ECE421
Intro. to Machine Learning

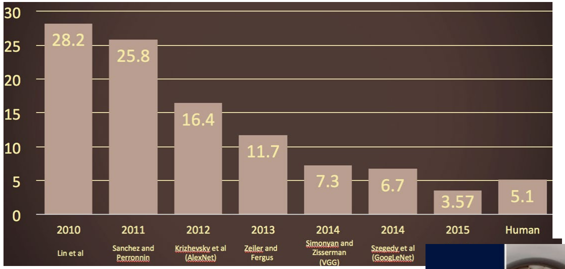
Winter 2022

Ben Liang

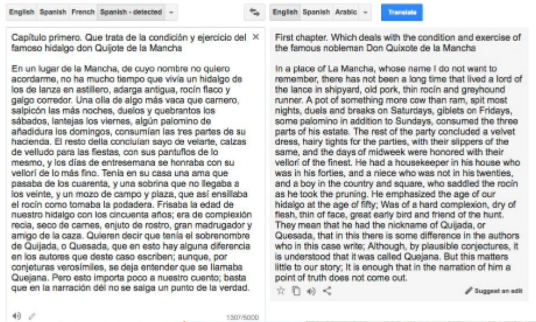
Ashish Khisti

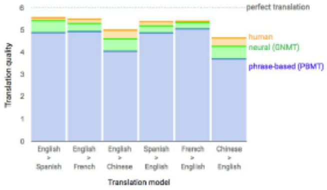
ImageNet Challenge

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

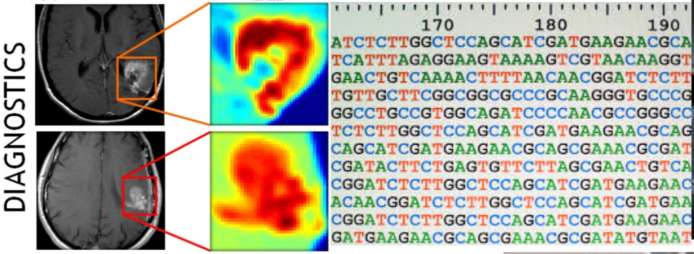


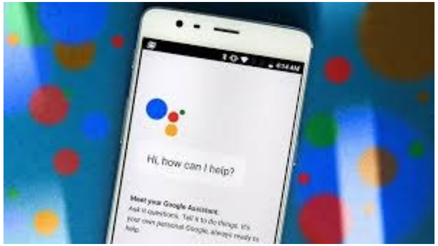
- Natural language processing

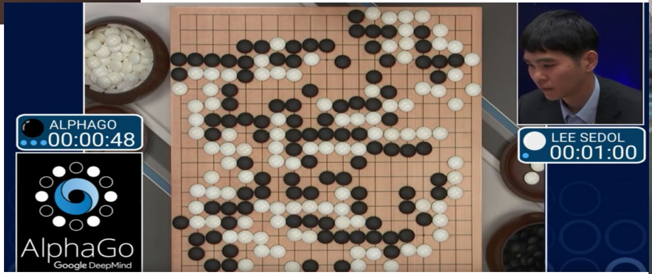


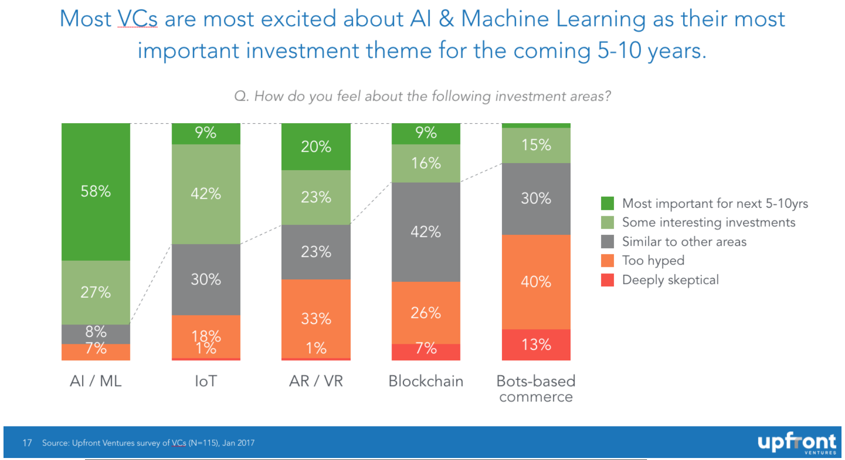


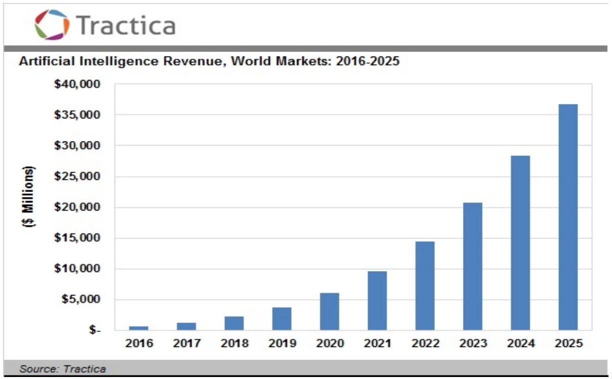
Google's Neural Machine Translation (Wu et al. 2016)

