MACHINE LEARNING (COURSERA)

SUPERVISED LEARNING

x := EXPLANATORY FEATURES

y := TARGET FEATURE

h := HYPOTHESIS FUNCTION

J := COST FUNCTION

θ := PARAMETERS (BETAS)

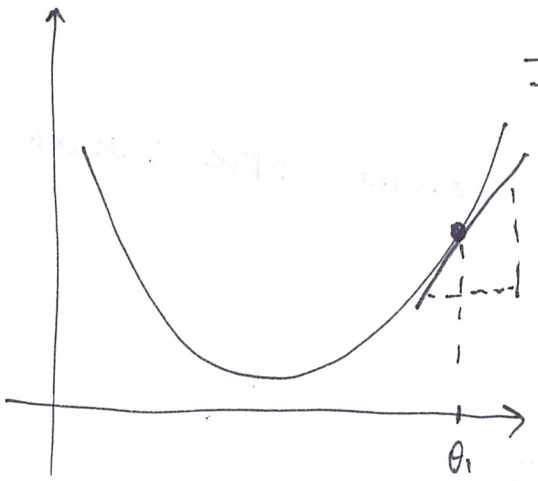
$$\begin{cases} \text{MINIMIZE}_{\theta_0,\theta_1} \dfrac{1}{2m} \displaystyle\sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2 \\ J(\theta_0,\theta_1) = \dfrac{1}{2m} \displaystyle\sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2 \quad \text{"MEAN SQUARED ERROR"} \\ h\left(x^{(i)}\right) = \theta_0 + \theta_1 \cdot x^{(i)} \end{cases}$$

* LINEAR REGRESSION UNIVARIATE

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0,\theta_1) \quad \text{"GRADIENT DESCENT"}$$

$$\begin{cases} temp\_0 := \theta_0 - \alpha \dfrac{\partial}{\partial \theta_0} J(\theta_0,\theta_1) \\ temp\_1 := \theta_1 - \alpha \dfrac{\partial}{\partial \theta_1} J(\theta_0,\theta_1) \\ \theta_0 := temp\_0 \\ \theta_1 := temp\_1 \end{cases}$$

"BATCH": EACH STEP OF GRADIENT DESCENT USES ALL TRAINING EXAMPLES.



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

SLOPE POSITIVE (≥ 0)

$\therefore \theta_1 := \theta_1 - \alpha \cdot$ (POSITIVE NUMBER)

So θ_1 IS GOING TO THE LEFT.

* α SMALL => CONVERGE IS SLOW
* α LARGE => PASS OVER MINIMUM OR DIVERGE!

FOR MULTIPLE FEATURES :

$$h_\theta(x) = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \dots + \theta_n x_n \quad , x_0 = 1$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \quad , \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1} \quad \therefore \boxed{h_\theta(x) = \theta^T \cdot x}$$

FEATURE SCALING

FOR FASTER PROCESS OF GRADIENT DESCENT, IT MAY BE A GOOD IDEA TO SCALE EVERY FEATURE INTO THE FOLLOWING RANGE: $-1 \leq x_i \leq 1$

MEAN NORMALIZATION

$$x_i \leftarrow \frac{x_i - \mu_i}{s_i}$$

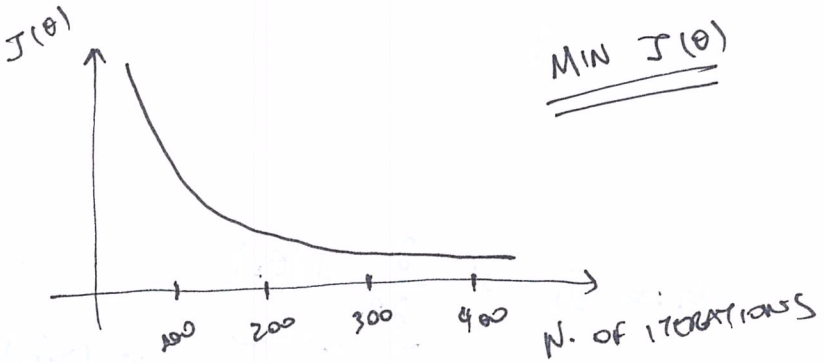
$\mu_i :=$ AVG VALUE OF x_i IN TRAINING SET

$s_i :=$ RANGE (MAX − MIN)

MIN NORMALIZATION

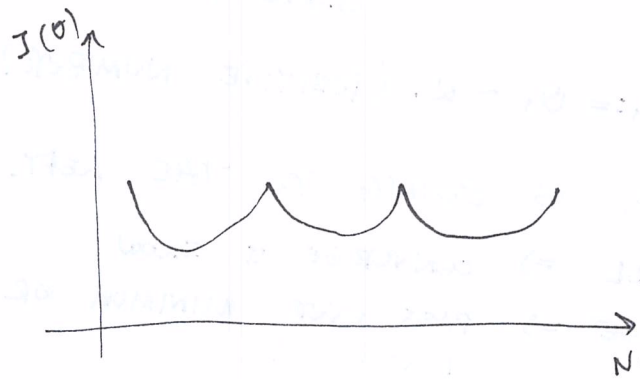
$$x_i \leftarrow \frac{x_i - (\text{MIN } x_i)}{s_i}$$

