ECE421
Intro. to Machine Learning

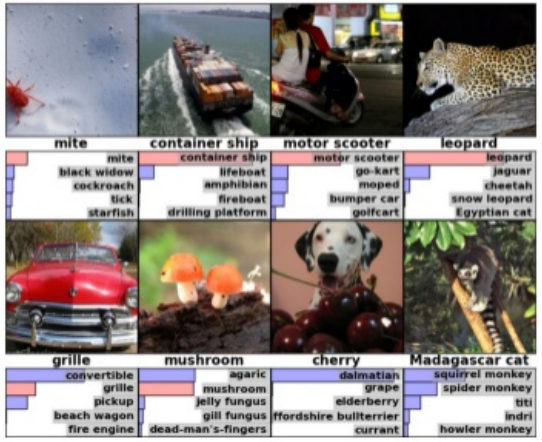
Fall 2019

Ashish Khisti

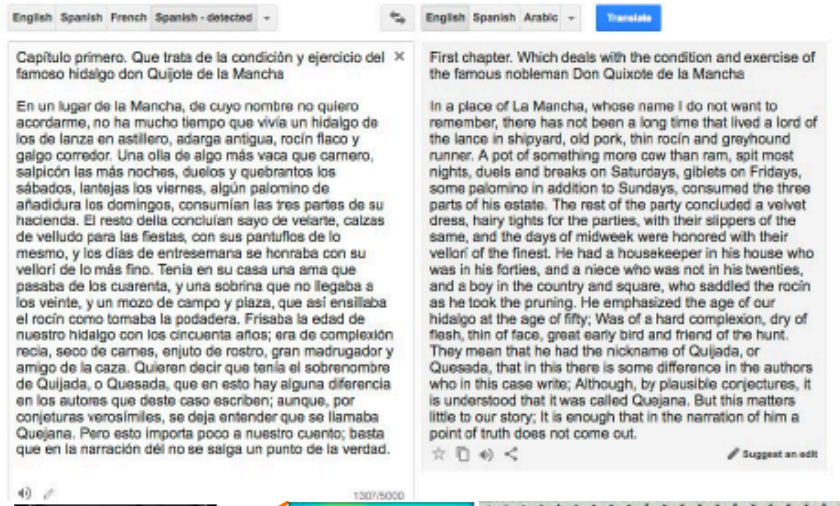
ImageNet Challenge

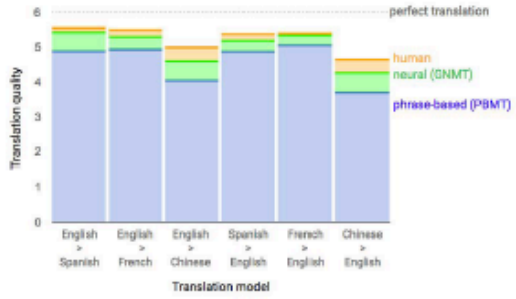
IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

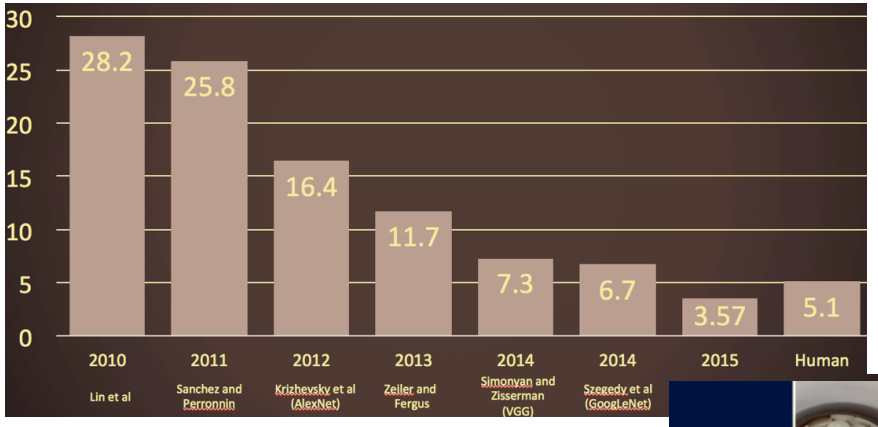


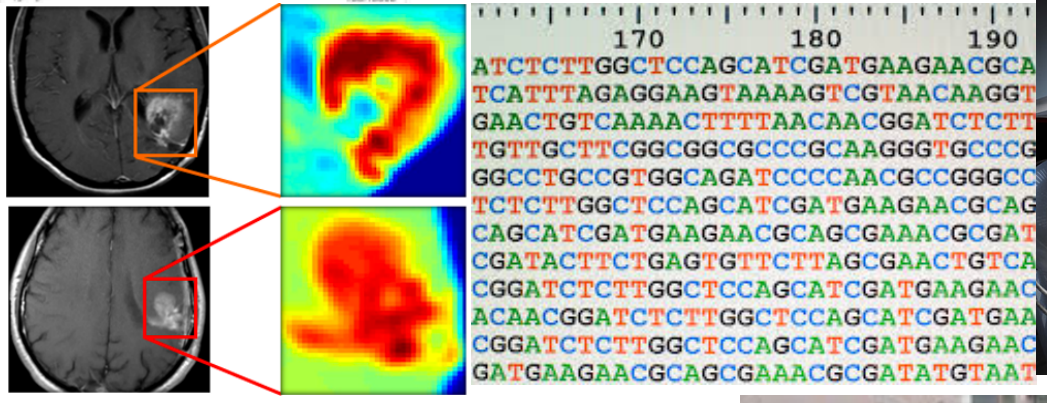
- Natural language processing

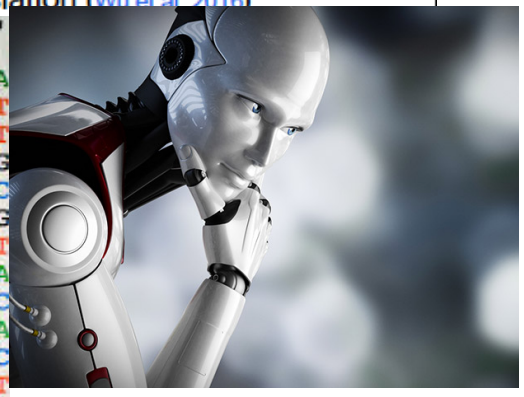


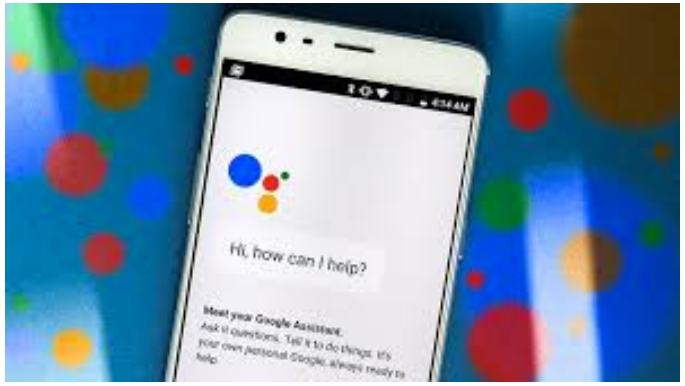


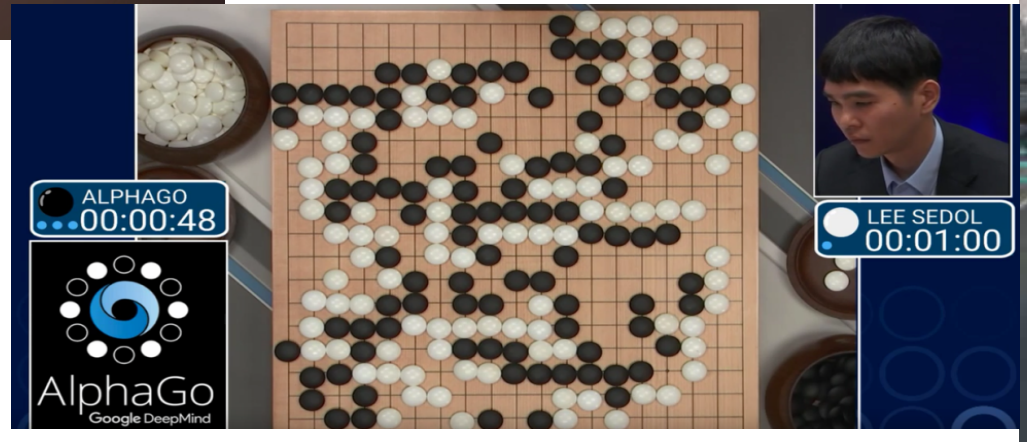
Google's Neural Machine Translation (Wu et al. 2016)





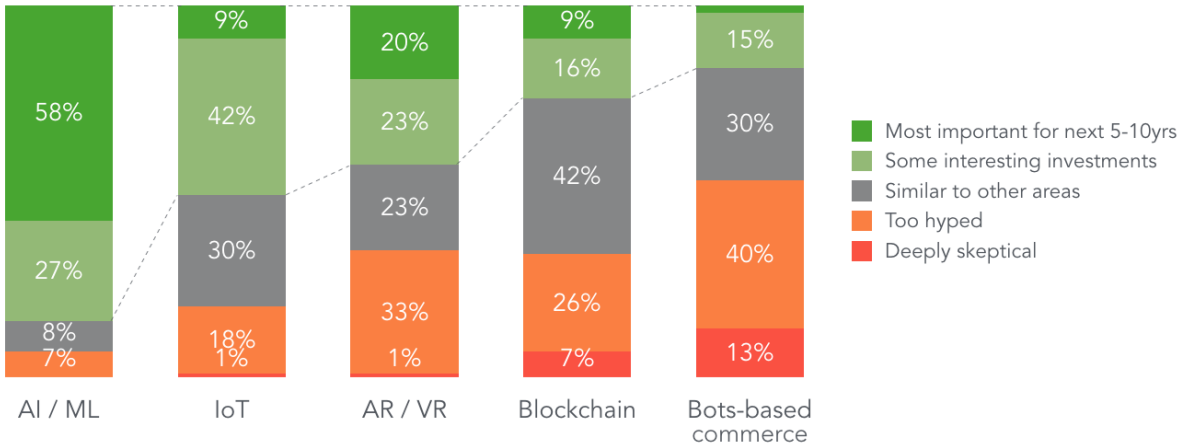


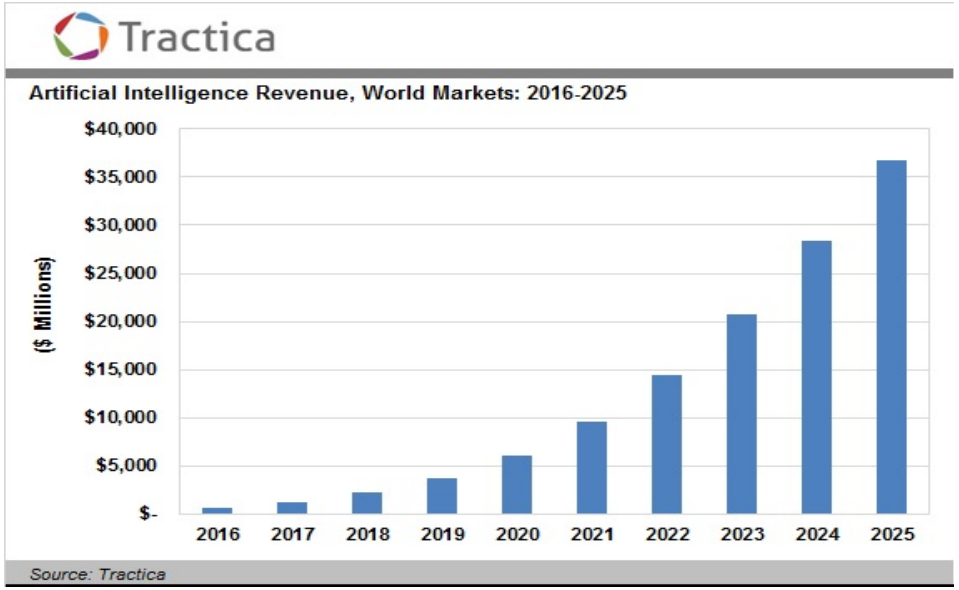




Most VCs are most excited about AI & Machine Learning as their most important investment theme for the coming 5-10 years.

Q. How do you feel about the following investment areas?

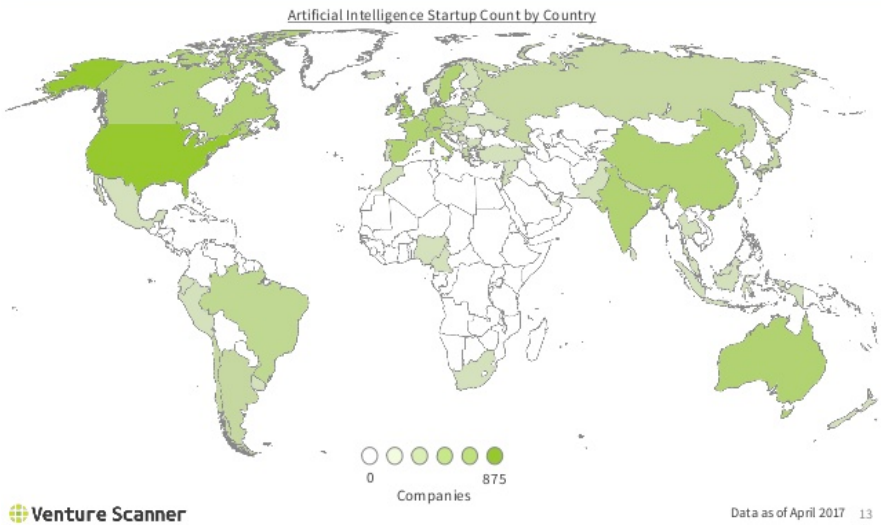


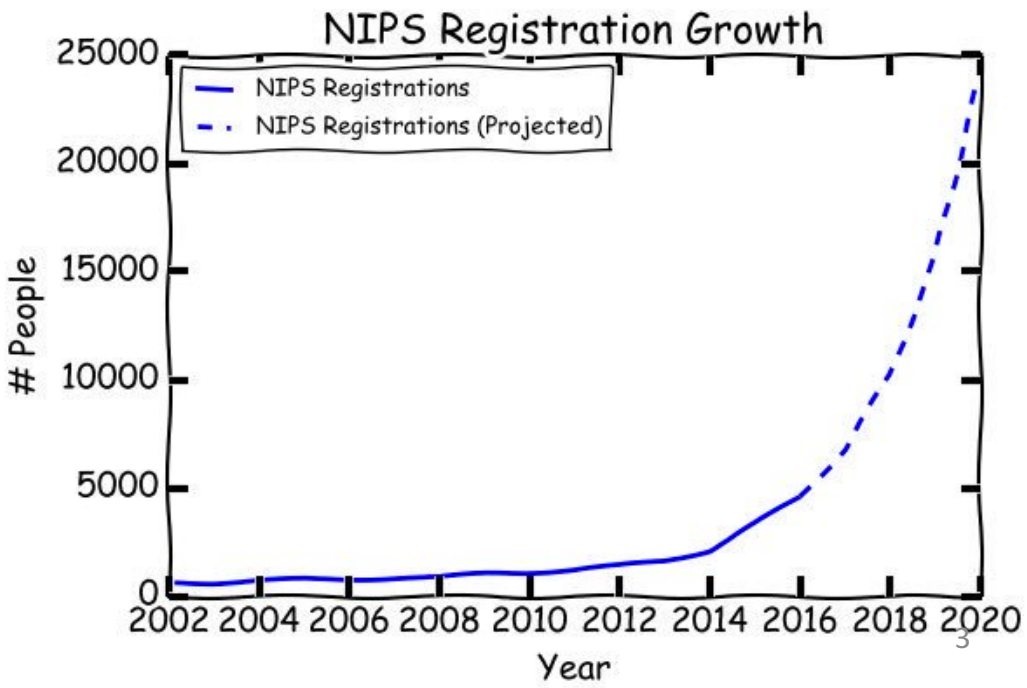


17 Source: Upfront Ventures survey of VCs (N=115), Jan 2017

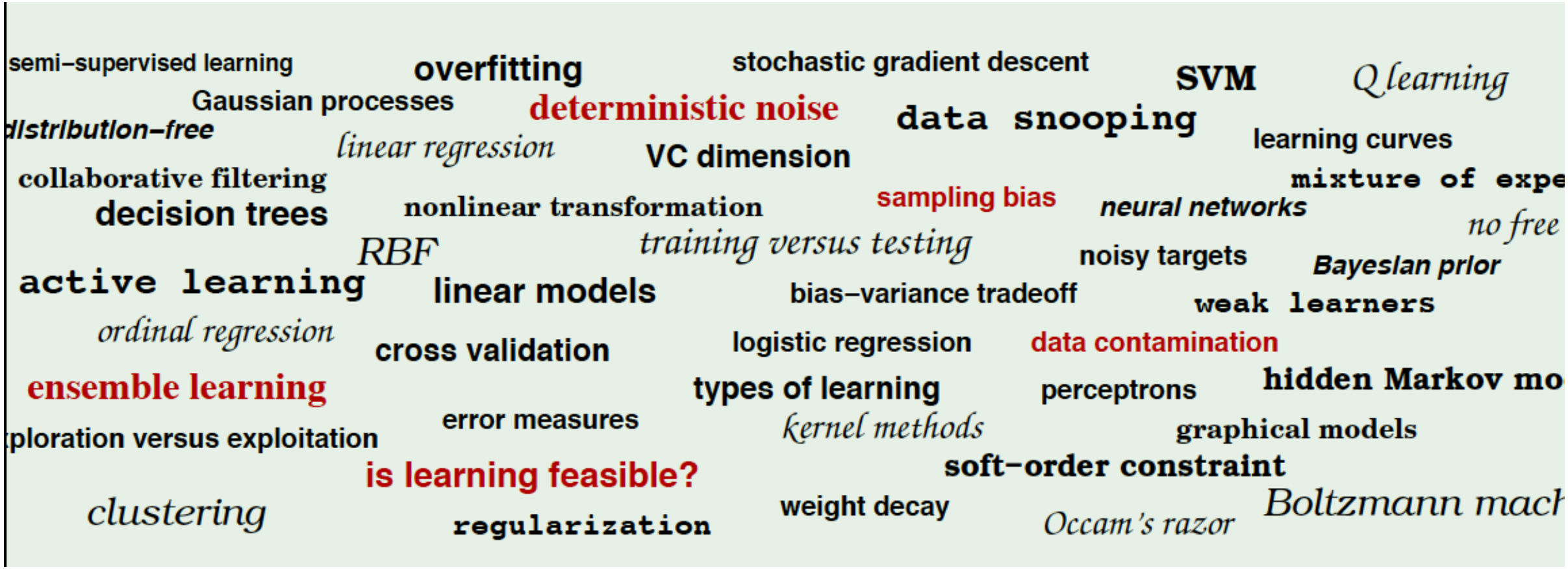
upfront VENTURES

Artificial intelligence startups are a global phenomenon





Machine Learning Jargon



Overview

Learning Methods

Supervised Learning

- Linear Models
- Neural Networks
- Support Vector Machines

Unsupervised Learning

- Clustering
- Density Estimation
- Principal Component Anal.

Learning Theory

- PAC Learning
- VC Dimension
- Bias Variance Tradeoff

Learning Principles

- Regularization
- Validation

Linear Classification

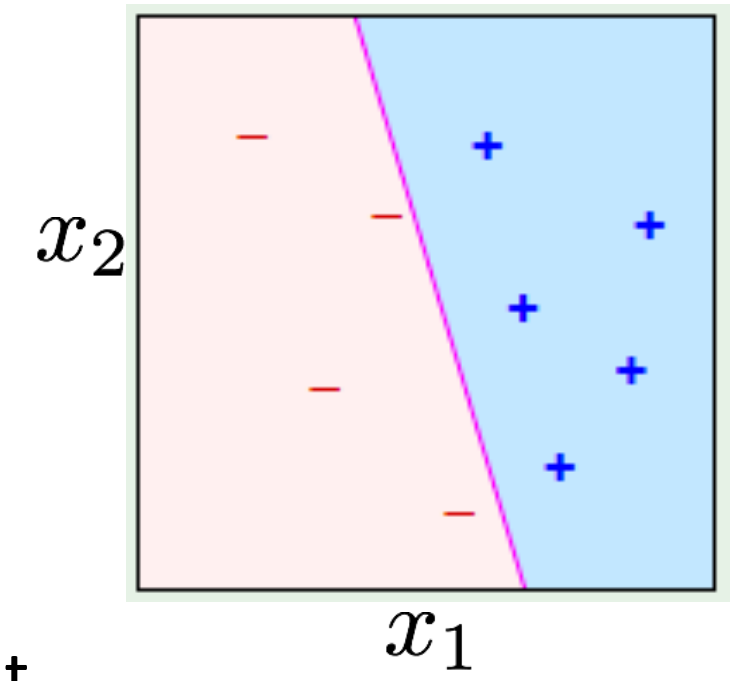
- **Given** Training Samples: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$

$$\mathbf{x}_i \in \mathbb{R}^d, \qquad y_i \in \{\pm 1\}$$

- **Determine** a *classification rule*

$$y = sign\left(\sum_{j=0}^{d} w_j x_j\right)$$

  to **minimize** classification error



- Perceptron Learning Algorithm
- Logistic Regression and Gradient Descent

Linear Regression

- **Given** Training Samples: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$

  $\mathbf{x}_i \in \mathbb{R}^d, \qquad y_i \in \mathbb{R}$

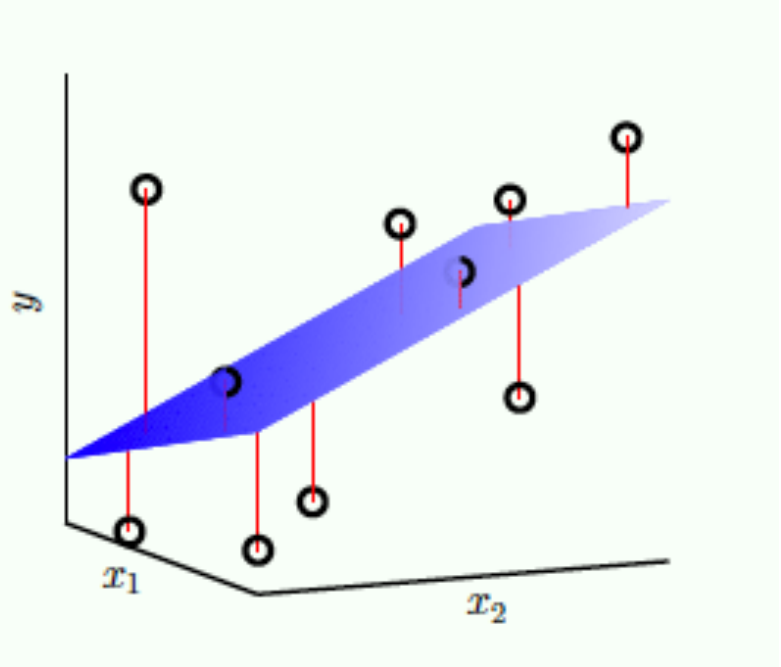
- **Determine** a *regression rule:* $\hat{y} = \sum_{i=0}^{d} w_i x_i$

  $\mathbf{x} = (x_1, \ldots, x_d)$

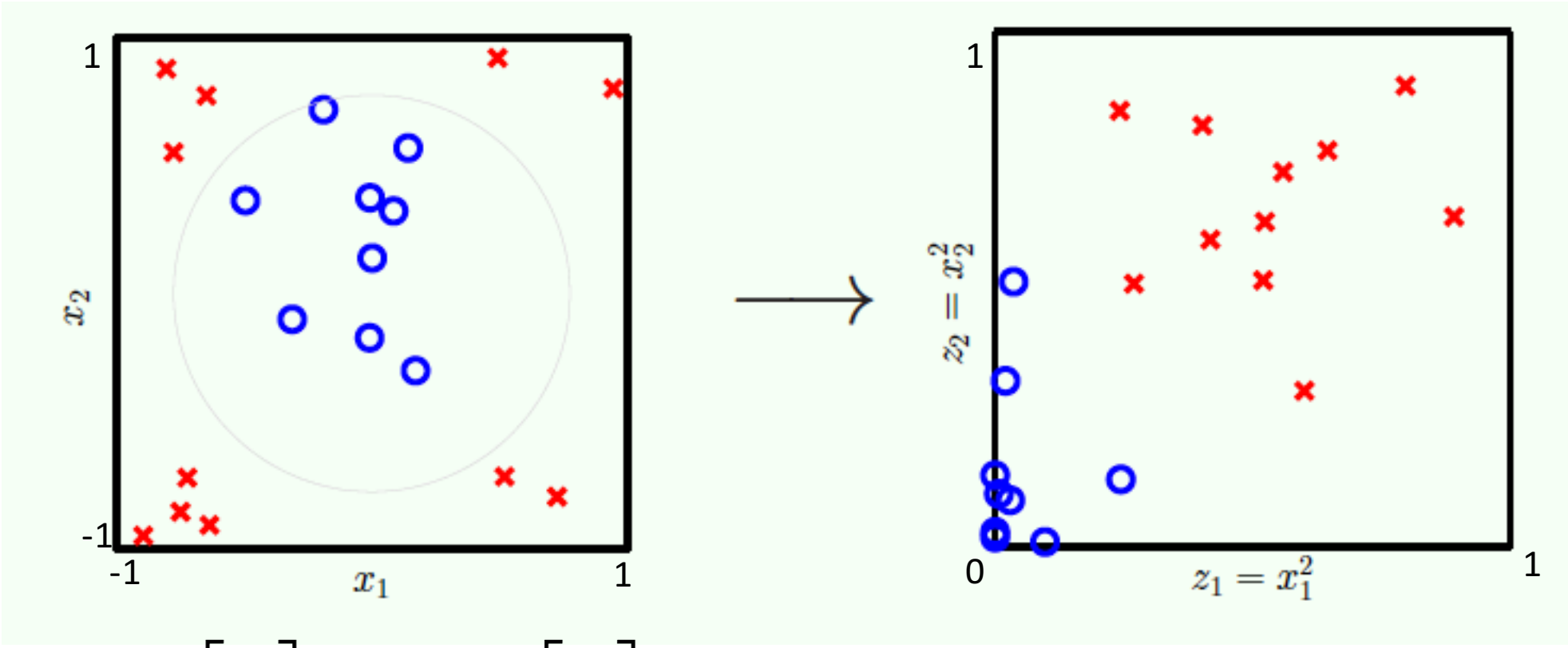
- **Minimize** the prediction error**:**

$$\sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

- Least Squares and its variations



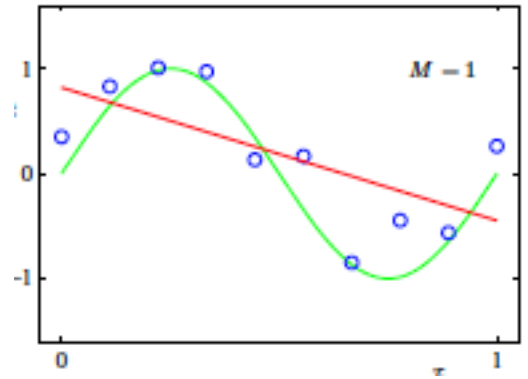
Non-Linear Transformations of Data



$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \Longrightarrow \mathbf{z} = \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$

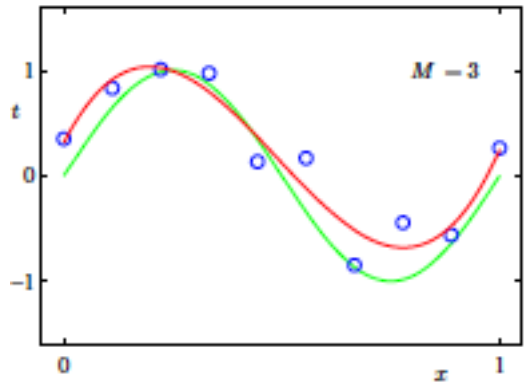
$$\hat{y} = \text{sign}\left(\mathbf{w}^T \mathbf{z}\right)$$

Non-Linear Transformations of Data



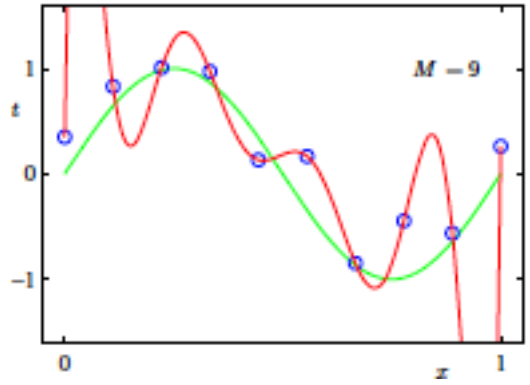
$$\mathbf{z} = \begin{bmatrix} 1 & x \end{bmatrix}^T$$

First Order Polynomial



$$\mathbf{z} = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix}^T$$

Third Order Polynomial



$$\mathbf{z} = \begin{bmatrix} 1 & x & x^2 & \dots & x^9 \end{bmatrix}^T$$

9th Order Polynomial

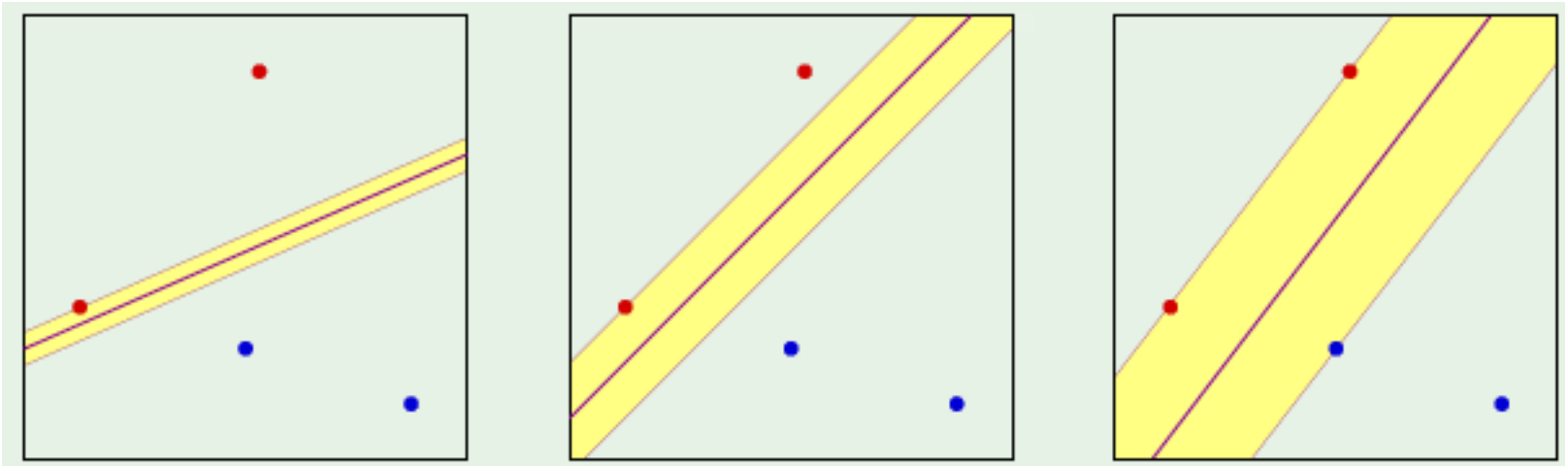
Under-fitting

$$\hat{y} = \mathbf{w}^T \mathbf{z}$$

Overfitting

- Higher Order Non-Linearity:
  - Better Fit to Training Data
  - Less Robust to Noise (Overfitting)

Support Vector Machines

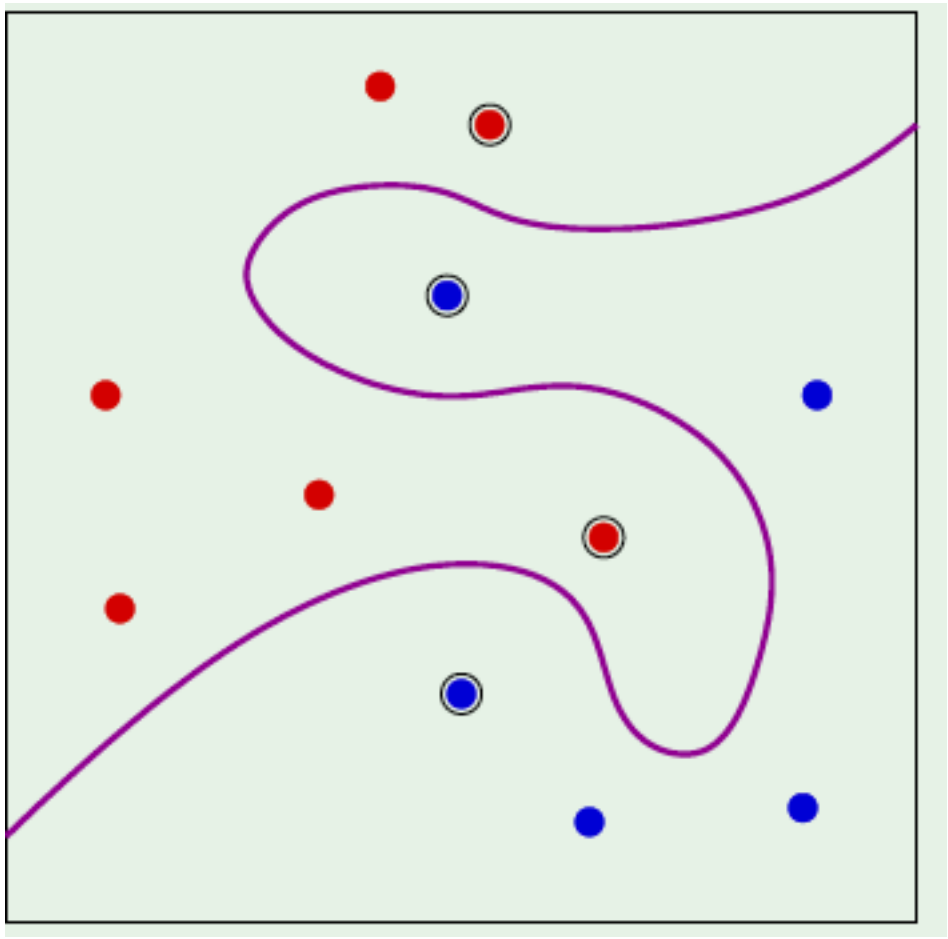


Thin Margin Classifier

Fat Margin Classifier

- Quadratic Programming: Maximizing Margin in Linear Classification
- Lagrange Duality Framework for identifying Support Vectors

