

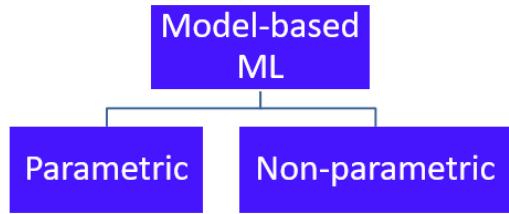
Statistical Methods in AI (CSE/ECE 471)

Lecture-22: Panel Discussion

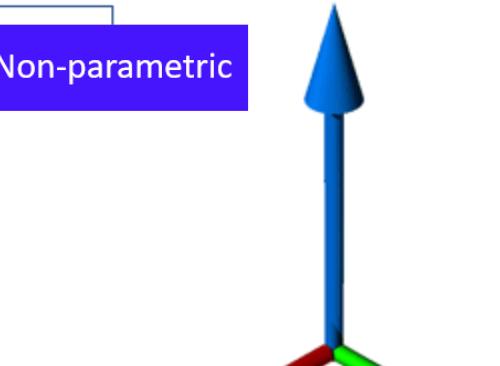
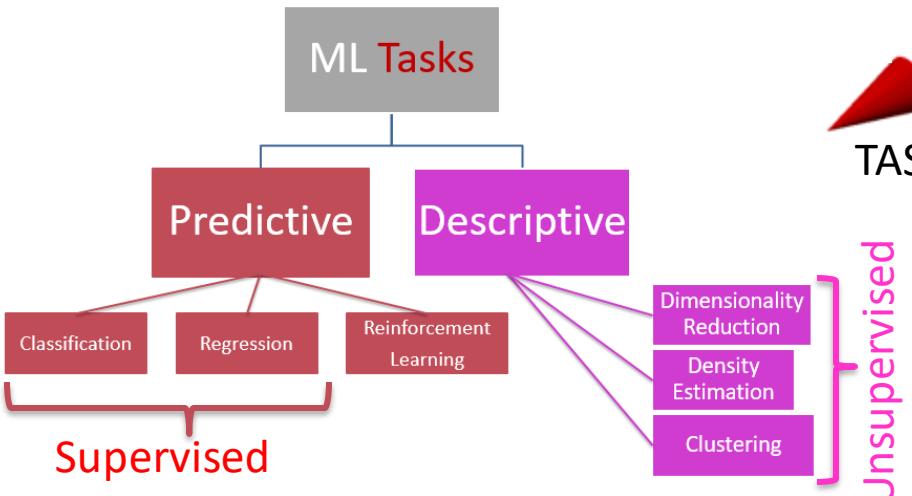
Ravi Kiran

