ECE421
Intro. to Machine Learning

Winter 2022

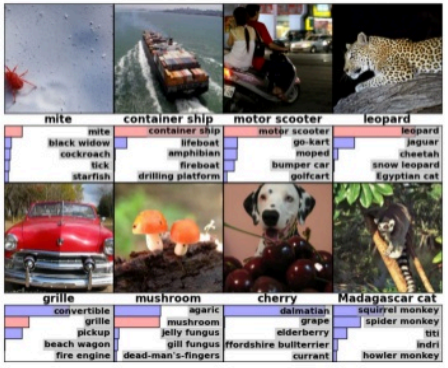
Ashish Khisti (Sec 1)

Christopher Lucasius (Sec 2)

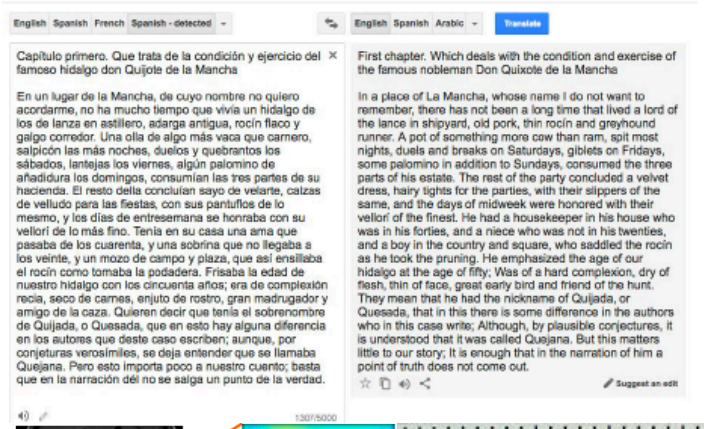
ImageNet Challenge

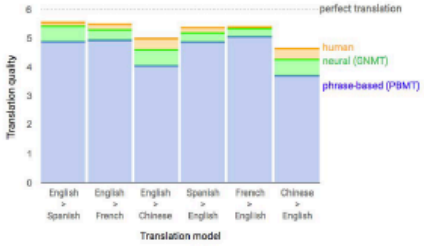
IMAGENET

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

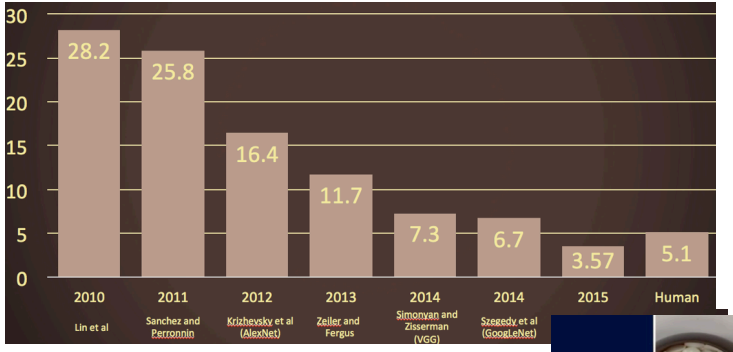


- Natural language processing

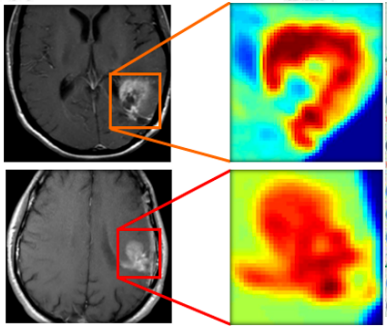


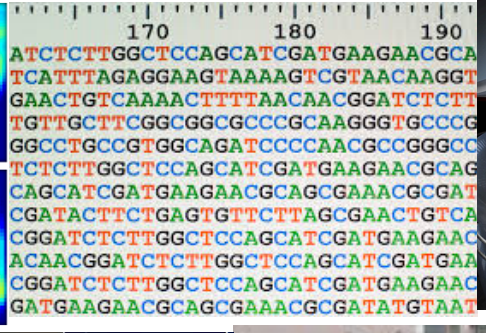


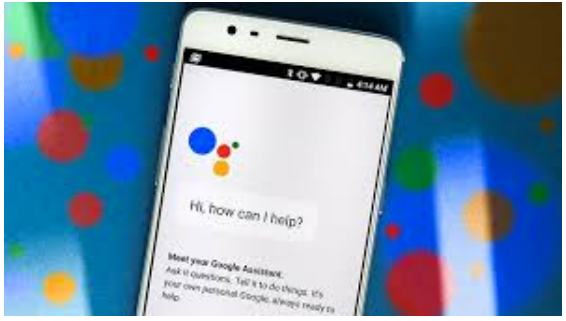
Google's Neural Machine Translation (Wu et al. 2016)