SVMs with Non-Linear Transforms



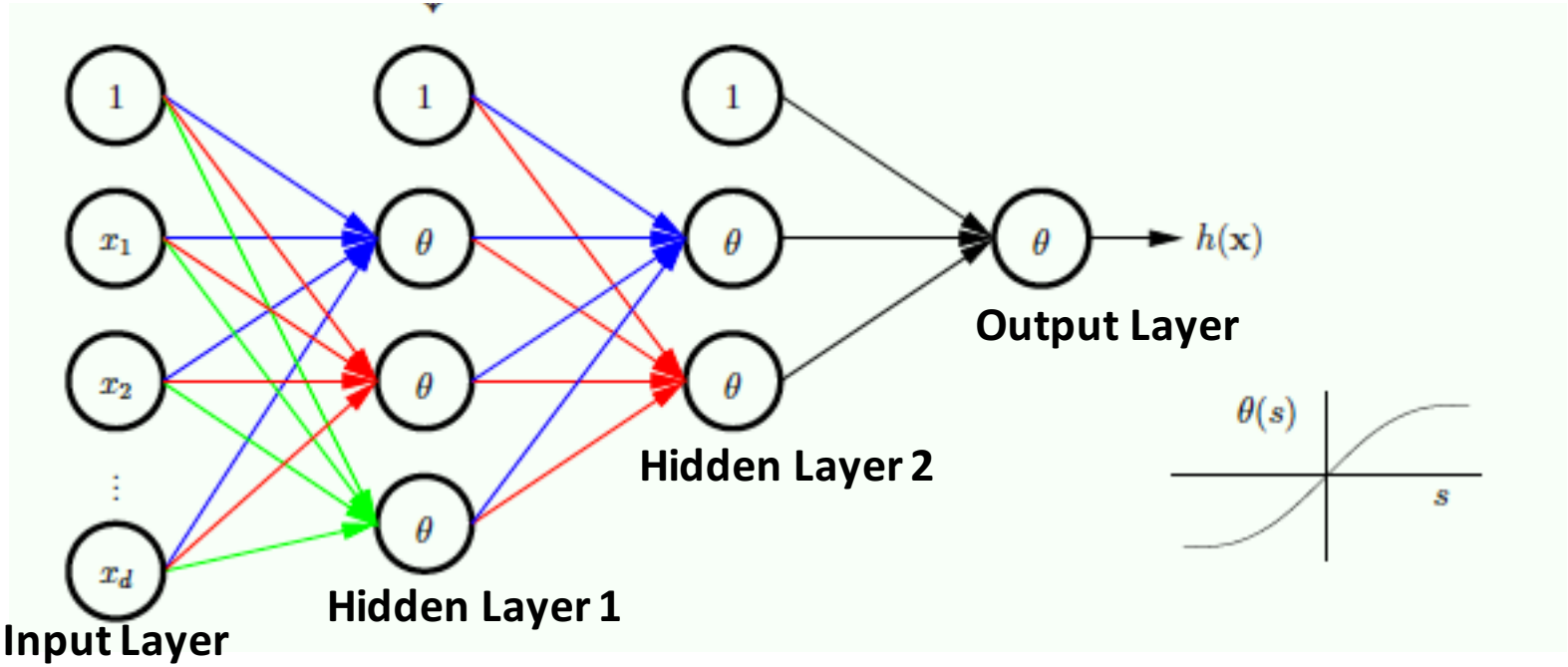
- Radial Basis Functions

$$\Phi(\mathbf{x}) = \left[\exp\left\{-\gamma(||\mathbf{x} - \mathbf{x_n}||^2)\right\}\right]_{1 \leq n \leq N}$$

- Kernel Trick for Efficient Computation

- Bounds on Generalization Error

Neural Networks - Architecture



$$h_i^{(1)} = \theta\left(\sum_{j=0}^{d} w_{ji}^{(1)} \cdot x_j\right)$$

Output `i' (Hidden Layer 1)

Weights:Layer 1

Inputs

$$h_i^{(2)} = \theta\left(\sum_{j=0}^{d} w_{ji}^{(2)} \cdot h_j^{(1)}\right)$$

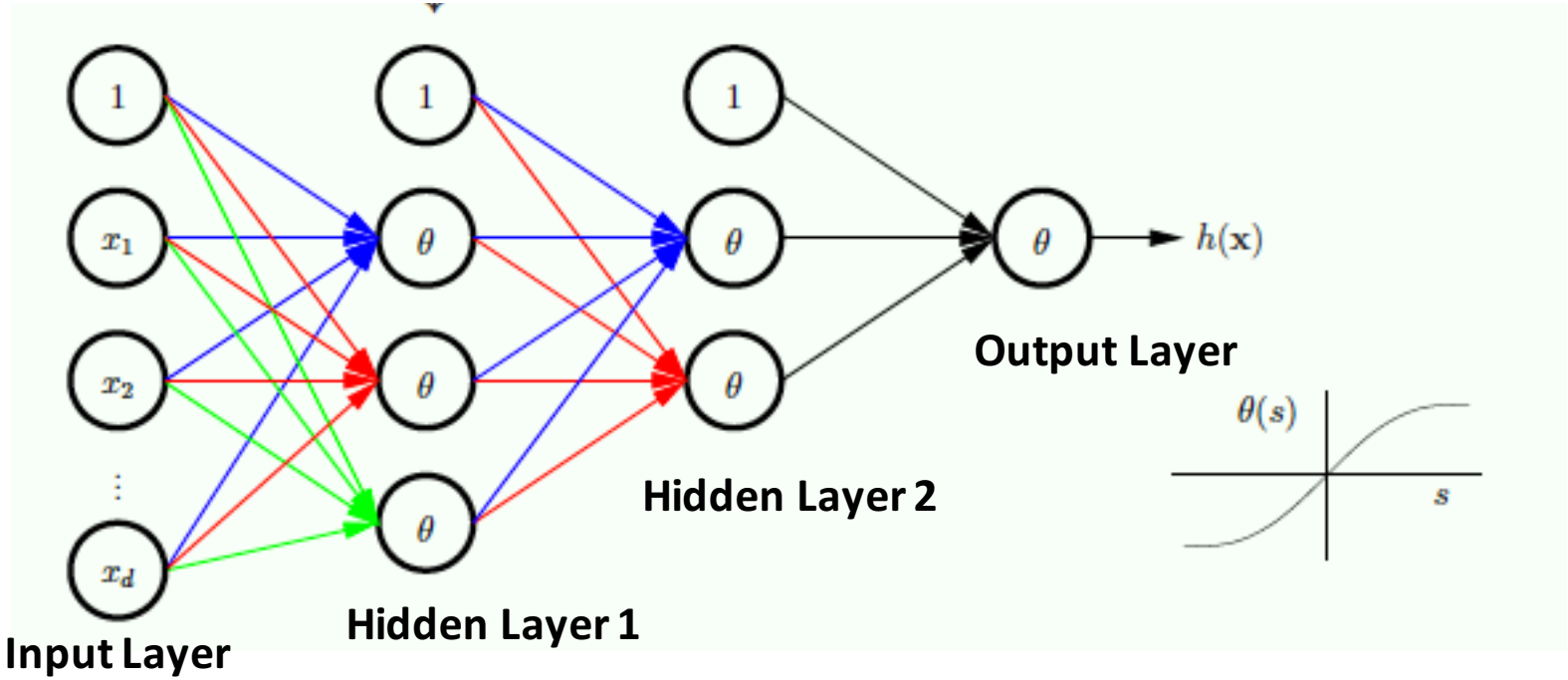
Output `i' (Hidden Layer 2)

Weights: Layer 2

$$y = sign\left(\sum_{j=0}^{d} w_{ji}^{(3)} \cdot h_j^{(2)}\right)$$

Output

Neural Networks - Architecture



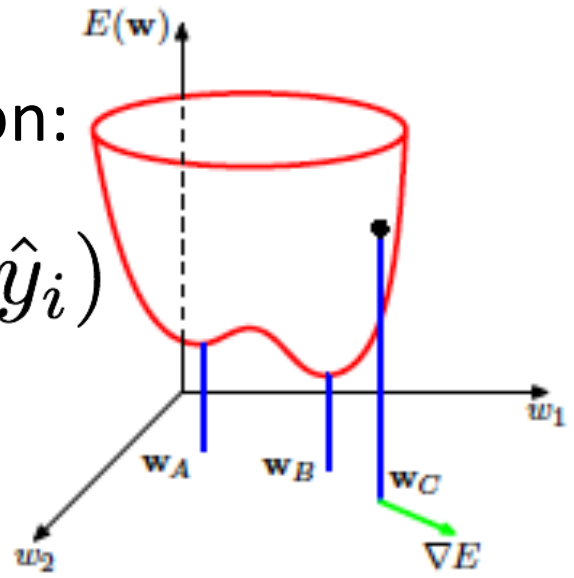
$$y = \text{sign}\left[W_3^T \left\{\theta\left(W_2^T \left\{\theta(W_1^T \cdot \mathbf{x})\right\}\right\}\right)\right]$$

Universal Approximation Theorem: A Neural Network with **one hidden layer** can approximate any "reasonable" non-linear function with sufficiently many hidden units.

Neural Network - Learning

- **Given** Training Samples: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$
- **Fix:**
  - Number of Hidden Units per layer
  - Nonlinearity: θ
- Learn: Weights: $w_{i,j}^{k}$
- **Minimize** Loss Function:

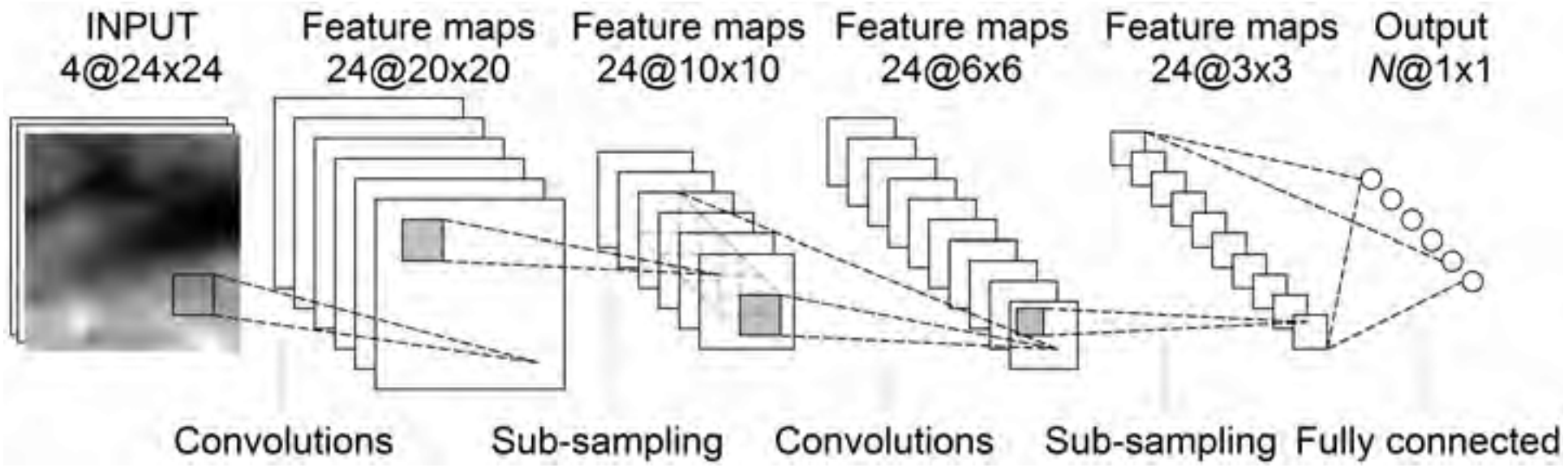
$$E(\mathbf{w}) = \sum_{i=1}^{n} \ell_{\mathbf{w}}(y_i, \hat{y}_i)$$



Training Procedure

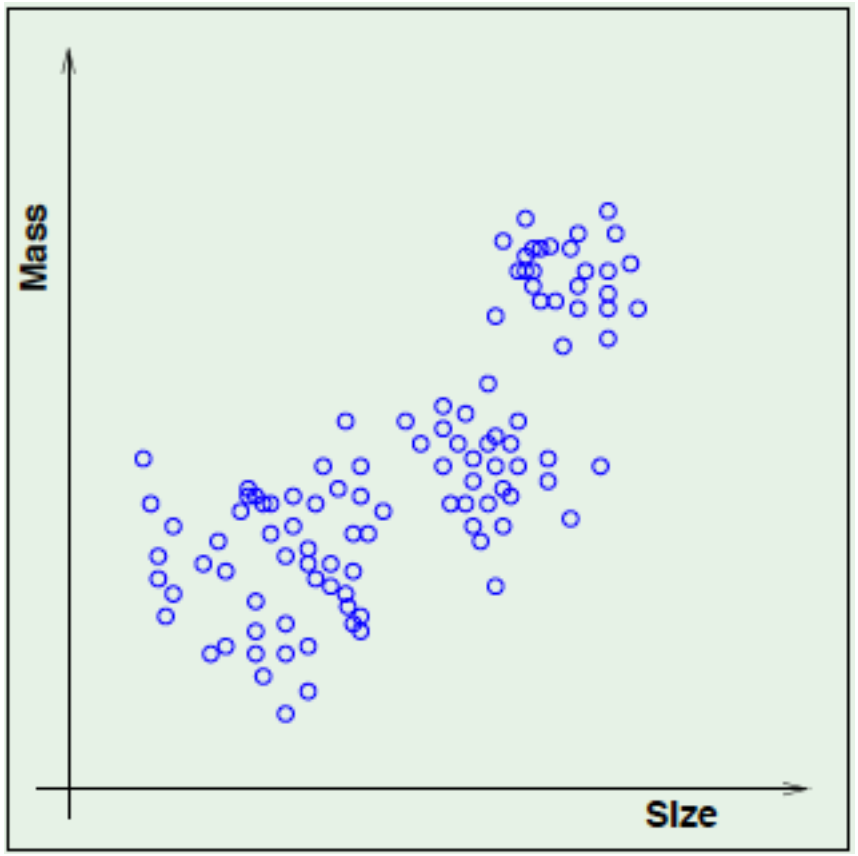
- Stochastic Gradient Descent
- Backpropagation Algorithm
- Early Stopping Rule
- Dropout and Regularization
- *Convolutional Neural Networks.*

Convolutional Neural Networks



- Image statistics are translation invariant (objects and viewpoint translates)
- Expect low-level features to be local (e.g. edge detector)
- Expect high-level features learned to be coarser

Unsupervised Learning



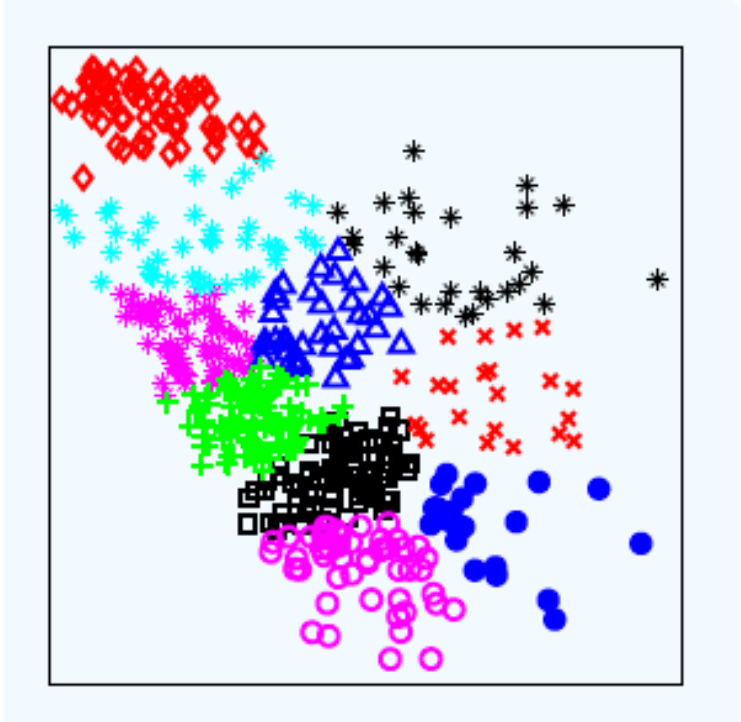
Training Data:

$$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$$

No Labels!

Still Need to do Learning!

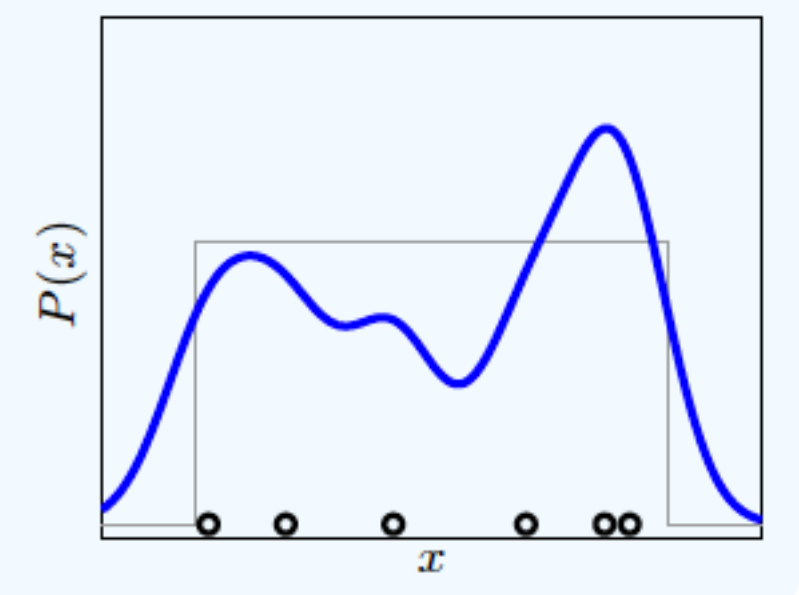
k-Means Clustering



- Cluster Centers: $\mu_1, \mu_2, \ldots, \mu_k$
- Partitions: $S_1, S_2, \ldots, S_k$
- Minimize : $\sum_{j=1}^{N} \|\mathbf{x}_j - \mu(\mathbf{x}_j)\|^2$

Lloyd's Algorithm

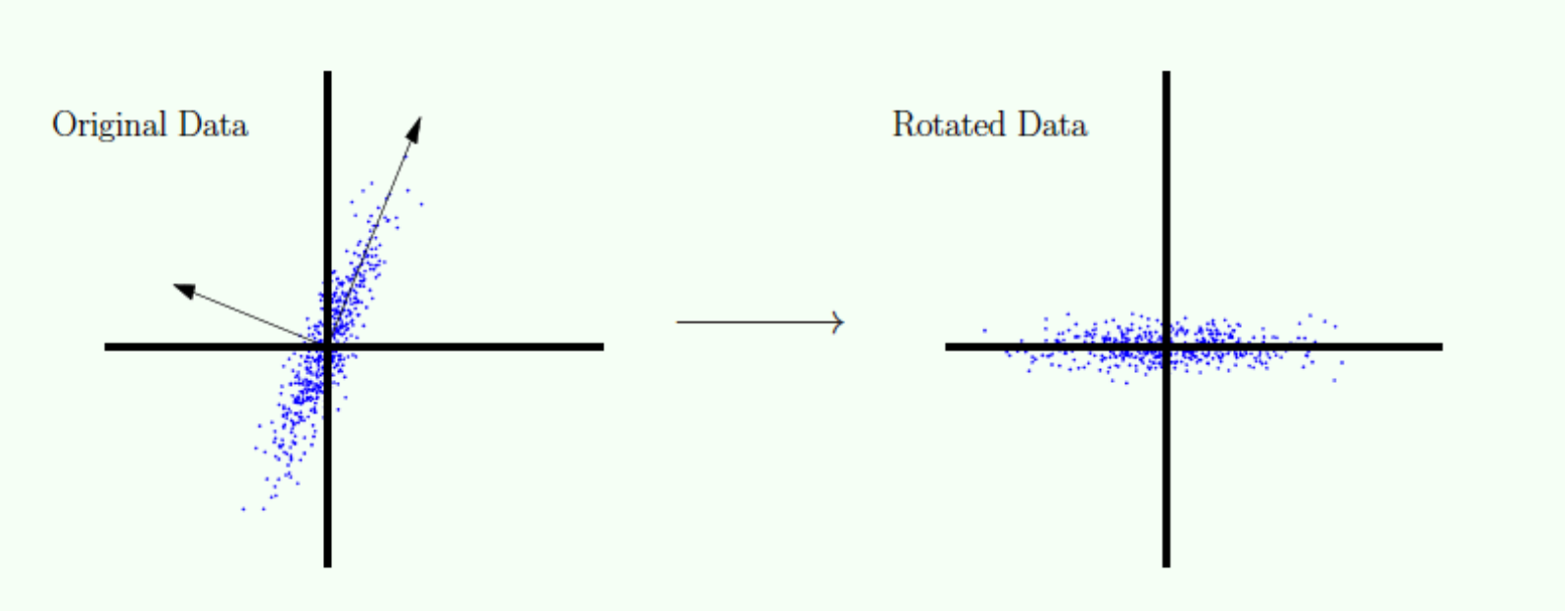
Density Estimation - Gaussian Mixture Models



$$f(\mathbf{x}) = \sum_{j=1}^{k} w_j \cdot \mathcal{N}(\mathbf{x}; \mu_j, \Sigma_j)$$

- Estimating Probability Density Function
- Expectation Maximization Algorithm (EM) for GMMs
- General EM (time permitting) for Maximum Likelihood solution

Principal Component Analysis



- Dimensionality Reduction
- Identify directions with large variance
- Singular Value Decomposition
- Non-Linear Extension: AutoEncoders

Theory

Training and Testing Errors

- Training Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$
- Ground Truth (Not Known): $y = f(\mathbf{x})$
- Output of Learning: $\hat{y} = h(\mathbf{x})$
- Training (In Sample) Error

$$E_{\text{in}}(h) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left[y_i \neq h(\mathbf{x}_i)\right]$$

- Testing (Out of Sample) Error

$$E_{\text{out}}(h) = E_{\mathbf{x},y}\left[\mathbb{I}\left[y \neq h(\mathbf{x})\right]\right]$$

Probably Approximately Correct (PAC) Learning

- Fixed Hypothesis Class

$$\mathcal{H} = \{h_1, h_2, \ldots, h_M\}, \qquad h \in \mathcal{H}$$

- The following bound holds with probability: $1 - \delta$
- $$E_{\text{out}}(h) \leq E_{\text{in}}(h) + \sqrt{\frac{1}{2N} \log \frac{2M}{\delta}}$$

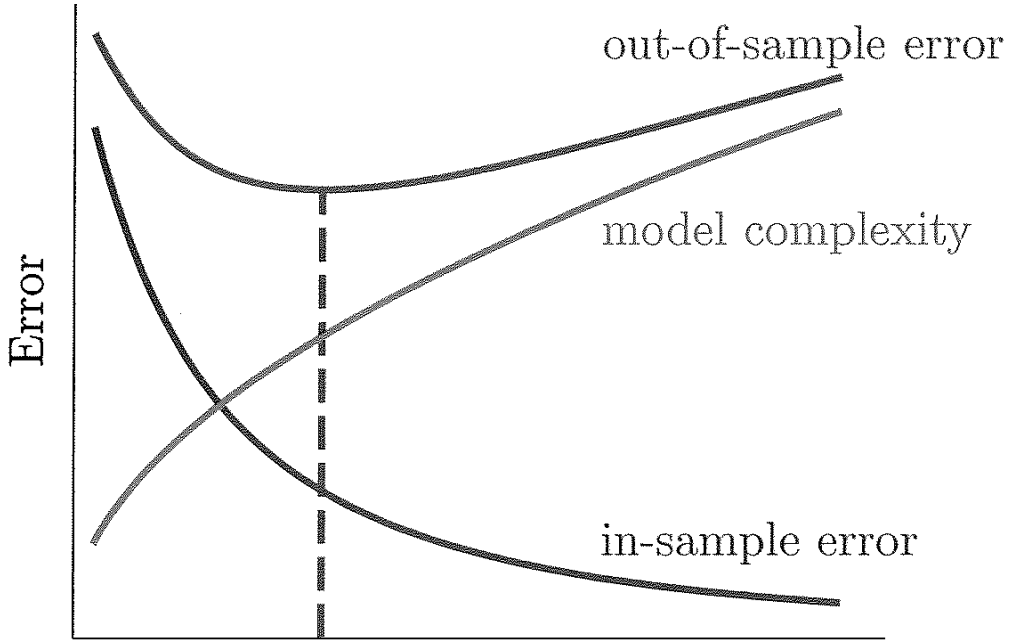
Complexity of Hypothesis Class

Number of Training Examples

$$|\mathcal{H}| = \infty \Longrightarrow M \leftarrow \text{VC Dimension}(\mathcal{H})$$

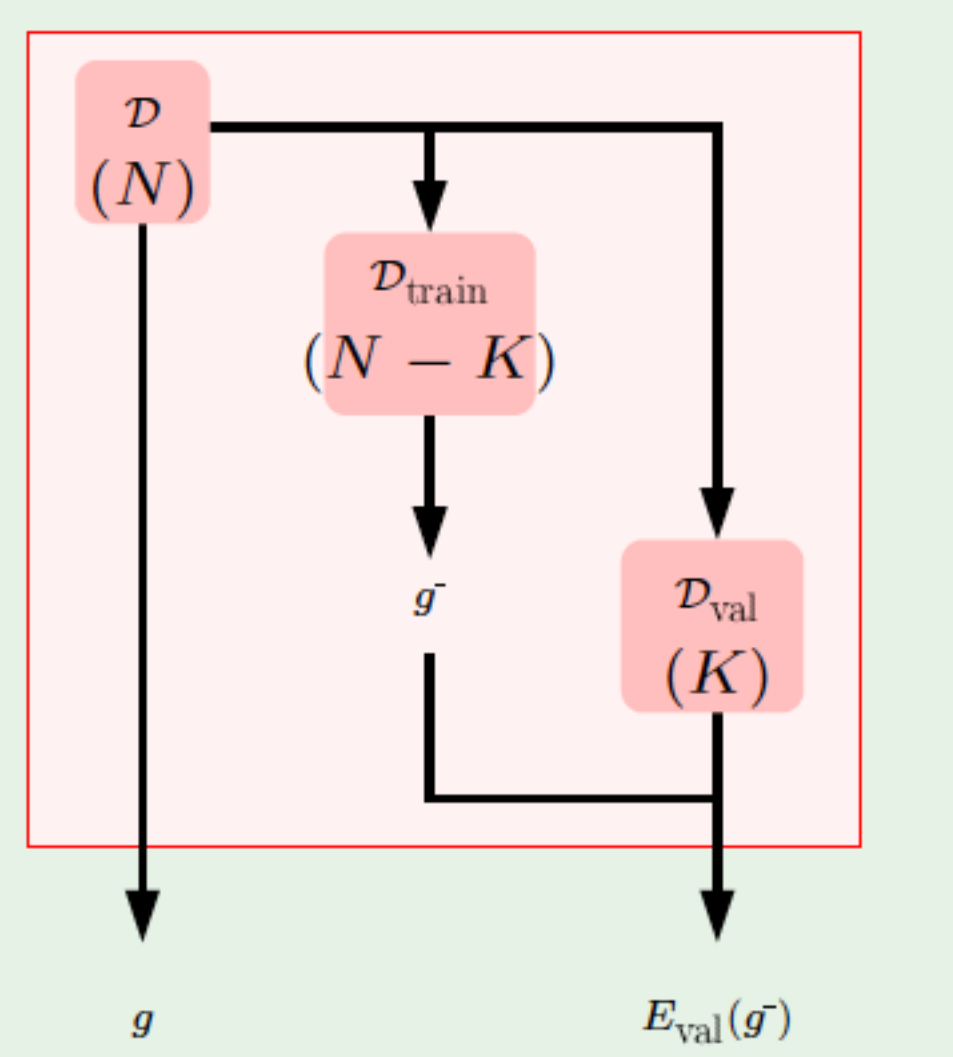
Vapnik Chervonenkis (VC) dimension

- VC Dimension provides a natural measure for complexity of class
- Linear Models: VC Dimension = dimension + 1
- Neural Networks (roughly) = # of Weight Parmeters



Techniques

Validation

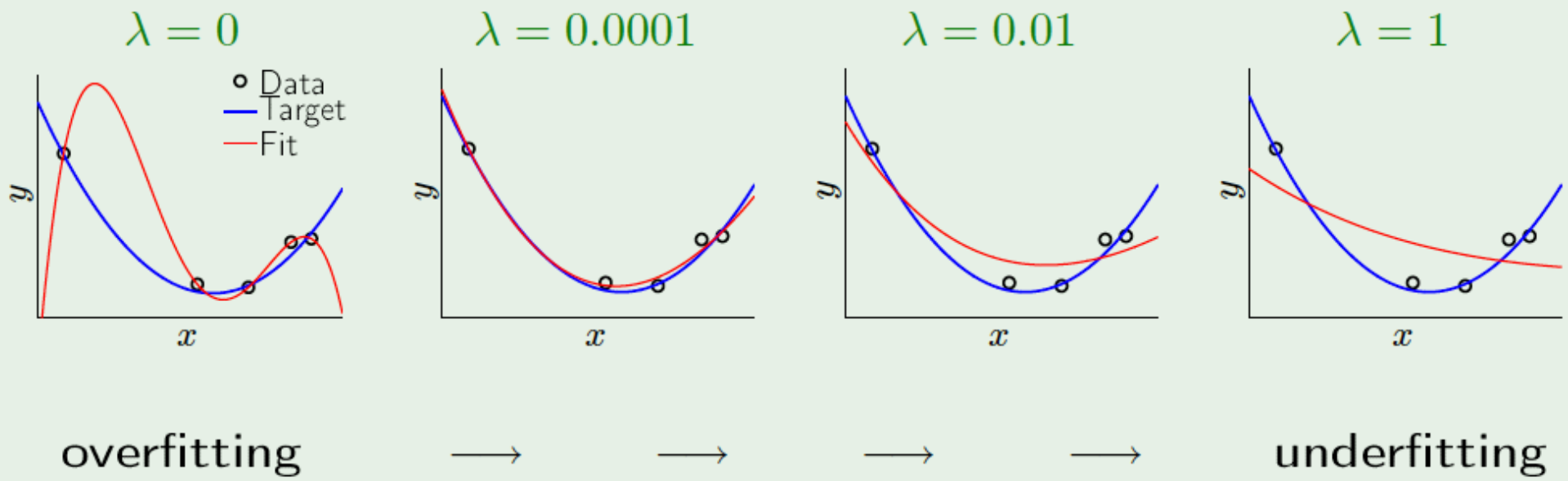


- Divide Training Set into 2 Parts
  - Learning Set
  - Validation Set
- Learning Set: Train Model
- Validation Set: Estimate Test Error
- Applications of Validation
  - Model Selection
  - Selection of Hyperparameters (e.g., regularization coefficient)
  - Number of Training Steps (Early Stopping)
- Cross Validation

Regularization

$y = \mathbf{w}^T \cdot \mathbf{z}, \quad \mathbf{z} = \begin{bmatrix} 1 & x & \dots, x^M \end{bmatrix}^T$

Minimizing $E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N} \mathbf{w}^\intercal \mathbf{w}$ for different λ's:



- Weight Decay Method
- Neural Networks: Drop Out Method

Logistics

Course Staff

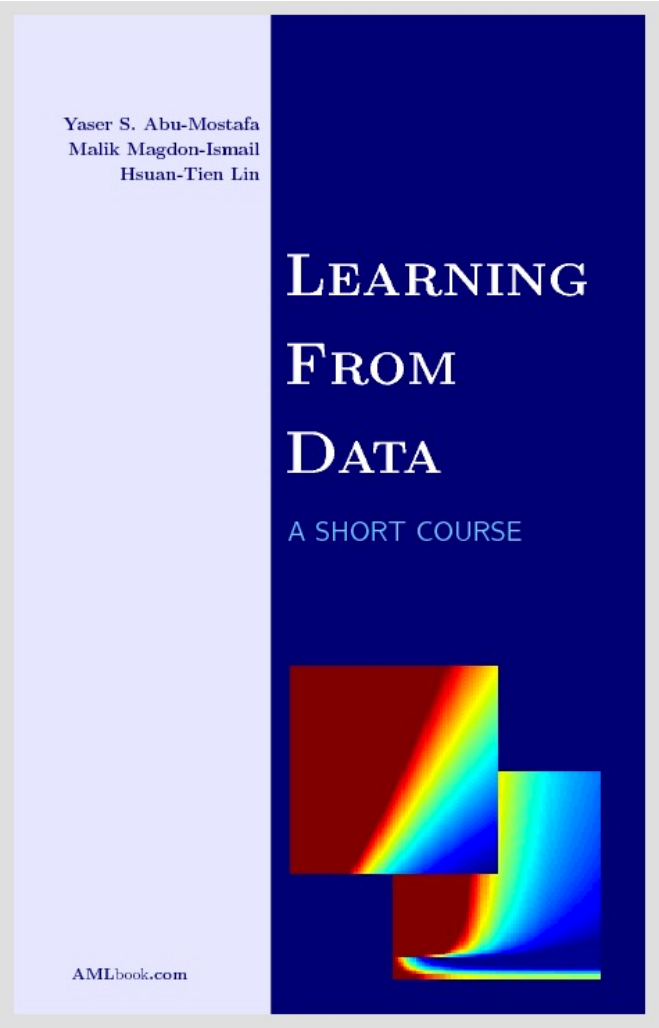
Instructor:

- Ashish Khisti

- Lectures
  - Mondays 5-6 (MC252)
  - Wednesdays 4-6 (MC252)

- Tutorials
  - Tutorial 1: Thursdays 1-3pm (GB304)
  - Tutorial 2: Thursdays 4-6pm (BA1230)

Course Textbook



- **Required** Textbook:
- "Learning from Data" Available at U of T bookstore
  - http://www.amlbook.com
  - Supplementary Chapters and Appendices
  - Slides from other courses (Caltech, RPI etc)
  - Video Lectures (Prof. Abu-Mostafa)
  - Discussion Forum
- Deep Learning by Goodfellow et.al (Free Online book)
- Recommended Textbook
  - Machine Learning and Pattern Recognition by Christopher Bishop (Text for CSC411)

Tentative Schedule

1	September 9-13	**Intro/Linear Classification**	Tut 1			
2	September 16-20	**Linear Regression, Regularization, Gradient Descent**	Tut 2	**Assnt 1 Posted**		
3	September 23-27	**Logistic Regression**	Tut 3			
4	September 30-Oct 4	**Gradient Descent**	Tut 4	**Assnt 1 due**		
5	Oct 7-11	**Multilayer Perceptron, Backpropagation**	Tut 5	**Assnt 2 Posted**		
6	Oct 14-18	**Thanksgiving and Mid-term Week**	Tut 6	**Mid-Term Exam**	**Wed Oct 16th**	
7	Oct 21-25	Deep Neural Networks	Tut 7			
8	Oct 28 – Nov 1	**Unsupervised Learning: Clustering and Density Estimation**	Tut 8	**Assnt 2 Due**		
9	Nov 4-8	**EM Algorithm, PCA**	Tut 9	**Assnt 3 Posted**		March 17th - Drop Date
10	Nov 11-15	**PAC Learning - 1 (Finite Hypothesis)**	Tut 10			
11	Nov 18-22	**PAC Learning - 2 (VC Dim, Bias-Variance)**	Tut 11	**Assnt 3 Due**		
12	Nov 25-29	**Validation, Generalization, Bootstrapping**	Tut 12			
13	Dec 2-6	**Support Vector Machine**		Last Day of Class – Dec 4th		