GRADIENT DESCENT & LEARNING RATE



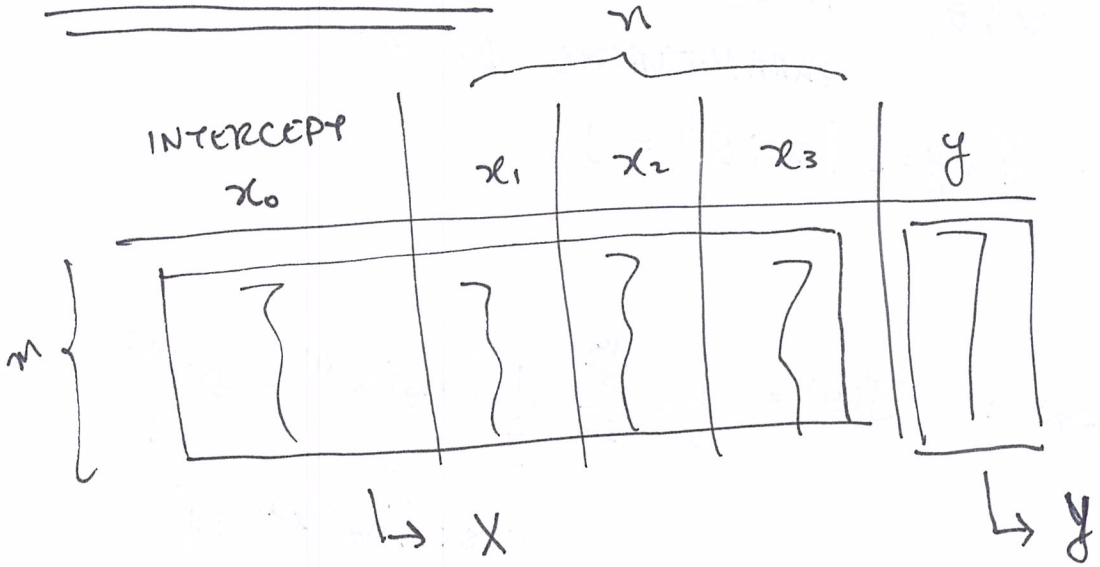
MIN $J(\theta)$

* GRADIENT DESCENT IS WORKING IF VALUE OF $J(\theta)$ IS GETTING SMALLER ALONG THE ITERATIONS.



* USE LEARNING RATE SMALLER (α)

NORMAL EQUATION



$$X \in \mathbb{R}^{m \times (n+1)}$$

$$y \in \mathbb{R}^{m}$$

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

GRADIENT DESCENT

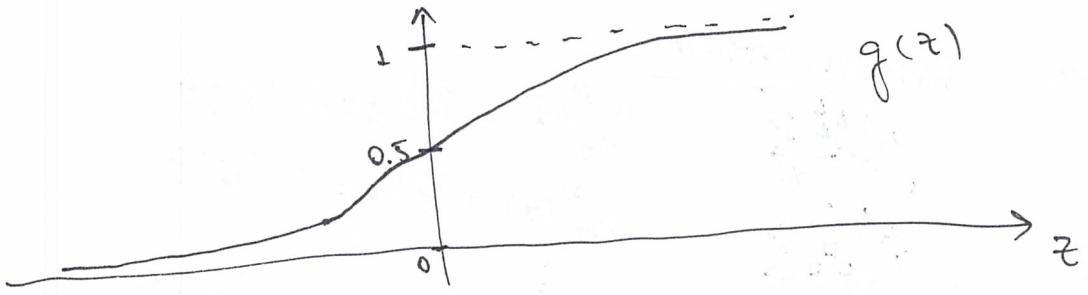
- NEED TO CHOOSE α
- NEEDS MANY ITERATIONS
- WORKS WELL EVEN WHEN n IS LARGE

NORMAL EQUATION

- NO NEED TO CHOOSE α
- DON'T NEED TO ITERATE
- NEED TO COMPUTE $(X^T X)^{-1}$
- SLOW IF n IS LARGE ($n > 10\,000$)

LOGISTIC REGRESSION MODEL

$$0 \leq h_\theta(x) \leq 1 \quad , \quad h_\theta(x) = g(\theta^T x) \quad , \quad g(z) = \frac{1}{1+e^{-z}} \quad (z = \theta^T x)$$



$$\therefore h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} = \text{ESTIMATED PROBABILITY THAT } y=1 \text{ ON INPUT } x$$

SIGMOID FUNCTION = LOGISTIC FUNCTION

$h_\theta(x) = \mathbb{P}(y=1 \mid x;\theta)$ "PROBABILITY THAT $y=1$, GIVEN x, PARAMETERIZED BY θ"

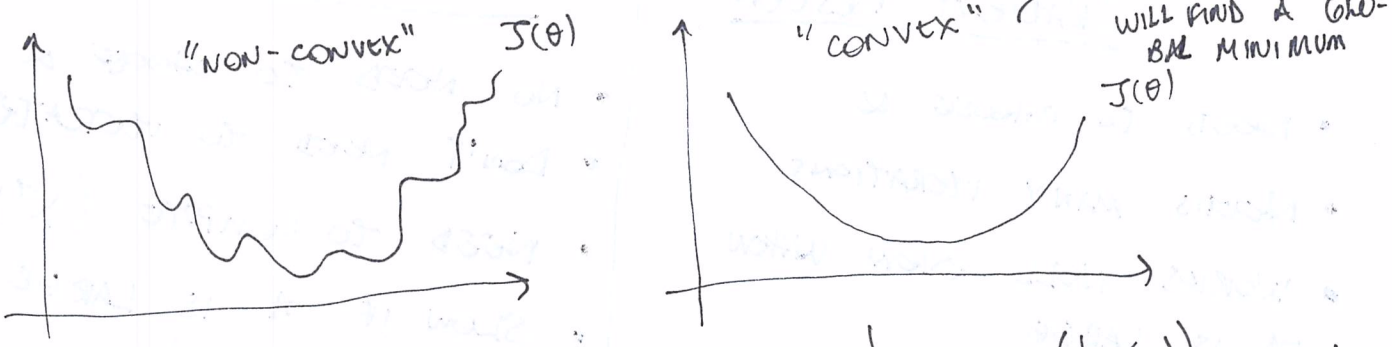
$$\mathbb{P}(y=0 \mid x;\theta) + \mathbb{P}(y=1 \mid x;\theta) = 1$$

COST FUNCTION

LINEAR REGRESSION : $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \underbrace{\frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}_{\text{cost}(h_\theta(x^{(i)}), y)}$

FOR LOGISTIC REGRESSION : $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$ NON LINEAR

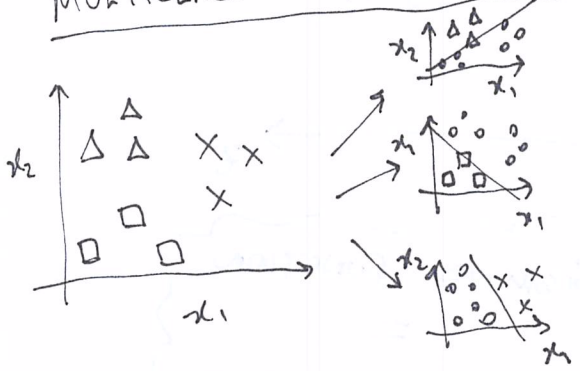
So $J(\theta)$ WILL BE NON-CONVEX (WITH MULTIPLE LOCAL MIN.)