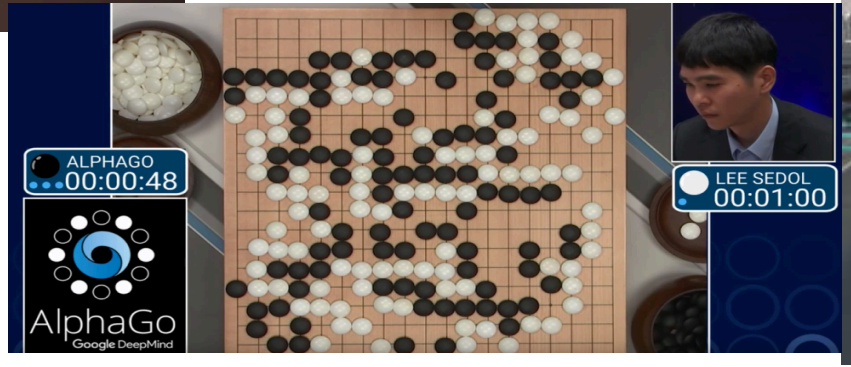


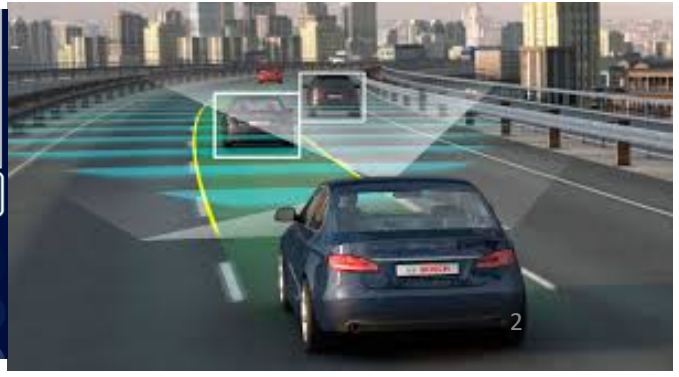
DIAGNOSTICS



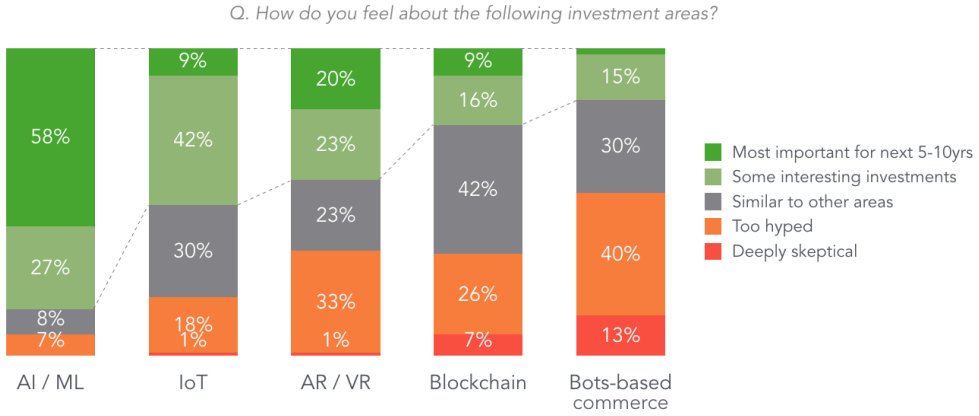


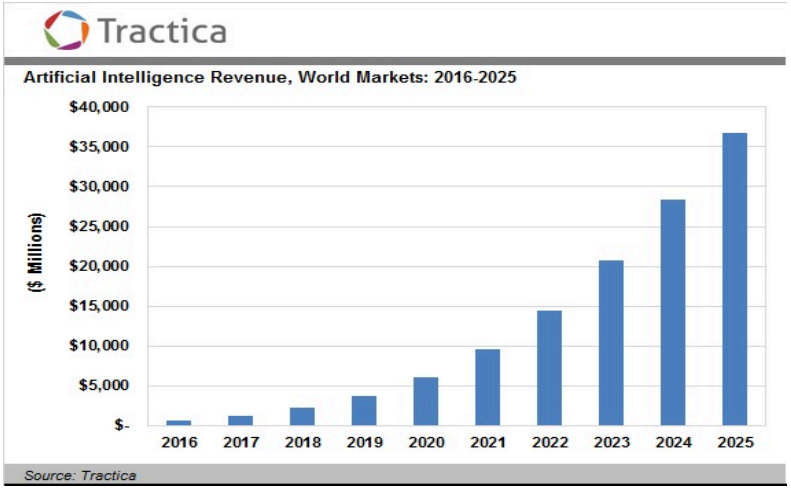






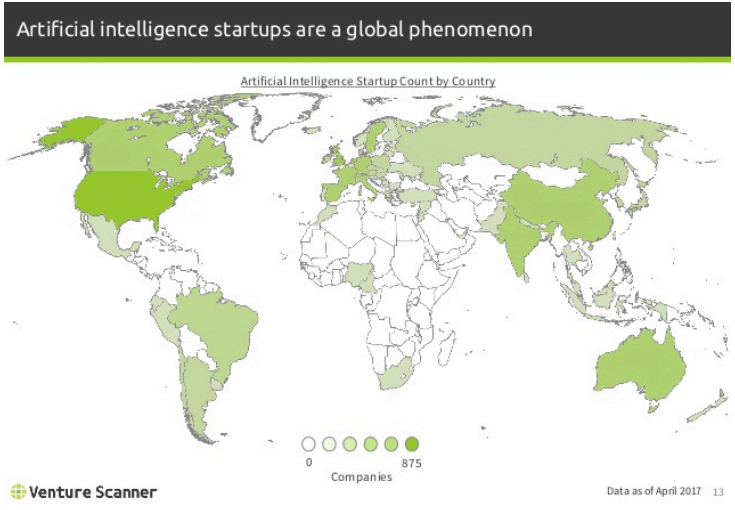
Most VCs are most excited about AI & Machine Learning as their most important investment theme for the coming 5-10 years.