Pre-requisites

- Undergraduate Course in Probability (Official Pre-Req.)
  - Bayes Theorem, Union Bound, Gaussian Distributions
- Linear Algebra (Strongly Recommended)
  - Vector Space Concepts, Matrices
- Programming
  - We will use Python and Tensor Flow Package for our assignments

Grade Composition

- Mid Term: Wed Oct 16th (30%)
- Programming Assignment (3 x 10% = 30%)
- Final Exam: 40% in Exam Week

- Location of Mid-Term: Exam Center
  - MC252 & RW117

Tensor flow Assignments

- **Python based** ML Library released by Google in 2015
- Automatic Training for Neural Networks
- GPU Support (Not Required for Assignments in this courses)

- Installation through Anaconda Environment is Recommended (See Installation Guide on Course Webpage)

- Tons of Resources!
  - Tensorflow.Org Tutorials
  - CS231n Stanford Tutorial (http://cs231n.stanford.edu/)
  - See Course Webpage for a simple tutorial (Updated, Use Chrome Browser)

Tensor flow Example (https://www.tensorflow.org)

```
import tensorflow as tf
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)
```

Initialize Computational Graph

```
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), reduction_indices=[1]))

train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

Loss Function and Optimizer

```
sess = tf.InteractiveSession()

tf.global_variables_initializer().run()