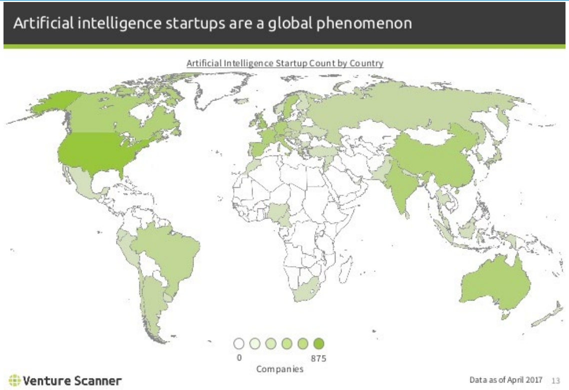


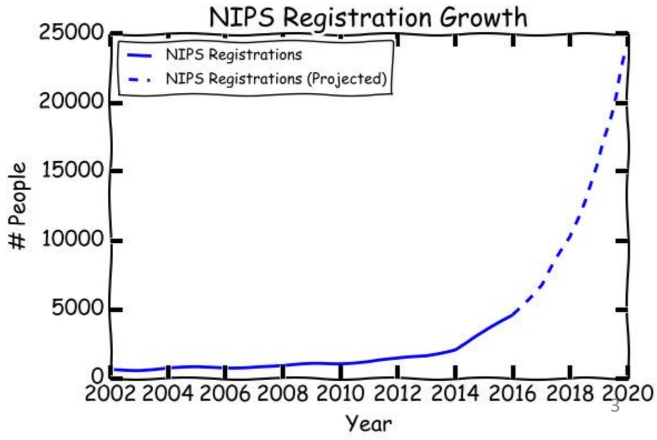




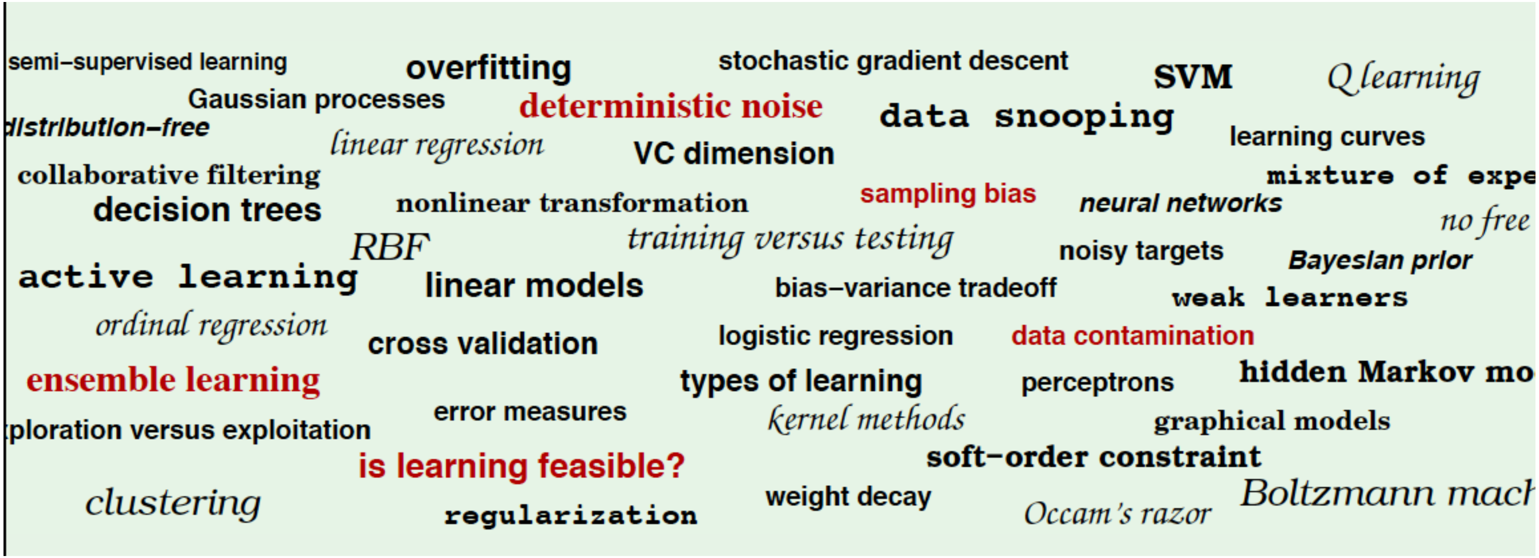

Most VCs are most excited about AI & Machine Learning as their most important investment theme for the coming 5-10 years.
Q. How do you feel about the following investment areas?
58%
27%
8%
7%
9%
42%
30%
18%
1%
20%
23%
23%
33%
1%
9%
16%
42%
26%
7%
15%
30%
40%
13%
AI / ML
IoT
AR / VR
Blockchain
Bots-based commerce
Most important for next 5-10yrs
Some interesting investments
Similar to other areas
Too hyped
Deeply skeptical
17 Source: Upfront Ventures survey of VCs (N=115), Jan 2017
upfront VENTURES


Tractica
Artificial Intelligence Revenue, World Markets: 2016-2025
$40,000
$35,000
$30,000
$25,000
$20,000
$15,000
$10,000
$5,000
$-
($ Millions)
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
Source: Tractica