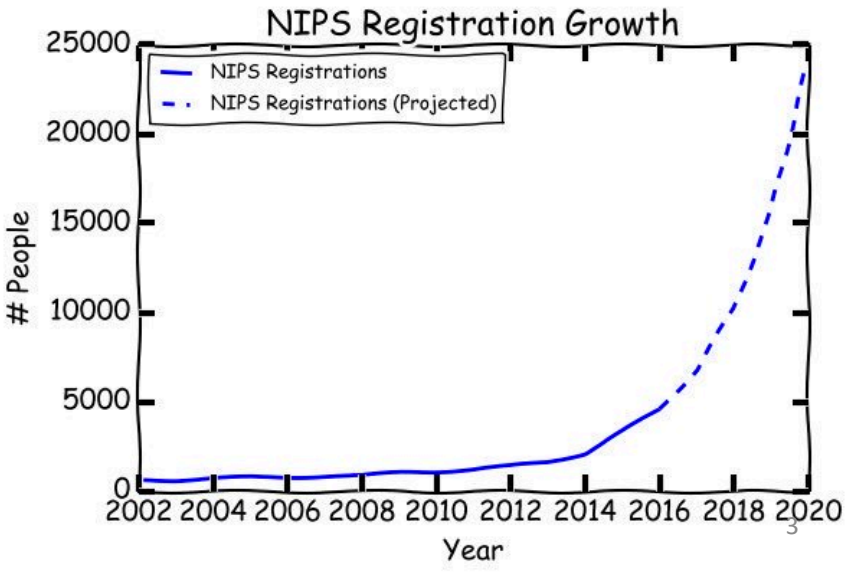




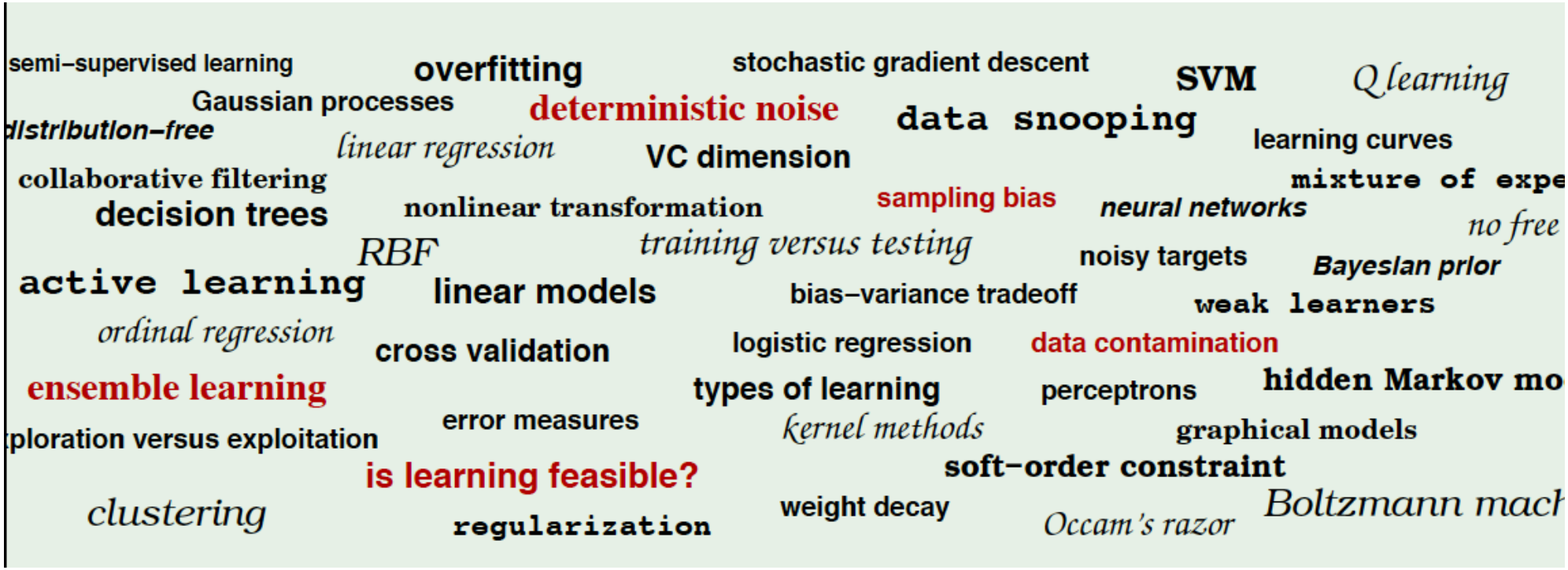
17 Source: Upfront Ventures survey of VCs (N=115), Jan 2017

upfront VENTURES





Machine Learning Jargon



Machine Learning

Develop computational systems to adaptively improve their performance with experience accumulated from the observed data.

Data Driven Approach

Overview

Learning Methods

Supervised Learning

- Linear Models
- Neural Networks
- Support Vector Machines

(Prediction)

Unsupervised Learning

- Clustering
- Density Estimation
- EM algorithm

(No Prediction)

Learning Theory

- PAC Learning
- VC Dimension
- Bias Variance Tradeoff

Learning Principles

- Regularization
- Validation

Linear Classification

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

Binary Label

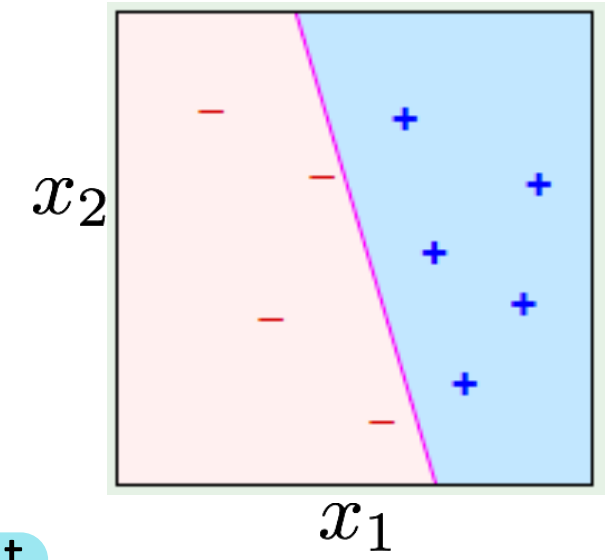
$$\mathbf{x}_i \in \mathbb{R}^d, \qquad y_i \in \{\pm 1\}$$

Observation Vector — Binary Label

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

  $\mathbf{x}_i \in \mathbb{R}^d, \qquad y_i \in \mathbb{R}$ (Real Number)

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

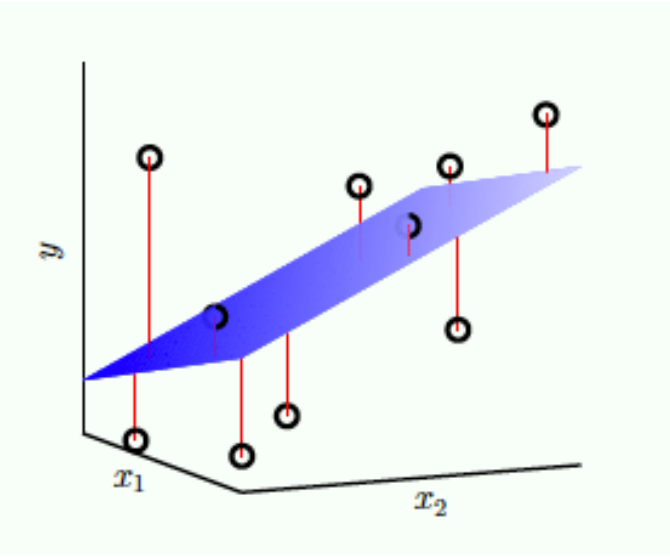
  $\mathbf{x} = (x_1, \ldots, x_d)$

  w_i : Learning Weight

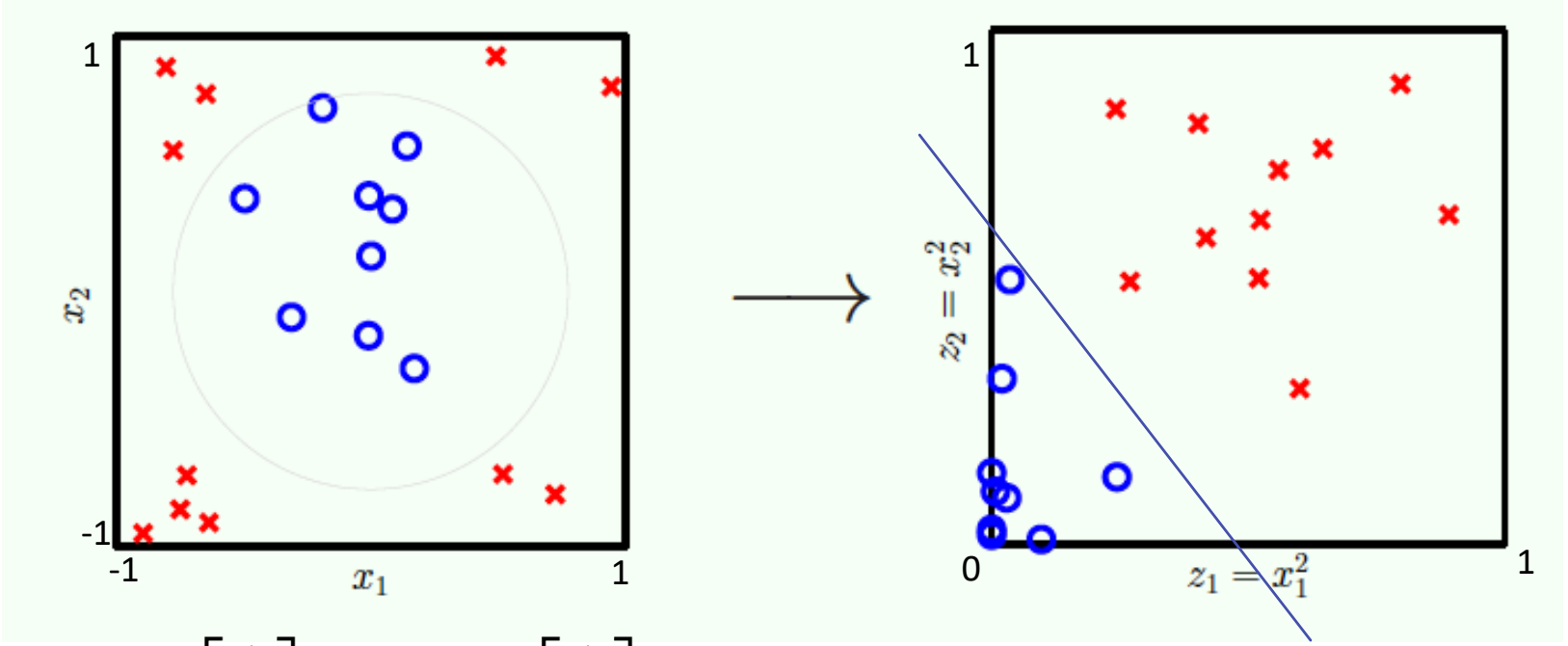
- **Minimize** the prediction error:

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

- Least Squares and its variations



Non-Linear Transformations of Data

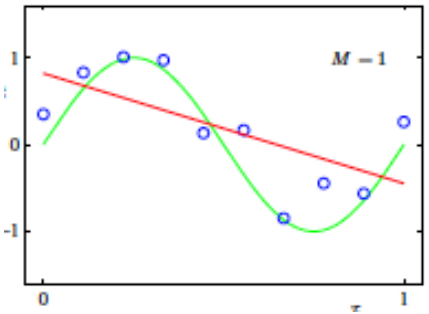


$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \Longrightarrow \mathbf{z} = \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$

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

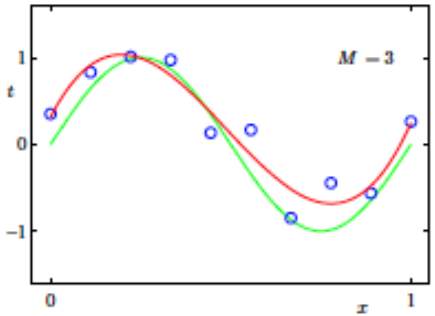
Nonlinear Transforms, Feature Vectors, Kernel Methods

Non-Linear Transformations of Data



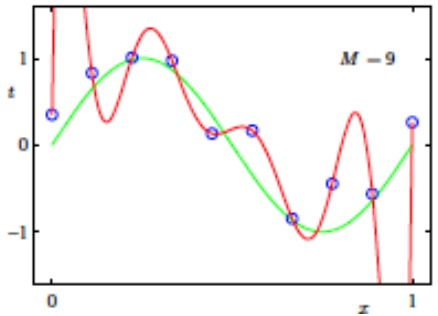
$$\mathbf{z} = \begin{bmatrix} 1 & x \end{bmatrix}^T$$

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

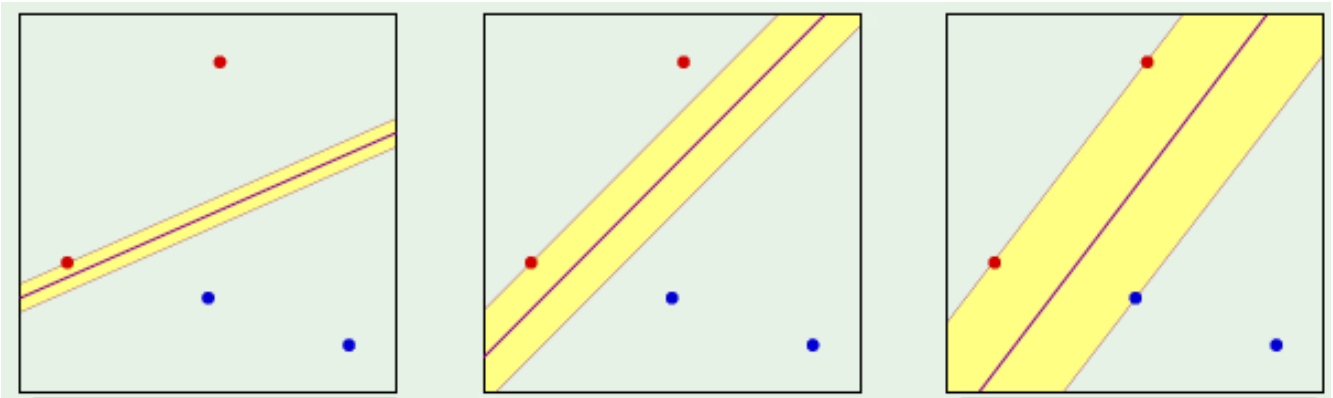
Under-fitting

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

Overfitting

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

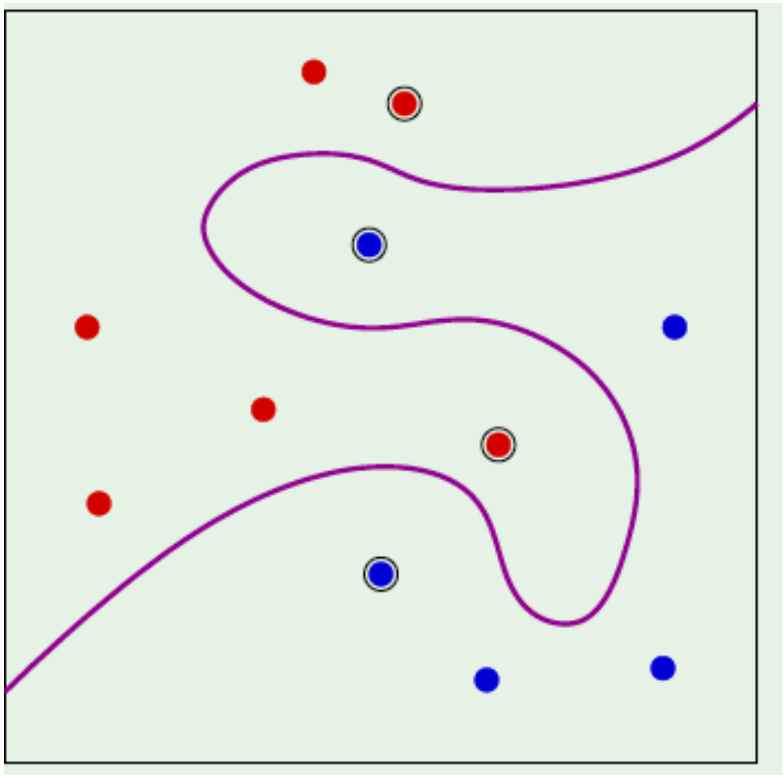
Support Vector Machines (Find Maximum Margin)