for _ in range(1000):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})I
```

Training Routing

ECE421 Course Description

An Introduction to the basic theory, the fundamental algorithms, and the computational toolboxes of machine learning. The focus is on a balanced treatment of the practical and theoretical approaches, along with hands on experience with relevant software packages. Supervised learning methods covered in the course will include: the study of linear models for classification and regression, neural networks and support vector machines. Unsupervised learning methods covered in the course will include: principal component analysis, k-means clustering, and Gaussian mixture models. Theoretical topics will include: bounds on the generalization error, bias-variance tradeoffs and the Vapnik-Chervonenkis (VC) dimension. Techniques to control overfitting, including regularization and validation, will be covered

ECE421: Intro. To Machine Learning

Week #1:

What is Machine learning.

Develop computational systems that adaptively improve performance with experience from accumulated data.

Example 1: Recommendation Systems.

Problem: Design a system that recommends new movies to users based on their preferences

Approach 1: Human Experts.

An expert views each movie & collects various attributes
- Actors?
- block buster?
- Action/Comedy?

Expert Interviews each user.
- Like comedy?
- fav. actors?
- Prefer block busters.

$\Rightarrow$ find suitable recommendations.

Cons: tedious, bias, preferences change etc.

Approach 2: Machine Learning

- data driven
- uses past user ratings.

1) User Preference Vector

$$\underline{x}_i = (x_{i1}, x_{i2}, \ldots x_{id}) \in \mathbb{R}^d$$

for each $i \in \mathcal{U}$

2) Movie attribute vector

$$\underline{y}_j = (y_{j1}, y_{j2}, \ldots y_{jd}) \in \mathbb{R}^d$$

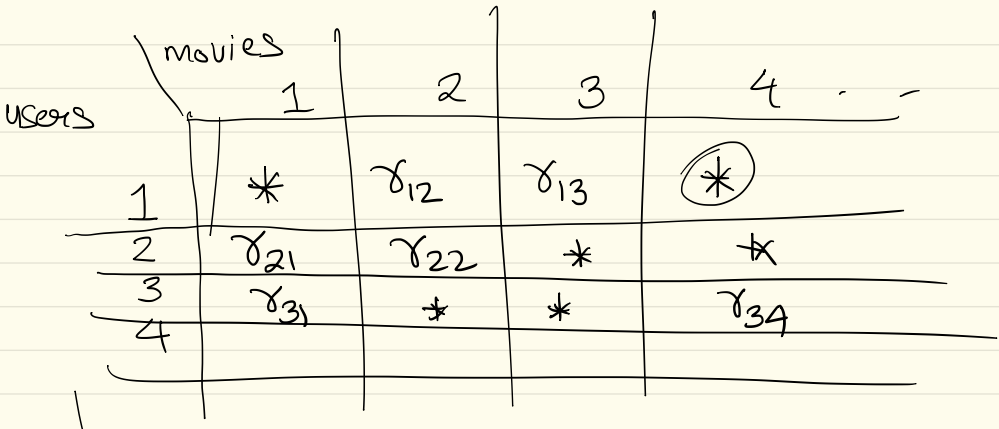
for each $j \in \mathcal{M}$

i = user index j = movie index

$\mathcal{U}$ = set of users $\mathcal{M}$ = set of Movies.

3) Data Set

r_{ij} = rating by user i for movie j



S = Set of (user, movie) pairs for which ratings are available.

$= \{(i,j) : r_{ij}$ is available in our data set $\}$

4) Proposed Model

Rating Estimate $\hat{r}_{ij} = \underline{x}_i^T \underline{y}_j$

$\underline{x}_i = (x_{i1}, x_{i2}, \ldots x_{id})$

$\underline{y}_j = (y_{j1}, y_{j2}, \ldots y_{jd})$

$\hat{r}_{ij} = \sum_{l=1}^{d} x_{il}\, y_{jl}$

How to compute $\underline{x}_i, \underline{y}_j$?

✓ Model Parameters

$$\Omega = \{ \{\underline{x}_i\}_{i \in U}, \{\underline{y}_j\}_{j \in M} \}$$

✓ Loss Function

$$E_{in}(\Omega) = \sum_{(i,j) \in S} (r_{ij} - \underbrace{\underline{x}_i^T \underline{y}_j}_{= \hat{r}_{ij}})^2$$

Training Phase

✓ $$\Omega^* = \arg\min_{\Omega} E_{in}(\Omega)$$

find model parameters that minimize loss function.

- Gradient descent
- EM Algorithm

Output $\{\underline{x}_i\}_{i \in U}$ $\{\underline{y}_j\}_{j \in M}$

✓ for all users & Movies

Prediction Phase: $\hat{r}_{ij} = \underline{x}_i^T \underline{y}_j$

Cons: Models are not easily interpretable.

new movies/ users cannot be recommended

Sampling bias: users who have watched too many movies will influence others.

Example 2: Credit Approval

Bank needs to decide whether or not to approve the credit of user

user attribute vector

$\underline{x} = (\text{age}, \text{gender}, \text{salary}, \ldots) \in \mathbb{R}^d$

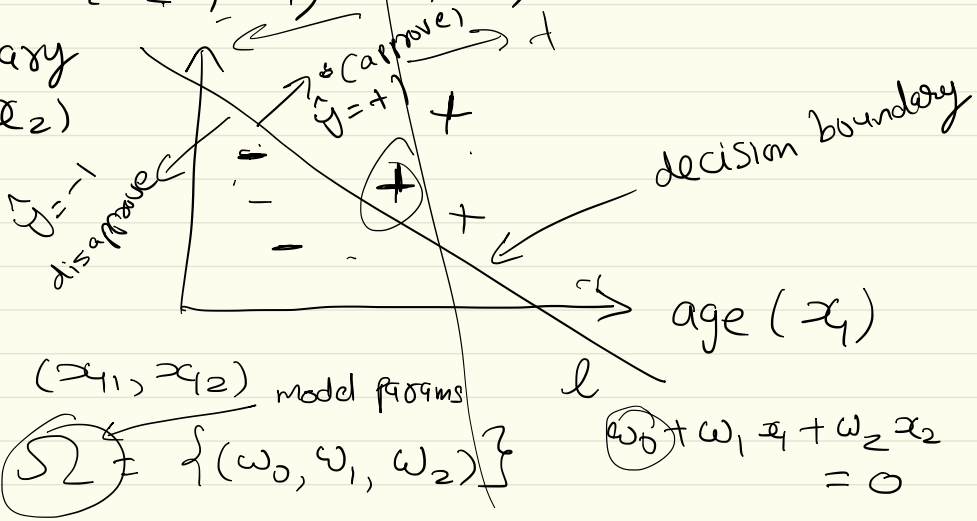
output $y = \begin{cases} +1 & \text{approve} \\ -1 & \text{dis-approve} \end{cases}$

Proposed Model: Linear classification

$d = 2 \quad \underline{x} = (\text{age}, \text{salary})$

Historical Data:

$\mathcal{D} = \{(\underline{x}_1, y_1), (\underline{x}_2, y_2), \ldots (\underline{x}_N, y_N)\}$



$\underline{x}_1 = (x_{11}, x_{12})$ model params

$\Omega = \{(\omega_0, \omega_1, \omega_2)\}$

$\omega_0 + \omega_1 x_1 + \omega_2 x_2 = 0$

(Binary) Linear Classification

weight vector: $\underline{w} \in \mathbb{R}^d$ $= (w_1, w_2, \ldots w_d)$

Constant $b \in \mathbb{R}$

Given Input $\underline{x} = (x_1, x_2, \ldots x_d) \in \mathbb{R}^d$

If $\sum_{i=1}^{d} w_i x_i > b \Rightarrow \hat{y} = +1$

else $\hat{y}_i = -1$

Model Parameters: $\Omega = \{\underline{w}, b\}$

Training Set: $\{(\underline{x}_1, y_1), \ldots (\underline{x}_N, y_N)\}$

$\underline{x}_i = (x_{i1}, x_{i2}, \ldots x_{id}) \in \mathbb{R}^d$

$y_i \in \{-1, +1\}$

Training Phase: Compute Ω

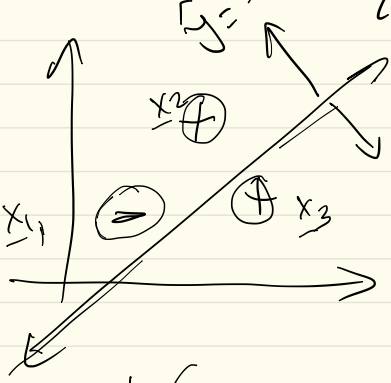
Loss function:

Average Number of Errors

$$E_{in}(\mathcal{L}) = \frac{1}{N} \sum_{i=1}^{N} 1\{y_i \neq \hat{y}_i\}$$

$1(\cdot) \equiv$ indicator function

$$1(t) = \begin{cases} 1 & t \text{ is true} \\ 0 & t \text{ is false} \end{cases}$$



$E_{in} =$

$y_1 = -1 \quad \hat{y}_1 = -1$

$y_2 = +1 \quad \hat{y}_2 = -1$

$y_3 = +1 \quad \hat{y}_3 = +1$

$$E_{in} = \frac{1}{3}(0+1+0) = \frac{1}{3}$$

Perceptron Learning Algorithm

Linear Classification

Weight vector $\underline{w} = (w_1, w_2, \ldots w_d)$

Constant $b \in \mathbb{R}$

Change in Notation

Expanded dimension

$\underline{x} = (x_0 = 1, x_1, x_2, \ldots x_d) \in \mathbb{R}^{d+1}$

$\underline{w} = (w_0 = -b, w_1, w_2, \ldots w_d) \in \mathbb{R}^{d+1}$

decision Rule $w_1 x_1 + w_2 x_2 + \ldots w_d x_d$

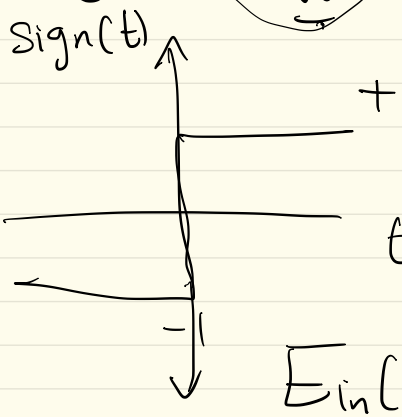
$\gtrless b$ (predict $\hat{y}=+1$ above, $\hat{y}=-1$ below)

$-b + w_1 x_1 + w_2 x_2 + \ldots w_d x_d \gtrless 0$ (predict $\hat{y}=+1$ above, $\hat{y}=-1$ below)

$\underline{w}^T \underline{x}$

Decision Rule: $\underline{w}^T \underline{x} \gtrless 0$ ($\hat{y}=+1$ / $\hat{y}=-1$)

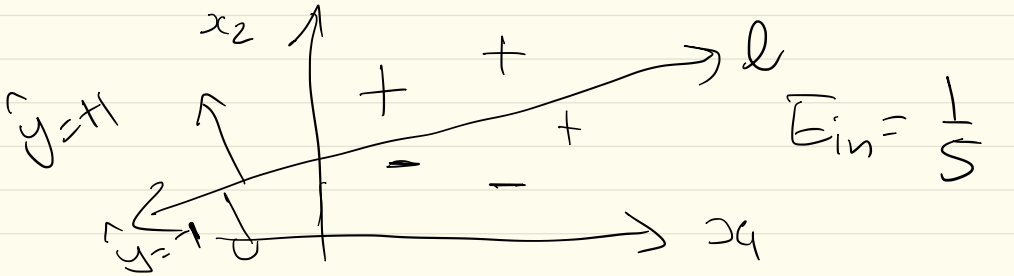
$$\hat{y} = h_{\underline{w}}(\underline{x}) = \text{sign}(\underline{w}^T \underline{x})$$



Goal:
select $\underline{w} \in \mathbb{R}^{d+1}$ that minimizes

$$E_{in}(\underline{w}) = \frac{1}{N} \sum_{i=1}^{N} 1\{y_i \neq h_{\underline{w}}(\underline{x}_i)\}$$

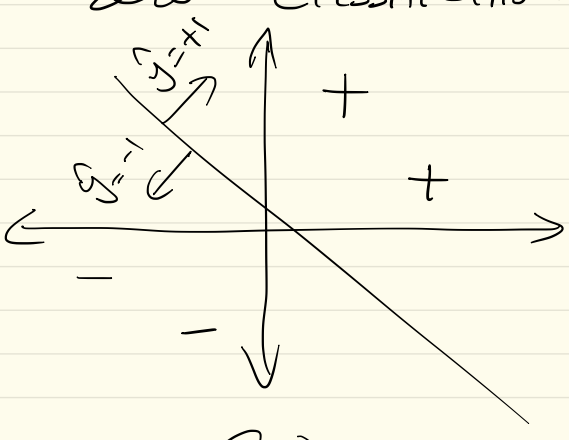
Training set:
$\{(\underline{x}_1, y_1), \ldots (\underline{x}_N, y_N)\}$



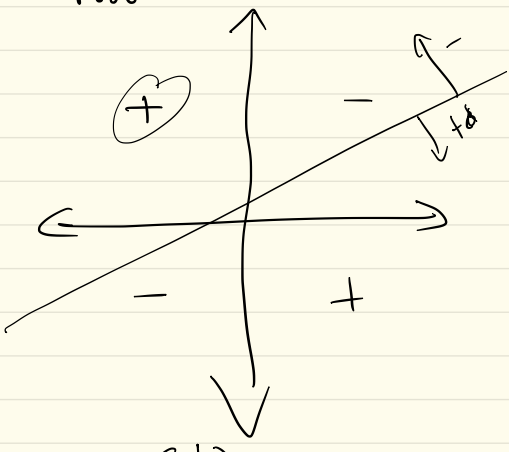
$E_{in} = \frac{1}{5}$

Separable Data sets:

A data set is separable if there is a line that achieves zero classification error.



(a)
Separable



(b)
Not separable.

If the data set is linearly separable then Perceptron learning Algorithm find $\underline{w} \in \mathbb{R}^{d+1}$ with $E_{in}(\underline{w}) = 0$

If the data set is Not Separable

minimizing $E_{in}(\underline{w})$ is NP-Hard in general. However Perceptron algorithm can be extended to non-separable Data sets to attain good Heuristics.

Knowing whether a data set is seperable or not is not NP-Hard.

Perceptron Learning Algorithm

Assume: $\mathcal{D} = \{(\underline{x}_1, y_1), \ldots (\underline{x}_N, y_N)\}$ is linearly separable

$\underline{x}_i \in \mathbb{R}^{d+1}$ $y_i \in \{-1, +1\}$

Output: $\underline{w} \in \mathbb{R}^{d+1}$ that achieves $E_{in}(\underline{w}) = 0$

Step 1: Initialize $\underline{w}$ in an arbitrary manner

Step 2: Compute $E_{in}(\underline{w})$

$$= \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\{y_i \neq h_w(\underline{x}_i)\}$$

If $E_{in}(\underline{w}) = 0$ Stop.
& output $\underline{w}$.

Step 3 let $(\underline{x}_n, y_n)$ be any mis-classified point
i.e $y_n \neq h_w(\underline{x}_n)$

$$\underline{w} \leftarrow \underline{w} + y_n \underline{x}_n$$

$\hookrightarrow \{-1, +1\}$

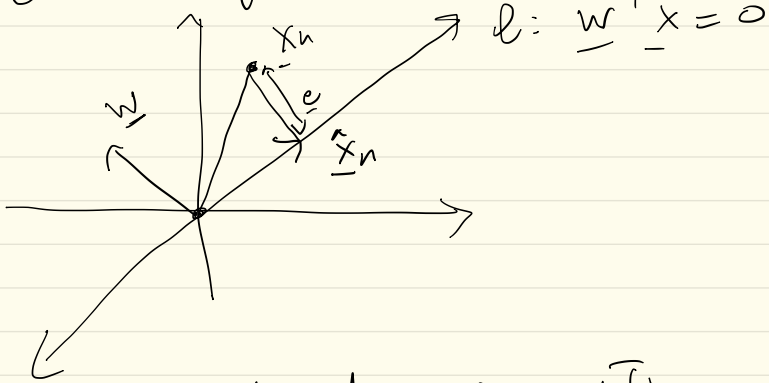
Go to Step 2

Next lecture

1) Geometric Intuition for Perceptron Alg.
2) Convergence
3) Extension to non separable data sets

Consider a linear classifier

$$\hat{y} = \text{sign}(\underline{w}^T \underline{x})$$



the decision boundary $\ell: \underline{w}^T \underline{x} = 0$ is shown above.

Note that $\underline{w}$ points perpendicular to ℓ.

let $\underline{x}_n$ be any point, we want to show $\text{dist}(\underline{x}_n, \ell) = \dfrac{|\underline{w}^T \underline{x}_n|}{\|\underline{w}\|}$

Let $\underline{u}$ be a unit vector along $\underline{w}$

let us decompose

$$\underline{x}_n = \hat{\underline{x}}_n + \underline{e},$$

where $\hat{\underline{x}}_n$ is a vector along l (i.e projection of $\underline{x}_n$ along l) and $\underline{e}$ is a vector perpendicular to l (i.e. parallel to $\underline{u}$)

We can express $\underline{e} = \|\underline{e}\| . \underline{u}$

also note that

$$\text{dist}(\underline{x}_n, l) = \|\underline{e}\|$$

Now consider

$$\underline{x}_n = \hat{\underline{x}}_n + \underline{e}$$

$$\underline{w}^T \underline{x}_n = \underline{w}^T \hat{\underline{x}}_n + \underline{w}^T \underline{e}$$

but $\underline{w}^T \hat{\underline{x}}_n = 0$ as $\underline{w} \perp \hat{\underline{x}}_n$

$\therefore$ $\underline{w}^T \underline{x}_n = \underline{w}^T \underline{e}$

Note. $|\underline{w}^T \underline{x}_n| = |\underline{w}^T \underline{e}|$

Since $\underline{w}$ & $\underline{e}$ are collinear,

$$|\underline{w}^T \underline{e}| = \|\underline{w}\| \, \|\underline{e}\|$$

thus $|\underline{w}^T \underline{x}_n| = \|\underline{w}\| \, \|\underline{e}\|$

$$\therefore \quad \|\underline{e}\| = \frac{|\underline{w}^T \underline{x}_n|}{\|\underline{w}\|}$$

$$\therefore \text{dist}(\underline{x}_n, l) = \frac{|\underline{w}^T \underline{x}_n|}{\|\underline{w}\|}$$

Week 2

Binary Linear Classification

$$\mathcal{D} = \{(\underline{x}_1, y_1), (\underline{x}_2, y_2), \ldots (\underline{x}_N, y_N)\}$$

$$\underline{x}_i = (x_{i0} = 1, x_{i1}, x_{i2}, \ldots x_{id}) \in \mathbb{R}^{d+1}$$

$$y_i \in \{-1, +1\}$$

Model Parameters

weight vector $\underline{w} \in \mathbb{R}^{d+1}$

Classification Rule

$$\hat{y} = h_{\underline{w}}(\underline{x}) = \text{sign}(\underline{w}^T \underline{x})$$

$$= \begin{cases} +1 & \underline{w}^T \underline{x} > 0 \\ -1 & \underline{w}^T \underline{x} < 0 \end{cases}$$

Loss Function

$$E_{in}(\underline{w}) =$$

Loss Function

$$E_{in}(\underline{w}) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}\{y_n \neq \hat{y}_n\}$$

Given training set $\mathcal{D}$,
minimize $E_{in}(\underline{w})$

Perceptron Learning Algorithm

Input: Training set $\mathcal{D}$ that is linearly separable

Output: $\underline{w} \in \mathbb{R}^{d+1}$ that achieves $E_{in}(\underline{w}) = 0$

Initialization: Initialize $\underline{w}$ in an arbitrary manner

e.g. $\underline{w} = (0, 0, \ldots 0) \in \mathbb{R}^{d+1}$

Step 1: check if $E_{in}(\underline{w}) = 0$?

If yes STOP & output $\underline{w}$

Step 2: let $(\underline{x}_n, y_n)$ be any point in $\mathcal{D}$ that is mis-classified (including any point on boundary i.e. $\underline{w}^T \underline{x}_n = 0$)

If $y_n = +1$ then $\underline{w} \leftarrow \underline{w} + \underline{x}_n$

If $y_n = -1$ then $\underline{w} \leftarrow \underline{w} - \underline{x}_n$

$\underline{w} \leftarrow \underline{w} + y_n \cdot \underline{x}_n$

Go to Step 1

(In 1950's by Rosenblatt)

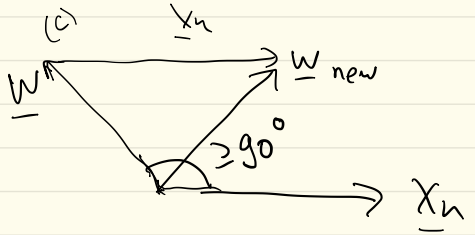
Update Rule:

Geometric Intuition

let $(\underline{x}_n, y_n)$ be a mis-classified point

Case 1: $y_n = +1$ $\quad \hat{y}_n = -1$

Recall: $\hat{y}_n = \text{sign}(\underline{w}^T \underline{x}_n)$

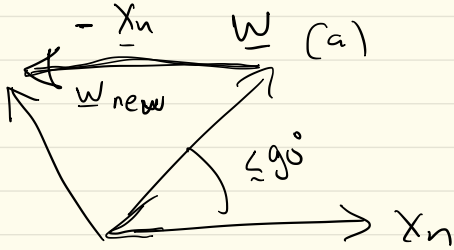


$\underline{w}^T \underline{x}_n \leq 0$

$\underline{w}_{new} \leftarrow \underline{w} + \underline{x}_n$

"$\underline{w}_{new}$" moves closer to $\underline{x}_n$

Case 2: $y_n = -1$, $\quad \hat{y}_n = +1$



$\underline{w}^T \underline{x}_n \geq 0$

$\underline{w}_{new} = \underline{w} - \underline{x}_n$

Algebraic Viewpoint

y_n	$w^T x_n$	classification	$y_n \cdot w^T x_n$
$+1$	> 0	✓	> 0
$+1$	< 0	X	< 0
-1	> 0	X	< 0
-1	< 0	✓	> 0

If $(\underline{x}_n, y_n)$ is correctly classified $y_n \underline{w}^T \underline{x}_n > 0$

If $(\underline{x}_n, y_n)$ is misclassified $y_n \underline{w}^T \underline{x}_n < 0$

Claim: If $(\underline{x}_n, y_n)$ is mis-classified in Step 2 and

$$\underline{w}_{new} = \underline{w} + y_n \cdot \underline{x}_n$$

$$y_n \underline{w}_{new}^T \underline{x}_n > y_n \cdot \underline{w}^T \underline{x}_n$$

Proof: $\underline{w}_{new}^T = \underline{w}^T + y_n \cdot \underline{x}_n^T$

$$\underline{w}_{new}^T \underline{x}_n = \underline{w}^T \underline{x}_n + y_n \underline{x}_n^T \underline{x}_n$$

$$y_n \underline{w}_{new}^T \underline{x}_n = y_n \underline{w}^T \underline{x}_n + \underbrace{y_n^2}_{=+1} \underline{x}_n^T \underline{x}_n \quad (\to 0)$$

$$\underline{x}_n^T \underline{x}_n = \|\underline{x}_n\|^2$$

Convergence of Perceptron Alg.

let $\underline{w}^*$ be the output of PLA.

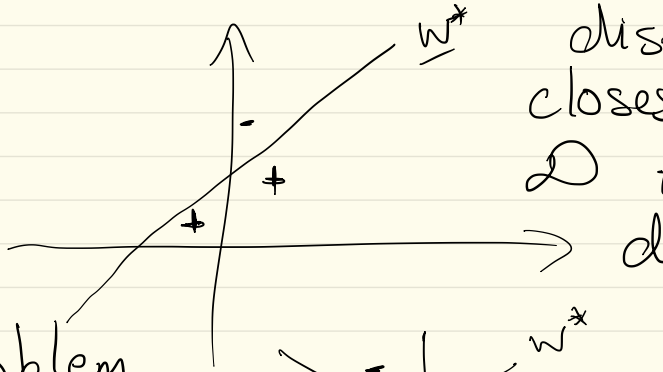
then PLA will stop in at-most

$$T = \frac{R^2}{\rho^2} \quad \text{steps where}$$

$$R = \max_{1 \le n \le N} \|\underline{x}_n\| \leftarrow \text{Radius of } \mathcal{D}$$

$$\rho = \min_{1 \le n \le N} \frac{|\underline{w}^{*T} \underline{x}_n|}{\|\underline{w}^*\|} \Big\} \text{ margin}$$

margin: distance between closest point in $\mathcal{D}$ to the decision boundary.



Problem 1.3 (textbook)

Pocket Algorithm

Heuristic that applies to non-separable data sets.

Idea: Keep the best weight vector upto current iteration aside

If the current perceptron update yields a better weight vector, i.e lower value of E_{in}, then replace the best weight vector with current one.

Stop after sufficiently many iterations.

Pocket Algorithm

Initialize: let $\hat{w} = w(0)$ be an arbitrary weight vector.

for $t = 0, 1, 2, \ldots T-1$ ($T = 10,000$)

- run one update of the perceptron Algorithm to obtain $w(t+1)$
- Evaluate $E_{in}(w(t+1))$
- If $E_{in}(w(t+1)) < E_{in}(\hat{w})$ ← replace $\hat{w} = w(t+1)$

End for

Linear Regression

$$\mathcal{D} = \{ (\underline{x}_1, y_1), (\underline{x}_2, y_2), \ldots (\underline{x}_N, y_N) \}$$

$$\underline{x}_i = (x_{i0} = 1, x_{i1}, x_{i2}, \ldots x_{id}) \in \mathbb{R}^{d+1}$$

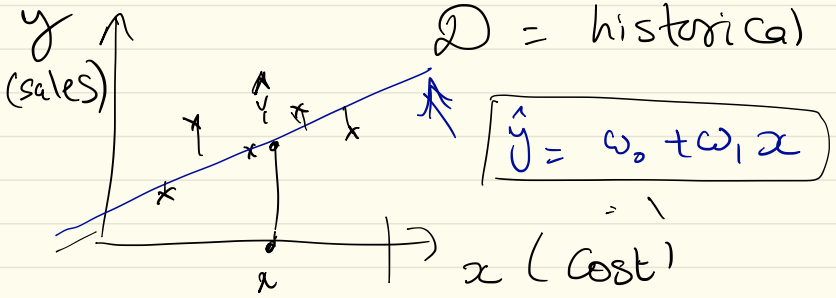
$$y_i \in \mathbb{R}$$

Given $\underline{x} = (x_0 = 1, x_1, x_2, \ldots x_d) \in \mathbb{R}^{d+1}$

output: $\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d$

Model Params $\underline{w} = (w_0, w_1, w_2, \ldots w_d) \in \mathbb{R}^{d+1}$

Example: $x \equiv$ advertising cost in 1 week
$y \equiv$ sales in that week
$\mathcal{D} =$ historical data



$\hat{y} = w_0 + w_1 x$

$w_0 \equiv$ Sales when there is no advertising

$w_1 \equiv$ increase in sales per unit increase in advertisement

Example 2:

$$\underline{X} = \begin{bmatrix} x_0 = 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \text{Cost/week in Internet adv} \\ \text{cost/week in TV adv} \\ \text{Cost/week in newspaper adv} \end{bmatrix}$$

$$\mathcal{D} = \{(\underline{x}_1, y_1), (\underline{x}_2, y_2), \ldots (\underline{x}_N, y_N)\}$$

Linear Prediction

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$$
$$= \underline{X}^T \underline{W}$$

Notation: Scalar variable: x

Vector: $\underline{X} = \begin{bmatrix} x_0 = 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \in \mathbb{R}^{d+1}$

$$\underline{X}^T = [x_0, x_1 \ldots x_d]$$

Matrix $\mathcal{X} = \begin{bmatrix} x_{11} & x_{12} \ldots x_{1M} \\ x_{21} & x_{22} \ldots x_{2M} \\ \vdots \\ x_{N1} & x_{N2} \ldots x_{NM} \end{bmatrix}$

Linear Regression : Problem Formulation

Dataset : $\{(\underline{x}_1, y_1), \dots (\underline{x}_N, y_N)\}$

$\underline{x}_i \in \mathbb{R}^{d+1}$, $y_i \in \mathbb{R}$

Data Matrix

$$\mathcal{X} = \begin{bmatrix} \underline{x}_1^T \\ \underline{x}_2^T \\ \vdots \\ \underline{x}_N^T \end{bmatrix} \in \mathbb{R}^{N \times (d+1)}$$

observation vector

$$\underline{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

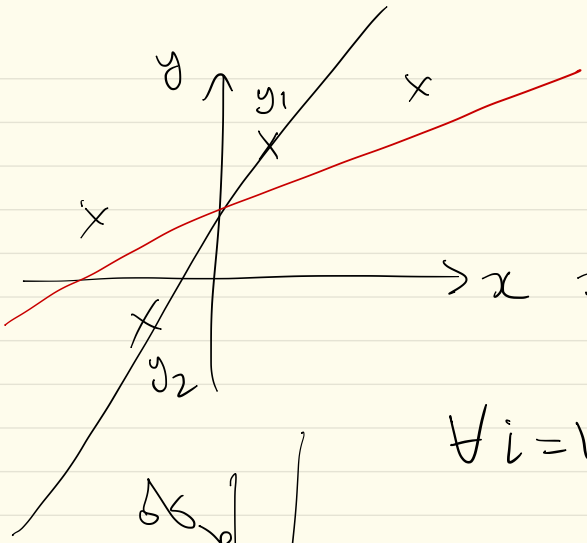
Weight vector

$$\underline{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \in \mathbb{R}^{(d+1)}$$

Prediction Vector

$$\underline{\hat{Y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{bmatrix} = \begin{bmatrix} \underline{x}_1^T \underline{w} \\ \underline{x}_2^T \underline{w} \\ \vdots \\ \underline{x}_N^T \underline{w} \end{bmatrix}$$

Given $\mathcal{X}, \underline{Y}$ find $\underline{w}$



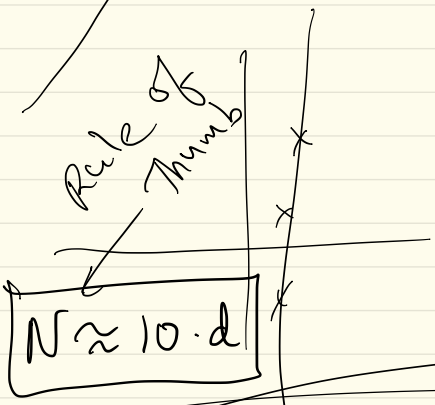
Ideally

$$y_i = \hat{y}_i$$

$$\forall i = 1, 2, \dots N$$

$$y_i = \underline{x}_i^T \underline{w}$$

for $i = 1, 2, \dots N$



$$y_1 = w_0 + x_{11} w_1 + x_{12} w_2 \dots + x_{1d} w_d$$

$$y_2 = w_0 + x_{21} w_1 + x_{22} w_2 + \dots + x_{2d} w_d$$

$$\vdots$$

$$y_N = w_0 + x_{N1} w_1 + x_{N2} w_2 + \dots + x_{Nd} w_d$$

unknowns: $w_0, w_1 \dots w_d \Rightarrow$ $N \leq d+1$

# of Eq's $\leq$ # of variables.

In practice $N \gg d$, so setting $\underline{y} = \hat{\underline{y}}$ is not feasible

Loss Function

$$E_{in}(\underline{w}) = \frac{1}{N} \| \underline{y} - \hat{\underline{y}} \|^2$$

$$= \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

$$= \frac{1}{N} \sum_{i=1}^{N} (y_i - \underline{x}_i^T \underline{w})^2$$

(Squared Error loss function)

$$\underline{w}^* = \arg\min_{\underline{w} \in \mathbb{R}^{d+1}} E_{in}(\underline{w})$$

$$= \arg\min_{\underline{w} \in \mathbb{R}^{d+1}} \| \underline{y} - \underbrace{X \underline{w}}_{= \hat{\underline{y}}} \|^2$$

Analytic Soln

Least Squares Solution

Loss Function

$$f(\underline{w}) = \| \underline{Y} - \underline{\underline{X}}\,\underline{w} \|^2$$

- $\underline{w}$: weight vector
- $\underline{Y}$: obs. vector
- $\underline{\underline{X}}$: data Matrix

Special case: $d = 1$

$$\underline{x}_i = \begin{pmatrix} x_{i0} = 1 \\ x_{i1} \end{pmatrix} \qquad \underline{w} = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}$$

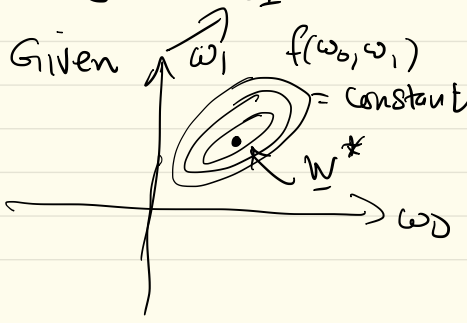
$$f(\underline{w}) = \sum_{i=1}^{N} (y_i - \underline{x}_i^T \underline{w})^2$$

$$\Downarrow$$

$$f(w_0, w_1) = \sum_{i=1}^{N} (y_i - w_0 - x_{i1} w_1)^2$$

Given

"Convex function"



$$\underset{\underline{w} \in \mathbb{R}^{d+1}}{\text{minimize}} \; f(\underline{w})$$

$$\nabla f(\underline{w}) = \begin{bmatrix} \frac{\partial f}{\partial \omega_0} \\ \frac{\partial f}{\partial \omega_1} \\ \vdots \\ \frac{\partial f}{\partial \omega_d} \end{bmatrix} = 0$$

$$f(\underline{w}) = \|\underline{y} - \underline{X}\underline{w}\|^2$$

Gradient:

$$g(\underline{w}) = \underline{w}^T \underline{v} = \omega_0 v_0 + \omega_1 v_1 + \cdots + \omega_d v_d = \underline{v}^T \underline{w}$$

$$\nabla g(\underline{w}) = \underline{v} \longrightarrow \begin{bmatrix} \frac{\partial g}{\partial \omega_0} \\ \frac{\partial g}{\partial \omega_1} \\ \vdots \end{bmatrix} = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \end{bmatrix} = \underline{v}$$

$$g(\underline{w}) = \underline{w}^T A \underline{w}$$

$$\nabla_{\underline{w}} g(w) = \nabla_w (\underline{w}^T A \underline{w}) = (A + A^T)\underline{w}$$