LOGISTIC REGRESSION : $\text{COST}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)), & y=1 \\ -\log(1-h_\theta(x)), & y=0 \end{cases}$

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

MULTICLASS CLASSIFICATION



$h_\theta^{(0)}(x) = \mathbb{P}(y=0 \mid x;\theta)$

$h_\theta^{(1)}(x) = \mathbb{P}(y=1 \mid x;\theta)$

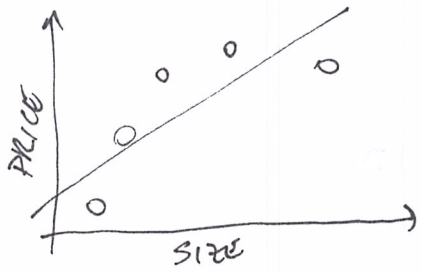
$\vdots$

$h_\theta^{(n)}(x) = \mathbb{P}(y=n \mid x;\theta)$

PREDICTION = $\max_i \left(h_\theta^{(i)}(x)\right)$

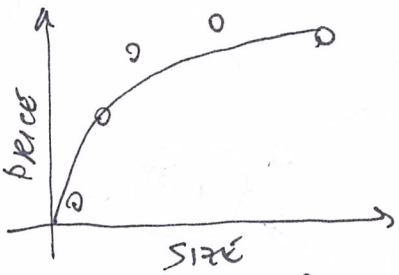
OVERFITTING

"LINEAR REGRESSION EXAMPLE"



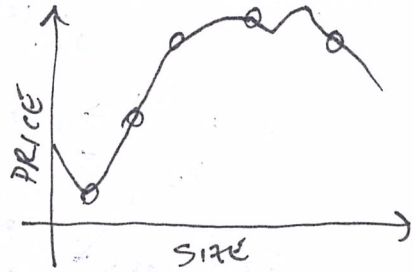
$\theta + \theta_1 x$

"UNDERFIT" / "HIGH BIAS"



$\theta_0 + \theta_1 x + \theta_2 x^2$

"JUST RIGHT"



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

"OVERFIT" / "HIGH VARIANCE"

DEF.: IF TOO MANY FEATURES, THE MODEL MAY FIT THE TRAINING SET TOO WELL ($J(\theta) \approx 0$), BUT FAIL TO GENERALIZE TO NEW EXAMPLE

MEANING:

- BIAS: MODEL HAS A BIAS, IN A WAY THAT DOESN'T FIT TO THE DATA (OR LOOK).
- VARIANCE: MODEL ~~TOO~~ IS VARIABLE, IN A WAY THAT IT CAN FIT PERFECTLY AT ANY DATA.

SOLVING OVERFITTING: SOME OPTIONS

1. REDUCE NUMBER OF FEATURES (MANUALLY / MODEL SELECTION)
2. REGULARIZATION (REDUCES MAGNITUDE/VALUES OF PARAMETERS θ)

FORMULA

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

REDUCES INFLUENCE OF PARAMETERS θ (RISE COST BASED ON SUMMARIZING θ & λ)

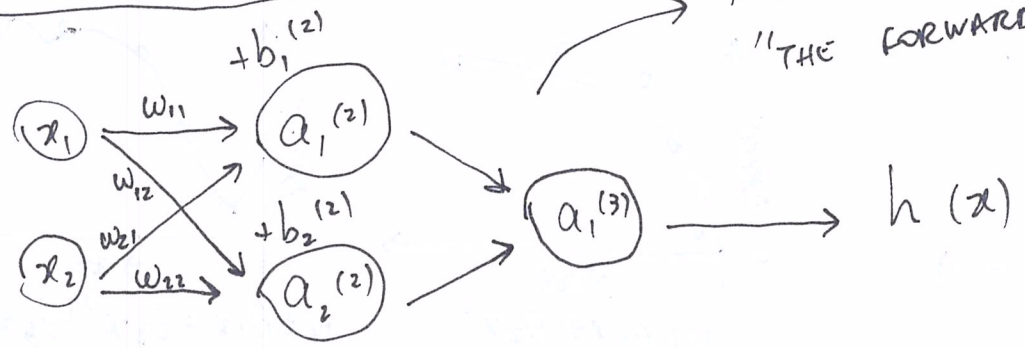
λ := REGULARIZATION PARAMETER

NEURAL NETWORKS

MOTIVATION: NON-LINEAR HYPOTHESES AND NUMBER OF FEATURES ARE TOO LARGE. EX.: IMAGE RECOGNITION.

MOREOVER: LINEAR METHODS ARE "WEAK" (MAKE STRONG ASSUMPTIONS) AND CAN ONLY EXPRESS RELATIVELY SIMPLE FUNCTIONS OF INPUTS.

MODEL REPRESENTATION



THIS PROCESS IS CALLED "THE FORWARD PROPAGATION"

LAYER 1 "INPUT LAYER" — LAYER 2 "HIDDEN LAYER" — LAYER 3 "OUTPUT LAYER"

$$a_1^{(2)} = g\left(w_{11}^{(1)} \cdot x_1 + w_{21}^{(1)} \cdot x_2 + b_1^{(2)}\right)$$

$$a_2^{(2)} = g\left(w_{12}^{(1)} \cdot x_1 + w_{22}^{(1)} \cdot x_2 + b_2^{(2)}\right)$$

* $z_1^{(2)} = w_{11}^{(1)} \cdot x_1 + w_{21}^{(1)} \cdot x_2 + b_1^{(2)}$

$a_i^{(j)}$:= "ACTIVATION" OF UNIT i IN LAYER j

$w_{ik}^{(j)}$:= "WEIGHTS" OF LAYER j FROM UNIT i TO UNIT k

$g(\cdot)$:= ACTIVATION FUNCTION

MULTICLASS CLASSIFICATION

$y^{(i)}$ RANGES, FOR EXAMPLE, AS: $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