Center for Visual Information Technology (CVIT), IIIT Hyderabad





ALGORITHMS

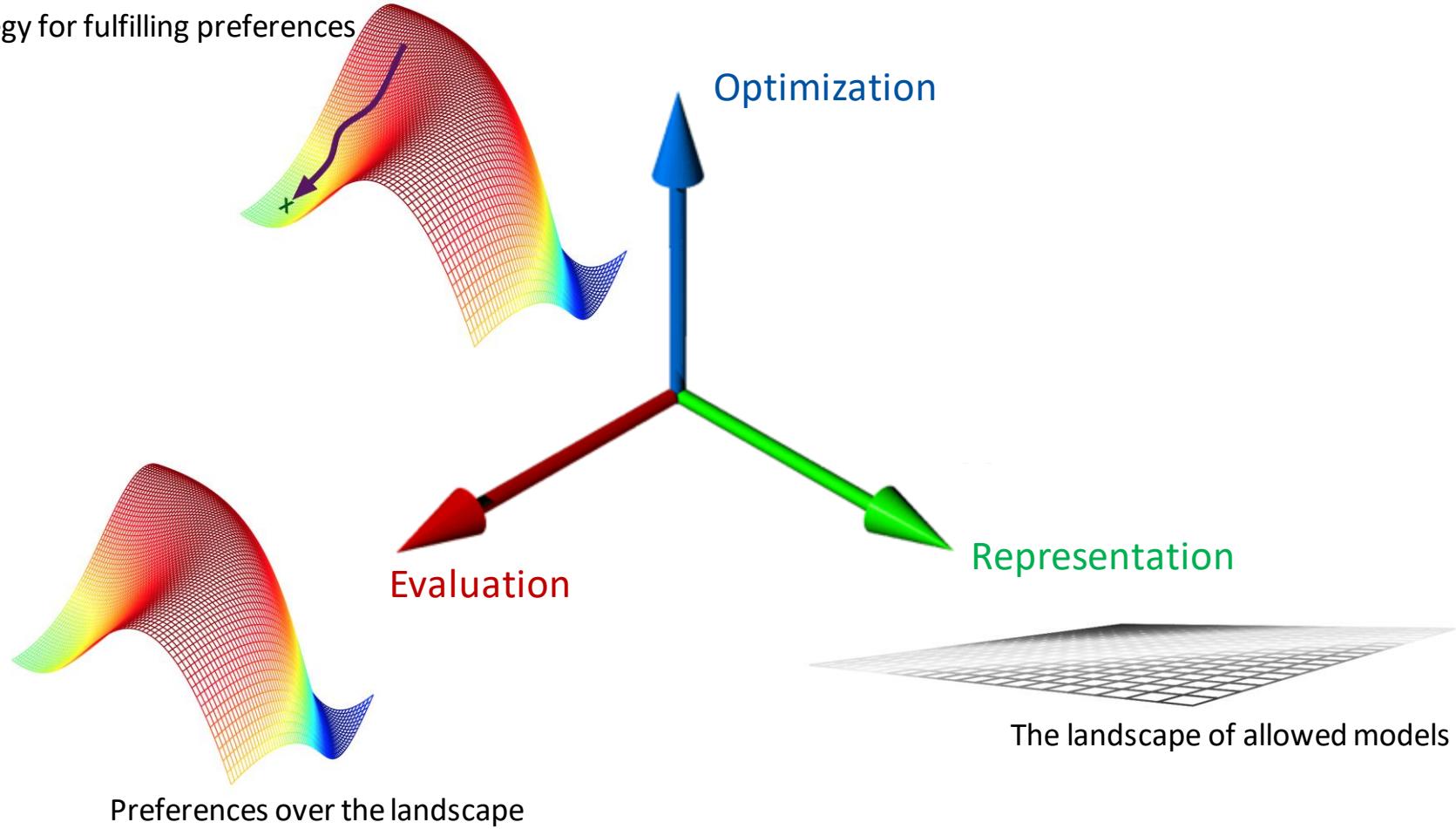


TASKS

DATA

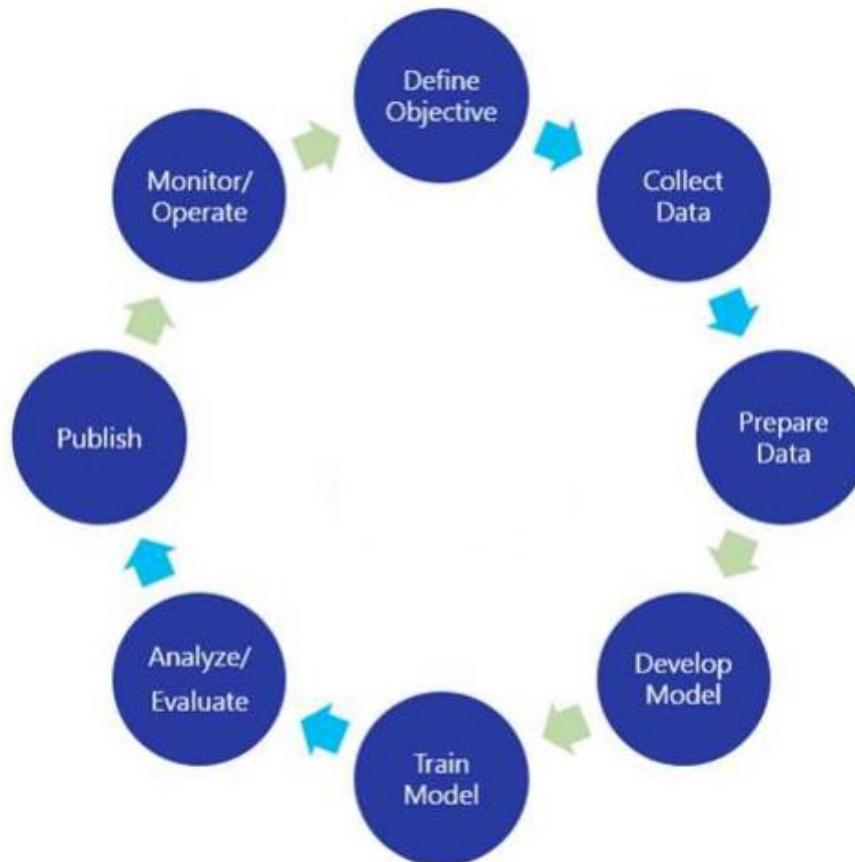
- Fully Observed
- Partially Observed
 - Some variables systematically not observed (e.g. 'topic' of a document)
 - Some variables missing some of the time (e.g. 'faulty sensor' readings)
- Actively collect / sense data (e.g. exploration robots)

