: feature scaling/mean normalization
2. Calculate covariance matrix
3. Compute eigenvectors of matrix Sigma
   (`[U, _, _] = svd(Sigma);`)
4. `z = U' * x` where
   x := original data z := dim. reduced data
5. principal components k (~ new axes in the data)
   Choose k to be smallest value so that:

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} \leq \cancel{0.01}\ \cancel{0.05}\ 0.10$$

ANOMALY DETECTION



- Fraud detection
- Manufacturing (e.g. engines)
- Monitoring machines in data center

ANOMALY DETECTION - GAUSSIAN DISTRIBUTION

Anomaly detection algorithm

1. Choose features x_i that you think might be indicative of anomalous examples.
2. Fit parameters $\mu_1, \ldots, \mu_n, \sigma_1^2, \ldots, \sigma_n^2$

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

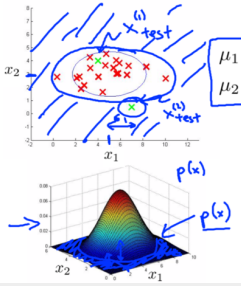
$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \mu_j)^2$$

3. Given new example x, compute $p(x)$:

$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

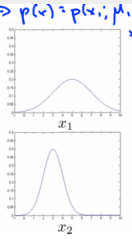
Anomaly if $p(x) < \varepsilon$

Anomaly detection example



$\mu_1 = 5, \sigma_1 = 2$

$\mu_2 = 3, \sigma_2 = 1$



$p(x_1; \mu_1, \sigma_1^2)$

$p(x_2; \mu_2, \sigma_2^2)$

$\varepsilon = 0.02$

$p(x_{test}^{(1)}) = 0.0426 \geq \varepsilon$

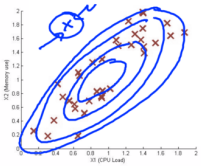
$p(x_{test}^{(2)}) = 0.0021 < \varepsilon$

Anomaly detection with the multivariate Gaussian

1. Fit model $p(x)$ by setting

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)(x^{(i)} - \mu)^T$$



2. Given a new example x, compute

$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

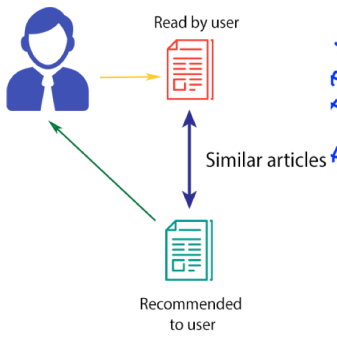
Flag an anomaly if $p(x) < \varepsilon$

RECOMMENDER SYSTEMS

- predict movie ratings (netflix, hulu, ...)
- predict music by style (spotify radio, last.fm, pandora)
- predict products that customers would buy (amazon)

CONTENT-BASED FILTERING

CONTENT-BASED FILTERING



Content-based recommender systems

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last 1	5	5	0	0
Romance forever 2	5	?	?	0
Cute puppies of love 3	?	4	0	?
Nonstop car chases 4	0	0	5	4
Swords vs. karate 5	0	0	5	?

$n_u = 4$, $n_m = 5$, $x_0 = 1$, $x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix}$, $n = 2$

For each user j, learn a parameter $\theta^{(j)} \in \mathbb{R}^3$. Predict user j as rating movie i with $(\theta^{(j)})^T x^{(i)}$ stars. $\theta^{(j)} \in \mathbb{R}^{n+1}$

$x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix}$ $\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$ $(\theta^{(1)})^T x^{(3)} = 5 \times 0.99 = 4.95$

Optimization algorithm:

$$\min_{\theta^{(1)},\dots,\theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

$J(\theta^{(1)}, \dots, \theta^{(n_u)})$

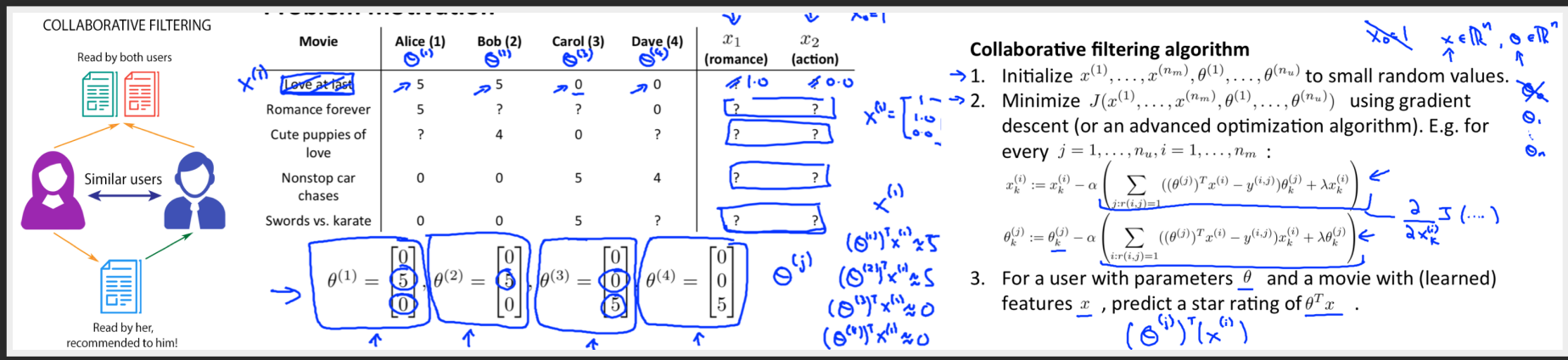
Gradient descent update:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} \quad (\text{for } k = 0)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \right) \quad (\text{for } k \neq 0)$$

$\frac{1}{m^{(j)}}$ $\frac{\partial}{\partial \theta_k^{(j)}} J(\theta^{(1)}, \dots, \theta^{(n_u)})$

COLLABORATIVE FILTERING



Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0		
Romance forever	5	?	?	0		
Cute puppies of love	?	4	0	?		
Nonstop car chases	0	0	5	4		
Swords vs. karate	0	0	5	?		

$\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \theta^{(2)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \theta^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}, \theta^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$

Collaborative filtering algorithm

1. Initialize $x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}$ to small random values.
2. Minimize $J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)})$ using gradient descent (or an advanced optimization algorithm). E.g. for every $j = 1, \ldots, n_u, i = 1, \ldots, n_m$:

$$x_k^{(i)} := x_k^{(i)} - \alpha \left(\sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta_k^{(j)} + \lambda x_k^{(i)} \right)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \right)$$

3. For a user with parameters θ and a movie with (learned) features x, predict a star rating of $\theta^T x$.

QUESTIONS?

SOURCES

- https://www.coursera.org/learn/machine-learning
- https://en.wikipedia.org/wiki/Machine_learning