Artificial intelligence startups are a global phenomenon
Artificial Intelligence Startup Count by Country
0
875
Companies
Venture Scanner
Data as of April 2017 13


NIPS Registration Growth
NIPS Registrations
NIPS Registrations (Projected)
25000
20000
15000
10000
5000
0
People
2002 2004 2006 2008 2010 2012 2014 2016 2018 2020
Year

Machine Learning Jargon



Machine Learning

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

Overview

Learning Methods

Supervised Learning

- Linear Models
- Neural Networks
- Support Vector Machines

Unsupervised Learning

- Clustering
- Density Estimation
- EM algorithm

Learning Theory

- PAC Learning
- VC Dimension
- Bias Variance Tradeoff

Learning Principles

- Regularization
- Validation

Linear Classification

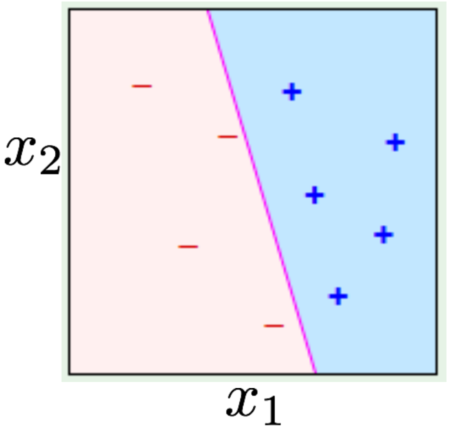
- **Given** Training Samples: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$