(Exercise) $= 2A\underline{w}$ if $A = A^T$

i.e. A is symmetric

$$f(\underline{w}) = \|\underline{Y} - \underline{\underline{X}}\,\underline{w}\|^2$$

$$\nabla f(\underline{w}) = 0$$

$$f(\underline{w}) = (\underline{Y} - \underline{\underline{X}}\,\underline{w})^T(\underline{Y} - \underline{\underline{X}}\,\underline{w})$$

$$= \underline{Y}^T\underline{Y} - \underline{Y}^T\underline{\underline{X}}\,\underline{w} - \underline{w}^T\underline{\underline{X}}^T\underline{Y} + (\underline{\underline{X}}\,\underline{w})^T(\underline{\underline{X}}\,\underline{w})$$

$$= \|Y\|^2 - 2\,\underline{w}^T\underline{\underline{X}}^T\underline{Y} + \underline{w}^T\underline{\underline{X}}^T\underline{\underline{X}}\,\underline{w}$$

$$\nabla f(\underline{w}) = \nabla_w\left(\|Y\|^2 - 2\,\underline{w}^T\underline{\underline{X}}^T\underline{Y} + \underline{w}^T\underline{\underline{X}}^T\underline{\underline{X}}\,\underline{w}\right)$$

$$= -2\,\underline{\underline{X}}^T\underline{Y} + 2\,\underline{\underline{X}}^T\underline{\underline{X}}\,\underline{w} = 0$$

$A = (\underline{\underline{X}}^T\underline{\underline{X}})$ $\quad A^T = \underline{\underline{X}}^T\underline{\underline{X}}$

$$\underline{\underline{X}} = \begin{bmatrix} \underline{x}_1^T \\ \underline{x}_2^T \\ \vdots \\ \underline{x}_N^T \end{bmatrix} \in \mathbb{R}^{N\times(d+1)}$$

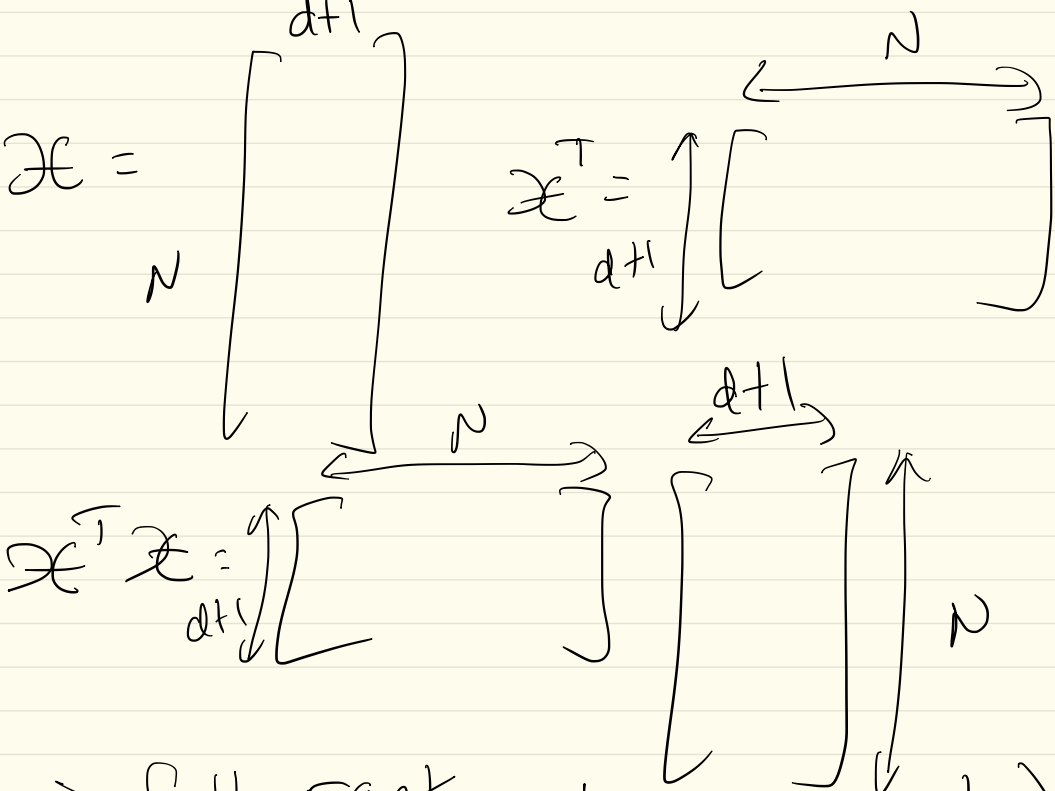
$$\underline{Y} \in \mathbb{R}^{N\times 1}$$

$$\underline{\underline{X}}^T\underline{\underline{X}} \in \mathbb{R}^{(d+1)\times(d+1)}$$

Solve for $\underline{w}$

$$\boxed{X^T X}\,\underline{w} = X^T \underline{y}$$

$M : (d+1) \times (d+1)$



$\Rightarrow$ full rank Matrix (usually)

$$\underline{w} = (\mathcal{X}^T\mathcal{X})^{-1}\mathcal{X}^T\underline{y}$$

$$\boxed{\hat{\underline{y}} = \mathcal{X}\underline{w}} = \mathcal{X}(\mathcal{X}^T\mathcal{X})^{-1}\mathcal{X}^T\underline{y}$$

Least Square Solution in Linear Regression.

$\mathcal{X}$: $N \times (d+1)$ Matrix

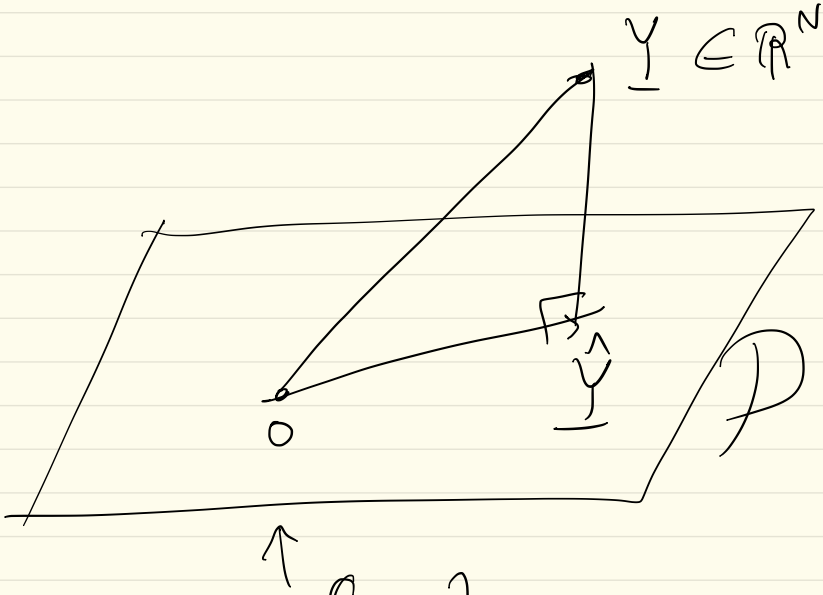
Projection Matrix

Geometric Interpretation

$\mathcal{X} = [\underline{Q}_0\ \underline{Q}_1 \cdots \underline{Q}_d]$ Columns of $\mathcal{X}$

$$\hat{\underline{y}} = \mathcal{X}\underline{w} = \underline{Q}_0 w_0 + \underline{Q}_1 w_1 + \cdots + \underline{Q}_d w_d$$

$\in$ Span $\{Q_0, Q_1, \cdots Q_d\}$



$\uparrow$

Col-Span $\{\underline{x}\}$

$=$ Span $\{\underline{Q}_0, \underline{Q}_1, \ldots \underline{Q}_d\}$

$$\hat{\underline{Y}} = \underbrace{\underline{x}(\underline{x}^T\underline{x})^{-1}\underline{x}^T}_{\text{projection Matrix}}\underline{Y}$$

Regularized Least Squares

$$\underline{w}_\lambda^* = \arg\min_{\underline{w} \in \mathbb{R}^{d+1}} \left\{ \|\underline{Y} - \underline{\underline{X}}\,\underline{w}\|^2 + \lambda \|\underline{w}\|^2 \right\}$$

λ = Regularization coefficient

λ can be selected emperically through cross-validation

$$f(\underline{w}) = \|\underline{Y} - \underline{\underline{X}}\,\underline{w}\|^2 + \lambda \|\underline{w}\|^2$$

$$\nabla f(\underline{w}) = 0$$

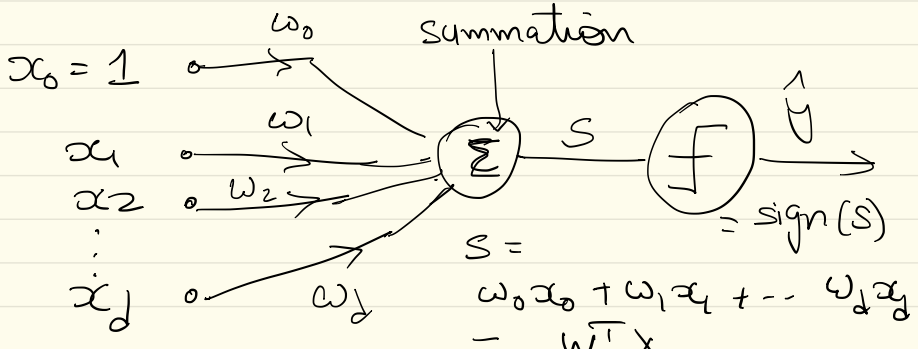
$$\underline{w}_\lambda^* = \left(\underline{\underline{X}}^T \underline{\underline{X}} + \lambda I\right)^{-1} \underline{\underline{X}}^T \underline{Y}$$

Logistic Regression

Linear Classification

Data Vector $\underline{x} \in \mathbb{R}^{d+1}$ $\quad y \in \{-1, +1\}$

$$\hat{y} = \text{sign}(\underline{w}^T \underline{x})$$



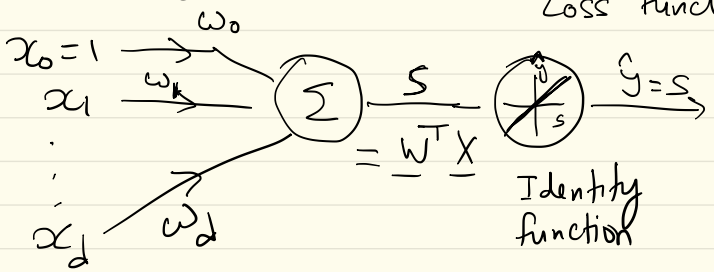
Loss Function $= 1\{y \neq \text{sign}(s)\}$

Linear Regression

Data Vector $\underline{x} \in \mathbb{R}^{d+1}$ $\quad y \in \mathbb{R}$

$\hat{y} = \underline{w}^T \underline{x}$

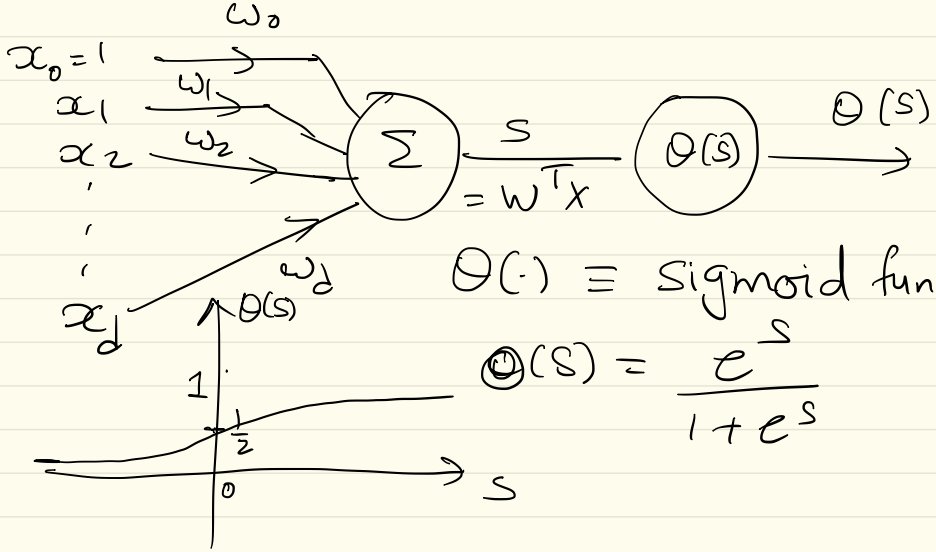
Loss Function $= (y - s)^2$



Binary Classification

$\underline{x} \in \mathbb{R}^{d+1}$ $\quad y \in \{-1, +1\}$

Model



$\theta(\cdot) \equiv$ sigmoid function

$$\theta(s) = \frac{e^s}{1+e^s}$$

Interpretation

$\theta(s) \underset{\hookrightarrow s = w^T x}{=} P_r(y=1 \mid \underline{x})$

$= \hat{P}_{\underline{w}}(1 \mid \underline{x})$

$$\hat{P}_{\underline{w}}(1 \mid \underline{x}) = \theta(\underline{w}^T \underline{x}) = \frac{e^{\underline{w}^T \underline{x}}}{1 + e^{\underline{w}^T \underline{x}}}$$

$$\hat{P}_{\underline{w}}(-1 \mid \underline{x}) = 1 - \hat{P}_{\underline{w}}(1 \mid \underline{x}) = \frac{1}{1+e^{\underline{w}^T \underline{x}}} = \frac{e^{-\underline{w}^T \underline{x}}}{1+e^{-\underline{w}^T \underline{x}}}$$

$$\hat{P}_{\underline{w}}(y|\underline{x}) = \frac{e^{y \cdot \underline{w}^T \underline{x}}}{1 + e^{y \cdot \underline{w}^T \underline{x}}}$$

$y \in \{-1, +1\}$

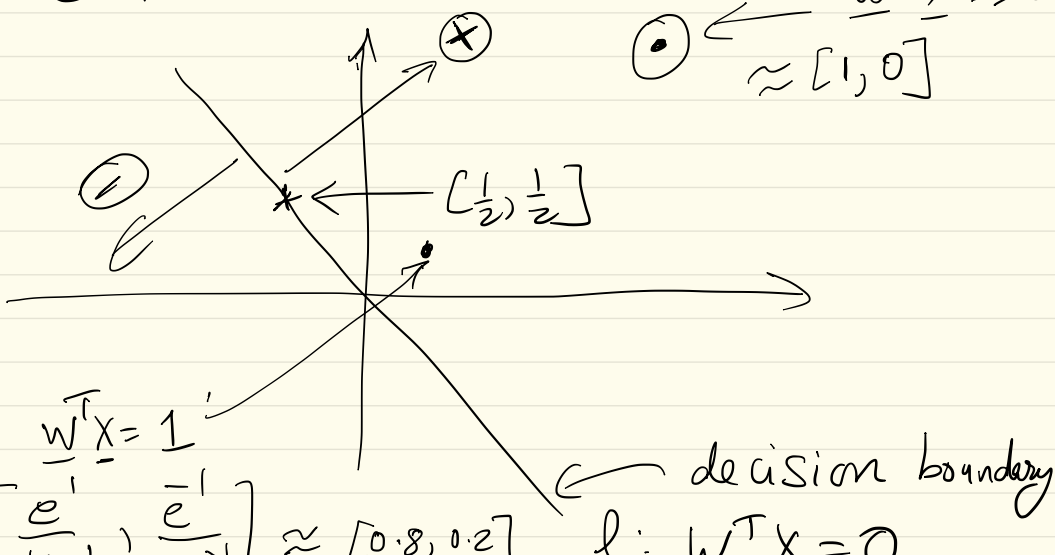
Recap: Input $\underline{x} \in \mathbb{R}^{d+1}$

Model Param: $\underline{w} \in \mathbb{R}^{d+1}$

Output: Probability Vector

$[\hat{P}_w(1|\underline{x}), \hat{P}_w(-1|\underline{x})]$

"Soft decision"



Training Set

$$\mathcal{D} = \{(\underline{x}_1, y_1), (\underline{x}_2, y_2), \ldots (\underline{x}_N, y_N)\}$$

Model Parameter: $\underline{w} \in \mathbb{R}^{d+1}$

Output: $[\hat{P}_w(1|\underline{x}), \hat{P}_w(-1|\underline{x})]$

Given $\mathcal{D}$ how to select $\underline{w}$? Which loss function to select?

Loss Function: $-\log \hat{P}_w[y_n|\underline{x}_n)$

$\underline{x}_n \rightarrow$ [Logistic Reg. Model $\underline{w}$] $\rightarrow [\hat{P}_w(1|\underline{x}_n), \hat{P}_w(-1|\underline{x}_n)]$

Log-loss Function

Example: A given input $\underline{x} \in \mathbb{R}^{d+1}$ generates the following output

$[\hat{P}_w(1|\underline{x}), \hat{P}_w(-1|\underline{x})] = [0.8, 0.2]$

$y = +1$ loss $= -\log 0.8 \simeq 0.22$

$y = -1$ loss $= -\log 0.2 \simeq 1.6$

Output vector $= [0.001, 0.999]$

$= [\hat{P}_w(1|\underline{x}), \hat{P}_w(-1|\underline{x})]$

$y = +1$ loss $= -\log 0.001 \approx 10$

$y = -1$ loss $\approx -\log 0.999 \approx 10^{-4}$

$E_{in}(\underline{w}) \equiv$ training Error

$$E_{in}(\underline{w}) = \frac{1}{N} \sum_{n=1}^{N} e_n(\underline{w})$$

where $e_n(\underline{w}) \equiv$ loss associated with $(\underline{x}_n, y_n)$

$$e_n(\underline{w}) = -\log \hat{P}_w(y_n | \underline{x}_n)$$

$$\hat{P}_w(y_n | \underline{x}_n) = \frac{e^{y_n \underline{w}^T \underline{x}_n}}{1 + e^{y_n \underline{w}^T \underline{x}_n}} = \frac{1}{1 + e^{-y_n \underline{w}^T \underline{x}_n}}$$

$$e_n(\underline{w}) = \log \frac{1}{\hat{P}_w(y_n|\underline{x}_n)} = \log\left(1 + e^{-y_n \underline{w}^T x_n}\right)$$

$$e_n(\underline{w}) = \log\left(1 + e^{-y_n \underline{w}^T \underline{x}_n}\right)$$

y_n	$\underline{w}^T \underline{x}_n$	$y_n \underline{w}^T \underline{x}_n$	$e_n(\underline{w})$
$+1$	$\gg 0$	$\gg 0$	small
$+1$	$\ll 0$	$\ll 0$	large
-1	$\gg 0$	$\ll 0$	large
-1	$\ll 0$	$\gg 0$	small

$$E_{in}(\underline{w}) = \frac{1}{N} \sum_{n=1}^{N} e_n(\underline{w})$$

$$= \frac{1}{N} \sum_{n=1}^{N} \log\left(1 + e^{-y_n \underline{w}^T \underline{x}_n}\right)$$

$$\underline{w}^* = \arg\min_{\underline{w} \in \mathbb{R}^{d+1}} E_{in}(\underline{w})$$

Logistic Regression

$$\mathcal{D} = \{(\underline{x}_1, y_1), (\underline{x}_2, y_2), \ldots (\underline{x}_N, y_N)\}$$

$\underline{x}_i \in \mathbb{R}^{d+1}$ $\quad y_i \in \{-1, +1\}$

Model Parameters : $\underline{w} \in \mathbb{R}^{d+1}$

Given $\underline{x} \in \mathbb{R}^{d+1}$

outputs $\left[\hat{P}_{\underline{w}}(1|\underline{x}),\ \hat{P}_{\underline{w}}(-1|\underline{x})\right]$

$$\hat{P}_{\underline{w}}(y|\underline{x}) = \frac{e^{y \cdot \underline{w}^T \underline{x}}}{1 + e^{y \cdot \underline{w}^T \underline{x}}} \qquad y \in \{-1, +1\}$$

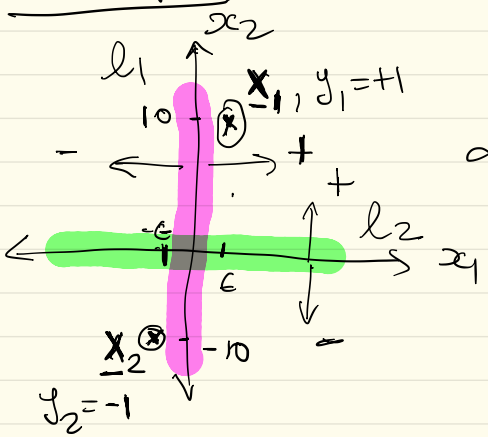
Loss Function

$$E_{in}(\underline{w}) = \frac{1}{N} \sum_{n=1}^{N} -\log \hat{P}_{\underline{w}}(y_n | \underline{x}_n)$$

$$= \frac{1}{N} \sum_{n=1}^{N} \log\left(1 + e^{-y_n \underline{w}^T \underline{x}_n}\right)$$

Example: $N = 2$ $d = 2$

$\epsilon \approx 10^{-3}$



$\mathcal{D} = \{(\underline{x}_1, y_1), (\underline{x}_2, y_2)\}$

$\underline{x}_1 = (1, \epsilon, 10)$ $y_1 = +1$

$\underline{x}_2 = (1, -\epsilon, -10)$ $y_2 = -1$

l_1 : $x_1 = 0$

l_1 : $\underline{w}^T \underline{x} = 0 \Rightarrow w_0 + w_1 x_1 + w_2 x_2 = 0$

$[w_0 \ w_1 \ w_2] = [0, 1, 0]$

evaluate loss Function $E_{in}(\underline{w})$
for (a) linear classification
(b) logistic Regression

(a)

$$E_{in}(\underline{w}) = \frac{1}{2} \sum_{n=1}^{2} 1\{y_n \neq \text{sign}(\underline{w}^T \underline{x}_n)\}$$

$= 0$

(b)

$$E_{in}(\underline{w}) = \frac{1}{2}\left(\log(1 + e^{-y_1 \underline{w}^T \underline{x}_1}) + \log(1 + e^{-y_2 \underline{w}^T \underline{x}_2})\right)$$

$$= \frac{1}{2}\left[\log(1 + e^{-1 \cdot \epsilon}) + \log(1 + e^{(-1)(-\epsilon)})\right] \approx 0.693$$

Regularization

l_2: $x_2 = 0$

$\underline{w} = (0, 0, 1)$

$\underline{x}_1 = (1, \epsilon, 10)$ $\quad$ $\underline{x}_2 = (1, -\epsilon, -10)$

$y_1 = +1$ $\quad$ $y_2 = -1$

linear classification

$$E_{in}(\underline{w}) = \frac{1}{2} \sum_{n=1}^{2} 1\{y_n \neq \text{sign}(\underline{w}^T \underline{x}_n)\}$$

$$= 0$$

logistic Regression

$$E_{in}(\underline{w}) = \frac{1}{2}\left(\log\left(1+e^{-y_1 \underline{w}^T \underline{x}_1}\right) + \log\left(1+e^{-y_2 \underline{w}^T \underline{x}_2}\right)\right)$$

$$= \frac{1}{2}\left[\log\left(1+e^{-1 \cdot 10}\right) + \log\left(1+e^{-(-1)(-10)}\right)\right]$$

$$= \log\left(1+e^{-10}\right) \approx 5 \times 10^{-4}$$

$\underline{w}' = (0, 0, 100) \Rightarrow 100\, x_2 = 0$

$\Rightarrow x_2 = 0 \Leftrightarrow l_2$

$E_{in}(\underline{w}') = E_{in}(\underline{w})$?

$$\min_{\underline{w}} \left[E_{in}(\underline{w}) + \lambda \|\underline{w}\|^2 \right]$$

$$E_{in}(\underline{w}) = \frac{1}{N} \sum_{n=1}^{N} \log\left(1 + e^{-y_n \underline{w}^T \underline{x}_n}\right)$$

$100 \underline{w}$

Maximum-Likelihood Viewpoint

Given: Training set

$$\mathcal{D} = \{(\underline{x}_1, y_1), (\underline{x}_2, y_2), \ldots (\underline{x}_N, y_N)\}$$

$\underline{x}_i \in \mathbb{R}^{d+1}$ $\quad y_i \in \{-1, +1\}$

Data vector sequence: $\underline{x}_1 \; \underline{x}_2, \ldots \; \underline{x}_N$

label sequence: $y_1, y_2, \ldots y_N$

logistic Regression Model assigns a probability to each y_i given $\underline{x}_i$

Select $\underline{w} \in \mathbb{R}^{d+1}$ that maximizes

$P(y_1 y_2 \ldots y_N \mid \underline{x}_1, \underline{x}_2 \ldots \underline{x}_N)$

Assume data samples are generated independently

$$P_{\underline{w}}(y_1, y_2 \cdots y_N \mid \underline{x}_1, \underline{x}_2, \cdots \underline{x}_N) = \prod_{n=1}^{N} \hat{P}_{\underline{w}}(y_n \mid \underline{x}_n)$$

select $\underline{w}$ that maximizes

$$\prod_{n=1}^{N} \hat{P}_{\underline{w}}(y_n \mid \underline{x}_n) \qquad \hat{P}_{\underline{w}}(y_n \mid \underline{x}_n) = \frac{1}{1+e^{-y_n \underline{w}^T \underline{x}_n}}$$

$$\underline{w}^* = \arg\max_{\underline{w}} \frac{1}{N} \prod_{n=1}^{N} \hat{P}_{\underline{w}}(y_n \mid \underline{x}_n)$$

$$= \arg\max_{\underline{w}} \frac{1}{N} \log \prod_{n=1}^{N} \hat{P}_{\underline{w}}(y_n \mid \underline{x}_n) = \arg\min_{\underline{w}} E_{in}(\underline{w})$$

$$= \arg\max_{\underline{w}} \frac{1}{N} \sum_{n=1}^{N} \log \hat{P}_{\underline{w}}(y_n \mid \underline{x}_n) \qquad -E_{in}(\underline{w})$$

Cross-Entropy Viewpoint

let $S = \{s_1, s_2, \ldots s_M\}$ be some discrete alphabets

let P & Q be two distributions on S

$$P = (P(s_1), P(s_2), \ldots P(s_M))$$

$$Q = (q(s_1), q(s_2), \ldots q(s_M))$$

Cross-Entropy function

$$CE(P, Q) = -\sum_{i=1}^{M=2} P(s_i) \log q(s_i)$$

log-loss function can be viewed as a cross-entropy.

consider

$$e_n(\underline{w}) = -\log \hat{P}_{\underline{w}}(y_n | \underline{x}_n)$$

$$= \{ -1\{y_n = +1\} \log \hat{P}_{\underline{w}}(1 | \underline{x}_n) - 1\{y_n = -1\} \cdot \log \hat{P}_{\underline{w}}(-1 | \underline{x}_n) \} \quad y_n \in \{-1, +1\}$$

$$S = \{-1, +1\}$$

$$CE(p, q) =$$

$$-p(-1) \log q(-1) - p(1) \log q(1)$$

$$- 1\{y_n = -1\} \log \hat{P}_w(-1 | x_n)$$

$$- 1\{y_n = +1\} \log \hat{P}_w(1 | x_n)$$

$$q = [\hat{P}_w(-1 | x_n), \hat{P}_w(1 | x_n)]$$

$$p = [1\{y_n = -1\}, 1\{y_n = +1\}]$$

$$= \begin{cases} [1, 0] & y_n = -1 \\ [0, 1] & y_n = +1 \end{cases}$$

ground truth probability

"distance" between output (q) & ground truth (p)

Training Set

$$\{(\underline{x}_1, \underline{q}_1), (\underline{x}_2, \underline{q}_2), \ldots (\underline{x}_N, \underline{q}_N)\}$$

$$\underline{q}_i = (q_i(-1), q_i(+1))$$

Output:

$$\underline{\hat{p}} = (\hat{P}_w(-1|\underline{x}_i), \hat{P}_w(1|\underline{x}_i))$$

loss function

$$= -q_i(-1) \log \hat{P}_w(-1|\underline{x}_i) - q_i(+1) \log \hat{P}_w(1|\underline{x}_i)$$

$$= CE(\underline{q}_i, \underline{\hat{p}})$$

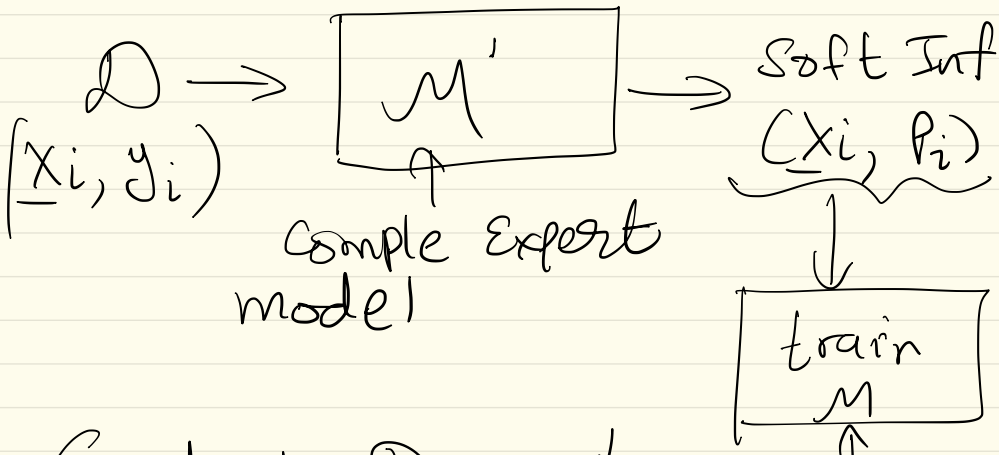
Knowledge Distillation (2016)

- limited Dataset $\mathcal{D}$
- train simple Model M

large & Complex Model M'

(Hinton 2016)

Pretrained Model : M'



Gradient Descent

$$\underline{w}^* = \arg\min_{\underline{w} \in \mathbb{R}^{d+1}} \frac{1}{N} \sum_{n=1}^{N} \log\left(1 + e^{-y_n \underline{w}^T \underline{x}_n}\right)$$

numerical methods

convex in $\underline{w}$

Linear Regression

$$\underline{w}^* = \arg\min_{\underline{w} \in \mathbb{R}^{d+1}} \frac{1}{N} \sum_{i=1}^{N} (y_i - \underline{w}^T \underline{x}_i)^2$$

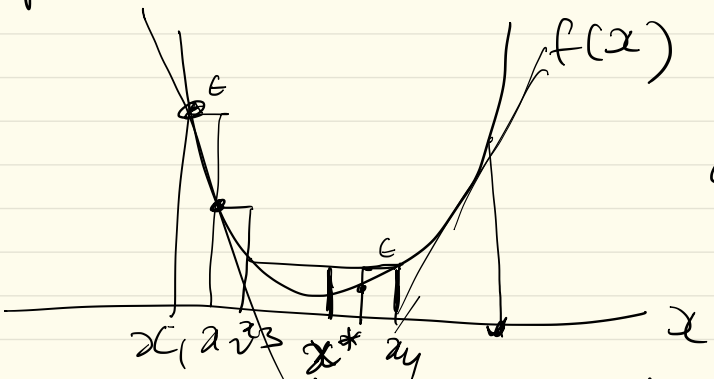
Gradient Descent

Given some function $f: \mathbb{R}^n \to \mathbb{R}$

$$\underset{\underline{X} \in \mathbb{R}^n}{\text{minimize}} \; f(\underline{X})$$

Assume $f(\underline{X})$ is a differentiable function, but a closed-form solution will not exist in general.

special case $n = 1$ $\quad f: \mathbb{R} \to \mathbb{R}$



$f(x)$ is a convex function

Compute x^* numerically.

Consider $f'(x) = \frac{\partial f}{\partial x}$

If $x = x^*$ $\quad f'(x) = 0$

if $x > x^*$ $\quad f'(x) > 0$

if $x < x^*$ $\quad f'(x) < 0$

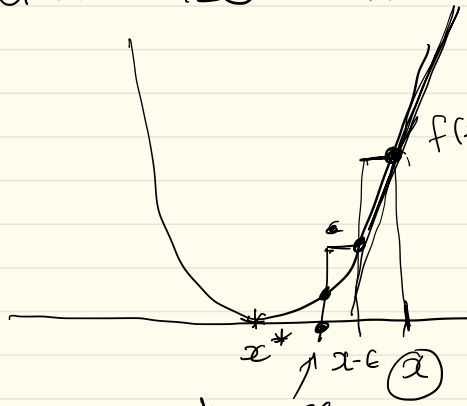
Gradient Descent

Given a function $f: \mathbb{R}^n \to \mathbb{R}$

$$\text{minimize} \quad f(\underline{x})$$
$$\underline{x} \in \mathbb{R}^n$$

Assume $f(\underline{x})$ is differentiable but a closed form solution to the minimum does not exist

Special case $n=1$ $\quad f: \mathbb{R} \to \mathbb{R}$



assume $f(x)$ is convex

$f'(x) = 0$

If $x = x^* \Rightarrow f'(x) = 0$

If $x > x^* \Rightarrow f'(x) > 0$ } $f(x)$ is increasing

If $x < x^* \Rightarrow f'(x) < 0$

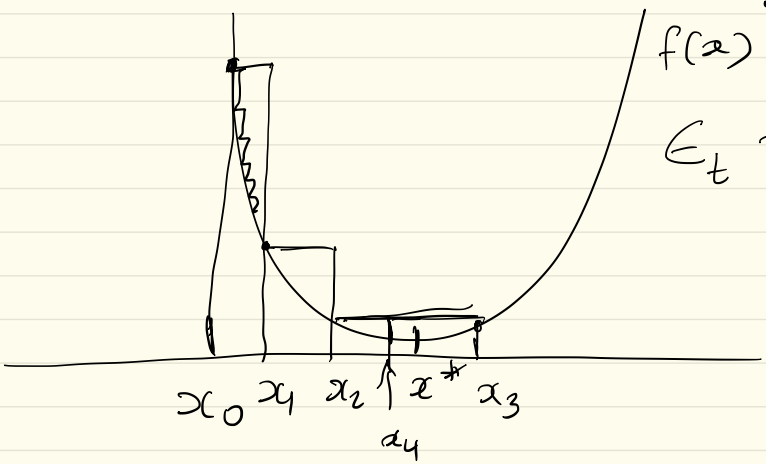
$f(x)$ is decreasing.

1. Initialize $x = x_0$

2. If $f'(x) \approx 0$ then stop & output x

3. If $f'(x) > 0$ then $x \leftarrow x - \epsilon$ goto step 2

4. If $f'(x) < 0$ then $x \leftarrow x + \epsilon$ goto step 2.



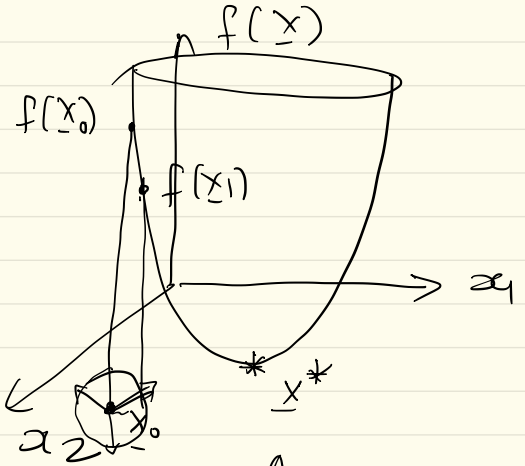
typically decrease ϵ as the iteration progresses.

large $\epsilon \Rightarrow$ less accuracy small $\epsilon \Rightarrow$ slow convergence

$f: \mathbb{R}^n \rightarrow \mathbb{R}$ $\quad n = 2$

$f(x_1, x_2)$

assume $f(\underline{x})$ is convex.



Given $\underline{x}_0$ is starting point

$$\underline{x}_1 = \underline{x}_0 + \epsilon \cdot \underline{u}$$

$\epsilon \equiv$ step size

$\underline{u} \equiv$ unit vector $\|\underline{u}\| = 1$

Assume ϵ is some small constant (e.g. 10^{-3})

How to find $\underline{u}$?

$f(\underline{x}_1) = f(\underline{x}_0 + \epsilon \cdot \underline{u})$

select $\underline{u}$ that minimizes $f(\underline{x}_0 + \epsilon \underline{u})$

Taylor Series Approximation

$$f(\underline{x}_1) \cong f(\underline{x}_0) + \epsilon \cdot \underline{u}^T \cdot \nabla f(\underline{x}_0) + O(\epsilon^2)$$

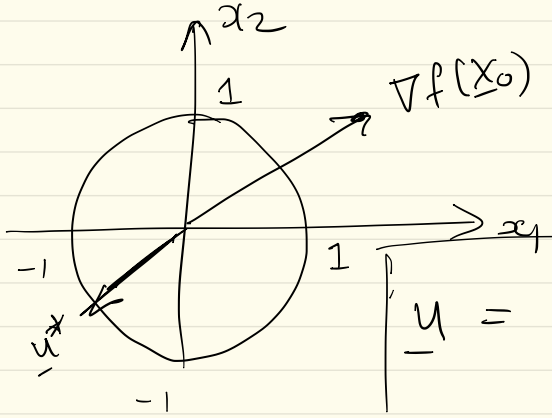
$$\underline{u} = \underset{\underline{u}:\|\underline{u}\|=1}{\arg\min} \left\{ f(\underline{x}_0) + \epsilon \cdot \underline{u}^T \cdot \nabla f(\underline{x}_0) \right\}$$

2^{nd} order terms.

$$\underline{u} = \arg\min \left(\underline{u}^T \cdot \nabla f(\underline{x}_0) \right)$$

$$\mathcal{U}: \|\underline{u}\| = 1$$

$$= \begin{bmatrix} \frac{\partial f(\underline{x}_0)}{\partial x_1} \\ \frac{\partial f(\underline{x}_0)}{\partial x_2} \end{bmatrix}$$



$$\underline{u} = - \frac{\nabla f(\underline{x}_0)}{\|\nabla f(\underline{x}_0)\|}$$

Given $\underline{x}_0 \longrightarrow \underline{x}_1$

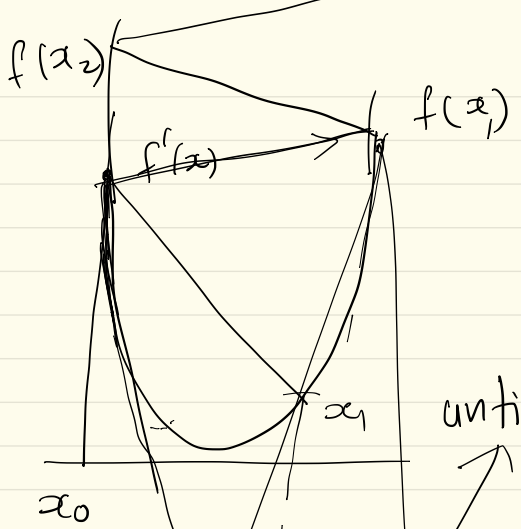
$$\underline{x}_{t+1} = \underline{x}_t + (\text{step-size})_t \cdot \underline{u}_t$$

$$u_t = - \frac{\nabla f(x_t)}{\|\nabla f(\underline{x}_t)\|}$$

$$(\text{step-size})_t = \|\nabla f(\underline{x}_t)\| \cdot \epsilon_t$$

$$\underline{x}_{t+1} = \underline{x}_t - \epsilon_t \nabla f(\underline{x}_t)$$

learning rate



until convergence

ϵ_t is very large

ϵ_t is too small

$$\epsilon_t \sim \frac{1}{t}$$

$t \equiv$ iteration index

guaranteed convergence

Gradient descent for convex functions

Initialize $\underline{x}_0$ in some arbitrary fashion

for $t = 0, 1, 2, \ldots$

- compute $\underline{g}_t = \nabla f(\underline{x}_t)$
- set $\underline{v}_t = -\underline{g}_t$
- update $\underline{x}_{t+1} = \underline{x}_t + \epsilon_t \cdot \underline{v}_t$

Continue

Announcements

1) Assignment 1: due oct 9th in class & online (on quercus)

2) Midterm: oct 16th (locations TBA)

Gradient Descent for Linear Regression

Loss function

$$E_{in}(\underline{w}) = \frac{1}{N} \sum_{n=1}^{N} e_n(\underline{w})$$

$$e_n(\underline{w}) = (\underline{w}^T \underline{x}_n - y_n)^2$$

$$\underline{w}^* = \arg\min_{\underline{w} \in \mathbb{R}^{d+1}} E_{in}(\underline{w})$$

$\Rightarrow$ find a closed-form (analytical) solution.

$$w_{LS}^* = (\underbrace{\underline{X}^T}_{d+1 \times N} \underbrace{\underline{X}}_{N \times d+1})^{-1} \underline{X}^T \underline{y}$$

Apply Gradient descent to compute $\underline{w}^*$

$\nabla_x f(x) = \begin{bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \end{bmatrix}$

Update Rule $\forall t \geq 0$

$$\underline{w}_{t+1} = \underline{w}_t - \epsilon_t \nabla E_{in}(\underline{w}_t)$$

$$E_{in}(\underline{w}) = \frac{1}{N} \sum_{n=1}^{N} \underbrace{(\underline{w}^T x_n - y_n)^2}_{= e_n(\underline{w})}$$

$$\nabla E_{in}(\underline{w}_t) = \nabla_{\underline{w}_t} \left(\frac{1}{N} \sum_{n=1}^{N} (\underline{w}_t^T \underline{x}_n - y_n)^2 \right)$$

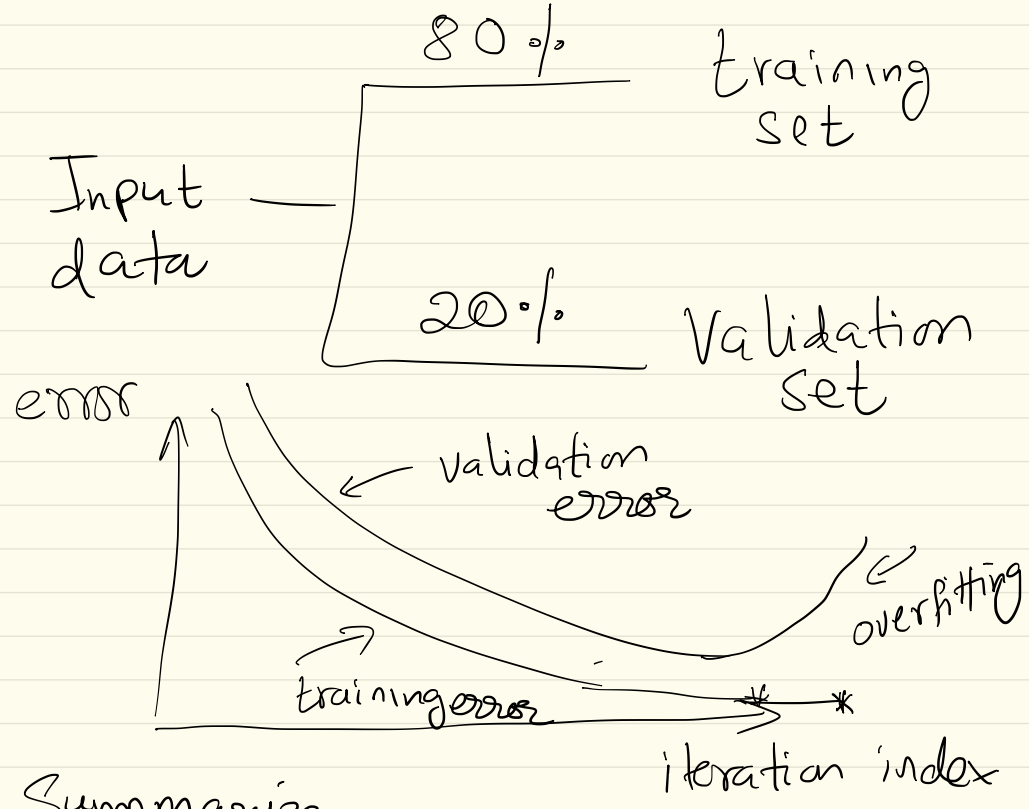
$$= \frac{1}{N} \sum_{n=1}^{N} \nabla_{\underline{w}_t} (\underline{w}_t^T \underline{x}_n - y_n)^2$$

$$= \frac{1}{N} \sum_{n=1}^{N} 2(\underline{w}_t^T \underline{x}_n - y_n) \nabla_{\underline{w}_t} \underbrace{(\underline{w}_t^T \underline{x}_n - y_n)}_{= \underline{x}_n^T \underline{w}_t}$$

$$= \frac{1}{N} \sum_{n=1}^{N} 2(\underline{w}_t^T \underline{x}_n - y_n) \underline{x}_n$$