* KIND OF AN INTRODUCTION TO SOFTMAX

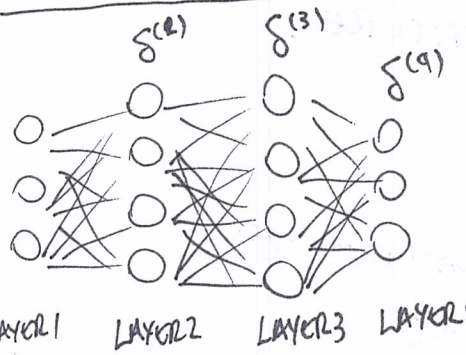
COST FUNCTION

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

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

- L = # TOTAL LAYERS
- s_l = # UNITS IN LAYER l
- K = # OUTPUT UNITS/CLASSES
- m = # OF OBSERVATIONS

BACK PROPAGATION



$$\delta_j^{(4)} = a_j^{(4)} - y_j \quad (\text{PREDICT MINUS OBSERV.})$$

$$\delta^{(3)} = \left(\Theta^{(3)}\right)^T \delta^{(4)} .* g'(z^{(3)})$$

$$\delta^{(2)} = \left(\Theta^{(2)}\right)^T \delta^{(3)} .* g'(z^{(2)})$$

"ALGORITHM"

TRAINING SET $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$

SET $\Delta_{ij}^{(l)} = 0$

FOR $i = 1$ TO m:

SET $a^{(1)} = x^{(i)}$

PERFORM FORWARD PROPAGATION TO COMPUTE $a^{(l)}$ $(l = 2, 3, \dots, L)$

USING $y^{(i)}$, COMPUTE $\delta^{(L)} = a^{(L)} - y^{(i)}$

COMPUTE $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$

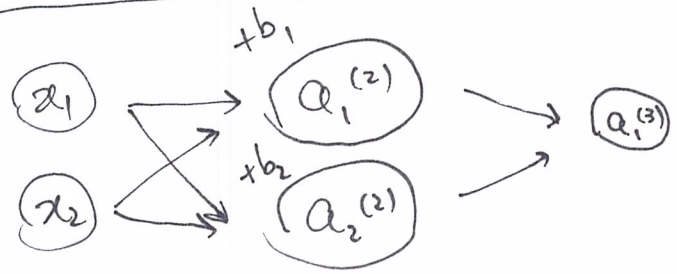
$$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \cdot \delta_i^{(l+1)}$$

$$\begin{cases} D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} & , j \neq 0 \\ D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} & , j = 0 \end{cases}$$

(DO NOT USE REGULARIZATION ON BIAS)

$$\therefore \boxed{\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}}$$

WEIGHTS RANDOM INITIALIZATION



$$\boxed{\Theta_{ij}^{(l)} = 0}, \forall i, j, l \Rightarrow$$

$$a_1^{(2)} = a_2^{(2)}, \quad \delta_1^{(2)} = \delta_2^{(2)},$$

$$\frac{\partial}{\partial \Theta_{01}^{(1)}} J(\Theta) = \frac{\partial}{\partial \Theta_{02}^{(1)}} J(\Theta)$$

IF ALL HIDDEN UNITS ARE IDENTICAL THEN IT WOULD BE LIKE A MODEL OF ONLY ONE FEATURE.

SUMMARY

* NUMBER OF INPUT UNITS = DIMENSION OF FEATURES $x^{(i)}$
* NUMBER OF OUTPUT UNITS = NUMBER OF CLASSES
* NUMBER OF HIDDEN UNITS PER LAYER = USUALLY THE MORE THE BETTER
* NUMBER OF HIDDEN LAYERS = DEFAULT IS 1, BUT USUALLY THE MORE THE BETTER (# OF HIDDEN UNITS EQUAL BETWEEN LAYERS)

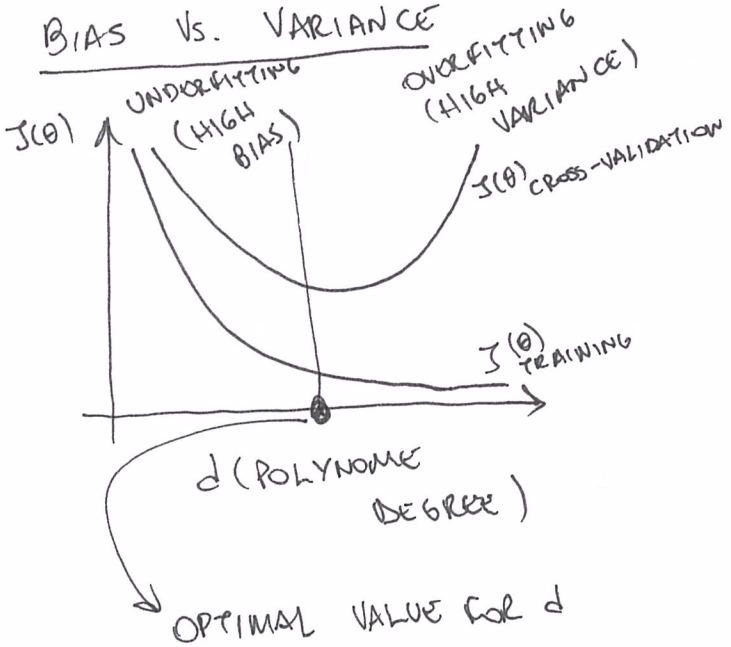
TRAINING A NN

1. RANDOMLY INITIALIZE THE WEIGHTS
2. IMPLEMENT FP TO GET $h_\Theta(x^{(i)})$ FOR ANY $x^{(i)}$
3. IMPLEMENT COST FUNCTION
4. IMPLEMENT BP TO COMPUTE PARTIAL DERIVATIVES
5. USE GRADIENT CHECKING TO CONFIRM THAT BP WORKS. THEN DISABLE GRADIENT CHECKING.
6. USE GRADIENT DESCENT OR A BUILT-IN OPTIMIZATION FUNCTION TO MIN $J(\Theta)$ WITH WEIGHTS.

MODEL SELECTION

1. OPTIMIZE THE PARAMETERS Θ (ESTIMATIONS) USING THE TRAINING SET FOR EACH POLYNOMIAL DEGREE
2. FIND THE POLYNOMIAL DEGREE $\underline{d}$ WITH THE LEAST ERROR USING CROSS-VALIDATION SET
3. ESTIMATE FINAL ERROR USING TEST SET.