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

- **Determine** a *classification rule*

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

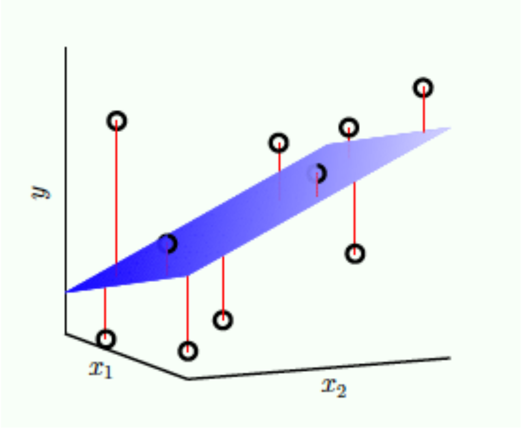
to **minimize** classification error



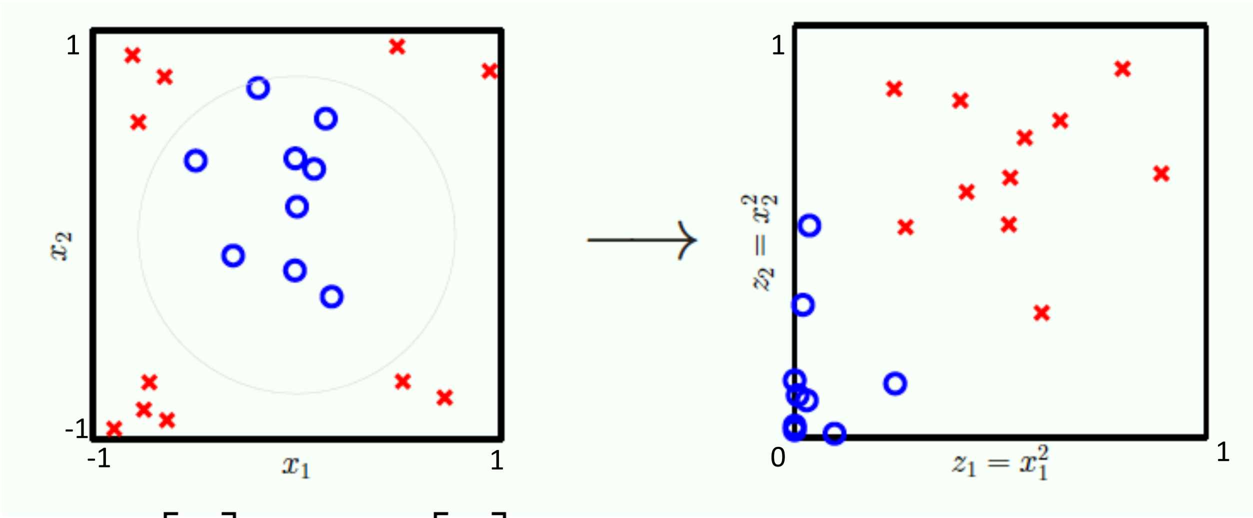
- Perceptron Learning Algorithm
- Logistic Regression and Gradient Descent

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

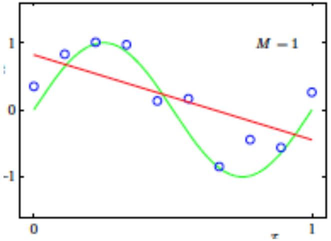


$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \Longrightarrow \mathbf{z} = \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$

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

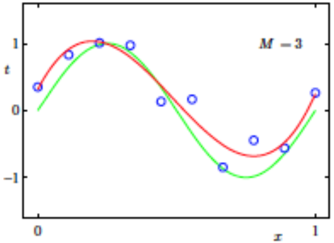
Nonlinear Transforms, Feature Vectors, Kernel Methods

Non-Linear Transformations of Data



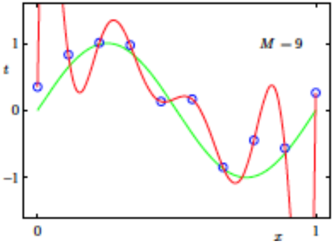
$$\mathbf{z} = \begin{bmatrix} 1 & x \end{bmatrix}^T$$