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

SVM(Support Vector Machine)

SVMs with Non-Linear Transforms

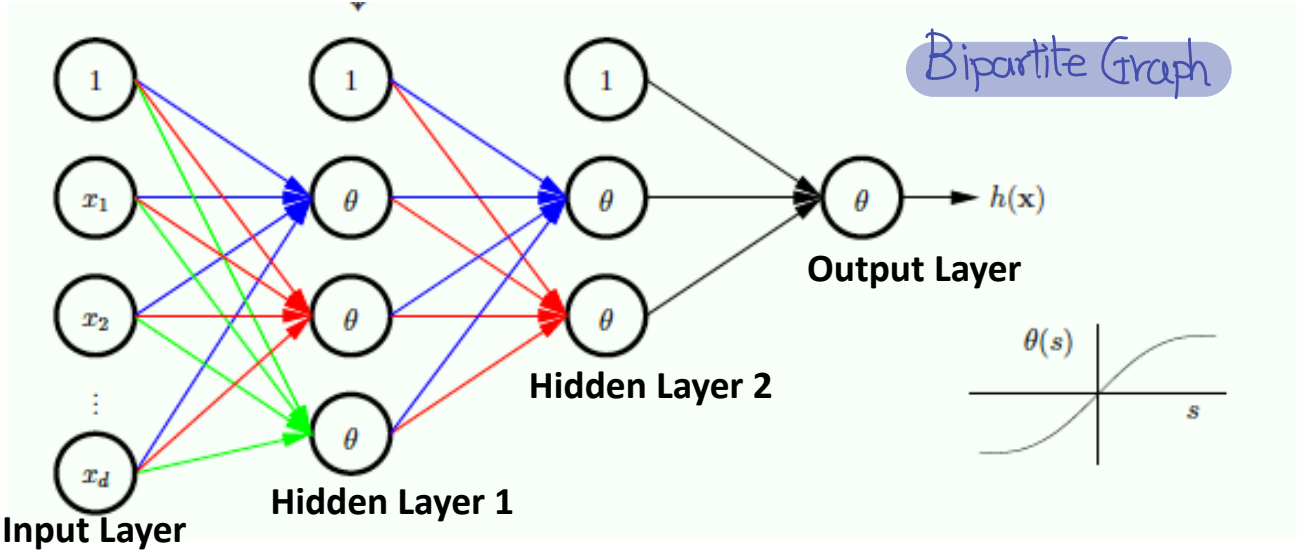


- Radial Basis Functions (RBF)

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

- 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)

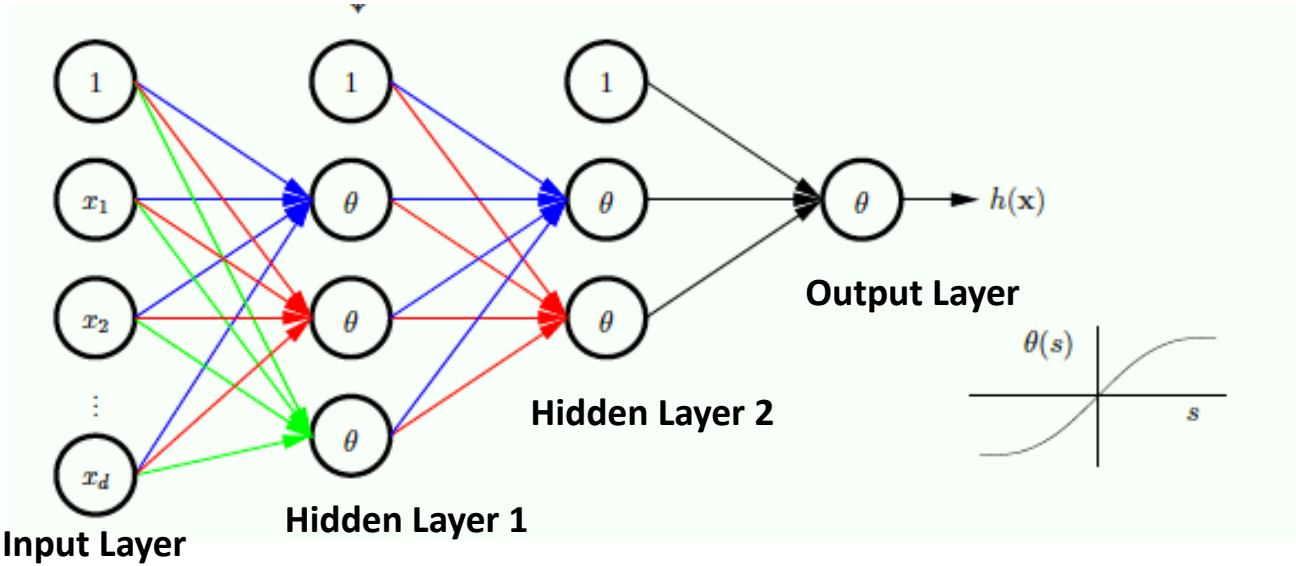
Weights: Layer 2

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

Output

θ : Non Linear Activation Function

Neural Networks - Architecture



Vector Form

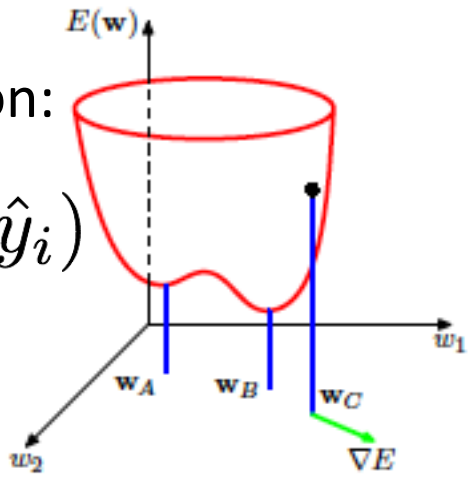
$$y = \text{sign}\left[W_3^T \left\{\theta\left(W_2^T \left\{\theta(W_1^T \cdot \mathbf{x})\right\}\right)\right\}\right]$$

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:

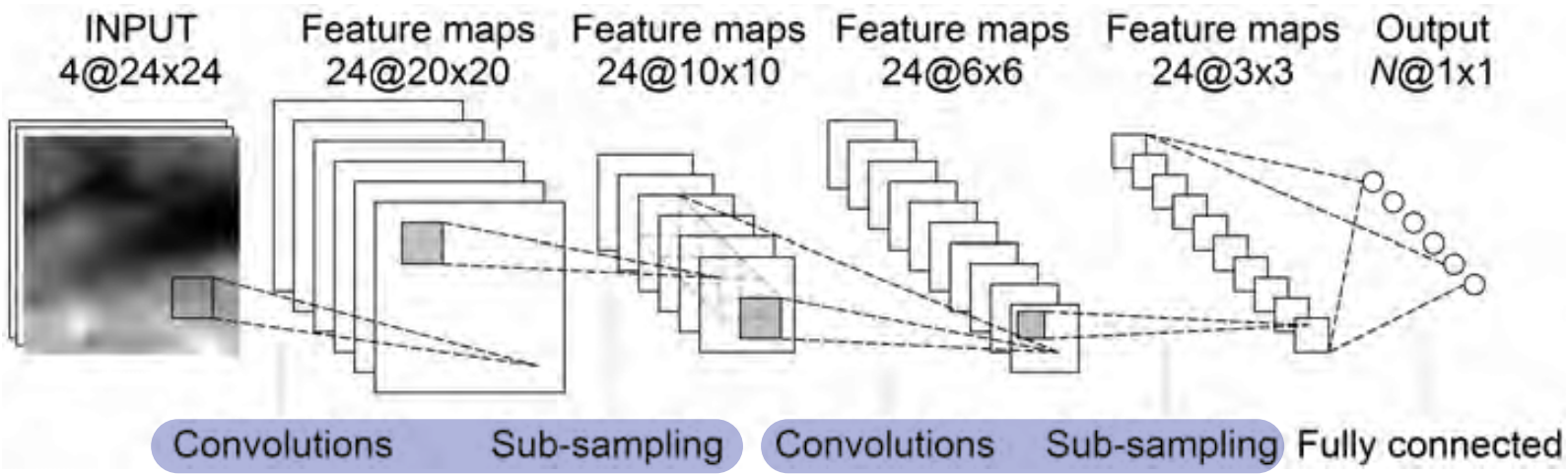
$$E(\mathbf{w}) = \sum_{i=1}^{n} \ell_{\mathbf{w}}(y_i, \hat{y}_i)$$



Training Procedure

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

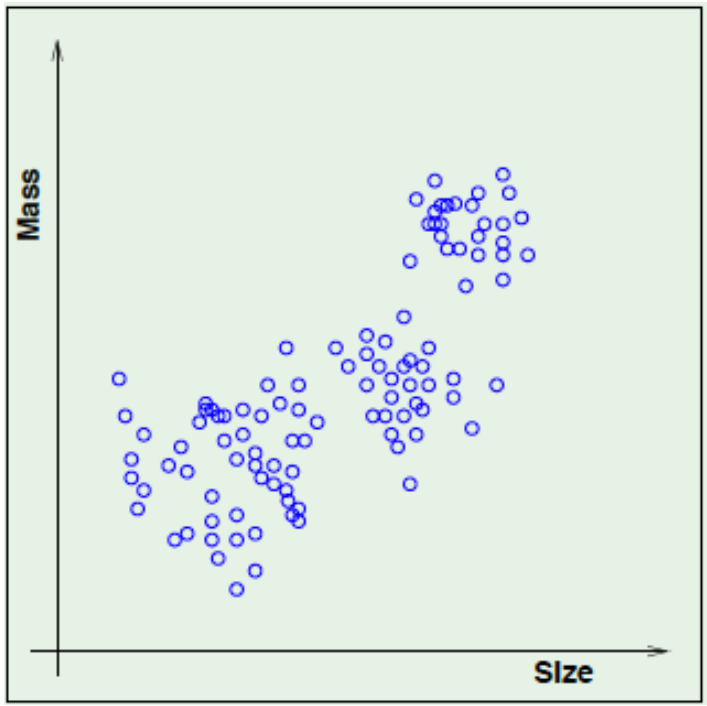
Convolutional Neural Networks



Filter, Feature Map

- 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



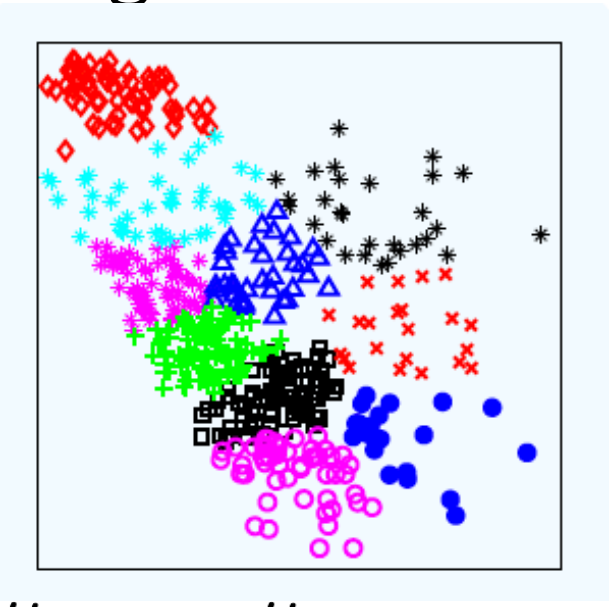
Training Data:

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

No Labels!

Still Need to do Learning!