BIAS Vs. VARIANCE



REGULARIZATION

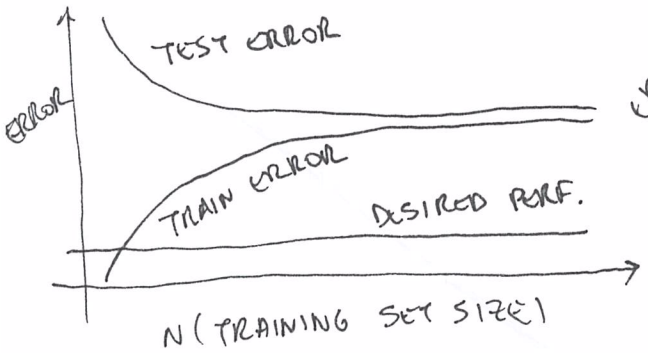
$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h(x^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{m}\sum_{j=1}^{n}\theta_j^2$$

- LARGE λ => HIGH BIAS (UNDERFIT)
  $\lambda = 10000 \Rightarrow \theta_1 \approx 0, \theta_2 \approx 0 \ldots h(x) \approx \theta_0$
- INTERMEDIATE λ => JUST RIGHT
- SMALL λ => HIGH VARIANCE (OVERFIT)
  $\lambda = 0$

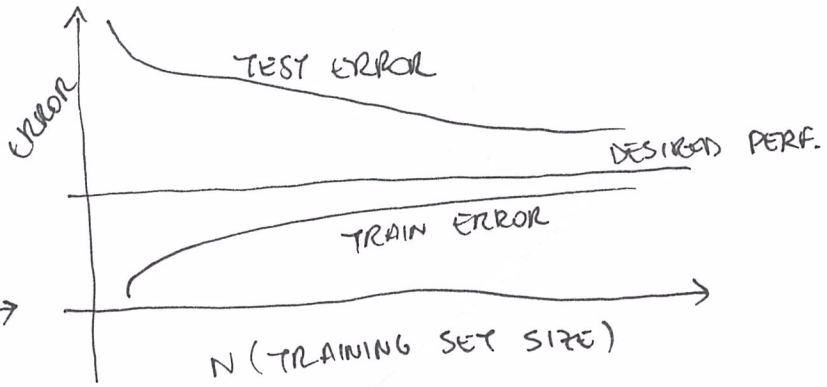
ITERATE THROUGH LIST OF LAMBDAS:
$\lambda \in \{0, 0.01, 0.02, 0.08, 0.16, 0.32, 0.64, 1.28, 2.56, 5.12, 10.24\}$

LEARNING CURVES

HIGH BIAS:



HIGH VARIANCE:



SUMMARY

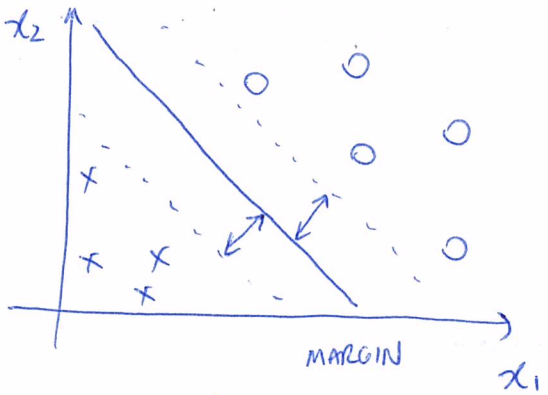
- GETTING MORE TRAINING EXAMPLES: FIXES HIGH VARIANCE
- TRYING SMALLER SETS OF FEATURES: FIXES HIGH VARIANCE
- ADDING FEATURES: FIXES HIGH BIAS
- ADDING POLYNOMIAL FEATURES: FIXES HIGH BIAS
- DECREASING λ: FIXES HIGH BIAS
- INCREASING λ: FIXES HIGH VARIANCE

SUPPORT VECTOR MACHINE

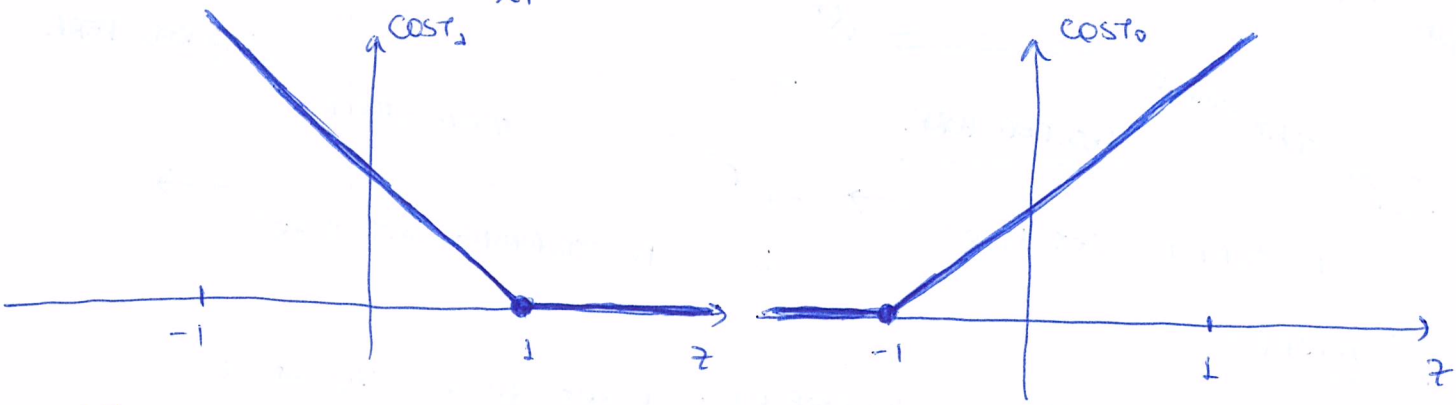
$$J(\theta) = \min_{\theta} C \cdot \sum_{i=1}^{m} \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

(COST FUNCTION OF SVM)

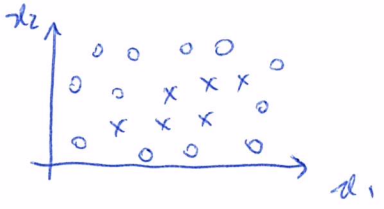
$$\text{S.t.} \begin{cases} \theta^T x^{(i)} \geq 1, & \text{if } y^{(i)} = 1 \\ \theta^T x^{(i)} \leq -1, & \text{if } y^{(i)} = 0 \end{cases}$$