First Order Polynomial



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

Third Order Polynomial



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

9th Order Polynomial

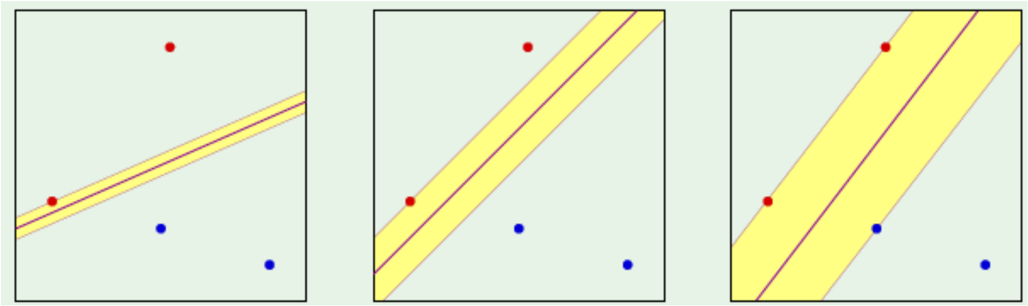
Under-fitting

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

Overfitting

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

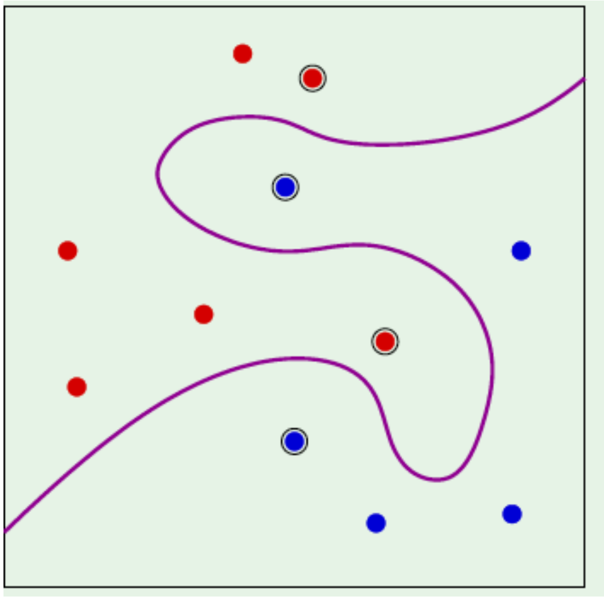
Support Vector Machines



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



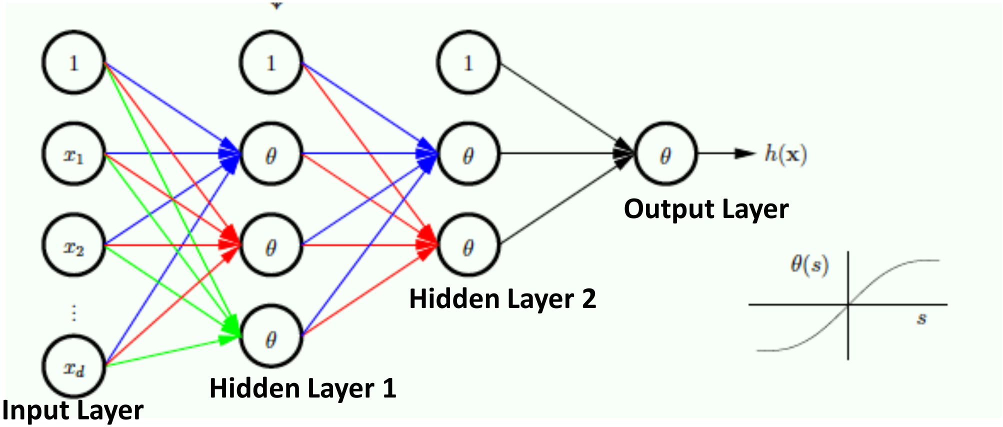
- Radial Basis Functions

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

- Kernel Trick for Efficient Computation

- Bounds on Generalization Error

Neural Networks - Architecture



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

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

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

Output \`i' (Hidden Layer 1)

Weights:Layer 1

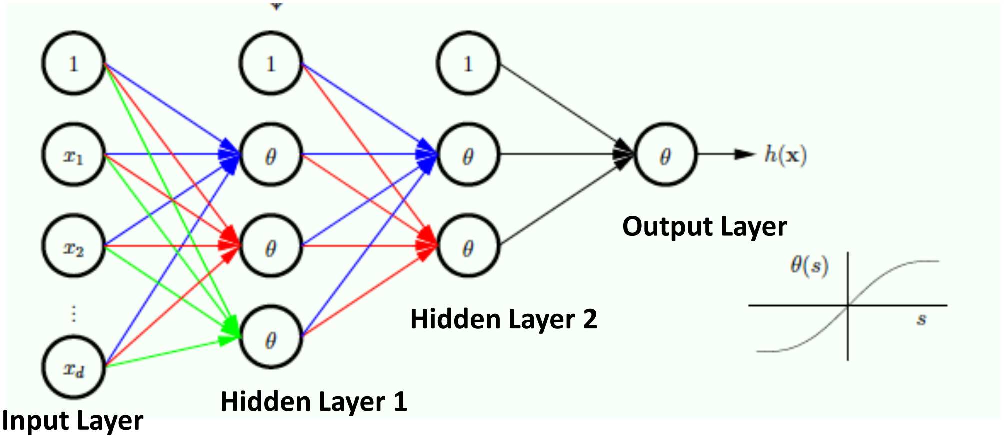
Inputs

Output \`i' (Hidden Layer 2)

Weights: Layer 2

Output

Neural Networks - Architecture



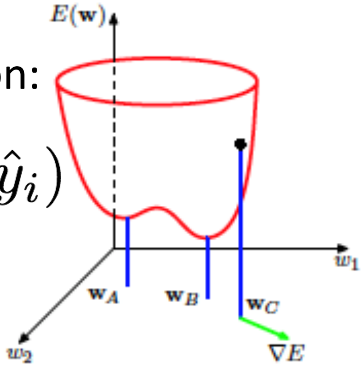
$$y = \text{sign}\left[W_3^T\left\{\theta\left(W_2^T\left\{\theta(W_1^T \cdot \mathbf{x})\right\}\right\}\right)\right]$$