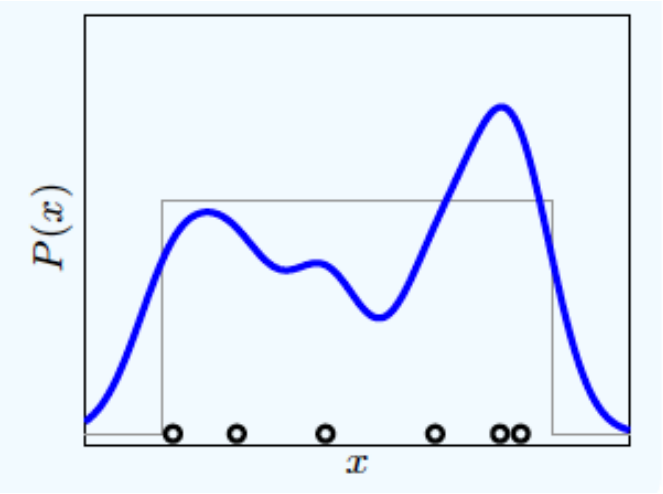
k-Means Clustering



- 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

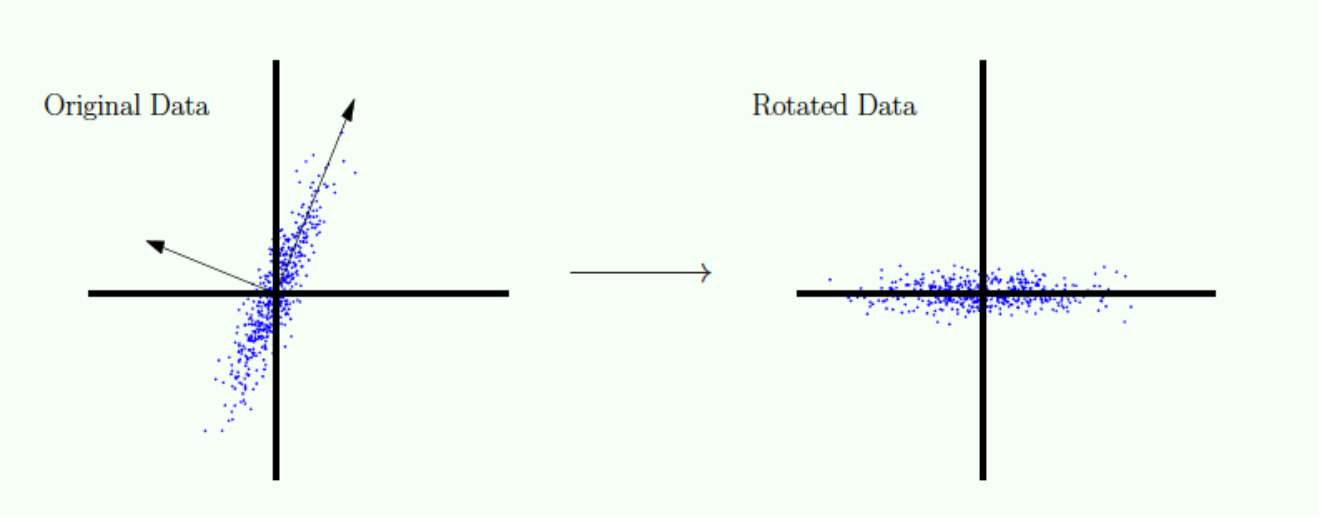
Density Estimation - Gaussian Mixture Models



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

- 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: $\hat{y} = h(\mathbf{x})$ Prediction
- 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

New Data Point

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

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: $1 - \delta$
- $$E_{\text{out}}(h) \leq E_{\text{in}}(h) + \sqrt{\frac{1}{2N} \log \frac{2M}{\delta}}$$