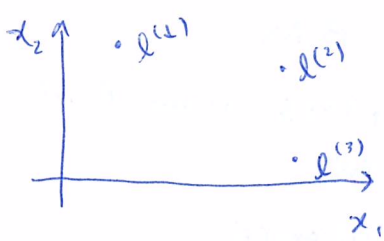


LINEAR DECISION BOUNDARY



KERNELS (FOR NON-LINEAR DECISION BOUNDARY)





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

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

"GAUSSIAN KERNEL"

$$J(\theta) = \min_{\theta} C \cdot \sum_{i=1}^{m} \left[y^{(i)} \text{cost}_1(\theta^T f_i) + (1 - y^{(i)}) \text{cost}_0(\theta^T f_i) \right] + \frac{1}{2} \sum_{j=1}^{m} \theta_j^2$$

(COST FUNCTION OF SVM FOR KERNELS)

SVM PARAMETERS

$C \left(= \frac{1}{\lambda}\right)$ { LARGE : LOW BIAS, HIGH VARIANCE
SMALL : HIGH BIAS, LOW VARIANCE

σ^2 { LARGE : FEATURES f_i VARY MORE SMOOTHLY. HIGH BIAS, LOW VARIANCE
SMALL : FEATURES f_i VARY LESS SMOOTHLY. LOW BIAS, HIGH VARIANCE

MULTI-CLASS CLASSIFICATION

TRAIN K SVMs, ONE TO DISTINGUISH $y = i$ FROM THE REST

FOR $i = 1, 2, \dots, K$: $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(K)}$

PICK CLASS i WITH LARGEST $(\theta^{(i)})^T x$

LOGISTIC REG. VS. SVMs

n = NUMBER OF FEATURES
m = NUMBER OF TRAINING EXAMPLES

- n LARGE : LOGISTIC REGRESSION OR SVM WITHOUT KERNEL
- n SMALL, m INTERMEDIATE : SVM WITH GAUSSIAN KERNEL
- n SMALL, m LARGE : LOGISTIC REG. OR SVM WITHOUT A KERNEL

UNSUPERVISED LEARNING

GIVEN AN UNLABELED DATASET, FIND "STRUCTURE" IN THE DATA. TRAINING SET OF FORM $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ WITHOUT LABELS $y^{(i)}$.

K-MEANS

0° STEP: DEFINE K RANDOM CENTROID

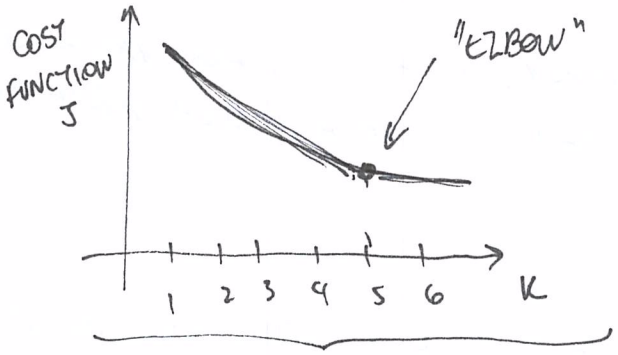
1° STEP: IDENTIFY EACH ELEMENT WH GROUP IS IN

2° STEP: RECALCULATE CENTROID BASED ON ALL ELEMENTS FROM THE GRO

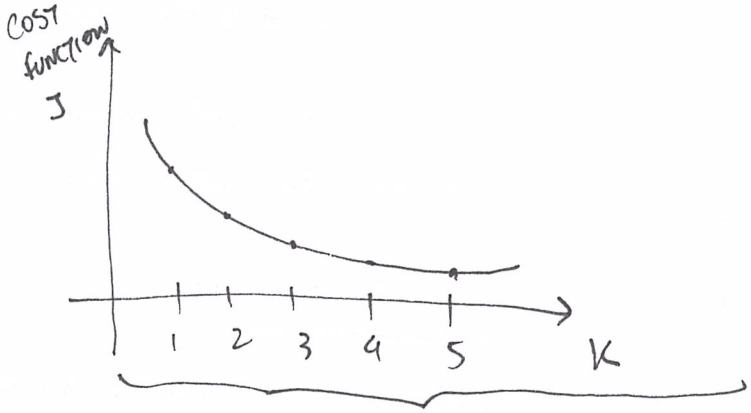
$$* \quad \min_k \left\| x^{(i)} - \mu_k \right\|^2$$

STEP0: SELECT K OBSERVATIONS AS RANDOM INITIAL POINTS

CHOOSING THE VALUE OF K



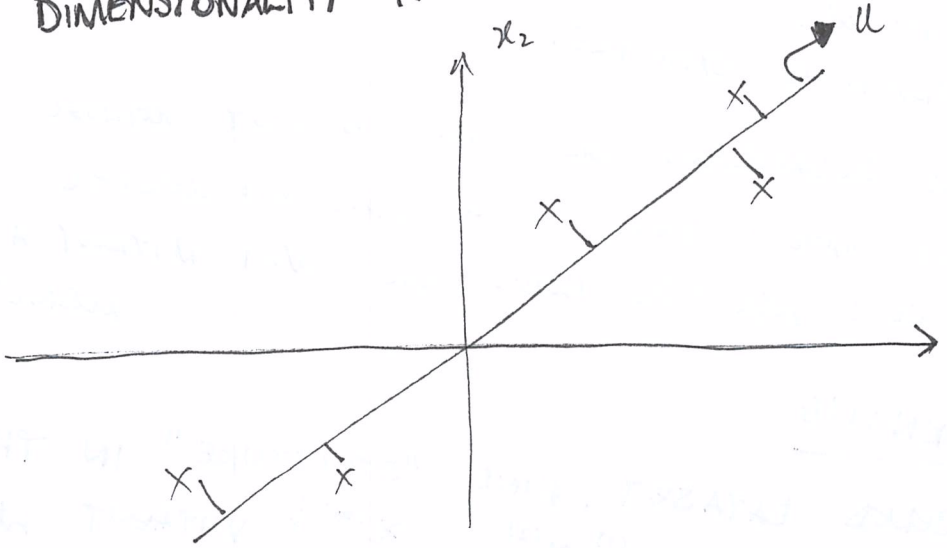
ELBOW METHOD



HERE THE ELBOW METHOD DOESN'T WORK

PCA : PRINCIPAL COMPONENT ANALYSIS

DIMENSIONALITY REDUCTION: DATA COMPRESSION & VISUALIZATION



- REDUCE 2-DIMENSION TO 1-DIMENSION
  BUT, YOU CAN
- REDUCE N-DIMENSION TO K-DIMENSION

* PCA IS NOT LINEAR REGRESSION
- → PCA IS TRYING TO OPTIMIZE THE SHORTEST DISTANCE ORTHOGONAL FROM THE DOTS TO THE LINE.
- → LINEAR REGRESSION MINIMIZE THE SQUARE ERROR BETWEEN POINT AND LINE.
- → PCA TREATS ALL FEATURES ALIKE, DIFF. FROM LINEAR REG.