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:

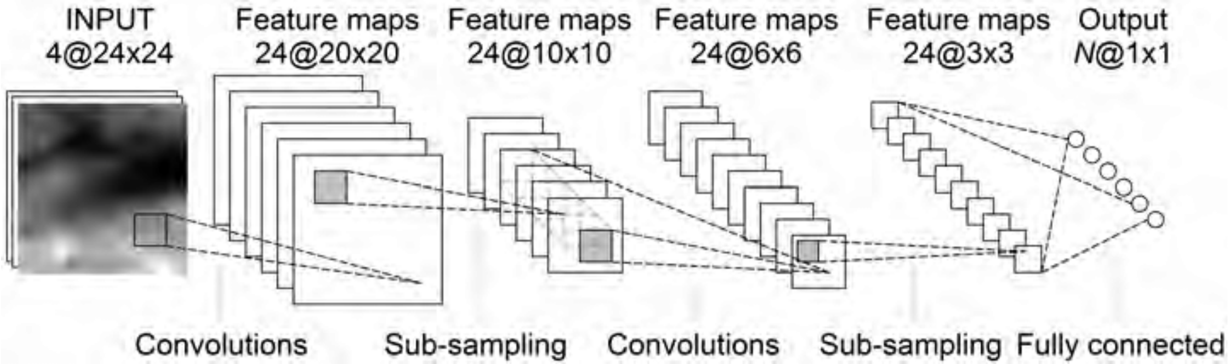
$$E(\mathbf{w}) = \sum_{i=1}^{n} \ell_{\mathbf{w}}(y_i, \hat{y}_i)$$



Training Procedure

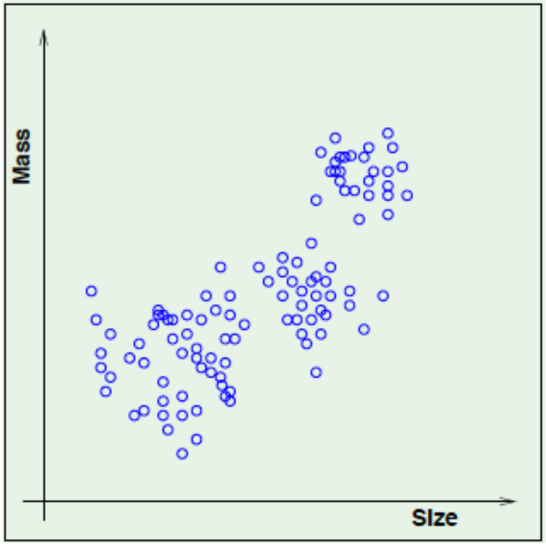
- Stochastic Gradient Descent
- Backpropagation Algorithm
- Early Stopping Rule
- Dropout and Regularization
- *Convolutional Neural Networks.*

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



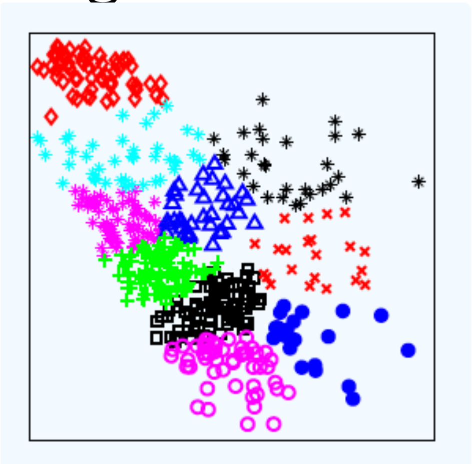
Training Data:

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

No Labels!

Still Need to do Learning!