Strategy for fulfilling preferences

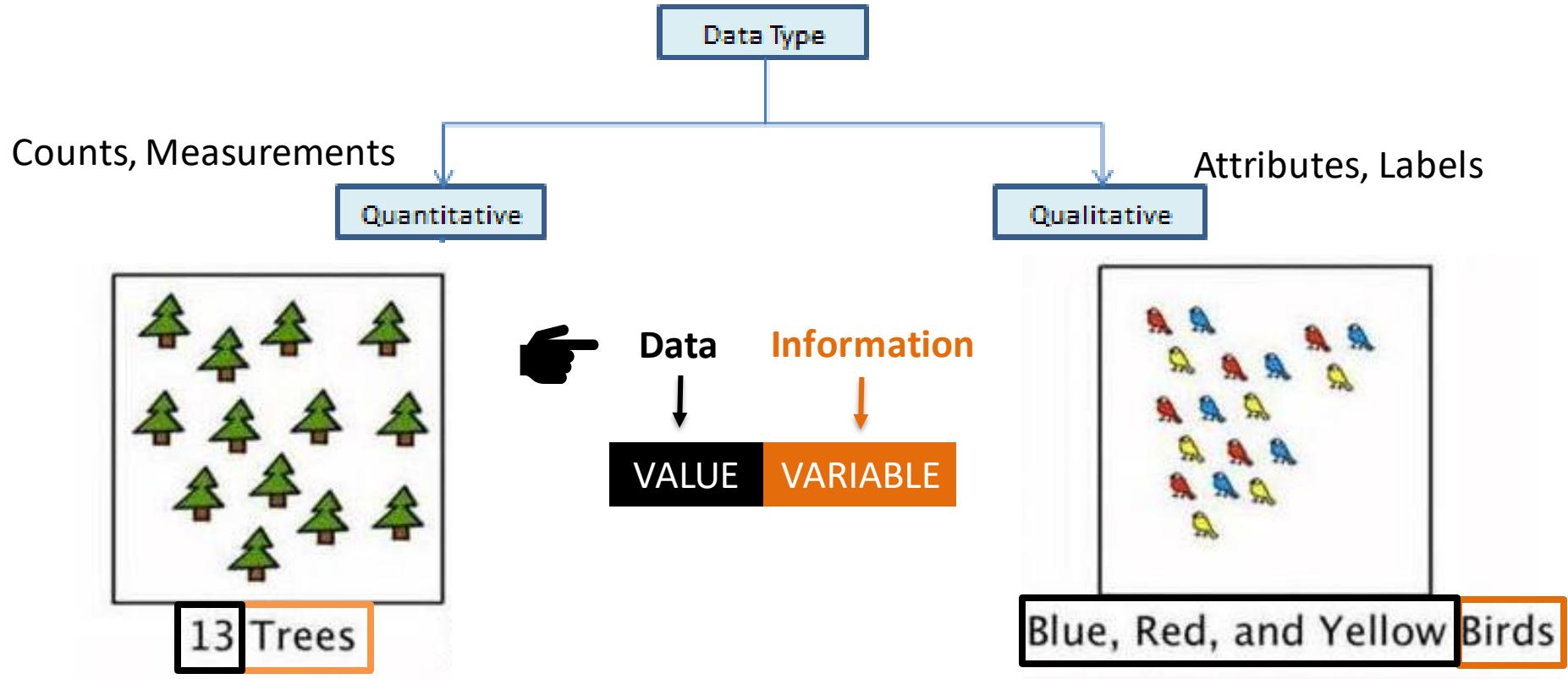


Preferences over the landscape

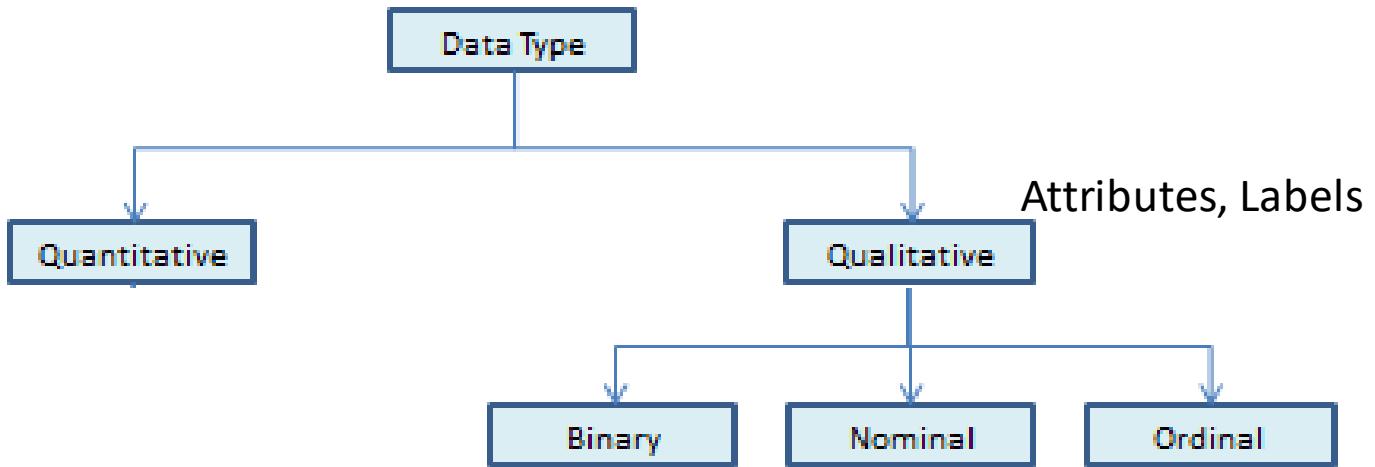
Workflow of a Machine Learning Problem



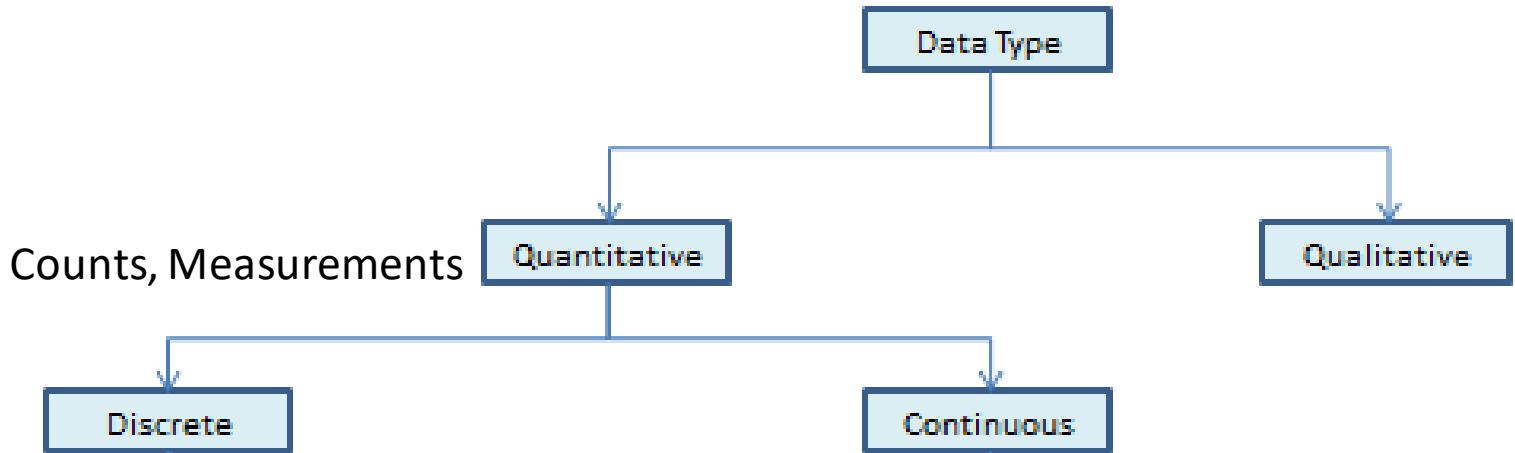
Taxonomy of data variables



Taxonomy of data

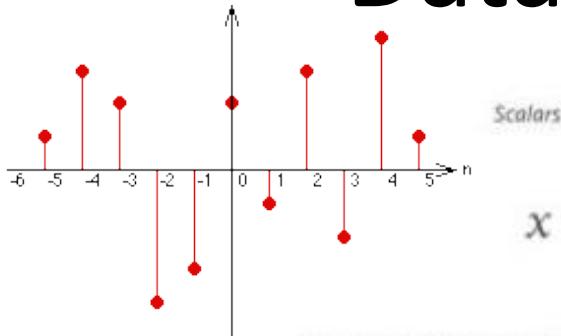


Taxonomy of data

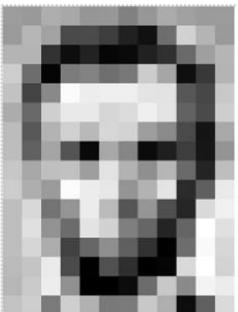


- # of CPU cores
- # of courses taken in a semester
- # of times word 'sale' appears in a doc