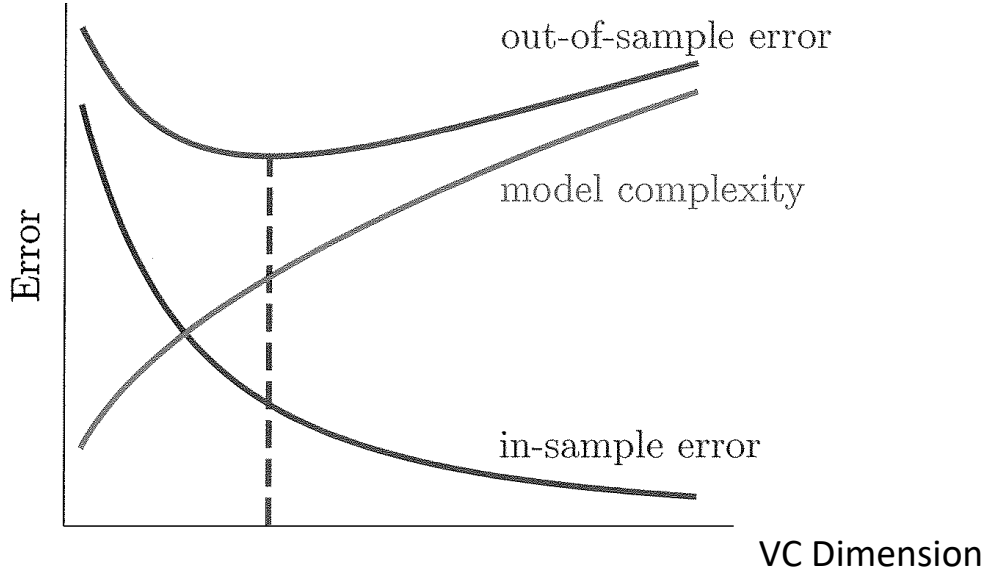
Complexity of Hypothesis Class

Number of Training Examples

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

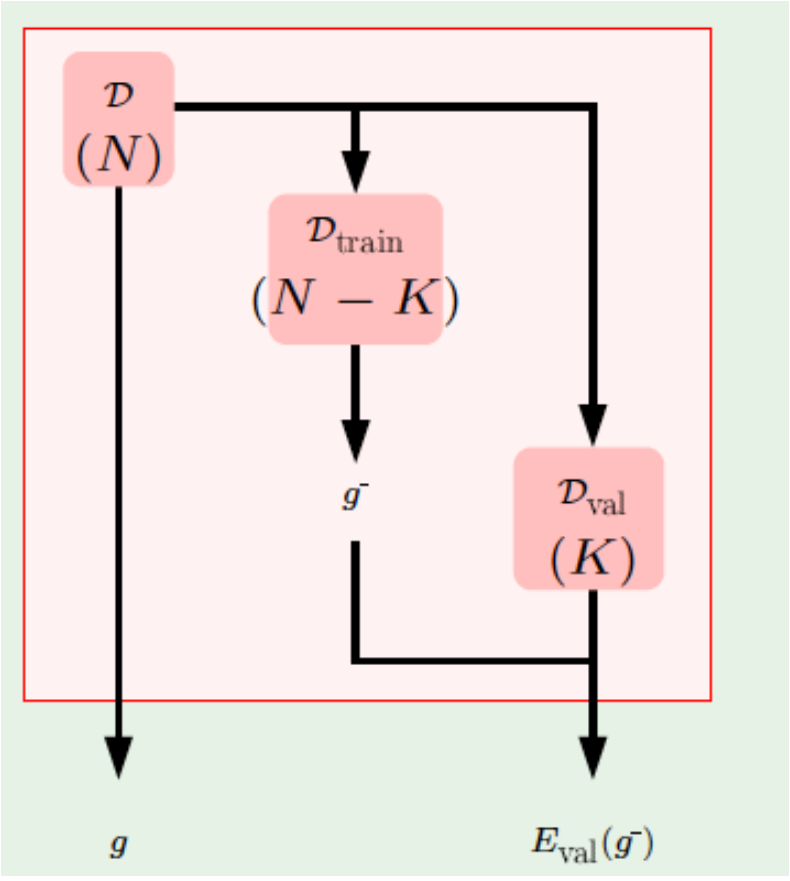
Vapnik Chervonenkis (VC) dimension

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



Techniques

Validation



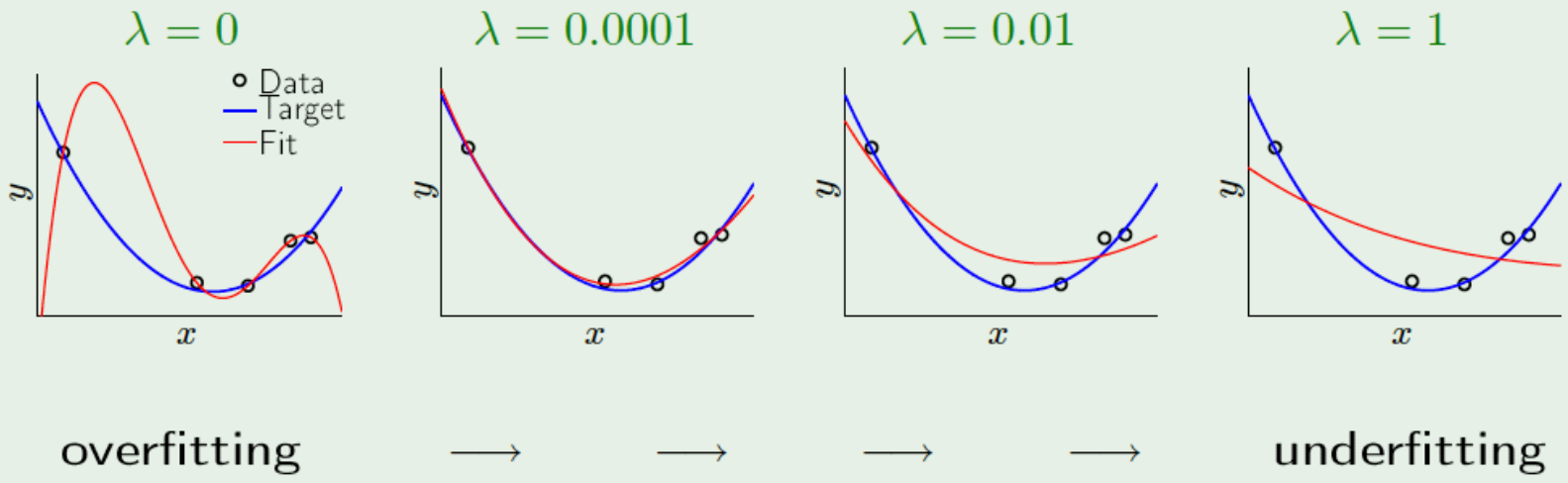
- Divide Training Set into 2 Parts
  - Learning Set
  - Validation Set

  Useful Technique In Practice
- 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

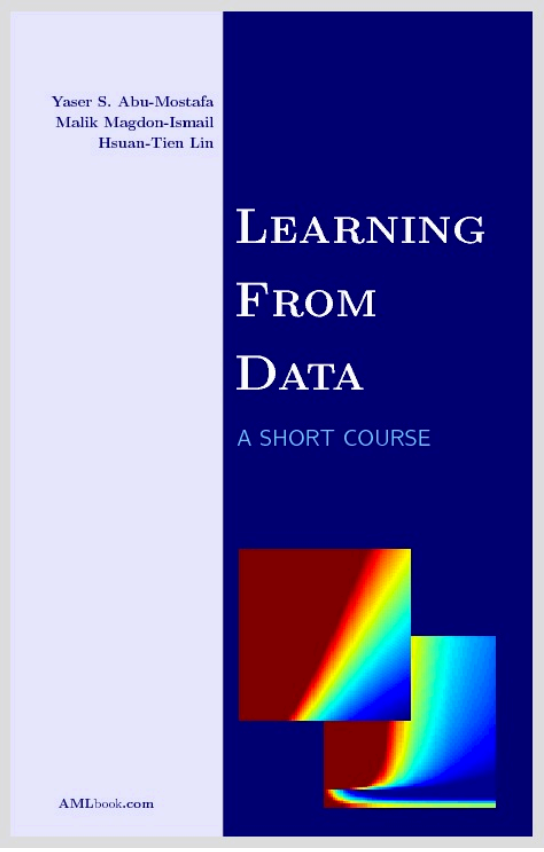
Course Staff

Instructors:

- Ashish Khisti (Section 1)
- Christopher Lucasius (Section 2)

- Tutorials (Mandatory)
  - See calendar timing

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

Week	Description
Week 1(Jan 9)	Intro/Linear Classification
Week 2 (Jan 16)	Linear Regression, Regularization
Week 3 (Jan 23)	Logistic Regression
Week 4 (Jan 30)	Gradient Descent
Week 5 (Feb 6)	Multiplayer Perceptron, Backpropagation
Week 6 (Feb 13)	Deep Learning
Week 7 (Feb 27)	Unsupervised Learning: Clustering and Density Estimation
Week 8 (Mar 6)	EM Algorithm
Week 9 (Mar 13	Support Vector Machine
Week 10 (Mar 20)	Support Vector Machine, PAC Learning
Week 11 (Mar 27)	PAC Learning, Bias-Variance Tradeoff
Week 12 (Apr 3)	Validation, Cross-Validation

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

- Homework: 10%
  Midterm: 25% March 6th, 2023
  Programming assignments: 7.5% * 4 = 30%
  Final exam: 35%

Learning Outcomes

- Fundamentals: basic theory and the fundamental algorithms
- Analysis: ML algorithms for classification, regression and unsupervised learning.
- Algorithm Design: using computational toolboxes of machine learning

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

Machine Learning

Computational system whose performance improves with accumulated data

Ex, recommandation systems (Netflix)

Given a list of movies and partial ratings given by users, you want to predict how a user will rate a movie he/she has not seen

Approach 1:

Based on human experts

An expert view all movies & categorize them

1. Actors
2. Blockbusters
3. Comedy/Actions

Interview each users

1. Like comedy/actions?
2. Favorite actors
3. Like blockbusters?

Tedious process, expensive, user preferences can change.

Approach 2:

Machine Learning

Visualize Dataset

users \ movies	1	2	3	4	5
1		r_{12}	r_{13}		
2	r_{21}	r_{22}			r_{25}
3	r_{31}			r_{34}	
4			r_{43}		
5					

r_{ij}, rating given by user i for movie j (if available)

S = set of available rating

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

$S = \{(1,2), (1,3), (2,1), (2,2), (2,5), \ldots\}$

Model (Cannot interpret the weights)

1) for each user i, **user preference vector**

$\underline{x_i} = (x_{i1}, x_{i2}, \ldots, x_{id}) \in \mathbb{R}^d$ → Hyperparameter

2) for each movie j, **movie attribute vector**

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

Function to compute an estimate for the rating given $\underline{x_i}$ & $\underline{y_j}$

Estimated Rating for User i and Movie j: $\hat{r}_{ij} = \underline{x_i^T y_j} = \sum_{l=1}^{d} x_{il} y_{jl}$

Model Parameters (Ω)

$\Omega = \{\{x_i\}_{i \in U}, \{y_j\}_{j \in M}\}$

U : set of users

M : set of movies

Training

Learn model parameters Ω given the training set

Define loss function (Training Error)

$$E_{in}(\Omega) = \sum_{(i,j) \in S} (r_{ij} - \hat{r}_{i,j})^2$$

E_{in}: Error In Sample

$\hat{r}_{i,j} = x_i^T y_j$

We will select Ω to minimize $E_{in}(\Omega)$

$$\Omega^* = \underset{\Omega}{\operatorname{argmin}} E_{in}(\Omega)$$

Optimal Model Parameter (Ω^*)

Training Algorithm, Gradient Descent

EM Algorithm

Prediction Function, $(i,j) \notin S$, $\hat{r}_{ij} = x_i^T y_j$

$\Omega^* = \{\{x_i\}_{i \in U}, \{y_j\}_{j \in M}\}$