$$\boxed{\underline{w}_{t+1} = \underline{w}_t - \frac{2\epsilon_t}{N} \sum_{n=1}^{N} (\underline{w}_t^T \underline{x}_n - y_n) \underline{x}_n}$$

as $t \to \infty$ $\quad w_t \to w_{LS}$



Summarize

Least-Squares Solution

(1) Can computationally Expensive

(2) Cause overfitting

(3) can be an overkill

Gradient Descent update for Linear Regression

$$\underline{w}_{t+1} = \underline{w}_t - \epsilon_t \frac{1}{N} \sum_{n=1}^{N} 2 \underline{x}_n (\underline{w}_t^T \underline{x}_n - y_n)$$

For each iteration require $\Theta(ND)$ computations

Stochastic Gradient Descent

at each iteration select one training example at random (uniformly) and compute

$$\underline{w}_{t+1} = \underline{w}_t - \epsilon_t \nabla e_n(\underline{w}_t)$$

$$n \sim \text{unif}\{1, 2, \ldots, N\}$$

$$\underline{w}_{t+1} = \underline{w}_t - 2\epsilon_t (\underline{w}_t^T \underline{x}_n - y_n) \cdot \underline{x}_n$$

Justification for SGD

Consider

$$\mathbb{E}_n\left[\nabla e_{(n)}(w_t)\right] = \nabla E_{in}(w_t)$$

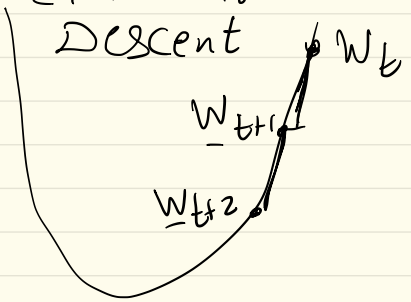
$n \in \text{Unif}\{1, 2, \ldots N\}$

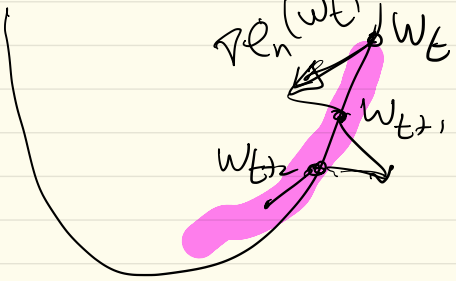
$$= \Pr(n=1)\,\nabla e_1(w_t) + \Pr(n=2)\,\nabla e_2(w_t) + \cdots + \Pr(n=N)\,\nabla e_N(w_t)$$

$$= \frac{1}{N}\left(\nabla e_1(w_t) + \nabla e_2(w_t) + \cdots + \nabla e_N(w_t)\right)$$

$$= \nabla E_{in}(w_t)$$

Exact Gradient Descent





Mini-Batch Gradient Descent

at each iteration draw M examples from the data set at random (without replacement)

let $S_k = \{i_1, i_2 \ldots i_M\}$ denote the set of examples randomly selected

update Rule

$$W_{k+1} = W_k - \frac{\epsilon_k}{M} \sum_{n=1}^{M} \nabla e_{(i_n)}(W_k)$$

$M = 1 \Rightarrow$ recover SGD

$M = N \Rightarrow$ Exact Gradient descent

$M = 32, 64$ etc.

Logistic Regression

$$E_{in}(\underline{w}) = \frac{1}{N} \sum_{n=1}^{N} e_n(\underline{w})$$

$$e_n(\underline{w}) = \log\left(1 + e^{-y_n \underline{w}^T x_n}\right)$$

SGD update

$$\underline{w}_{k+1} = \underline{w}_k - \epsilon_k \nabla e_n(\underline{w}_k)$$

$$n \in \text{unif}\{1, 2, \ldots N\}$$

$$\nabla e_n(\underline{w}_k) = \nabla_{\underline{w}_k}\left(\log\left(1 + e^{-y_n \underline{w}_k^T \underline{x}_n}\right)\right)$$

$$= \frac{1}{1 + e^{-y_n \underline{w}_k^T \underline{x}_n}} \nabla_{\underline{w}_k}\left(1 + e^{-y_n \underline{w}_k^T \underline{x}_n}\right)$$

$$= \frac{\nabla_{\underline{w}_k}\left(e^{-y_n \underline{w}_k^T \underline{x}_n}\right)}{1 + e^{y_n \underline{w}_k^T \underline{x}_n}} = \frac{e^{-y_n \underline{w}_k^T x_n}}{1 + e^{y_n \underline{w}_k^T \underline{x}_n}} \underbrace{\nabla_{\underline{w}_k}\left(-y_n \underline{w}_k^T \underline{x}_n\right)}_{= -y_n \underline{x}_n}$$

Update Rule: $k \geq 0$

$$\underline{w}_{k+1} = \underline{w}_k + \frac{\epsilon_k y_n \underline{x}_n}{1 + e^{y_n \underline{w}_k^T \underline{x}_n}}$$

$n \in \text{Unif}\{1, 2, \ldots N\}$

Special Cases

- Select a highly misclassified point

$$y_n \underline{w}_k^T \underline{x}_n \ll 0$$

$$e^{y_n \underline{w}_k^T \underline{x}_n} \approx 0$$

$$\underline{w}_{k+1} = \underline{w}_k + \epsilon_k y_n \underline{x}_n$$

Set $\epsilon_k = 1 \Rightarrow$ Perceptron Alg

$\rightarrow$ Alternately $y_n \underline{w}_k^T \underline{x}_n \gg 0 \Rightarrow \underline{w}_{k+1} \approx \underline{w}_k$

Multi-class Logistic Regression

Input Dataset

$$D = \{ (\underline{X}_1, y_1), (\underline{X}_2, y_2), \ldots (\underline{X}_N, y_N) \}$$

$\underline{X}_i \in \mathbb{R}^{d+1}$ $\quad y_i \in \{1, 2, \ldots C\}$

We will consider $C = 3$

let $\underline{W}(1), \underline{W}(2), \underline{W}(3) \in \mathbb{R}^{d+1}$ be weight vectors for classes '1', '2' & '3'.

$$\Omega = \{ \underline{W}(1), \underline{W}(2), \underline{W}(3) \}$$

(Model Parameters)

Output: $[\hat{P}_\Omega(1|\underline{X}), \hat{P}_\Omega(2|\underline{X}), \hat{P}_\Omega(3|\underline{X})]$

given $\underline{X} \in \mathbb{R}^{d+1}$

Soft Max - classification

$$\hat{P}_{\Omega}(i|\underline{X}) = \frac{e^{\underline{w}^T(i)\,\underline{X}}}{e^{\underline{w}^T(1)\underline{X}} + e^{\underline{w}^T(2)\underline{X}} + e^{\underline{w}^T(3)\underline{X}}}$$

$i \in \{1, 2, 3\}$

Loss Function

Log - loss function $(\underline{X}_n, y_n)$

$y_n \in \{1, 2, 3\}$

output $\left[\hat{P}_{\Omega}(1|\underline{X}_n), \hat{P}_{\Omega}(2|\underline{X}_n), \hat{P}_{\Omega}(3|\underline{X}_n)\right]$

$$e_n(\Omega) = -\log \hat{P}_{\Omega}(y_n|\underline{X}_n)$$

find $$\Omega^* = \arg\min_{\Omega} \frac{1}{N} \sum_{n=1}^{N} e_n(\Omega)$$

$\Omega^* = (\underline{w}^*(1), \underline{w}^*(2), \underline{w}^*(3))$ } using SGD

SGD update

Iteration #k

$$\Omega_k = (\underline{W}_k(1), \underline{W}_k(2), \underline{W}_k(3))$$

$$\underline{W}_{k+1}(l) = \underline{W}_k(l) - \epsilon_k \nabla_{W_k(l)} e_n(\Omega)$$

$$n \in \text{Unif}\{1, 2, \ldots N\}$$

$$l \in \{1, 2, 3\}$$

$$\underline{W}_{k+1}(1) = \underline{W}_k(1) - \epsilon_k \nabla_{W_k(1)} e_n(\Omega)$$

$$\underline{W}_{k+1}(2) = \underline{W}_k(2) - \epsilon_k \nabla_{W_k(2)} e_n(\Omega)$$

$$\underline{W}_{k+1}(3) = \underline{W}_k(3) - \epsilon_k \nabla_{W_k(3)} e_n(\Omega)$$

Standard SGD

Compute

$$\nabla_{w_k(1)} e_n(\Omega_k)$$

$$= \nabla_{\underline{w}_k(1)} \left(-\log \hat{p}_{\Omega_k}(y_n | \underline{x}_n) \right)$$

$$= \nabla_{w_k(1)} \left[-\log \frac{e^{\underline{w}_k^T(y_n) \cdot \underline{x}_n}}{e^{\underline{w}_k^T(1) \cdot \underline{x}_n} + e^{\underline{w}_k^T(2) \cdot \underline{x}_n} + e^{\underline{w}_k^T(3) \cdot \underline{x}_n}} \right]$$

$$= \nabla_{w_k(1)} \left[-\underline{w}_k^T(y_n) \underline{x}_n + \log \left(e^{\underline{w}_k^T(1) \underline{x}_n} + e^{\underline{w}_k^T(2) \underline{x}_n} + e^{\underline{w}_k^T(3) \underline{x}_n} \right) \right]$$

$y_n \in \{1, 2, 3\}$

$y_n = 1$: $\nabla_{w_k(1)} \left(\underline{w}_k^T(1) \cdot \underline{x}_n \right) = \underline{x}_n$

$y_n = 2, 3$ $\nabla_{w_k(1)} \left(\underline{w}_k^T(y_n) \cdot \underline{x}_n \right) = 0.$

Verify

$$\nabla_{w_k(l)} \left(\log \left(e^{w_k^T(1) x_n} + e^{w_k^T(2) \cdot x_n} + e^{w_k^T(3) \cdot x_n} \right) \right)$$

$$= \frac{e^{w_k^T(l) \cdot x_n}}{\sum_{j=1}^{3} e^{w_k^T(j) \cdot x_n}} \cdot x_n$$

Show that

$$\nabla_{w_k(l)} \left[-\log \hat{P}_{w_k}(x_n) \right]$$

$$= \begin{cases} -x_n + \dfrac{e^{w_k^T(l) \cdot x_n}}{\sum_{j=1}^{3} e^{w_k^T(j) x_n}}, & \text{if } y_n = l \\ \dfrac{e^{w_k^T(l) \cdot x_n}}{\sum_{j=1}^{3} e^{w_k^T(j) x_n}} & \text{if } y_n \neq l \end{cases}$$

Momentum Based Methods

Basic SGD update

$$g_t = \nabla e_n(\underline{W}_t)$$

$$\underline{W}_{t+1} = \underline{W}_t - \epsilon_t g_t$$

SGD + Momentum $\underline{V}_0 = 0$

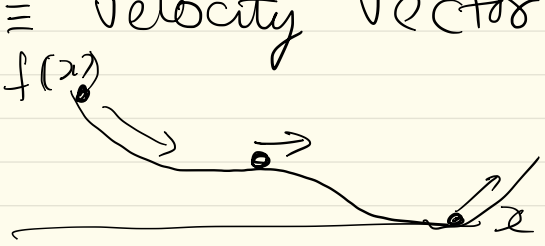
$$\underline{g}_t = \nabla e_n(\underline{W}_t)$$

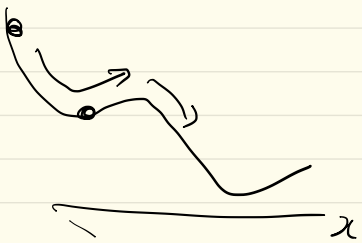
$$\underline{V}_t = -\epsilon_t \underline{g}_t + \mu \cdot \underline{V}_{t-1}$$

typically $\mu \approx 0.9$

$$\underline{W}_{t+1} = \underline{W}_t + \underline{V}_t$$

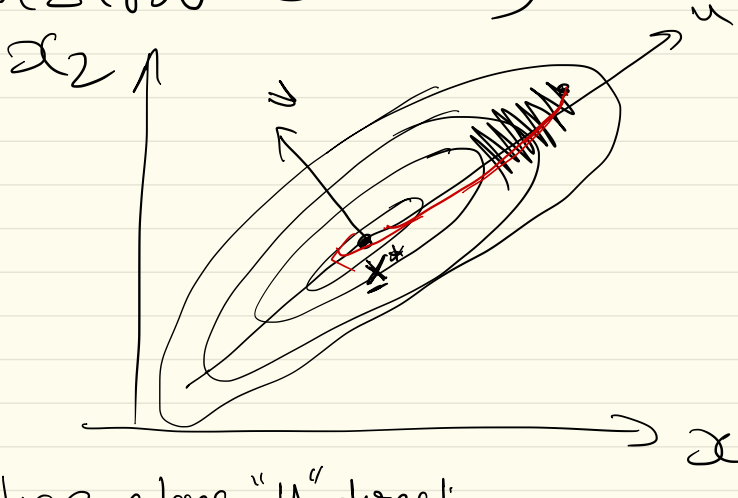
$\underline{V} \equiv$ velocity vector





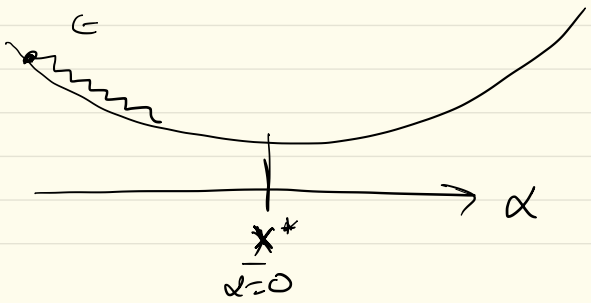
Even for convex function momentum based methods have provably faster convergence than gradient descent

Nestrov (1983)



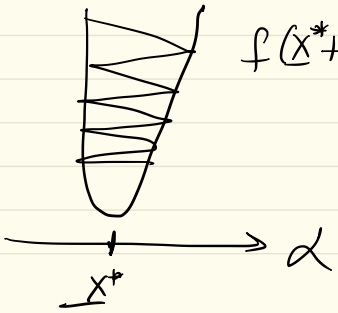
slice along "u" direction

$f(\underline{x}^* + \alpha \underline{u})$



slice along v direction

$f(\underline{x}^* + \alpha \underline{v})$



Announcements

1) Assignment #1: Due on Wed.
   - hard copy in class
   - code submission online
2) Mid-term - Next Wednesday
   - will cover upto momentum methods (first part of today's lecture)
3) Wed Oct 9th lecture by TA
   NO office hour on wed.
   - Extra OH: TBA

Momentum Method.

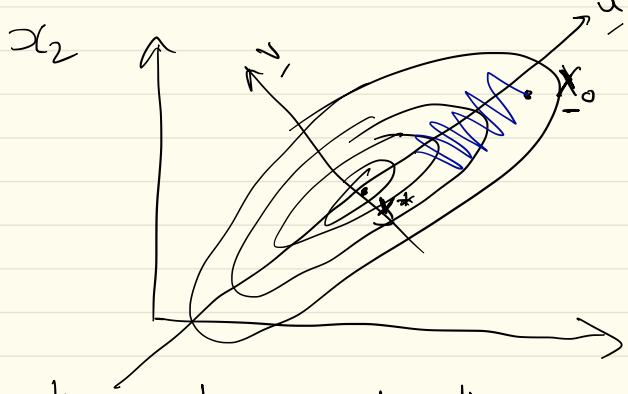
gradient vector: $\underline{g}_t = \nabla e_n(\underline{w}_t)$

velocity vector: $\underline{v}_t = -\epsilon_t \underline{g}_t + \mu \underline{v}_{t-1}$

$\underline{v}_0 = 0$

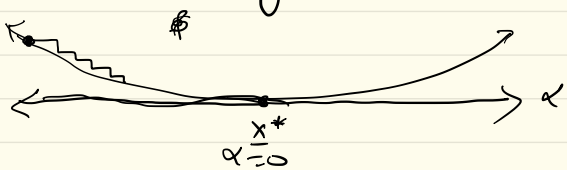
$\underline{w}_{t+1} = \underline{w}_t + \underline{v}_t$

Momentum methods lead to faster convergence even for convex functions

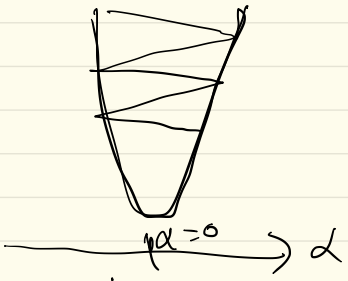


$f(x_1, x_2) =$ Constant

slice along v direction

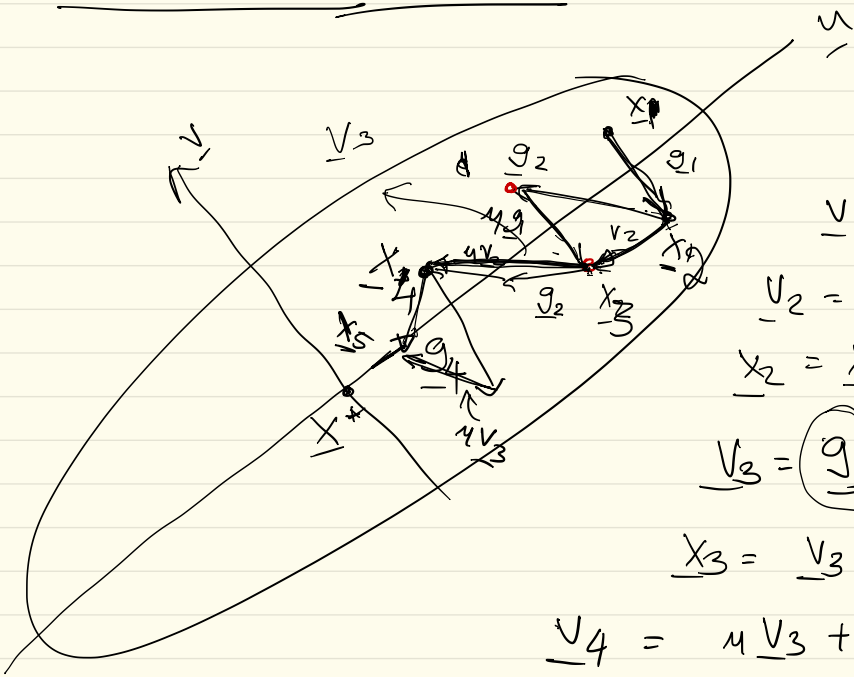


$f(\underline{x}^* + \alpha \underline{u})$



$f(\underline{x}^* + \alpha \underline{v})$

Momentum Method



$$v_1 = g_1$$

$$v_2 = g_2 + \mu g_1 \checkmark$$

$$x_2 = x_1 + v_2 \checkmark$$

$$v_3 = g_3 + \mu v_2$$

$$x_3 = v_3 + x_2$$

$$v_4 = \mu v_3 + g_4$$

$$x_4 = x_3 + v_4$$

Nestrov Momentum

$$v_t = \mu v_{t-1} - \epsilon_t \nabla C_n(w_t + \mu v_{t-1})$$

$$w_{t+1} = w_t + v_t$$

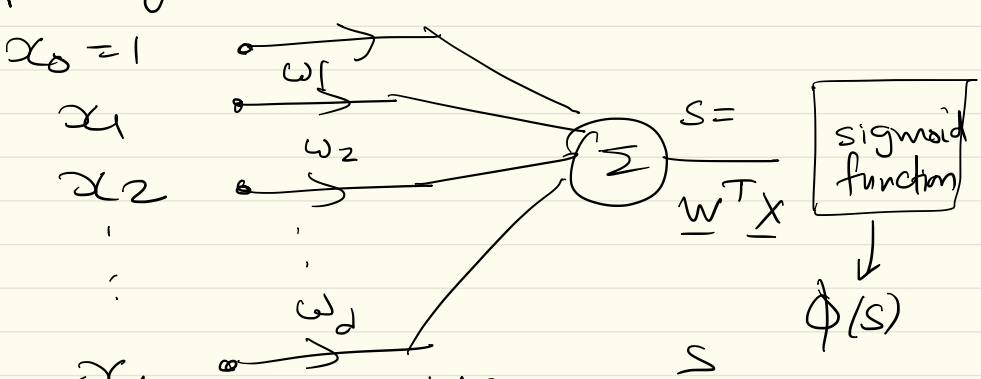
ADAM method (built into Tensorflow)

Neural Networks

Recall: Binary logistic Regression

"single layer network"

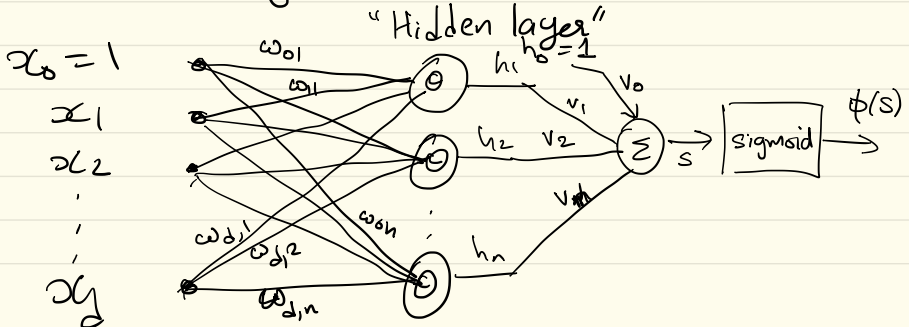
Input layer



$$\phi(s) = \frac{e^s}{1+e^s}$$

$$\phi(s) = \hat{P}_w(1 \mid \underline{x})$$

2-layer Network



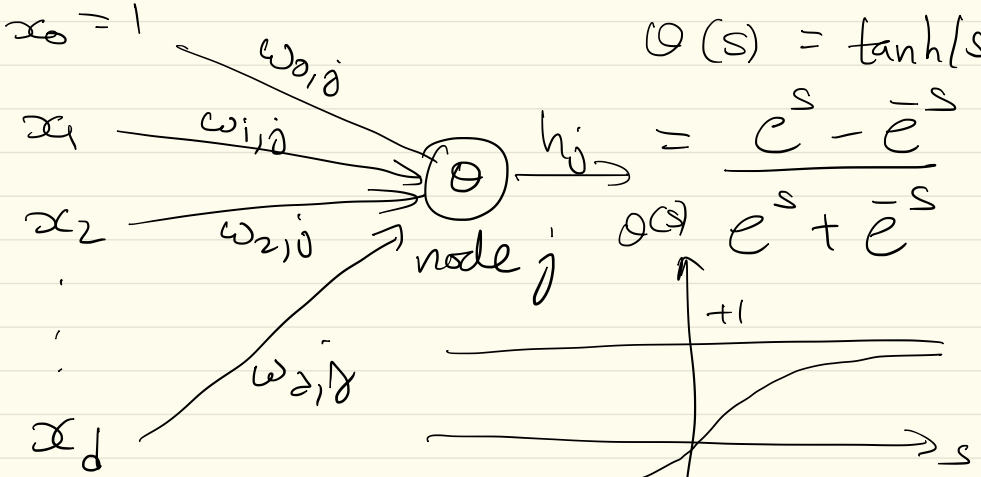
Θ is a non linear function.

$w_{i,j}$ ≡ weight between input node i & hidden layer node j

v_i = weight on the edge between hidden node i & output node

h_j ≡ output of hidden node j

$$h_j = \Theta\left(w_{0j} + \sum_{i=1}^{d} w_{ij}\, x_i\right)$$



$\Theta(s) = \tanh(s) = \dfrac{e^{s} - e^{-s}}{e^{s} + e^{-s}}$

$\Theta(s) \approx s$ if $|s| \ll 1$

$\Theta'(s) = 1 - \Theta^2(s)$

ReLu

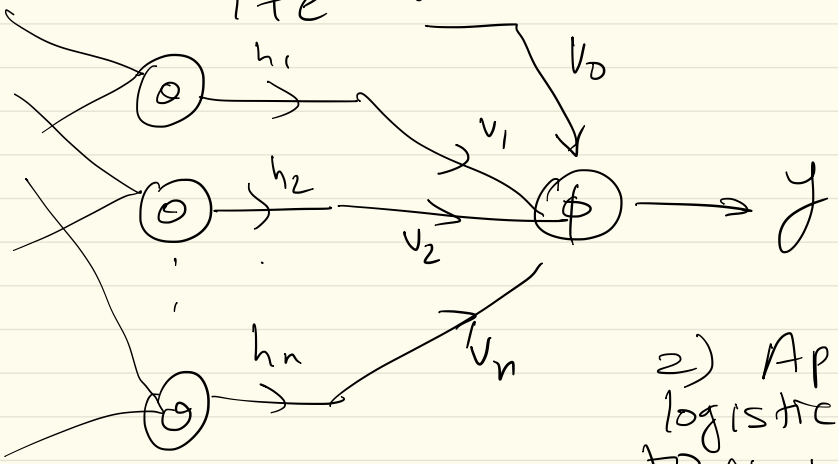
$\Theta(s) = \max(0, s)$

$$h_j = \theta(\underline{w}_j^T \underline{x})$$

$$\underline{w}_j = (w_{0j} \quad w_{1j} \quad \cdots \quad w_{dj})^T$$

$$y = \phi\left(v_0 + \sum_{j=1}^{n} v_j h_j\right)$$

$$\phi(s) = \frac{e^s}{1+e^s}$$



2) Apply logistic regression to $(h_0, h_1 \cdots h_n)$

Two-step Approach

1) Apply non-linear transforms

$h_j = \theta(\underline{w}_j^T x)$

Training of Neural Networks

Notation : L layers.

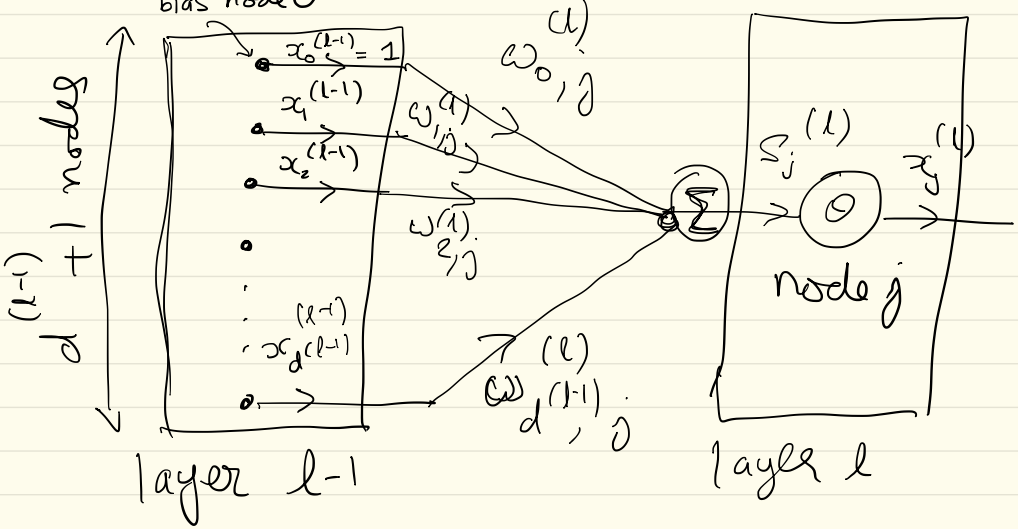
Input layer : $\ell = 0$

Hidden layers : $1 \leq \ell \leq L-1$

Output layer : $\ell = L$

$w_{i,j}^{(\ell)}$: weight connecting node i in layer $(\ell-1)$ & node j in layer ℓ

$d^{(\ell)}$: number of nodes (excluding bias node) in layer ℓ



$x_j^{(l)} \equiv$ Output from node j in layer l

$s_j^{(l)} \equiv$ input into node j in layer l

$$s_j^{(l)} = w_{0,j}^{(l)} + \sum_{i=1}^{d^{(l-1)}} w_{i,j}^{(l)} \cdot x_i^{(l-1)}$$

$$x_j^{(l)} = \Theta\left(s_j^{(l)}\right)$$

Vector Notation

$$\underline{s}^{(l)} = \begin{bmatrix} s_1^{(l)} \\ s_2^{(l)} \\ \vdots \\ s_{d^{(l)}}^{(l)} \end{bmatrix} \qquad \Theta\left(\underline{s}^{(l)}\right) = \begin{bmatrix} \Theta(s_1^{(l)}) \\ \Theta(s_2^{(l)}) \\ \vdots \\ \Theta(s_{d^{(l)}}^{(l)}) \end{bmatrix}$$

$$\underline{x}^{(l)} = \begin{bmatrix} 1 \\ \Theta(\underline{s}^{(l)}) \end{bmatrix}$$

$$\underline{W}^{(l)} = \left\{ w_{i,j}^{(l)} \right\} \quad 0 \le i \le d^{(l-1)}, \quad 1 \le j \le d^{(l)}$$

$$= \begin{bmatrix} w_{0,1}^{(l)} & w_{0,2}^{(l)} & \cdots & w_{0,d^{(l)}}^{(l)} \\ w_{1,1}^{(l)} & w_{1,2}^{(l)} & & w_{1,d^{(l)}}^{(l)} \\ \vdots & & & \\ w_{d^{(l-1)},1}^{(l)} & w_{d^{(l-1)},2}^{(l)} & & w_{d^{(l-1)},d^{(l)}}^{(l)} \end{bmatrix}$$

j^{th} column

Forward Propagation Alg.

j^{th} column of $\underline{W}^{(l)}$ is the collection of weights incident on node j in layer l.

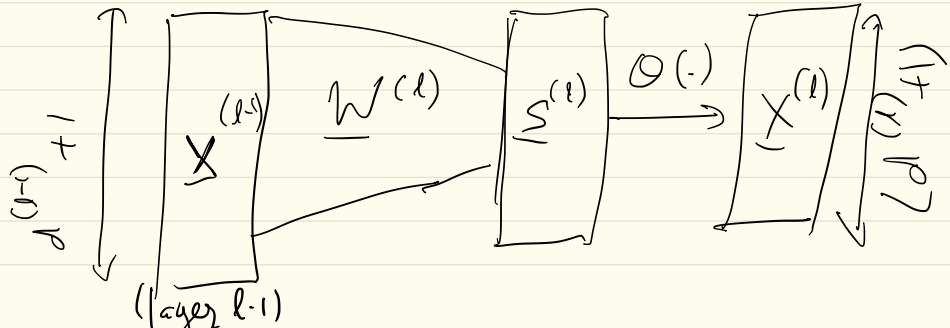
$$\underline{S}^{(l)} = \left[\underline{W}^{(l)}\right]^T \underline{X}^{(l-1)}$$

Input Vector: $\underline{X}^{(0)} = [1, x_1, x_2 \cdots x_d]$

$$\underline{X}^{(0)} \xrightarrow{\underline{W}^{(1)}} \underline{S}^{(1)} \xrightarrow{\theta(\cdot)} \underline{X}^{(1)}$$

$$\underline{X}^{(1)} \xrightarrow{\underline{W}^{(2)}} \underline{S}^{(2)} \xrightarrow{\theta(\cdot)} \underline{X}^{(2)}$$

$$\underline{X}^{(L-1)} \xrightarrow{\underline{W}^{(L)}} \underline{S}^{(L)} \xrightarrow{\theta(\cdot)} \underline{X}^{(L)}$$



Computational Complexity

$\underline{X}^{(l-1)} \xrightarrow{W^{(l)}} \underline{S}^{(l)} \xrightarrow{\theta(\cdot)} \underline{X}^{(l)}$

$=$ total number of edges between layer $(l-1)$ & layer l

$= (d^{(l-1)}+1) \cdot d^{(l)}$

Total Computational complexity

$= \Theta\left(\sum_{l=1}^{L} (d^{(l-1)}+1) d^{(l)} \right)$

$=$ # of edges in the neural network.

Typically a few millions

Introduced: Neural Networks.

Model Parameters

$$\Omega = \left\{ w_{i,j}^{(\ell)} \right\} \quad 1 \le \ell \le L$$

Input: $\underline{x}^{(0)} = [1, x_1, x_2, \ldots x_d]$

Output: $\underline{x}^{(L)}$ (computed using forward pass)

Loss Function

Regression $\longrightarrow$ Squared loss

$\underline{x}^{(L)} \longrightarrow x^{(L)} \in \mathbb{R}$

Loss function $g(x^{(L)}, y) = (x^{(L)} - y)^2$

Classification: log loss

$\underline{x}^{(L)} = (\hat{p}_1, \hat{p}_2, \ldots \hat{p}_c)$ $\quad g(\underline{x}^{(L)}, y=i) = -\log \hat{p}_i$

let $(\underline{x}_n, y_n)$ be a point in the training set

$$e_n(\Omega) = g(\underline{x}_n^{(L)}, y_n)$$

$$e.g. = (x_n^{(L)} - y_n)^2$$

(for squared loss)

select Ω to minimize the training error $= \frac{1}{N} \sum_{n=1}^{N} e_n(\Omega)$

Stochastic Gradient Descent

- In iteration #k, let Ω_k be the current choice of model params.
- Select a training point, say $(\underline{x}_n, y_n)$ at random.
- use forward pass to compute

$$e_n(\Omega_k) = g(\underline{x}_n^{(L)}, y_n)$$

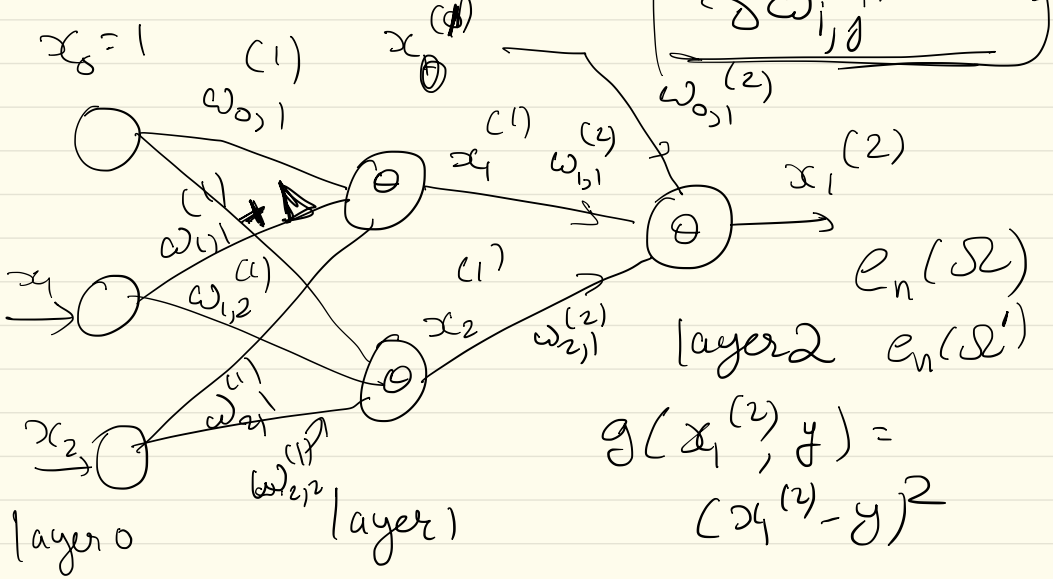
Compute $\boxed{\frac{\partial e_n(\Omega_k)}{\partial \omega_{i,j}^{(l)}}}$ $\forall i,j,l$

$\frac{\partial e_n(\Omega)}{\partial \omega_{1,1}^{(1)}} = \frac{e_n(\Omega') - e_n(\Omega)}{\Delta}$

$$\omega_{i,j}^{(l)} \leftarrow \omega_{i,j}^{(l)} - \epsilon \frac{\partial e_n(\Omega_k)}{\partial \omega_{i,j}^{(l)}}$$

- continue until convergence.

How to Compute $\boxed{\frac{\partial e_n(\Omega_k)}{\partial \omega_{i,j}^{(l)}}}$?



Training for Neural Networks

Last lecture: Stochastic Gradient Descent

In each iteration

- select one training example $(\underline{x}, y)$ at random
- forward pass: output $\underline{x}^L$
- loss: $e(\Omega) = g(\underline{x}^L, y)$
- weight update

$$w_{i,j}^{(l)} \leftarrow w_{i,j}^{(l)} - \epsilon \boxed{\frac{\partial e(\Omega)}{\partial w_{i,j}^{(l)}}}$$

(1986 Rumelhart, Hinton-Williams)

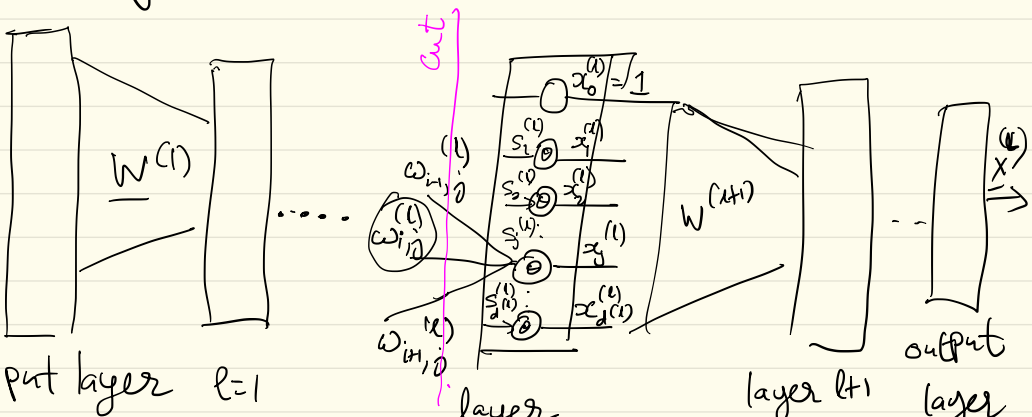
- Continue until convergence.

Back Propagation Algorithm

- exact computation of all derivatives
- Complexity is linear in size of neural network

$$\frac{\partial e}{\partial w_{ij}^{(l)}} = \frac{\partial e}{\partial s_j^{(l)}} \cdot \frac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}}$$

L-layer neural Network



$$e(\Omega) = g(x^{(L)}, y)$$

$$= (x^{(L)} - y)^2$$

Compute $\dfrac{\partial e(\Omega)}{\partial w_{ij}^{(l)}}$ $\forall i, j, l$

Given $s^{(l)} = (s_1^{(l)}, s_2^{(l)}, \ldots s_{d^{(l)}}^{(l)})$ & $W^{(l+1)}, W^{(l+2)}, \ldots W^{(L)}$ the output vector can be computed.

$e(\Omega) \equiv e(s_1^{(l)}, s_2^{(l)}, \ldots \boxed{s_j^{(l)}} \ldots s_{d^{(l)}}^{(l)}, W^{(l+1)}, W^{(l+2)}, \ldots W^{(L)})$

"Composition trick"

depends on $w_{ij}^{(l)}$

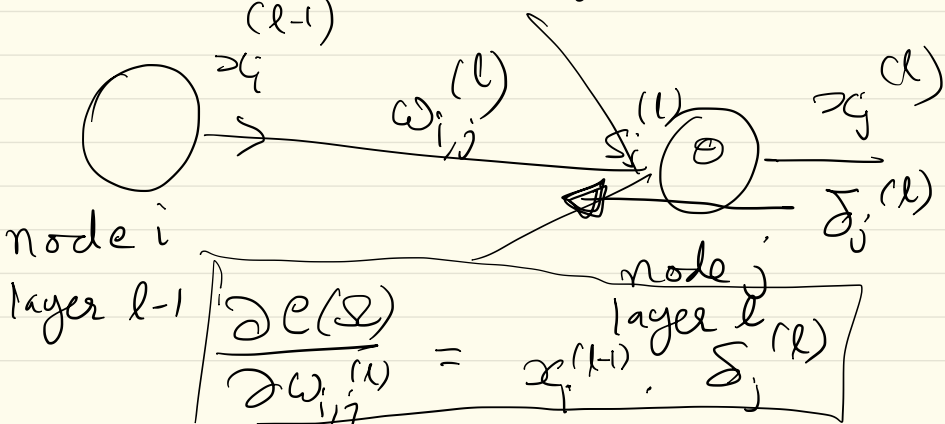
$$\frac{\partial e(s)}{\partial w_{i,j}^{(l)}} = \frac{\partial e(s)}{\partial s_j^{(l)}} \cdot \boxed{\frac{\partial s_j^{(l)}}{\partial w_{i,j}^{(l)}}} \qquad x_i^{(l-1)}$$

$$s_j^{(l)} = \sum_{k=0}^{d^{(l-1)}} x_k^{(l-1)} \cdot w_{k,j}^{(l)}$$

$$\frac{\partial s_j^{(l)}}{\partial w_{i,j}^{(l)}} = \sum_{k=0}^{d^{(l-1)}} \frac{\partial}{\partial w_{i,j}^{(l)}} \left(x_k^{(l-1)} \cdot w_{k,j}^{(l)} \right)$$

$$= x_i^{(l-1)}$$

define $\delta_j^{(l)} = \frac{\partial e(s)}{\partial s_j^{(l)}}$ Backward Message



$$\boxed{\frac{\partial e(s)}{\partial w_{i,j}^{(l)}} = x_i^{(l-1)} \cdot \delta_j^{(l)}}$$

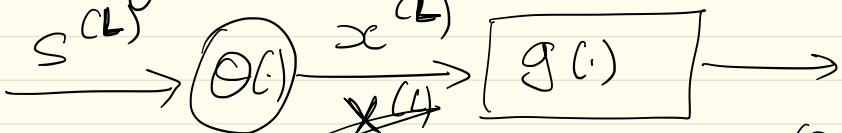
Use backward pass to

Compute $\delta_j^{(l)}$ $j = 1, 2, \ldots d^{(l)}$

$\underline{\delta}^{(L)} \longrightarrow \underline{\delta}^{(L-1)} \longrightarrow \underline{\delta}^{(L-2)} \cdots \underline{\delta}^{(1)} \rightarrow \underline{\delta}^{(1)}$

special case: Regression, Squared error

Output layer: $d^{(L)} = 1$ (X, y)



Compute: $\delta^{(L)} = \dfrac{\partial e(x)}{\partial s^{(L)}}$

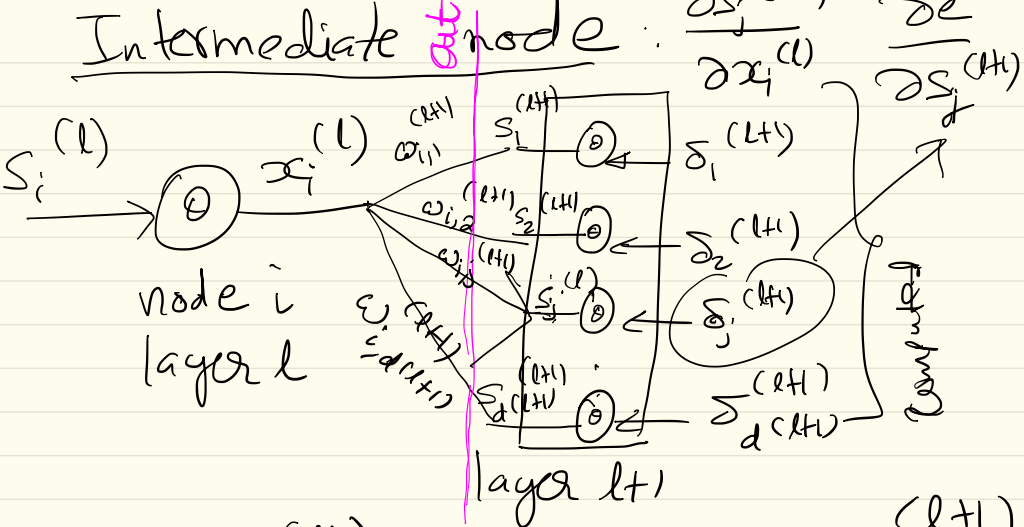
$= \dfrac{\partial e}{\partial x^{(L)}} \cdot \dfrac{\partial x^{(L)}}{\partial s^{(L)}} = 2(x^{(L)} - y) \cdot \theta'(s^{(L)})$

e.g. $\theta(s) = \tanh(s)$

check $\theta'(s) = \dfrac{\partial \theta}{\partial s} = 1 - \theta^2(s)$

$\delta^{(L)}$ at output layer $\longrightarrow$ easy computation.

Intermediate node: $\frac{\partial s_j^{(l+1)}}{\partial x_i^{(l)}}$ $\frac{\partial e}{\partial s_j^{(l+1)}}$



Given $\delta_1^{(l+1)}$, $\delta_2^{(l+1)}$, ... $\delta_{d(l+1)}^{(l+1)}$

& wish to compute $\delta_i^{(l)} = \frac{\partial e(s_2)}{\partial s_i^{(l)}}$

for $i = 1, 2, \ldots d^{(l)}$

$$\frac{\partial e(s_2)}{\partial s_i^{(l)}} = \frac{\partial e(s_2)}{\partial x_i^{(l)}} \cdot \underbrace{\frac{\partial x_i^{(l)}}{\partial s_i^{(l)}}}_{\theta'(s_i^{(l)})}$$

$$x_i^{(l)} = \theta(s_i^{(l)})$$

$$\frac{\partial x_i^{(l)}}{\partial s_i^{(l)}} = \theta'(s_i^{(l)})$$

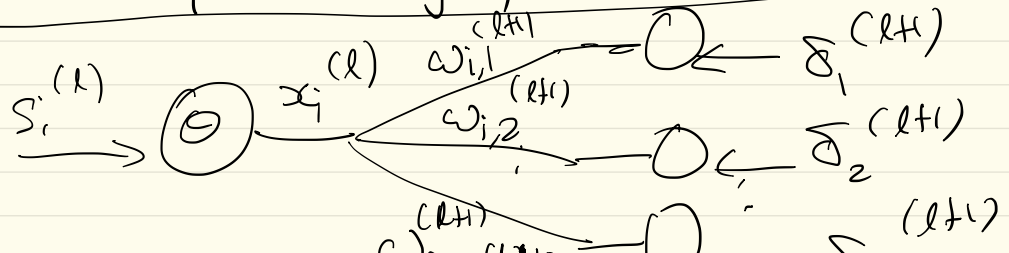
$$e(s_2) = e\Big(\underbrace{s_1^{(l+1)}, s_2^{(l+1)}, \ldots s_{d(l+1)}^{(l+1)}}_{\text{depend on } x_i^{(l)}}, W^{(l+2)}, \ldots W^{(L)}\Big)$$

$$\frac{\partial e(s)}{\partial x_i^{(l)}} = \sum_{j=1}^{d^{(l+1)}} \underbrace{\frac{\partial e(s)}{\partial s_j^{(l+1)}}}_{\delta_j^{(l+1)}} \cdot \frac{\partial s_j^{(l+1)}}{\partial x_i^{(l)}}$$

$$s_j^{(l+1)} = \sum_{k=0}^{d^{(l)}} x_k^{(l)} \cdot w_{k,0}^{(l+1)}$$

$$\frac{\partial s_j^{(l+1)}}{\partial x_i^{(l)}} = w_{i,0}^{(l+1)}$$

$$\boxed{\frac{\partial e(s)}{\partial x_i^{(l)}} = \sum_{j=1}^{d^{(l+1)}} \delta_j^{(l+1)} \cdot w_{i,j}^{(l+1)}}$$



$$\boxed{\delta_i^{(l)} = \frac{\partial e(s)}{\partial x_i^{(l)}} \cdot \theta'(s_i^{(l)})}$$

Backpropagation Algorithm

Input data point: $(\underline{x}, y)$

$\Omega = \{\underline{W}^{(1)}, \underline{W}^{(2)}, \ldots \underline{W}^{(L)}\}$

Output: $\frac{\partial e(\Omega)}{\partial w_{i,j}^{(l)}}$

$\underline{\delta}^{(l)} = (\delta_1^{(l)}, \delta_2^{(l)}, \ldots \delta_{d^{(l)}}^{(l)})^T$

for $l = L, L-1, \ldots 2, 1$.

---

1. Run Forward Pass to compute $\{\underline{s}^{(l)}, \underline{x}^{(l)}\}$ $l = 1, 2, \ldots L$ & $e(\Omega) = g(\underline{x}^{(L)}, y)$

2. $\underline{\delta}^{(L)} = \left[\nabla_{\underline{x}^{(L)}} g(\underline{x}^{(L)}, y)\right] \otimes \theta'(\underline{s}^{(L)})$ (element wise multiplication)

$\underline{s}^{(l)} = \begin{bmatrix} s_1^{(l)} \\ s_2^{(l)} \\ \vdots \\ s_{d^{(l)}}^{(l)} \end{bmatrix}$ $\underline{x}^{(l)} = \begin{bmatrix} 1 \\ \theta(\underline{s}^{(l)}) \end{bmatrix}$ $\theta'(\underline{s}^{(l)}) = \begin{bmatrix} \theta'(s_1^{(l)}) \\ \theta'(s_2^{(l)}) \\ \vdots \\ \theta'(s_{d^{(l)}}^{(l)}) \end{bmatrix}$