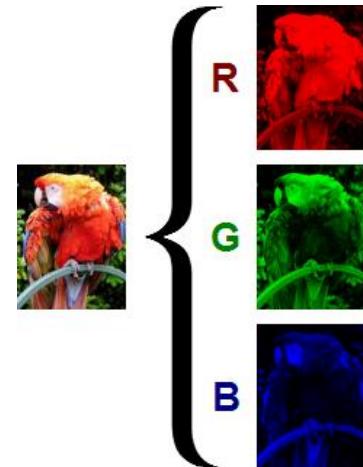
Data Representations

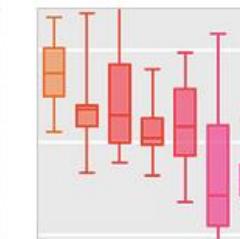
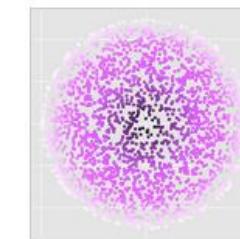
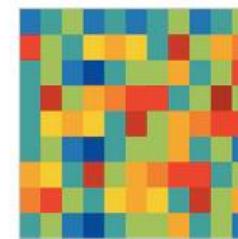
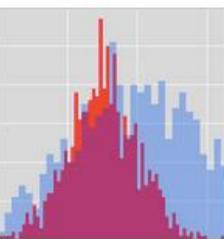
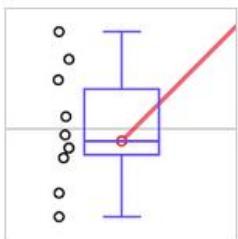
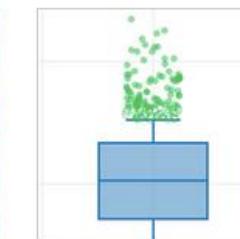
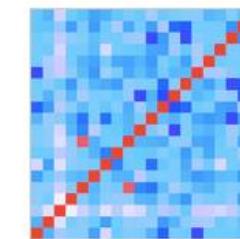
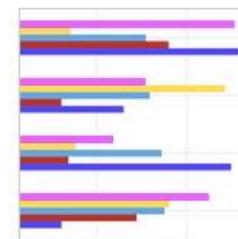
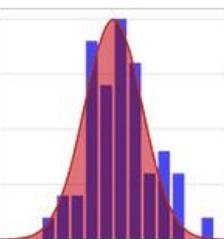
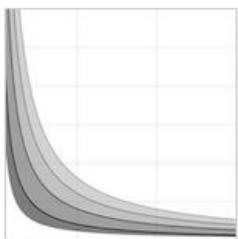
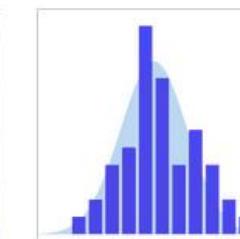
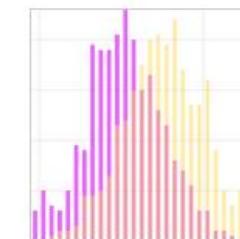
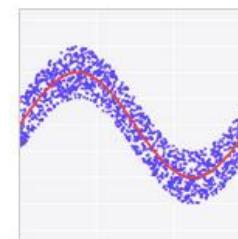
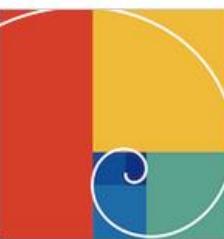
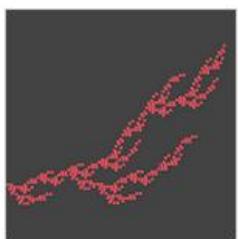
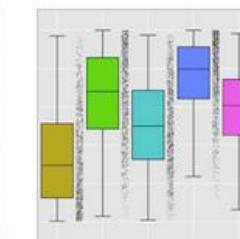
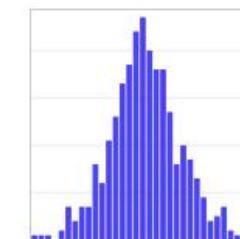
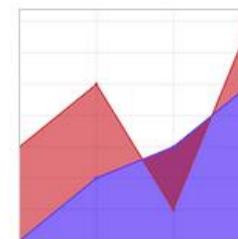
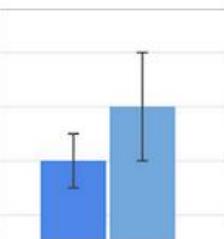
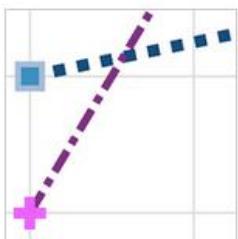


2-d image



$$X = \begin{bmatrix} x & \dots & x_N \end{bmatrix} = \begin{bmatrix} X_{1,1} & & X_{N,1} \\ \vdots & \ddots & \vdots \\ X_{1,M} & & X_{N,M} \end{bmatrix}$$