TO REDUCE DATA FROM n-DIMENSIONS TO K-DIMENSIONS:

- $\Sigma = \frac{1}{m} \sum_{i=1}^{n} (x^{(i)})(x^{(i)})^T$ "COVARIANCE MATRIX" = SIGMA
  ($x^{(i)}$: $n \times 1$, $(x^{(i)})^T$: $1 \times n$)

- SVD (SIGMA) → U "EIGENVECTORS" OF MATRIX Σ = SINGULAR VALUE DECOMPOSITION

$$U = \underbrace{\begin{bmatrix} | & | & | & & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \cdots & u^{(n)} \\ | & | & | & & | \end{bmatrix}}_{k}, \quad U \in \mathbb{R}^{n \times n}$$

$$z = \underbrace{\begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ | & | & & | \end{bmatrix}^T}_{n \times k} \underbrace{x}_{n \times 1} = \underbrace{\underbrace{\begin{bmatrix} — & (u^{(1)})^T & — \\ — & (u^{(2)})^T & — \\ & \vdots & \\ — & (u^{(k)})^T & — \end{bmatrix}}_{k \times n} \underbrace{x}_{n \times 1}}_{k \times 1}$$

PCA TRIES TO MINIMIZE: $\frac{1}{m} \sum_{i=1}^{m} \| x^{(i)} - x^{(i)}_{APPROX} \|^2$

WHERE $x^{(i)}_{APPROX}$ IS THE RESULT OF THE APPLICATION OF PCA.

$$\left(\underbrace{x_{APPROX}}_{n \times 1} = \underbrace{U_{REDUCE}}_{n \times k} \cdot \underbrace{z}_{k \times 1} \right)$$

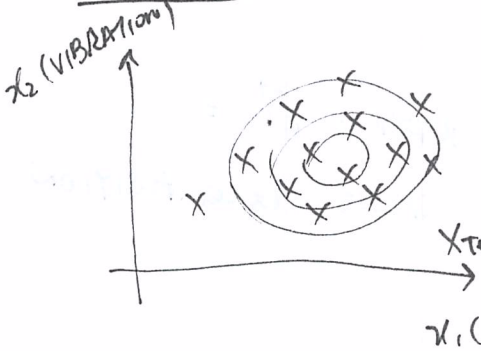
SUMMARY (WHY TO USE PCA?)

→ COMPRESS THE DATA TO TAKE LESS COMPUTER MEMORY

→ REDUCE DIMENSION OF INPUT DATA TO SPEED UP LEARNING ALGORITHM

→ TO VISUALIZE HIGH-DIMENSIONAL DATA ($k=2$ OR $k=3$)

ANOMALY DETECTION



$p(x_{TEST}) < \varepsilon \Rightarrow$ FLAG ANOMALY

$p(x_{TEST}) \geq \varepsilon \Rightarrow$ OK

→ LOWER PROBABILITY

DENSITY ESTIMATIO

ANOMALY DETECTION VS. SUPERVISED LEARNING

- VERY SMALL Nº OF POSITIVE EXAMPLES - LARGE Nº OF NEGATIVE EXAMPLES - MANY DIFF. TYPES OF ANOMALIES	- LARGE Nº OF POSITIVE AND NEGATIVE EXAMPLES - ENOUGH POSITIVE EXAMPLES FOR ALGORITHM GET SENSE OF WHAT POSITIVE EXAMPLES ARE LIKE.

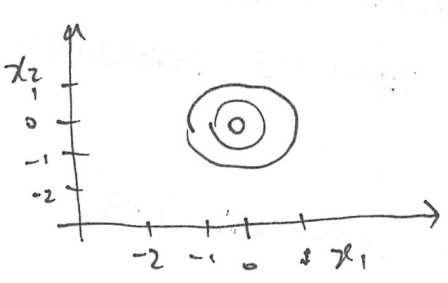
MULTIVARIATE GAUSSIAN (NORMAL) DISTRIBUTION

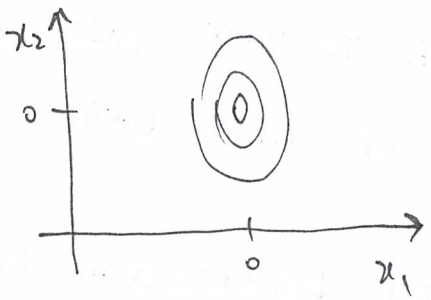
DON'T MODEL $p(x_1), p(x_2), \ldots$ SEPARATELY, MODEL $p(x)$ ALL IN ONE:

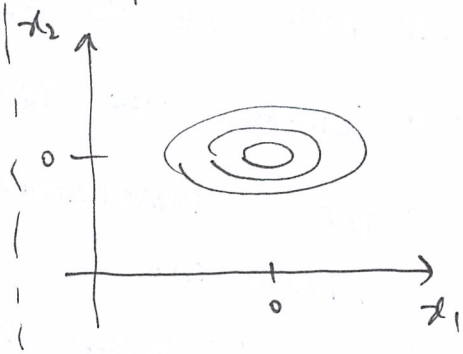
$\mu \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{n \times n}$ (COVARIANCE MATRIX)

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

$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ | $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 1 \end{bmatrix}$ | $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$