3. for $l = L-1, L-2, \ldots 1$:

$$\underline{\delta}^{(l)} = \left[\hat{\underline{W}}^{(l+1)} \cdot \underline{\delta}^{(l+1)} \right] \otimes \theta'(\underline{s}^{(l)})$$

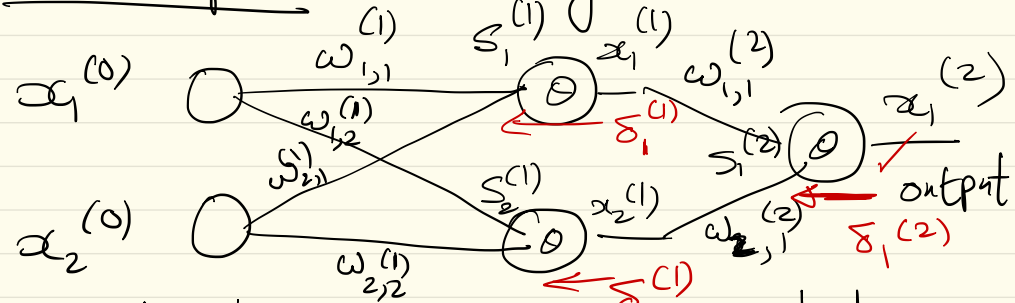
$$\hat{\underline{W}}^{(l+1)} = \begin{bmatrix} \omega_{1,1}^{(l+1)} & \omega_{1,2}^{(l+1)} & \cdots & \omega_{1,d^{(l+1)}}^{(l+1)} \\ \omega_{2,1}^{(l+1)} & \omega_{2,2}^{(l+1)} & \cdots & \omega_{2,d^{(l+1)}}^{(l+1)} \\ \vdots & & & \\ \omega_{d^{(l)},1}^{(l+1)} & \cdots & & \omega_{d^{(l)},d^{(l+1)}}^{(l+1)} \end{bmatrix}$$

end.

$$\underline{\delta}^{(L)} \to \underline{\delta}^{(L-1)} \cdots \to \underline{\delta}^{(l)} \cdots \to \underline{\delta}^{(1)}$$

$$\frac{\partial e(\Omega)}{\partial \omega_{i,j}^{(l)}} = x_i^{(l-1)} \cdot \delta_j^{(l)}$$

Example : 2-layer network



$$e(\Omega) = \left(x_1^{(2)} - y\right)^2$$

Compute $\partial e / \partial w_{i,j}^{(l)}$ (2)

$$\boxed{\frac{\partial e}{\partial x_1^{(0)}} ?}$$

$$\frac{\partial e(\Omega)}{\partial w_{1,1}^{(2)}} = x_1^{(1)} \delta_1$$

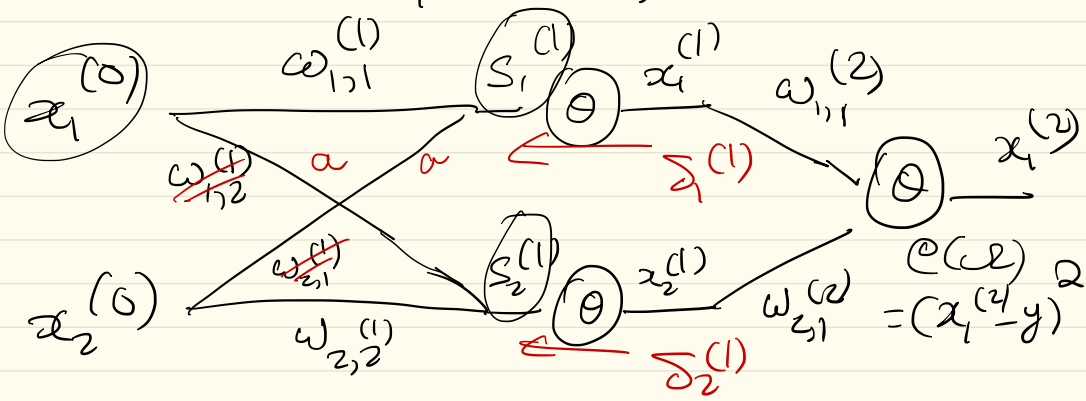
$$\frac{\partial e(\Omega)}{\partial w_{2,1}^{(1)}} = x_2^{(0)} \cdot \delta_1^{(1)}$$

$$\delta_1^{(2)} = \frac{\partial e(\Omega)}{\partial s_1^{(2)}} = \frac{\partial e}{\partial x_1^{(2)}} \cdot \frac{\partial x_1^{(2)}}{\partial s_1^{(2)}}$$

$$= 2\left(x_1^{(2)} - y\right) \cdot \theta'\left(s_1^{(2)}\right)$$

$$\delta_1^{(1)} = \frac{\partial e(\Omega)}{\partial x_1^{(1)}} \cdot \underbrace{\frac{\partial x_1^{(1)}}{\partial s_1^{(1)}}}_{\theta'\left(s_1^{(1)}\right)} = \delta_1^{(2)} \cdot w_{1,1}^{(2)} \cdot \theta'\left(s_1^{(1)}\right)$$

$$\delta_2^{(1)} = \delta_1^{(2)} \cdot \omega_{2,1}^{(2)} \cdot \theta'(s_2^{(1)})$$



$$\frac{\partial e(x)}{\partial x_1^{(0)}} = \frac{\partial e(x)}{\partial s_1^{(1)}} \cdot \frac{\partial s_1^{(1)}}{\partial x_1^{(0)}} + \frac{\partial e(x)}{\partial s_2^{(1)}} \cdot \frac{\partial s_2^{(1)}}{\partial x_1^{(0)}}$$

$$= \delta_1^{(1)} \cdot \omega_{1,1}^{(1)} + \delta_2^{(1)} \cdot \omega_{1,2}^{(1)}$$

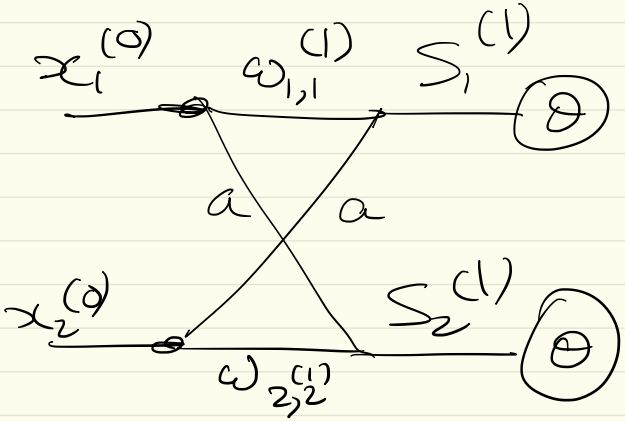
$$\frac{\partial e(x)}{\partial x_2^{(0)}} = \delta_1^{(1)} \cdot \omega_{2,1}^{(1)} + \delta_2^{(1)} \omega_{2,2}^{(1)}$$

Compute $\frac{\partial e(x)}{\partial a}$

$$\boxed{\omega_{1,2}^{(1)} = \omega_{2,1}^{(1)} = a}$$

$$\frac{\partial e(s)}{\partial a} = \frac{\partial e}{\partial s_1^{(1)}} \cdot \frac{\partial s_1^{(1)}}{\partial a} + \frac{\partial e}{\partial s_2^{(1)}} \cdot \frac{\partial s_2^{(1)}}{\partial a}$$

$$\delta_1^{(1)} \cdot x_2^{(0)} + \delta_2^{(1)} \cdot x_1^{(0)}$$



$$s_1^{(1)} = a \cdot x_2^{(0)} + \omega_{1,1}^{(1)} \cdot x_1^{(0)}$$

Implementation of Neural Networks

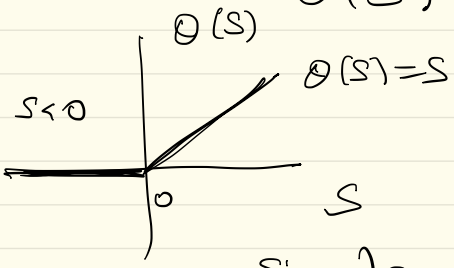
- Mid-term Avg: 61.5/100

1) choice of Activation functions

sigmoid $\theta(s) = \frac{e^s}{1+e^s}$

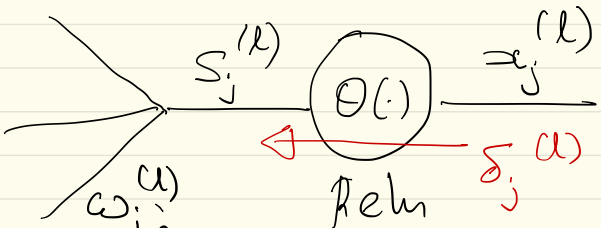
tanh $\theta(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$

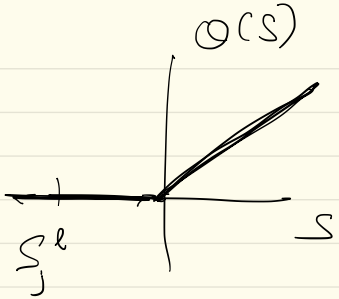
Relu. $\theta(s) = \max\{0, s\}$



- modern NNs leads to 6x improvement in speed of training
- Simple in computation
- Avoids non-linear effects.

Issues: (Dead-neurons)
~ 40% of hidden units are "dead-neurons"





$\longleftrightarrow$ large & negative

$$s_j^{(l)} < 0 \Rightarrow x_j^{(l)} = 0$$

also $\theta'(s_j^{(l)}) = 0$

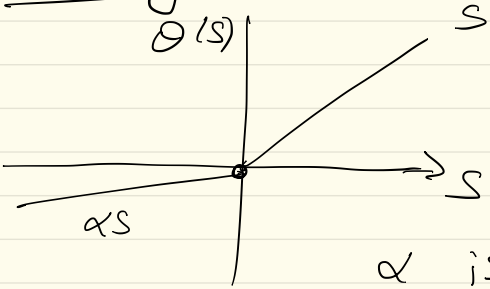
$$\delta_j^{(l)} = 0$$

$$\frac{\partial e}{\partial w_{i,j}^{(l)}} = 0$$

$$w_{i,j}^{(l)} \leftarrow w_{i,j}^{(l)} - \epsilon \underbrace{\frac{\partial e}{\partial w_{i,j}^{(l)}}}_{=0}$$

$\sim 40\%$ are "dead neurons"

Leaky Relu



$$\theta(s) = \begin{cases} s & s>0 \\ \alpha s & s<0 \end{cases}$$

$\alpha \approx 0.1$

α is set to a small constant to carry over -ve inputs forward.

Parametric Relu: α is not fixed but updated during training

---

2) Input Pre-processing

$$\mathcal{D} = \{(\underline{x}_1, y_1), \dots (\underline{x}_N, y_N)\}$$

$\underline{x}_i \in \mathbb{R}^d \quad \underline{x}_i = (x_i(1), x_i(2), \dots x_i(d))$

Zero mean - unit variance normalization

$$\underline{\mu} = \frac{1}{N}\sum_{n=1}^{N} \underline{x}_i \quad \text{(sample mean)}$$

$$\sigma(j) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i(j)-\mu(j))^2} \quad \text{(sample standard deviation)}$$

$j = 1, 2, \dots d$

$$\underline{X}_i(j) \leftarrow \frac{\underline{X}_i(j) - \underline{\mu}(j)}{\sigma(j)}$$

for each $j = 1, 2, \ldots d$ &
each $i = 1, 2, \ldots N$

Batch Norm: Normalization at the output of each hidden layer.

Data-Augmentation

Given $(\underline{X}, y)$ in training set, also include $(F(\underline{X}), y)$

$F \equiv$ scaling, cropping, flipping etc.

Weight Initialization

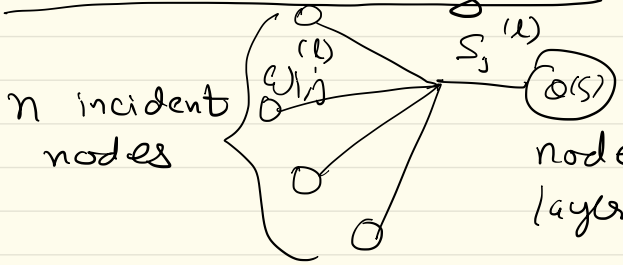
(Section 8.4 in Deep Learning Book)

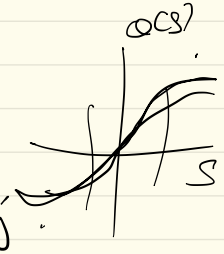
traditionally $w_{i,j}^{(l)} \sim N(0, \sigma^2)$

$\sigma^2 \approx 10^{-3}$

- this works on "Shallow networks"
- for deep networks with many layers, → Vanishing gradients

Xavier Initialization





$w_{i,j}^{(l)} \sim N\left(0, \frac{1}{n}\right)$

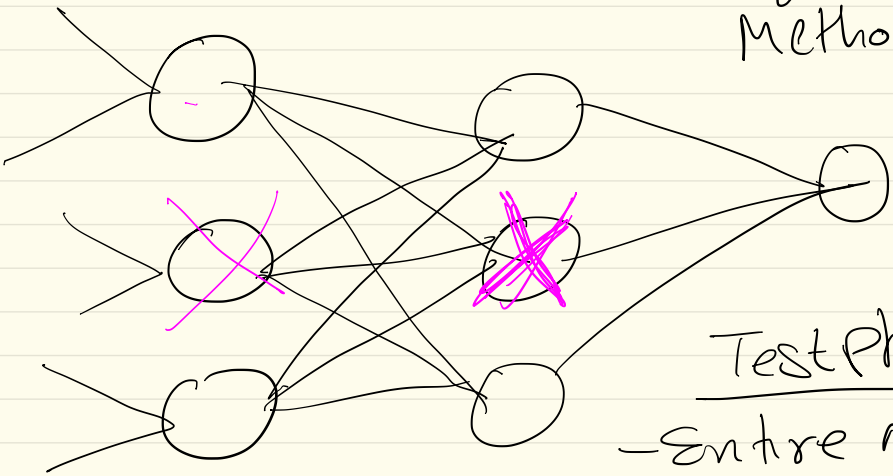
Relu $w_{i,j}^{(l)} \sim N(0, \frac{2}{n})$

$$s_j^{(l)} = \sum_{i=1}^{n} \underbrace{x_i^{(l-1)}}_{\text{unit variance \& independent}} \cdot \underbrace{w_{i,j}^{(l)}}_{\text{variance} \approx \frac{1}{n}}$$

$\text{Variance}(s_j^{(l)}) \approx 1$

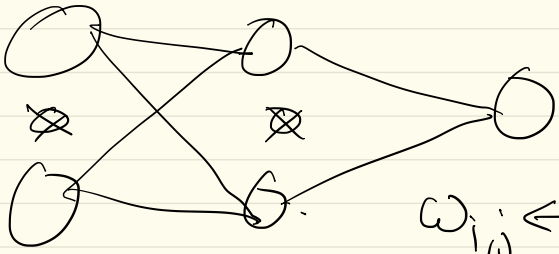
Drop out (D.L. book sec. 7.12)

- with probability P, set output of a given node to zero in the forward pass. "Regularization Method"



Test Phase:

- Entire network but scale the weights



$$w_{i,j} \leftarrow w_{i,j}(1-P)$$

Unsupervised Learning

Training Set

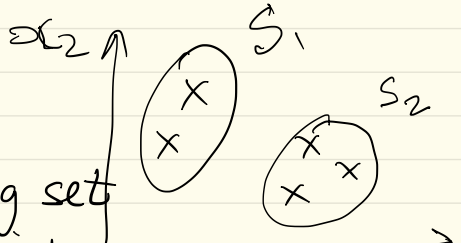
$\mathcal{D} = \{\underline{x}_1, \underline{x}_2, \ldots \underline{x}_N\}$ $\underline{x}_i \in \mathbb{R}^d$

1) clustering
2) Density Estimation
3) Dimensionality Reduction
   PCA → ECE367

} focus

} X

Clustering



$d=2$
$N=5$

Partition training set $\mathcal{D}$ into k disjoint clusters ($K \ll N$): $S_1, S_2, \ldots S_k$ such that elements within each cluster are close to each other

$S_i \subseteq \mathcal{D}$, $S_i \cap S_j = \emptyset$ (disjoint)

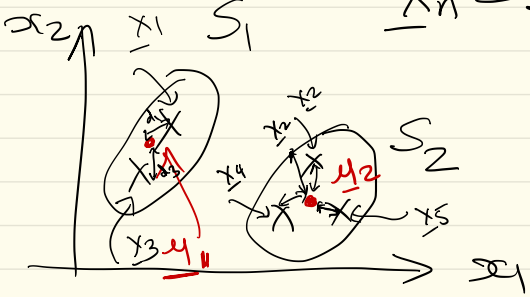
$S_1 \cup S_2 \cup \ldots S_k = \mathcal{D}$

Loss function for clustering.

For cluster S_i let $\underline{\mu}_i$ be the cluster center

Approximation Error for S_j

$$E_j = \sum_{\underline{x}_n \in S_j} \| \underline{x}_n - \underline{\mu}_j \|^2$$



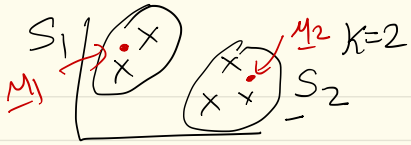
Total Approximation Error

$$E_{in}(S_1, S_2, \ldots S_k, \underline{\mu}_1, \underline{\mu}_2, \ldots \underline{\mu}_k)$$

$$= \sum_{j=1}^{k} E_j = \sum_{j=1}^{k} \sum_{\underline{x}_n \in S_j} \| \underline{x}_n - \underline{\mu}_j \|^2$$

$$= \sum_{n=1}^{N} \| \underline{x}_n - \underline{\mu}(\underline{x}_n) \|^2$$

$\underline{\mu}(\underline{x}_n)$: cluster center associated with $\underline{x}_n$



Optimal Clustering:

Given $\mathcal{D} = \{\underline{x}_1, \underline{x}_2, \ldots \underline{x}_N\}$

minimize $E_{in}(S_1, \ldots S_k, \underline{\mu}_1, \ldots \underline{\mu}_k)$

$S_1 \ldots S_k$

$\underline{\mu}_1, \ldots \underline{\mu}_k$ $= \sum_{n=1}^{N} \| \underline{x}_n - \mu(\underline{x}_n) \|^2$

N-P Hard

Good News: Efficient heuristics that are widely used in practice.

K-means Algorithm

Applications

1) Topic Discovery: N documents Partition into k clusters

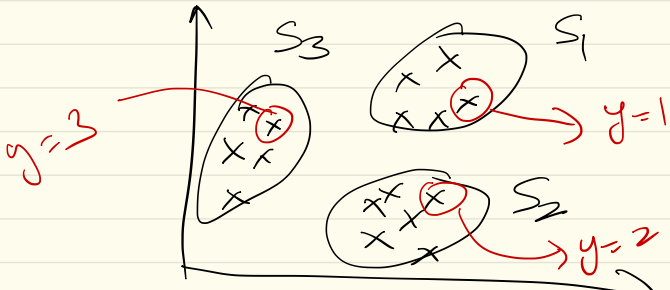
$\underline{x}_i \equiv$ histogram of words in document i

$\underline{\mu}_j \equiv$ mean histogram in cluster j

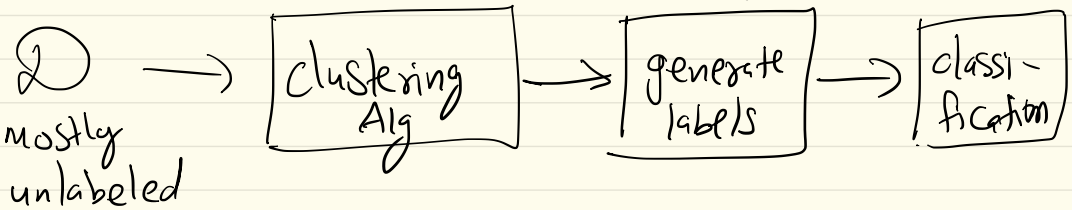
2) Recommendation Systems:

- Cluster Consumers based on some common preferences to provide new recommendations.

3) Semi-Supervised learning.



Most data-vectors have no labels, but a small number have labels



K-Means Algorithm

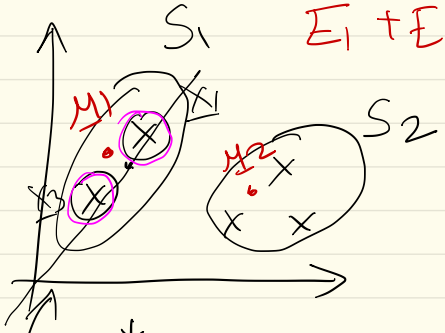
minimize $E_{in}(S_1, \ldots S_k, \underline{\mu}_1, \ldots \underline{\mu}_k)$

$S_1 \ldots S_k$

$\underline{\mu}_1, \ldots \underline{\mu}_k$

$= \sum_{j=1}^{k} E_j$

Problem 1: Given $S_1 \ldots S_k$ find $\underline{\mu}_1, \ldots \underline{\mu}_k$ to minimize E_{in}



select $\underline{\mu}_j$ in order to minimize E_j

$$\underline{\mu}_j^* = \arg\min_{\underline{\mu}_j \in \mathbb{R}^d} \underbrace{\sum_{\underline{x}_n \in S_j} \|\underline{x}_n - \underline{\mu}_j\|^2}_{E_j}$$

Example:

$$E_1(\underline{\mu}_1) = \|\underline{x}_1 - \underline{\mu}_1\|^2 + \|\underline{x}_3 - \underline{\mu}_1\|^2$$

$$\nabla E_1(\mu_1) = 0 \qquad \underline{\mu}_1^* = \frac{1}{2}(\underline{x}_1 + \underline{x}_3)$$

$$E_2(\mu_2) = \|\underline{x}_2 - \underline{\mu}_2\|^2 + \|\underline{x}_4 - \underline{\mu}_2\|^2 + \|\underline{x}_5 - \underline{\mu}_2\|^2$$

$\nabla E_2(\underline{\mu}_2) = 0 \qquad \underline{\mu}_2^* = \frac{1}{3}(\underline{x}_2 + \underline{x}_4 + \underline{x}_5)$

$S_1 = \{\underline{x}_1$
$S_2 = \{$

General Solution

$$\underline{\mu}_j^* = \frac{1}{|S_j|} \sum_{\underline{x}_n \in S_j} \underline{x}_n$$

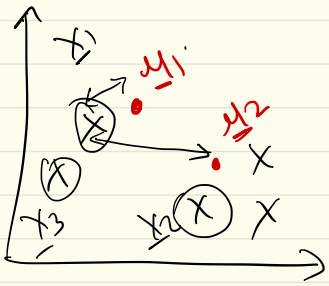
will minimize $E_j(\underline{\mu}_j)$

Problem 2: Given $\underline{\mu}_1, \underline{\mu}_2, \ldots \underline{\mu}_k$

→ find $S_1, S_2, \ldots S_k$ to

minimize $E_{in}(S_1, S_2, \ldots S_k, \underline{\mu}_1, \underline{\mu}_2 \ldots \underline{\mu}_k)$

$$= \sum_{n=1}^{N} \| \underline{x}_n - \underline{\mu}(\underline{x}_n) \|^2$$



For each point $\underline{x}_n$ we simply select the cluster center closest to $\underline{x}_n$ & assign $\underline{x}_n$ to that cluster

K-Means Alg (Lloyd 1956)

1. Initialize $\underline{\mu}_1, \underline{\mu}_2, \ldots \underline{\mu}_k$ in some fashion.
   e.g. (k-means ++ Initialization)
2. Construct $S_1, \ldots S_k$ by solving sub-problem #2
3. update $\underline{\mu}_1, \ldots \underline{\mu}_k$ by solving sub-problem #1
4. Repeat steps 2 & 3 until convergence.
   ($E_{in}(S_1 \ldots S_k, \mu_1 \ldots \mu_k)$ is sufficiently small $\rightarrow$ stop)

Alg. is known to converge to a local minima.

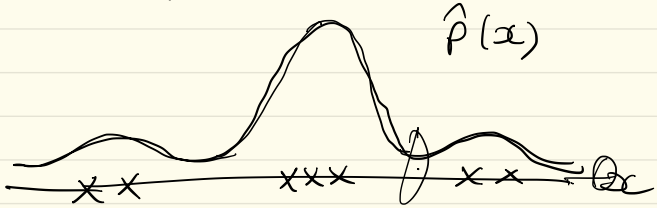
Density Estimation:

$\mathcal{D} = \{ \underline{x}_1, \underline{x}_2, \ldots \underline{x}_N \}$ $\underline{x}_i \overset{iid}{\sim} P(\underline{x})$

$P(\underline{x})$ is unknown to us.

Output $\hat{P}(\underline{x})$ an estimate of $P(\underline{x})$

$d=1$



$N = 7$

Applications: Anomaly detection.

Given Training set $\mathcal{D}$

Compute $\hat{P}(\underline{x})$

given query point $\underline{q}$ if $\hat{P}(\underline{q}) < \epsilon$

declare $\underline{q}$ to be an anomaly.

1) Histogram Method
2) Nearest neighbour
3) Gaussian Mixture model (EM algorithm)

Histogram Method

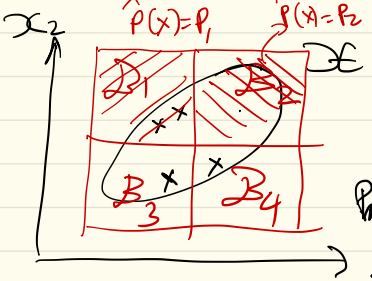
assume that each $\underline{x}$ lies in a bounded region e.g. $\underline{x} \in [0,1]^d$

i.e. $P(\underline{x}) = 0$ if $\underline{x} \notin \mathcal{X} \subseteq \mathbb{R}^d$

Cover $\mathcal{X}$ using uniform hypercubes $\mathcal{B}_1, \mathcal{B}_2, \ldots \mathcal{B}_k$ of constant volume V.

$N = 4$

$d = 2$



each $\mathcal{B}_i$ is a square with area of V.

~~$P_r(\underline{x} \in \mathcal{B}_1) = \frac{2}{4}$~~

$P_r(\underline{x} \in \mathcal{B}_2) = 0$

Step 1: Compute $P_r(\underline{x} \in \mathcal{B}_i) = \frac{N_i}{N}$

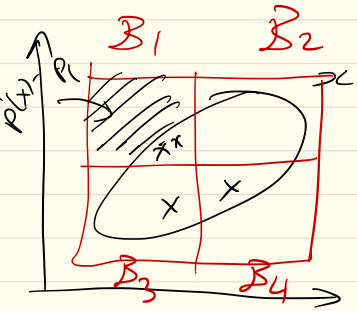
$i = 1, 2, 3, 4$

$N_i \equiv$ total # of points from $\mathcal{D}$ in $\mathcal{B}_i$

$N = N_1 + N_2 + \ldots N_k \equiv$ total # of training pts.

$P_r(\underline{x} \in \mathcal{B}_1) = \frac{2}{4}$ $\quad P_r(\underline{x} \in \mathcal{B}_2) = 0$

$P_r(\underline{x} \in \mathcal{B}_3) = P_r(\underline{x} \in \mathcal{B}_4) = \frac{1}{4}$



$$\boxed{Pr(\underline{x} \in B_1) = \frac{2}{4}}$$

$$Pr(\underline{x} \in B_2) = 0$$

$$Pr(\underline{x} \in B_3) = Pr(\underline{x} \in B_4) = \frac{1}{4}$$

Assume density function $\hat{P}(\underline{x})$ is constant on each B_i

i.e $\hat{P}(\underline{x}) = \begin{cases} P_1 & \forall \underline{x} \in B_1 \\ P_2 & \forall \underline{x} \in B_2 \\ P_3 & \forall \underline{x} \in B_3 \\ P_4 & \forall \underline{x} \in B_4 \end{cases}$

$P_2 = 0$, $P_3 = P_4 = \frac{1}{4V}$

find P_1, P_2, P_3, P_4

$$\int_{\underline{x} \in B_1} \underbrace{\hat{P}(\underline{x})}_{=P_1} d\underline{x} = Pr(\underline{x} \in B_1) = \frac{2}{4}$$

$$= P_1 \boxed{\int_{\underline{x} \in B_1} 1 \, d\underline{x}} \Rightarrow \text{area of } B_1 = V$$

$$\rightarrow \boxed{P_1 = \frac{2}{4V}}$$

Density Estimation

$\mathcal{D} = \{\underline{x}_1, \underline{x}_2, \ldots \underline{x}_N\}$ $P(x) = q$

$\underline{x}_i \in \mathbb{R}^d$ $\underline{x}_i \overset{iid}{\sim} P(\underline{x})$

Output an estimate, say $\hat{P}(\underline{x})$ of $P(\underline{x})$

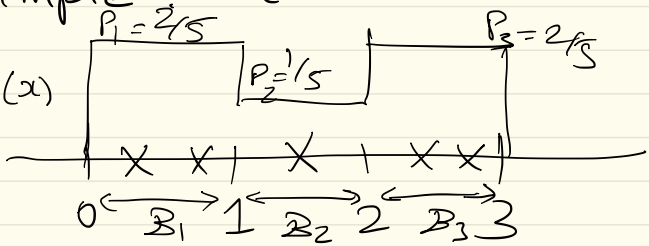
- Histogram Method

$$Pr(\underline{x} \in \mathcal{B}_i) = \frac{N_i}{N}$$

$$P_i = \frac{N_{N_i}}{N \cdot V}$$

V = Volume $(\mathcal{B}_i)$

Example: $d = 1$ $N = 5$



Assume: $x \notin \mathcal{X} = [0, 3]$ then $P(x) = 0$

$\mathcal{B}_1 = [0, 1]$ $\mathcal{B}_2 = [1, 2]$ $\mathcal{B}_3 = [2, 3]$

$Pr(x \in \mathcal{B}_1) = \frac{2}{5}$ $Pr(x \in \mathcal{B}_2) = \frac{1}{5}$ $Pr(x \in \mathcal{B}_3) = \frac{2}{5}$

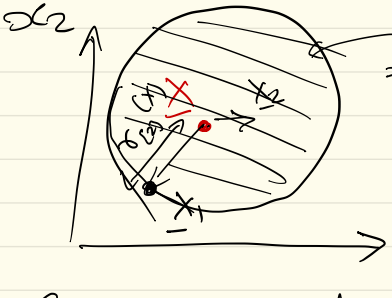
$Vol(\mathcal{B}_i) = 1$ $i = 1, 2, 3$

$\hat{P}(x) = \frac{2}{5}$ $x \in \mathcal{B}_1$ $\hat{P}(x) = \frac{1}{5}$ $x \in \mathcal{B}_2$ $\hat{P}(x) = \frac{2}{5}$ $x \in \mathcal{B}_3$

2) Nearest Neighbour Estimation

$\mathcal{D} = \{\underline{x}_1, \underline{x}_2, \ldots \underline{x}_N\}$ &

estimate $\hat{p}(\underline{x})$ for any given $\underline{x} \in \mathbb{R}^d$



area $= \pi r_k^2$
$= V_k(\underline{x})$

$d = 2$

Given $\underline{x}$
output $\hat{p}(\underline{x})$
$k = 2$

$k > 1$

Given $\underline{x}$ let

$\underline{x}_{[1]}, \underline{x}_{[2]}, \ldots \underline{x}_{[k]}$ be the k nearest neighbours in $\mathcal{D}$ to $\underline{x}$

$r_k(\underline{x}) = \|\underline{x} - \underline{x}_{[k]}\|$ & $V_k(\underline{x})$ be the volume of a sphere in $\mathbb{R}^d$ with radius $r_k(\underline{x})$

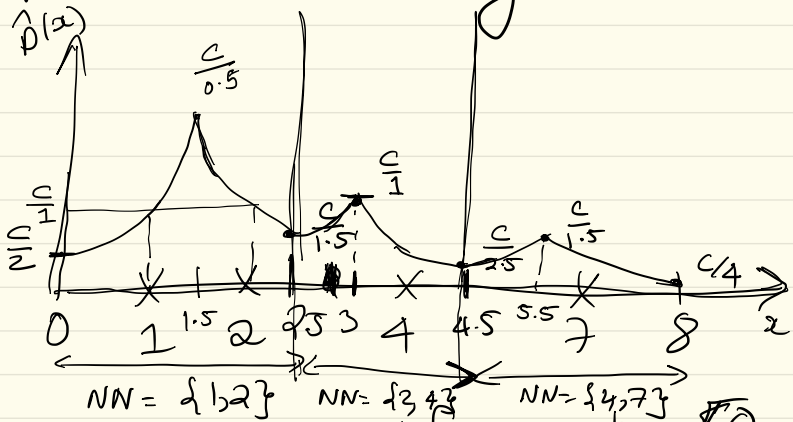
$$\hat{p}(\underline{x}) \propto \frac{1}{V_k(\underline{x})} \Rightarrow \hat{p}(\underline{x}) = \frac{c}{V_k(\underline{x})}$$

where $c \equiv$ normalizing constant

$d=1$

Example: $\mathcal{D} = \{x_1=1, x_2=2, x_3=4, x_4=7\}$

$k=2$ NN-density Estimation



$p(x) = 0$ if $x \notin [0, 8]$

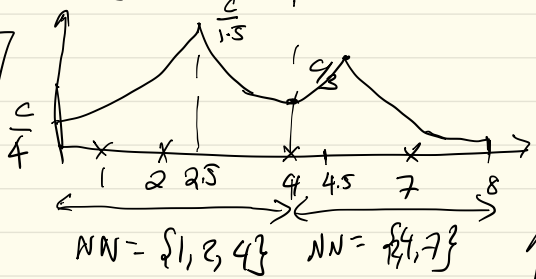
$x \in [0, 2.5]$: $NN = \{1, 2\}$

$x \in [2.5, 4.5]$ $NN = \{2, 4\}$

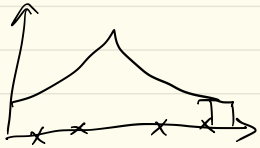
$x \in [4.5, 8]$: $NN = \{4, 7\}$

$k=1$?

$k=3$



$\mathcal{D} = \{1, 2, 4, 7\}$



$$K = E[(\underline{X} - \underline{\mu})(\underline{X} - \underline{\mu})^T]$$

So far ...

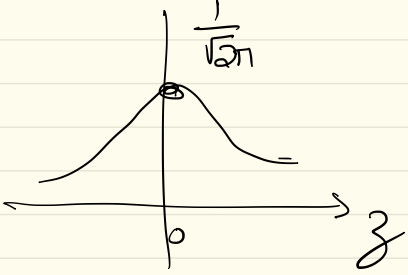
1) Histogram Method
2) K- Nearest Neighbour.

$$\underline{X} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

$$\underline{\mu} = \begin{bmatrix} E[X_1] \\ E[X_2] \end{bmatrix}$$

- Gaussian Mixture Models.

$$\phi(z) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{z^2}{2}\right\}$$



mean : 0

Variance : 1

$$Z \sim N(0, 1)$$

Two Jointly Gaussian Random Variables.

$\underline{X}_1, \underline{X}_2$: Jointly Gaussian

Covariance Matrix

Density Estimation

sec 6.3.3 & sec 6.4

Training Set

$\mathcal{D} = \{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_N\}$

$\underline{x}_i \in \mathbb{R}^d$ $\quad \underline{x}_i \overset{iid}{\sim} P(\underline{x})$

output $\hat{P}(\underline{x})$ an estimate of $P(\underline{x})$

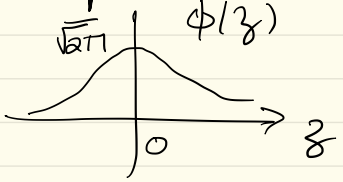
Previous Methods

1) Histogram 2) K-Nearest Neighbour

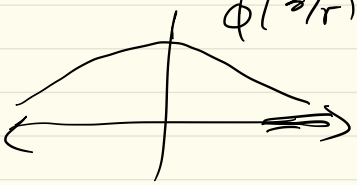
Parzen Window Estimation using Gaussian kernels.

Gaussian Density function $Z \sim N(0,1)$

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2}$$

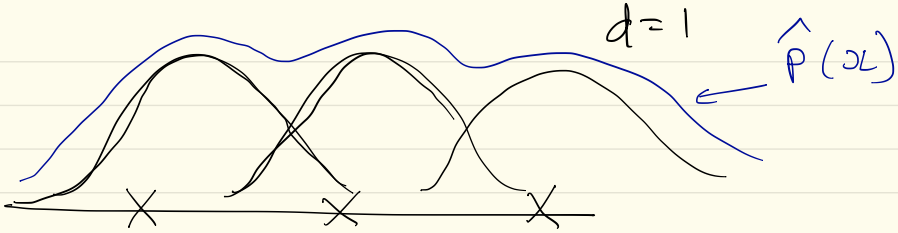


$\phi\left(\frac{z}{r}\right)$



"width = r" Gaussian Kernal

$r > 0$

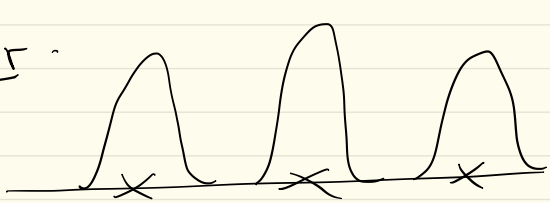


$$\hat{p}(x) = \frac{1}{K} \sum_{n=1}^{N} \phi\left(\frac{x - x_i}{\gamma}\right)$$

$K \equiv$ normalization constant

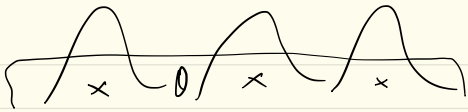
$\gamma \equiv$ kernel width.

small γ.



large γ

Choice of γ can be important in practice. : Validation to select "γ"



Validation set

$$\mathcal{V} = \{\hat{x}_1, \hat{x}_2, \ldots \hat{x}_M\}$$

not included in $\mathcal{D}$.

for different values of γ,

compute $$\hat{p}_\gamma(x) = \frac{1}{K}\sum_{n=1}^{N} \phi\left(\frac{x - x_i}{\gamma}\right)$$

evaluate $\hat{p}_\gamma(x)$ on $\mathcal{V}$ for each choice of γ.

Now Evaluate

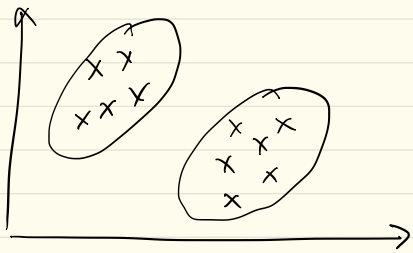
$$-\log \hat{p}_\gamma(\hat{x}_1, \hat{x}_2 \ldots \hat{x}_M) = -\log \prod_{m=1}^{M} \hat{p}_\gamma(\hat{x}_m)$$

$$= -\sum_{m=1}^{M} \log \hat{p}_\gamma(\hat{x}_m)$$

Loss function $= -\sum_{m=1}^{M} \log \hat{p}_\gamma(\hat{x}_m)$

select γ to minimize this loss function

Gaussian Mixture Model



$d = 2$

$K = 2$ Gaussians density functions.

Multivariate Gaussian Density function

Univariate Gaussian: $Z \sim N(\mu, \sigma^2)$

$$f_Z(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(z-\mu)^2}{2\sigma^2}\right\}$$

$\mu = E[Z]$ (mean)

$\sigma^2 = E[(z-\mu)^2]$ (variance)

Bivariate Gaussian Density function

$X_1 \sim N(m_1, \sigma_1^2)$ $\quad X_2 \sim N(m_2, \sigma_2^2)$

$E[X_1] = m_1$ $\quad E[X_2] = m_2$

$\text{var}(X_1) = \sigma_1^2$ $\quad \text{var}(X_2) = \sigma_2^2$

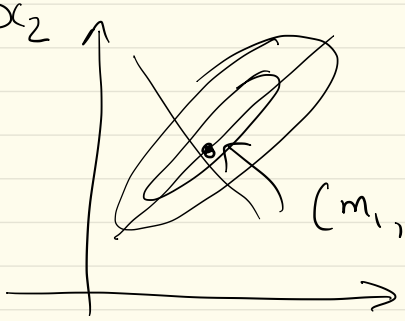
Covariance:

$$\sigma_{12} = E\left[(X_1 - m_1)(X_2 - m_2)\right]$$

auto correlation coefficient $\rho = \dfrac{\sigma_{12}}{\sigma_1 \cdot \sigma_2}$

$$-1 \le \rho \le 1$$

$$f_{X_1, X_2}(x_1, x_2) = \frac{1}{2\pi \sigma_1 \sigma_2 \sqrt{1-\rho^2}} \exp\left\{ \left[\frac{(x_1 - m_1)^2}{\sigma_1^2} + \frac{(x_2 - m_2)^2}{\sigma_2^2} - 2\rho \left(\frac{x_1 - m_1}{\sigma_1}\right)\left(\frac{x_2 - m_2}{\sigma_2}\right) \right] \Big/ 2(1-\rho^2) \right\}$$



$f(x_1, x_2)$ = Constant

= constant

$$\left(\frac{x_1 - m_1}{\sigma_1}\right)^2 + \left(\frac{x_2 - m_2}{\sigma_2}\right)^2 - 2\rho \left(\frac{x_1 - m_1}{\sigma_1}\right)\left(\frac{x_2 - m_2}{\sigma_2}\right)$$

Multivariate Gaussian Density

$\underline{X} = [x_1, x_2, \ldots x_d]^T$

x_i is Gaussian $\quad \underline{\mu} = [m_1, m_2 \ldots m_d]^T$

$E[x_i] = m_i$

Covariance Matrix

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2d} \\ \vdots & & & \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_{dd} \end{bmatrix}$$