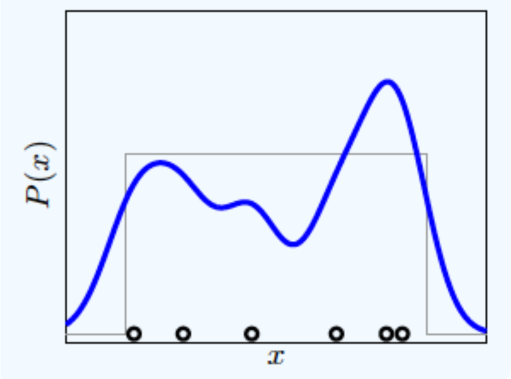
k-Means Clustering



- 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

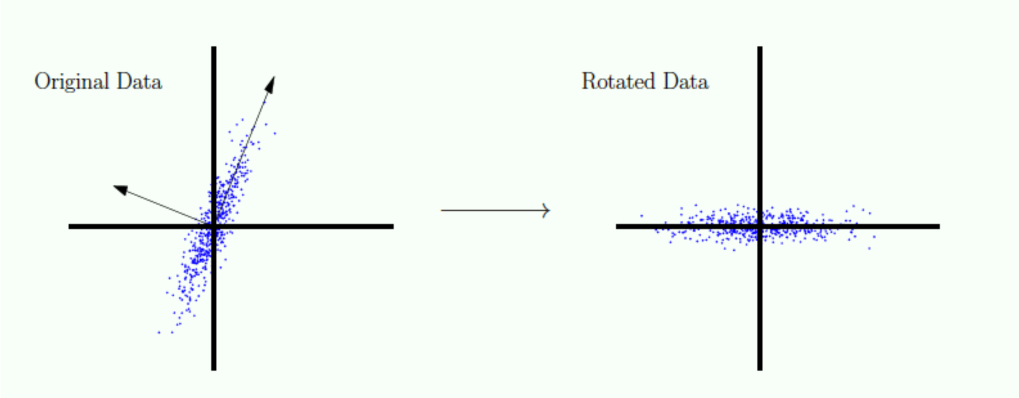
Density Estimation - Gaussian Mixture Models



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

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

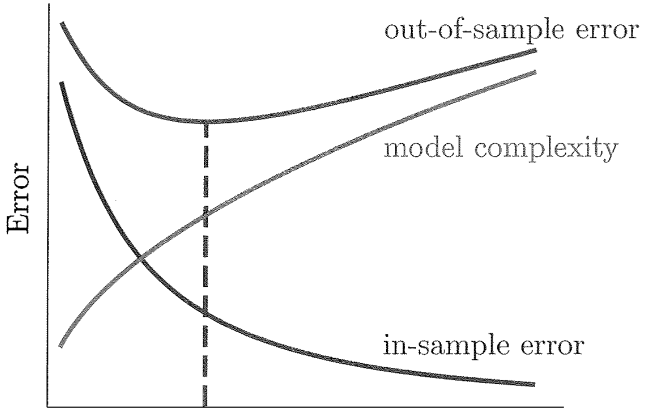
Complexity of Hypothesis Class

Number of Training Examples

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

Vapnik Chervonenkis (VC) dimension

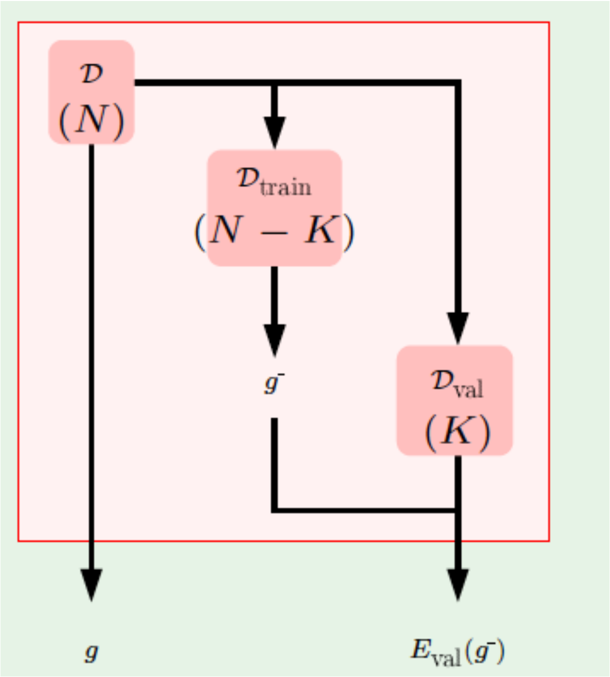
- VC Dimension provides a natural measure for complexity of class
- Linear Models: VC Dimension = dimension + 1
- Neural Networks (roughly) = # of Weight Parmeters