CONTENT BASED RECOMMENDATIONS

MOVIE	U1	U2	U3	U4	x_1 (ROMANCE)	x_2 (ACTION)
A	5	5	0	0	0.9	0
B	5	?	?	0	1.0	0.01
C	?	4	0	?	0.99	0
D	0	0	5	?	0.1	1.0

$x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix}$, $x^{(2)} = \begin{bmatrix} 1 \\ 1.0 \\ 0.01 \end{bmatrix}$, ... FEATURES FOR EACH MOVIE BASED ON USER EXPERIENCES / RATINGS

$\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$, ... PARAMETER OF PREFERENCE FOR EACH USER (ESTIMATED)

PROBLEM FORMULATION (LEARN $\theta^{(j)}$)

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

- $r(i,j) = 1$, IF USER j HAS RATED MOVIE i (0 OTHERWISE)
- $y(i,j) = ?$, IF RATING BY USER j ON MOVIE i NOT DEFINED
- $\theta^{(j)}$ PARAMETER VECTOR FOR USER j
- $x^{(i)}$ FEATURE VECTOR FOR MOVIE i
- PREDICTED RATING: $(\theta^{(j)})^T (x^{(i)})$ FOR USER j AND MOVIE i

COLLABORATIVE FILTERING (PROBLEM FORMULATION TO LEARN $x^{(i)}$)

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

COLLABORATIVE FILTERING ALGORITHM

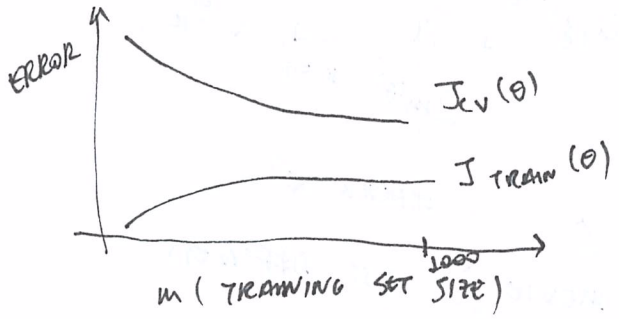
① INITIALIZE $x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}$ TO SMALL RANDOM VALUES

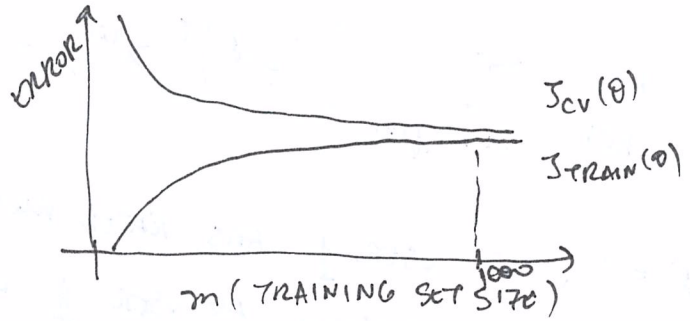
② MINIMIZE $J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$ USING GRADIENT DESCENT. E.G.:

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

LEARNING WITH LARGE DATASETS

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$





ON THESE CASES, THE FIRST INDICATES THAT m SHOULD BE HIGHER THAN 1000, AND THE SECOND IT SEEMS THAT 1000 IS SUFFICIENT. (ERROR-DRIVEN)

- BATCH GRADIENT DESCENT: USE ALL m EXAMPLES IN EACH ITERATION
- STOCHASTIC GRADIENT DESCENT: USE 1 EXAMPLE IN EACH ITERATION
- MINI-BATCH GRADIENT DESCENT: USE B EXAMPLES IN EACH ITERATION

ONLINE LEARNING

GIVEN SOME CHARACTERISTICS $\underline{\underline{x}}$ WANT TO LEARN TO PREDICT $\underline{\underline{y}}$ BASED ON PARAMETERS $\underline{\underline{\theta}} := p(y=1 \mid x; \theta)$

REPEAT FOREVER {
GET (x, y) CORRESPONDING TO USER
UPDATE θ USING (x, y):

$$\theta_j := \theta_j - \alpha (h_\theta(x) - y) \cdot x_j \quad (j = 0, \dots, n)$$

}

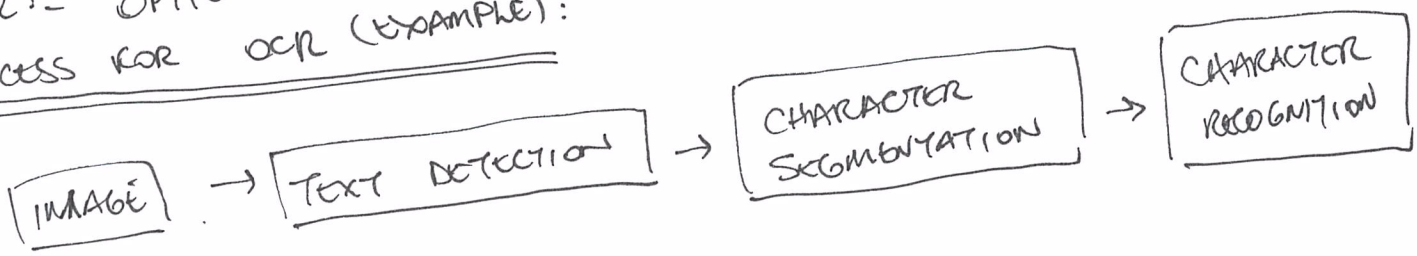
FOR EACH OBSERVATION RETRAIN THE PARAMETERS θ.

MACHINE LEARNING PIPELINE & PHOTO OCR

SYSTEM WITH MANY STAGES / COMPONENTS, SEVERAL OF WHICH MAY USE A ML TECHNIQUE.

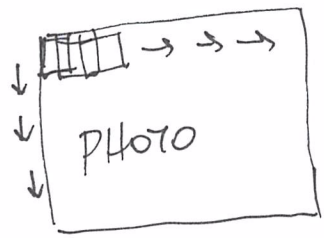
OCR := OPYICAL CHARACTER RECOGNITION

PROCESS FOR OCR (EXAMPLE):



SLIDING WINDOWS

TO COMPLETE THE FIRST STEP (TEXT DETECTION) IT'S USED THE TECHNIQUE OF SLIDING WINDOWS:



EACH RECTANGLE TRIES TO IDENTIFY IF THERE ARE ANY TYPE OF TEXT ON IT (SUPERVISED LEARNING) AND AFTER EACH ITERATION CHANGES SIZE OF RECTANGLE.