$$\sigma_{ij} = E[(x_i - m_i)(x_j - m_j)]$$

$$= \sigma_{ji}$$

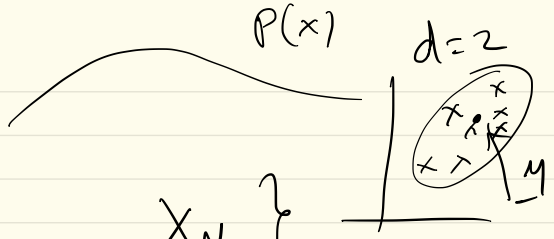
$\underline{X} \sim N(\underline{\mu}, \Sigma)$

Density function

$$f(X) = \frac{1}{(2\pi)^{d/2}\sqrt{|\Sigma|}} \exp\left\{ - \frac{(\underline{x} - \underline{\mu})^T \Sigma^{-1} (\underline{x} - \underline{\mu})}{2} \right\}$$

(Pg 240 - Leon-Garcia Text)

Example:



$\mathcal{D} = \{\underline{x}_1, \underline{x}_2, \ldots \underline{x}_N\}$

$\underline{x}_i \overset{iid}{\sim} p(\underline{x})$ it is known that $p(\underline{x})$ is Gaussian.

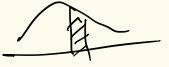
$p(\underline{x})$ is of the form $\mathcal{N}(\underline{\mu}, \Sigma)$

We only need to estimate mean μ, & covariance matrix Σ from $\mathcal{D}$.

$$\hat{\underline{\mu}} = \frac{1}{N} \sum_{n=1}^{N} \underline{x}_n$$ (Empirical mean)

$$\hat{\Sigma} = \frac{1}{N} \sum_{n=1}^{N} (\underline{x}_n - \mu)(\underline{x}_n - \mu)^T$$

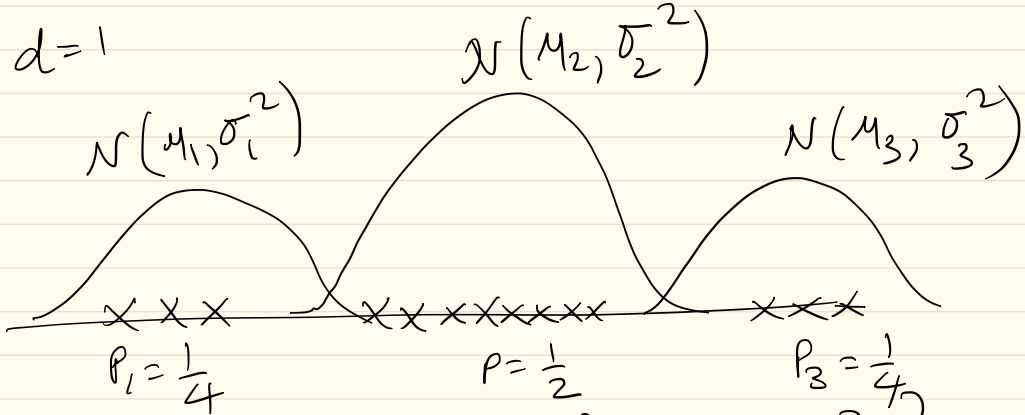
$$\Sigma = E[(\underline{x} - \underline{\mu})(\underline{x} - \underline{\mu})^T]$$



Given $P(x)$

Gaussian Mixture Model

$d=1$



$$\hat{P}(x) = \frac{1}{4} \frac{1}{\sqrt{2\pi}\sigma_1^2} \exp\left\{ - \frac{(x-\mu_1)^2}{2\sigma_1^2} \right\}$$

$$+ \frac{1}{2} \cdot \frac{1}{\sqrt{2\pi}\sigma_2^2} \exp\left\{ - \frac{(x-\mu_2)^2}{2\sigma_2^2} \right\}$$

$$+ \frac{1}{4} \frac{1}{\sqrt{2\pi}\sigma_3^2} \exp\left\{ - \frac{(x-\mu_3)^2}{2\sigma_3^2} \right\}$$

$$= \frac{1}{4} \mathcal{N}(x; \mu_1, \sigma_1^2) + \frac{1}{2} \mathcal{N}(x; \mu_2, \sigma_2^2) + \frac{1}{4} \mathcal{N}(x; \mu_3, \sigma_3^2)$$

Gaussian Mixture Model

Multivariate Gaussian

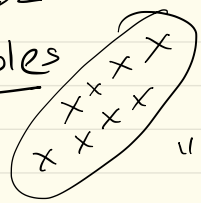
$$P_j(\underline{X}) = \mathcal{N}(\underline{X}; \underline{\mu}_j, \underline{\Sigma}_j)$$

$\underline{\mu}_j$: mean vector

$\underline{\Sigma}_j$: covariance matrix

$d = 2$

samples



"single cluster"

GMM

$$\hat{P}(\underline{X}) = \omega_1 \cdot P_1(\underline{X}) + \omega_2 \cdot P_2(\underline{X}) + \dots \omega_K P_K(\underline{X})$$

$$P_j(\underline{X}) = \mathcal{N}(\underline{X}; \underline{\mu}_j, \underline{\Sigma}_j)$$

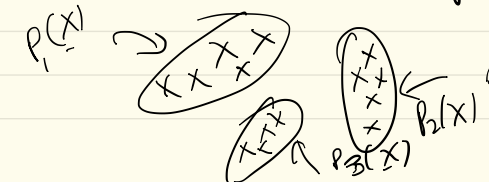
$\omega_1, \omega_2, \dots \omega_K \equiv$ weights associated with each Gaussian

$\omega_i \geq 0$, $\sum_{i=1}^{K} \omega_i = 1$

samples of $\hat{P}(\underline{X})$

$d = 2$

$K = 3$



Given

$$\mathcal{D} = \{\underline{x}_1, \underline{x}_2, \ldots \underline{x}_N\}, \quad \underline{x}_i \in \mathbb{R}^d$$

Estimate Parameters of Gaussian Mixture Model.

$$\hat{P}_{\Omega}(\underline{x}) = \sum_{j=1}^{K} \omega_j \cdot \mathcal{N}(\underline{x}; \underline{\mu}_j, \Sigma_j)$$

Model Parameters:

$$\Omega = \{\omega_j, \underline{\mu}_j, \Sigma_j\}_{j=1}^{K}$$

find Ω that is the "best-fit" for $\mathcal{D}$.

Maximum-Likelihood Solution

find Ω that maximizes

$$\hat{P}_{\Omega}(\underline{x}_1, \underline{x}_2, \ldots \underline{x}_N) = \prod_{n=1}^{N} \hat{P}_{\Omega}(\underline{x}_n)$$

$$\Omega^* = \text{argmax} \prod_{n=1}^{N} \hat{P}_{\Omega}(\underline{x}_n)$$

$$\Omega^* = \arg\max_{\Omega} \log \prod_{n=1}^{N} \hat{P}_{\Omega}(\underline{x}_n)$$

$$= \arg\max_{\Omega} \sum_{n=1}^{N} \log \hat{P}_{\Omega}(\underline{x}_n)$$

$$= \arg\min_{\Omega} \frac{1}{N} \sum_{n=1}^{N} \underbrace{-\log \hat{P}_{\Omega}(\underline{x}_n)}_{e_n(\Omega)}$$

$$e_n(\Omega) = -\log \hat{P}_{\Omega}(\underline{x}_n)$$

$$= -\log \left(\sum_{j=1}^{K} \omega_j \, \mathcal{N}(\underline{x}_n; \underline{\mu}_j, \underline{\Sigma}_j) \right)$$

$$\Omega^* = \arg\min_{\Omega} \frac{1}{N} \sum_{n=1}^{N} e_n(\Omega)$$

Gradient Descent for $\Omega = \{ \omega_j, \underline{\mu}_j, \underline{\Sigma}_j \}_{j=1}^{K}$ $\quad \sum \omega_j = 1, \ \omega_j \geq 0$

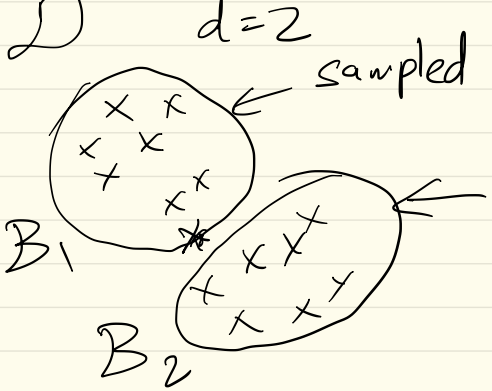
Expectation - Maximization Alg

$$\Omega^* = \underset{\Omega}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} e_n(\Omega)$$

Auxiliary Variables

$$\mathcal{B}_1, \mathcal{B}_2, \ldots \mathcal{B}_K \subseteq \mathcal{D}$$

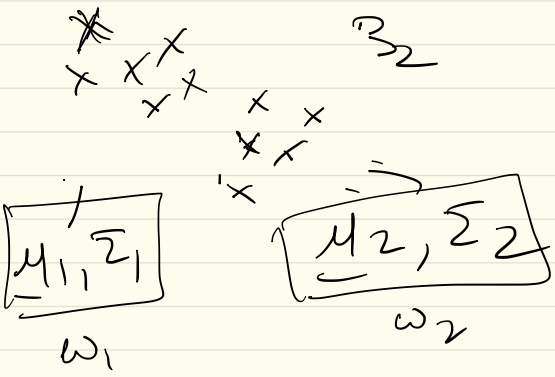
$\mathcal{B}_i \equiv$ set of points in $\mathcal{D}$ associated with cluster # i and hence sampled from $P_i(\underline{x})$



Sub-problem 1:

Suppose $B_1, B_2, \ldots B_K$ are given 1-e for each $\underline{x}_n \in \mathcal{D}$ we know the cluster associated with it

find $\Omega = \{\omega_j, \underline{\mu}_j, \Sigma_j\}_{j=1}^{K}$



Use Empirical Estimates

$N_j \equiv$ # of points in B_j

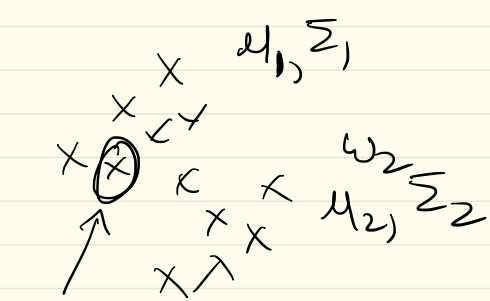
$$\omega_j = \frac{N_j}{N} \quad / \quad \underline{\mu}_j = \frac{1}{N_j} \sum_{\underline{x}_n \in B_j} \underline{x}_n$$

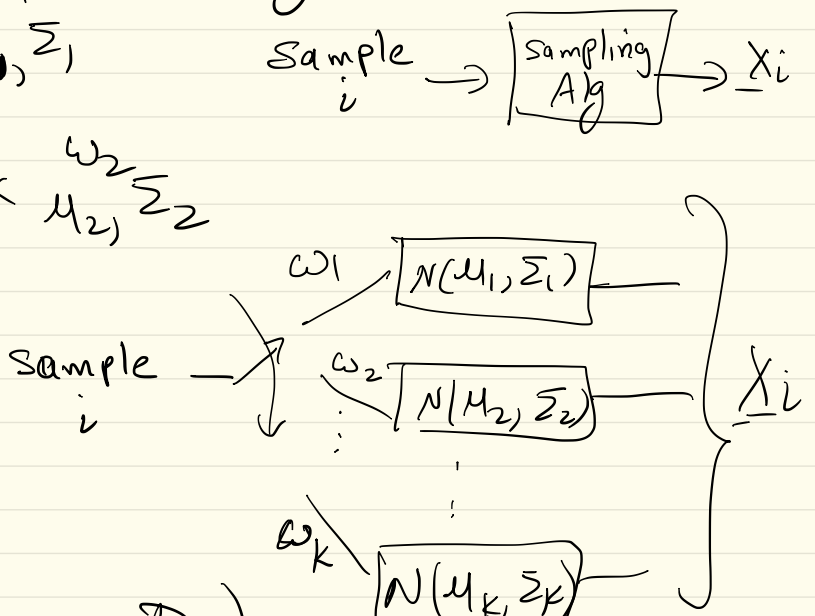
$$\Sigma_j = \frac{1}{N_j} \sum_{\underline{x}_n \in B_j} (\underline{x}_n - \underline{\mu}_j)(\underline{x}_n - \underline{\mu}_j)^T$$

Sub-problem #2

Suppose $\Omega = \{\omega_j, \underline{\mu}_j, \Sigma_j\}_{j=1}^{K}$ is known. find $\mathcal{R}_1, \ldots \mathcal{R}_K$

i.e for each $\underline{x}_n$, determine the cluster ω_i it belongs to.





Compute

$Pr(\underline{x}_n \in \mathcal{R}_j)$ for each $j = 1, 2, \ldots K$

& select the $\mathcal{R}_j$ with largest value

Gaussian Mixture Model & EM Algorithm

$\mathcal{D} = \{\underline{x}_1, \underline{x}_2, \cdots \underline{x}_N\}$, $\underline{x}_i \in \mathbb{R}^d$

$$\hat{P}_\Omega(\underline{x}) = \sum_{j=1}^{k} \omega_j \cdot \underbrace{N(\underline{x}; \underline{\mu}_j, \Sigma_j)}_{\text{multivariate Gaussian Density}}$$

$\omega_j \geq 0$

$\sum_{j=1}^{k} \omega_j = 1$

Given $\mathcal{D}$: find $\hat{P}_\Omega(\underline{x})$ that is "best-fit" for $\mathcal{D}$.

Loss-Function

Maximum-Likelihood Solution

$$E_{in}(\Omega) = \frac{1}{N} \sum_{n=1}^{N} -\log \hat{P}_\Omega(\underline{x}_n)$$

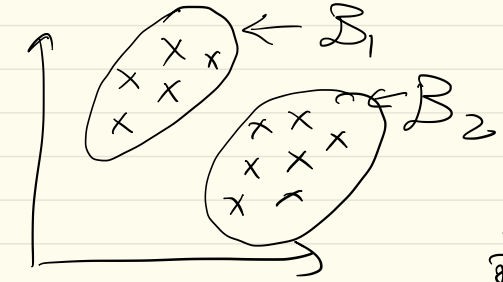
$\Omega = \{\omega_j, \underline{\mu}_j, \Sigma_j\}_{j=1}^{k}$ $\times$ SGD. $\times$ EM Algorithm.

Expectation-Maximization Algorithm

Auxiliary Variables

$B_1, B_2, \ldots B_k \subseteq D$

B_i = Set of points in D belonging to "cluster = i" and hence sampled from $P_i(\underline{x}) = N(x; \underline{\mu}_i, \Sigma_i)$



Assume:

B_i are disjoint

$\bigcup_{i=1}^{k} B_i = D$.

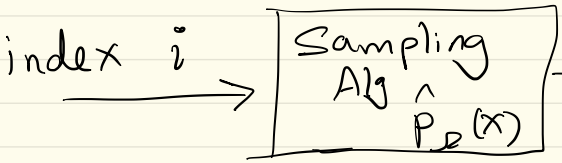
$$f(x_i) = \hat{P}_\Omega(x_i)$$

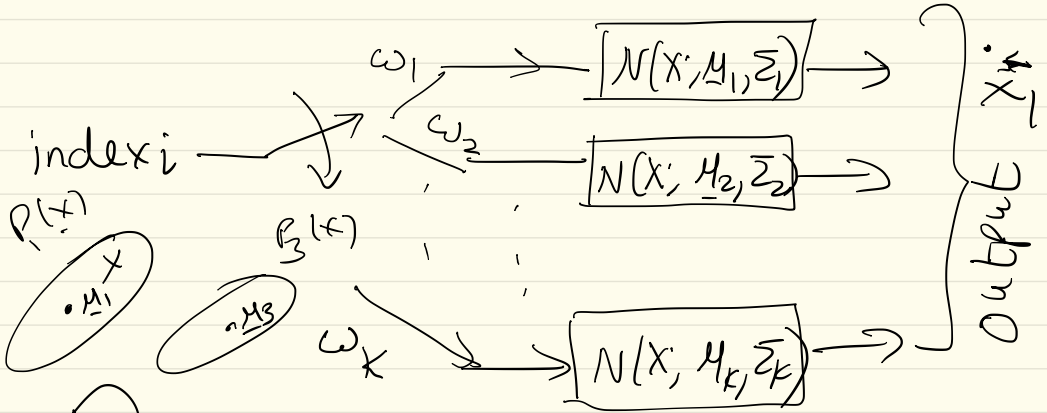
↑ density of x_i

Thought Experiment

Assuming each $x_i \in \mathcal{D}$ is sampled independently from

$$\hat{P}_\Omega(x) = \sum_{j=1}^{k} \omega_j \, \mathcal{N}(x; \mu_j, \Sigma_j)$$





$$\Pr(x_i \text{ belongs to cluster } j) = \omega_j$$

$$\Pr(x_i \in \mathcal{B}_j) = \omega_j$$

$$f(x_i \mid x_i \in \mathcal{B}_j) = \mathcal{N}(x_i; \mu_j, \Sigma_j)$$

Sub-Problem 1

Suppose $\mathcal{B}_1, \mathcal{B}_2, \ldots \mathcal{B}_K$ are given & Identify

$$\Omega = \{\omega_j, \underline{\mu}_j, \Sigma_j\}_{j=1}^{K}$$

Last-Lecture

$$\omega_j = \frac{N_j}{N}$$

N_j = # of Points in $\mathcal{B}_j$

$$\underline{\mu}_j = \frac{1}{N_j} \sum_{\underline{x}_n \in \mathcal{B}_j} \underline{x}_n$$

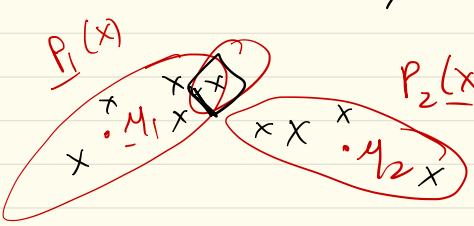
$$\Sigma_j = \frac{1}{N_j} \sum_{\underline{x}_n \in \mathcal{B}_j} (\underline{x}_n - \underline{\mu}_j)(\underline{x}_n - \underline{\mu}_j)^T$$

(Empirical Estimates)

Sub-problem 2

Given $\Omega = \{\omega_j, \underline{\mu}_j, \Sigma_j\}_{j=1}^{K}$

find $\mathcal{Z}_1, \mathcal{Z}_2, \ldots \mathcal{Z}_K$



$P_1(\underline{x}) = \mathcal{N}(\underline{x}; \underline{\mu}_1, \Sigma_1)$

$P_2(\underline{x}) = \mathcal{N}(\underline{x}; \underline{\mu}_2, \Sigma_2)$

$\vdots$

$P_K(\underline{x}) = \mathcal{N}(\underline{x}; \underline{\mu}_K, \Sigma_K)$

Equivalently

Given any $\underline{x}_n \in \mathcal{D}$ find the "most-likely cluster" associated with $\underline{x}_n$ "Compute"

Posterior Probability

$$\Pr(\text{cluster} = j \mid \underline{x}_n)$$

for each $j = 1, 2, \ldots K$

~~[0.8, 0.2]~~

$\underline{x}_n \in \mathcal{Z}_j$

assign $\underline{x}_n \in \mathcal{Z}_{j^*}$ if $j^* = \arg\max_{j \in \{1,2,\ldots K\}} \Pr(\text{cluster} = j \mid \underline{x}_n)$

Baye's Rule

$$Pr(\text{cluster} = j \mid \underline{x}_n)$$

$$= \frac{\hat{p}(\underline{x}_n \mid \text{cluster} = j) \cdot Pr(\text{cluster} = j)}{\hat{p}(\underline{x}_n)}$$

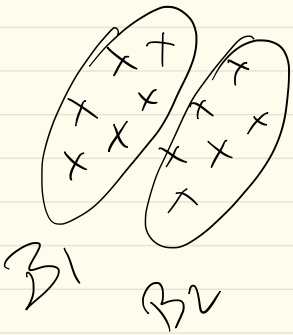
$$= \frac{N(\underline{x}_n; \underline{\mu}_j, \Sigma_j) \cdot \omega_j}{\underbrace{\hat{p}_{GZ}(\underline{x}_n) = \sum_{j=1}^{K} \omega_j N(\underline{x}_n; \underline{\mu}_j, \Sigma_j)}_{\text{independent of } j}}$$

$$j^* = \underset{j \in \{1,2,\ldots K\}}{\arg\max} \; \omega_j \cdot N(\underline{x}_n; \underline{\mu}_j, \Sigma_j)$$

$\underline{x}_n \in \mathcal{D}$.

EM - Algorithm

- Start with some arbitrary choice of $B_1, B_2, \ldots B_K$



Step 1:
Given $B_1 - B_K$
use sub-problem #1
to compute

$$\Omega = \{\omega_j, \mu_j, \Sigma_j\}_{j=1}^{K}$$

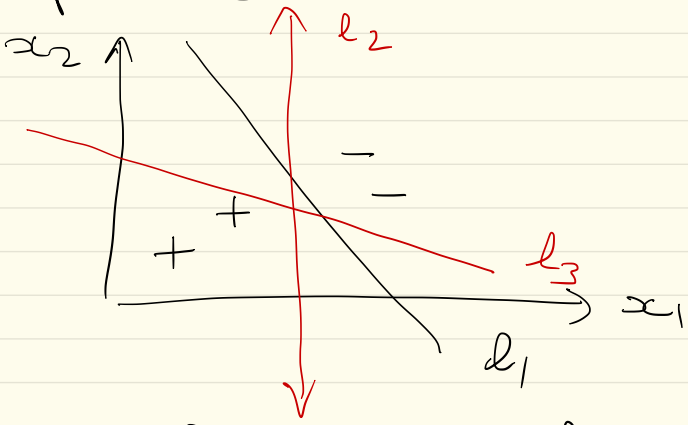
Step 2: Given $\Omega = \{\omega_j, \mu_j, \Sigma_j\}_{j=1}^{K}$
use sub-problem #2 to compute
$B_1 \ldots B_K$

- Alternate between Steps #1 & #2 until convergence.

Support Vector Machines

- Binary Linear Classification

Separable dataset



SVM finds a linear classifier with largest "margin"

Training Set

$$\mathcal{D} = \{(\underline{x}_1, y_1), (\underline{x}_2, y_2), \dots (\underline{x}_N, y_N)\}$$

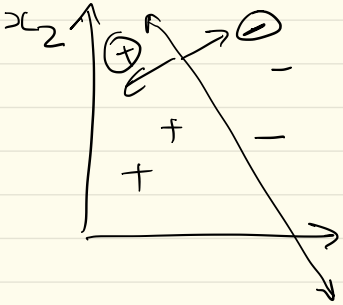
$\underline{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$

decision Rule:

$$\hat{y} = \text{sign}(\underline{w}^T \underline{x} + b)$$

$\ell: \underline{w}^T \underline{x} + b$, $\underline{w} \in \mathbb{R}^d$, $b \in \mathbb{R}$

assume $\mathcal{D}$ is linearly separable data-set & ℓ achieves perfect classification.



Given any $\underline{x}_n \in \mathbb{R}^d$

if $y_n = +1 \Rightarrow \underline{w}^T \underline{x}_n + b > 0$

if $y_n = -1 \Rightarrow \underline{w}^T \underline{x}_n + b < 0$

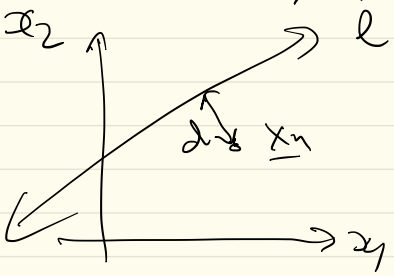
$$y_n(\underline{w}^T \underline{x}_n + b) > 0$$

$\forall n = 1, 2, \dots N$

Margin of ℓ: $\underline{w}^T \underline{x} + b$

distance between $\underline{x}_n$ & ℓ

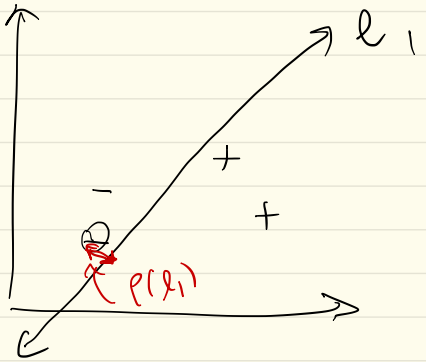
$$\text{dist}(\underline{x}_n, \ell) = \frac{|\underline{w}^T \underline{x}_n + b|}{\|\underline{w}\|}$$



$$= \frac{y_n (\underline{w}^T \underline{x}_n + b)}{\|\underline{w}\|}$$

margin of ℓ:

$$\rho(\ell) = \min_{1 \le n \le N} \text{dist}(\underline{x}_n, \ell)$$



Idea in SVM:

find ℓ that maximizes $\rho(\ell)$

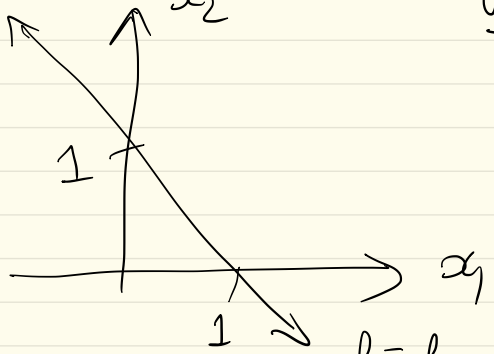
Normalization of $(\underline{w}, b)$

$l_1: x_1 + x_2 - 1 = 0$

$\underline{w} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad b = -1$

$l_2: 5x_1 + 5x_2 - 5 = 0$

$\underline{w} = \begin{bmatrix} 5 \\ 5 \end{bmatrix} \quad b = -5$



$(\underline{w}, b)$ & $(\alpha \underline{w}, \alpha b)$

$l = l_2$ represent the same decision boundary for any non-zero α.

Given l: $(\underline{w}^T \underline{x} + b)$

define

$$\delta = \min_{1 \le n \le N} y_n (\underline{w}^T \underline{x}_n + b)$$

Without loss of generality we can select $(\underline{w}, b)$ such that $\delta = 1$.

Proof: Note that since l is a perfect classifier, $y_n(\underline{w}^T \underline{x}_n + b) > 0$ $\forall n$. Hence $\delta > 0$

Suppose $\delta \neq 1$. Define

$$\tilde{\underline{w}} = \frac{\underline{w}}{\delta}, \quad \tilde{b} = \frac{b}{\delta}$$

$\delta > 0$, $\delta \neq 1$

$\tilde{l}: (\tilde{\underline{w}}^T \underline{x} + \tilde{b})$ Clearly $\tilde{l} = l$.

Claim: $\min_{1 \le n \le N} y_n (\tilde{\underline{w}}^T \underline{x}_n + \tilde{b}) = 1$

$$\min_{1 \le n \le N} y_n \left(\tilde{\underline{w}}^T \underline{x}_n + \tilde{b} \right)$$

$$= \min_{1 \le n \le N} y_n \left(\frac{\underline{w}^T \underline{x}_n}{s} + \frac{b}{s} \right)$$

$$= \min_{1 \le n \le N} \frac{y_n \left(\underline{w}^T \underline{x}_n + b \right)}{s} = \frac{1}{s}$$

but $\delta = \min_{1 \le n \le N} y_n \left(\underline{w}^T \underline{x}_n + b \right)$

From now Assume that for any l: $\left(\underline{w}^T \underline{x} + b \right)$ $\underline{w}$ & b are such that

$$\min_{1 \le n \le N} y_n \left(\underline{w}^T \underline{x}_n + b \right) = 1$$

Margin

$\ell: \underline{w}^T \underline{x} + b$

$$\rho(\ell) = \min_{1 \le n \le N} \frac{y_n (\underline{w}^T \underline{x}_n + b)}{\|\underline{w}\|}$$

$$= \frac{\min_{1 \le n \le N} y_n (\underline{w}^T \underline{x}_n + b)}{\|\underline{w}\|}$$

$$= \frac{1}{\|\underline{w}\|} = \frac{1}{\sqrt{\underline{w}^T \underline{w}}}$$

Problem A

maximize $\frac{1}{\sqrt{\underline{w}^T \underline{w}}}$ $\Rightarrow$ minimize $\frac{1}{2} \underline{w}^T \underline{w}$

subject to $\min_{1 \le n \le N} y_n (\underline{w}^T \underline{x}_n + b) = 1$

Variables: $\underline{w} \in \mathbb{R}^d$, $b \in \mathbb{R}$

$$\min_{1 \le n \le N} \underbrace{y_n(\underline{w}^T \underline{x}_n + b)}_{= a_n} = 1$$

minimize $\frac{1}{2} \underline{w}^T \underline{w}$ ← objective

Problem

subject to

$$y_n(\underline{w}^T \underline{x}_n + b) \geq 1$$

$\forall\, n = 1, 2, \ldots N$

with equality for at-least one $n \in \{1, 2, \ldots N\}$

$\min\{a_1, a_2, \ldots a_N\} = 1$

original constraint

$a_1 \geq 1, \; a_2 \geq 1 \; \ldots \; a_N \geq 1$

with equality for at-least some a_n

Problem B

Variables: $\underline{w}, b$

minimize $\frac{1}{2}\underline{w}^T\underline{w}$

$\forall n = 1, 2, \ldots N$

$$y_n(\underline{w}^T\underline{x}_n + b) \geq 1$$

~~with Equality for at-least one value of $n \in \{1, 2 \ldots N\}$~~

Solution of Problem A: $(\underline{w}^*, b^*)$

objective: $\frac{1}{2}\underline{w}^{*T}\underline{w}^*$

Solution to Problem B: $(\hat{\underline{w}}, \hat{b})$

objective: $\frac{1}{2}\hat{\underline{w}}^T\hat{\underline{w}}$

is smaller than $\frac{1}{2}\underline{w}^{*T}\underline{w}^*$

Problem A:

objective: minimize $\frac{1}{2} \underline{w}^T \underline{w}$

such that $y_n(\underline{w}^T \underline{x}_n + b) \geq 1$

$\forall n = 1, 2, \ldots N$, equality for at-least one n

$\Longrightarrow$ optimal (w^*, b^*) given SVM.

Problem B:

minimize $\frac{1}{2} \underline{w}^T \underline{w}$

Such that $y_n(\underline{w}^T x_n + b) \geq 1$

$\forall n = 1, 2, \ldots N$

optimal $(\hat{\underline{w}}, \hat{b})$

then $\frac{1}{2} \hat{\underline{w}}^T \hat{\underline{w}} \leq \frac{1}{2} w^{*T} \underline{w}^*$

in general

Relaxation of Problem A to Problem B is Exact.

Proof: let $(\hat{\underline{w}}, \hat{b})$ be a solution for problem B

Suppose: $y_n(\hat{\underline{w}}^T \underline{x}_n + \hat{b}) > 1$

$\forall n = 1, 2, \ldots N$

i.e $\Theta = \min_{1 \le n \le N} [y_n(\hat{\underline{w}}^T \underline{x}_n + \hat{b})]$

$\Theta > 1$.

Consider $\underline{w}^* = \frac{\hat{\underline{w}}}{\Theta}$, $b^* = \frac{\hat{b}}{\Theta}$ ✓

Claim: (1) $\frac{1}{2} \underline{w}^{*T} \underline{w}^* < \frac{1}{2} \hat{\underline{w}}^T \hat{\underline{w}}$

2) $(\underline{w}^*, b^*)$ is feasible in problem A

$$\frac{1}{2} \underline{w}^{*T} \underline{w}^{*}$$

$$= \frac{1}{2} \left(\frac{\hat{\underline{w}}}{\theta}\right)^T \left(\frac{\hat{\underline{w}}}{\theta}\right)$$

$$= \frac{1}{2} \frac{\hat{\underline{w}}^T \hat{\underline{w}}}{\theta^2}$$

$$< \frac{1}{2} \hat{\underline{w}}^T \hat{\underline{w}}$$

Thus:
optimal $(\underline{w}, b)$ in problem B is also the optimal $(\underline{w}, b)$ in problem A

$\theta > 1$

$$\min_{1 \le n \le N} y_n \left(\underline{w}^{*T} x_n + b^*\right)$$

$$= \min_{1 \le n \le N} y_n \left(\left(\frac{\hat{\underline{w}}}{\theta}\right)^T x_n + \frac{\hat{b}}{\theta}\right)$$

$$= \min_{1 \le n \le N} \frac{y_n\left(\hat{\underline{w}}^T x_n + \hat{b}\right)}{\theta}$$

$$= \frac{\min_{1 \le n \le N} y_n\left(\hat{\underline{w}}^T x_n + \hat{b}\right)}{\min_{1 \le n \le N} y_n\left(\hat{\underline{w}}^T x_n + \hat{b}\right)} = 1.$$

$l: (\underline{w}^T x + b)$

Given $\mathcal{D} = \{(\underline{x}_1, y_1) \dots (\underline{x}_N, y_N)\}$

minimize $\frac{1}{2} \underline{w}^T \underline{w}$

such that $y_n (\underline{w}^T x_n + b) \geq 1$

$\forall n \in \{1, 2, \dots N\}$

$\frac{1}{2} \underline{w}^T \underline{w}$ ← Quadratic in $\underline{w}$

$y_n (\underline{w}^T \underline{x}_n + b)$ is linear in $(\underline{w}, b)$

Quadratic Programming Problem.

Quadratic Programming

variable: $\underline{u} \in \mathbb{R}^L$

objective: $\boxed{\frac{1}{2}\underline{u}^T Q \underline{u} + \underline{p}^T \underline{u}}$

minimize

$Q \in \mathbb{R}^{2L}$ matrix

$\underline{p} \in \mathbb{R}^L$ vector

Constraints: $l_i \le u_i \le h_i$

$i = 1, 2, \ldots L$

$\underline{a}_m^T \underline{u} \ge c_m$

$m = 1, 2, \ldots M$

Quadratic solver

Input: $Q, \underline{p}, \{l_i, h_i\}_{i=1}^{d}, \{\underline{a}_m, c_m\}_{m=1}^{M}$

SVM is a Quadratic Programming Problem

SVM: minimize $\frac{1}{2}\underline{w}^T\underline{w}$

such that $y_n(\underline{w}^T\underline{x}_n + b) \geq 1$

variables: $(\underline{w}, b)$ $\forall n \in \{1, 2, \ldots N\}$

Quadratic Programming

minimize $\frac{1}{2}\underline{u}^T Q \underline{u} + p^T \underline{u}$

such that $a_m^T \underline{u} \geq c_m \quad m = 1, 2, \ldots M$

Identify $\underline{u}, Q, \underline{p}, \{a_m, c_m\}$

$$\underline{u} = \begin{bmatrix} \underline{w} \\ b \end{bmatrix} \in \mathbb{R}^{d+1} \qquad a_m = [y_m \underline{x}_m^T \quad y_m]^T$$

$$c_m = 1$$

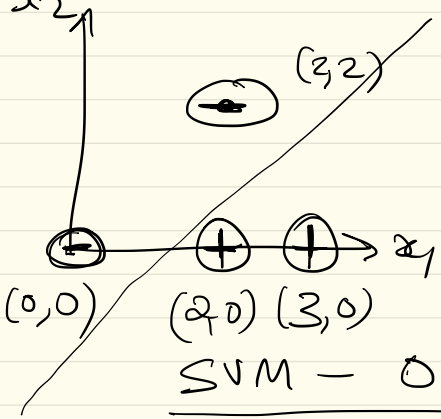
$$Q = \left[\begin{array}{c|c} I & 0 \\ \hline 0 & 0 \end{array}\right] \qquad p = \underline{0}$$

Example $N = 4$

$X_1 = (0, 0)$ $y_1 = -1$ | $X_3 = (2, 0)$ $y_3 = +1$

$X_2 = (2, 2)$ $y_2 = -1$ | $X_4 = (3, 0)$ $y_4 = +1$

i.e. $\omega_1 x_1 + \omega_2 x_2 + b = 0$



find a linear classification boundary with max margin.

SVM – optimization

minimize : $\frac{1}{2}(\omega_1^2 + \omega_2^2)$

such that $y_n(\omega_1 x_{1n} + \omega_2 x_{2n} + b) \geq 1$

$\forall\, n \in \{1, 2, 3, 4\}$

$n=1$: $-1(0 \cdot \omega_1 + 0 \cdot \omega_2 + b) \geq 1$

$\therefore \boxed{-b \geq 1}$

$n=2$ $\boxed{-1(2\omega_1 + 2\omega_2 + b) \geq 1}$

$n=3$ $\boxed{1(2\omega_1 + 0 \cdot \omega_2 + b) \geq 1}$ $\xrightarrow{n=4}$ $\boxed{3\omega_1 + b \geq 1}$

$\ell: x_1 - x_2 - 1 = 0$

minimize $\frac{1}{2}(w_1^2 + w_2^2)$

$n=1$: $-b \geq 1$ — (1)

$n=2$ $-(2w_1 + 2w_2 + b) \geq 1$ — (2)

$n=3$ $2w_1 + b \geq 1$ — (3)

$n=4$ $3w_1 + b \geq 1$ — (4)

(1) ⊕ (3) $2w_1 \geq 2 \Rightarrow w_1 \geq 1$

(2) ⊕ (3) $-2w_2 \geq 2 \Rightarrow w_2 \leq -1$

select $w_1 = +1$ & $w_2 = -1$

$b \leq -1$ $b \leq -1$ $b \geq -1$

$b \geq -2$

$b = -1$ satisfies all constraints is our solution

$w_1^* = +1$, $w_2^* = -1$ $b^* = -1$