VC Dimension

Techniques

Validation

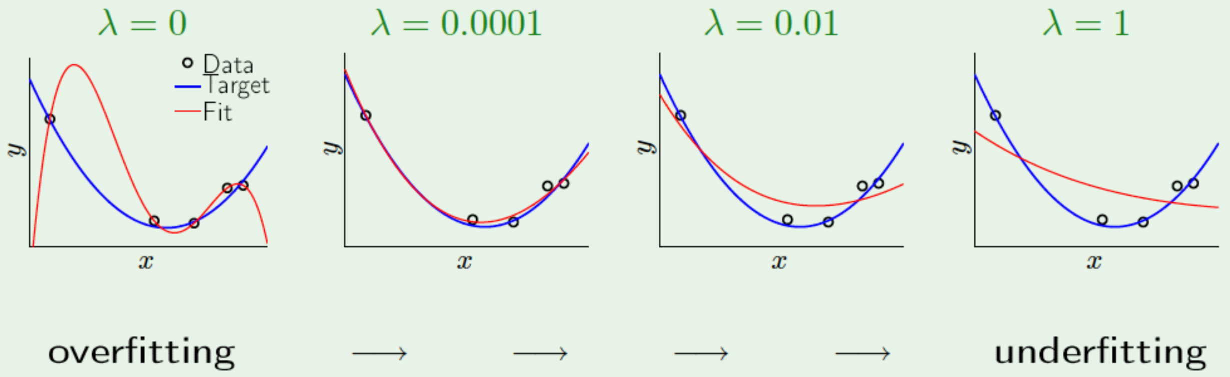


- Divide Training Set into 2 Parts
  - Learning Set
  - Validation Set
- Learning Set: Train Model
- Validation Set: Estimate Test Error
- Applications of Validation
  - Model Selection
  - Selection of Hyperparameters (e.g., regularization coefficient)
  - Number of Training Steps (Early Stopping)
-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

Regularization

$y = \mathbf{w}^T \cdot \mathbf{z}, \quad \mathbf{z} = [1 \quad x \quad \ldots, x^M]^T$

Minimizing $E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N}\, \mathbf{w}^{\mathsf{T}}\mathbf{w}$ for different λ's:



- Weight Decay Method
- Neural Networks: Drop Out Method

Logistics

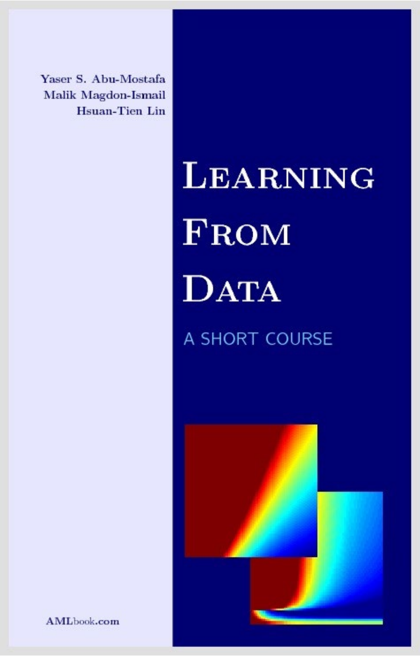
Course Staff

Instructors:

- Ben Liang (Section 1)
- Ashish Khisti (Section 2)

- Tutorials
  - Tutorial 1: Fridays 9 - 11 am
  - Tutorial 2: Fridays 10 - 12 pm
  - Tutorial 3: Mondays 12 - 2 pm
  - Tutorial 4: Tuesdays 9 - 11 am

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 of Jan 10	Intro/Linear Classification		
Jan 17	Linear Regression, Regularization	Tut 1	Assignment 1 posted
Jan 24	Logistic Regression	Tut 2	
Jan 31	Gradient Descent	Tut 3	**Assignment 1 due**, Assignment 2 posted
Feb 7	Multiplayer Perceptron, Backpropagation	Tut 4	
Feb 14	Deep Learning	Tut 5	
Feb 21	**Reading Week**		
Feb 28	Unsupervised Learning: Clustering and Density Estimation	Tut 6	**Assignment 2 due**, Assignment 3 posted
Mar 7	EM Algorithm	Tut 7	**Midterm Exam**
Mar 14	Support Vector Machine	Tut 8	
Mar 21	Support Vector Machine, PAC Learning	Tut 9	
Mar 28	PAC Learning	Tut 10	
Apr 4	PAC Learning, Bias-Variance Tradeoff	Tut 11	**Assignment 3 due**
Apr 11	Validation, Cross-Validation		Last day of class: Apr 14

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: 20% (March 7th)
- Homeworks: 10%
- Programming Assignment (3 x 15% = 45%)
- Final Exam: 25% in Exam Week

- Assignments will be done individually.

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

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