Supervised Learning





Classification

Binary

Multi-class

Multi-label

Structure

$\{0,1\}$

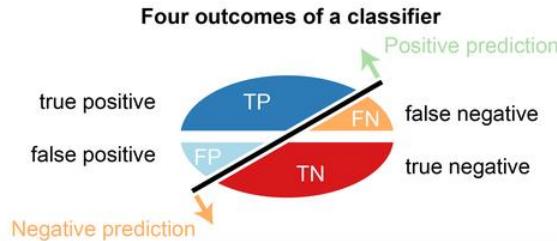
1-of-K

n-of-K

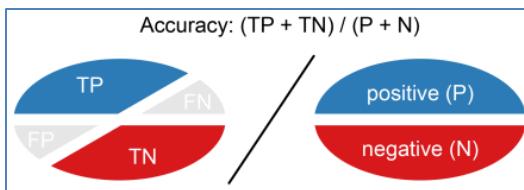


E.g. graph/sequence

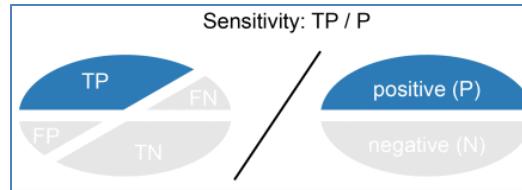
Summary of Measures



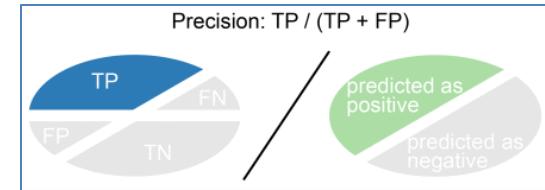
n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
55		110	



% of correct predictions



% of + class correctly predicted
[aka Recall / TPR]



correct prediction of + class

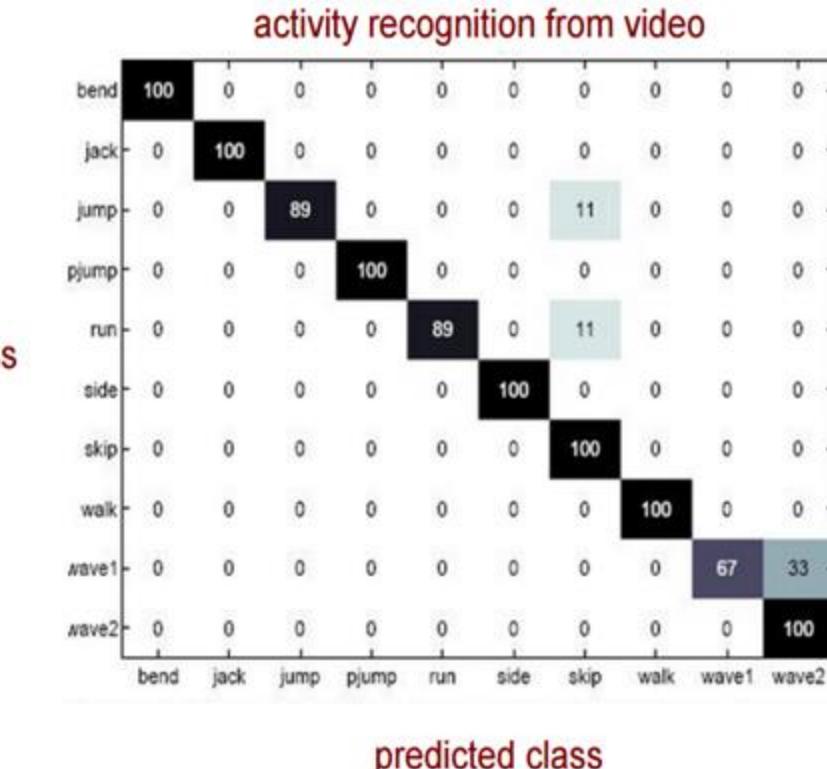


% of - class incorrectly predicted

Multi-class problems - Confusion matrix

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

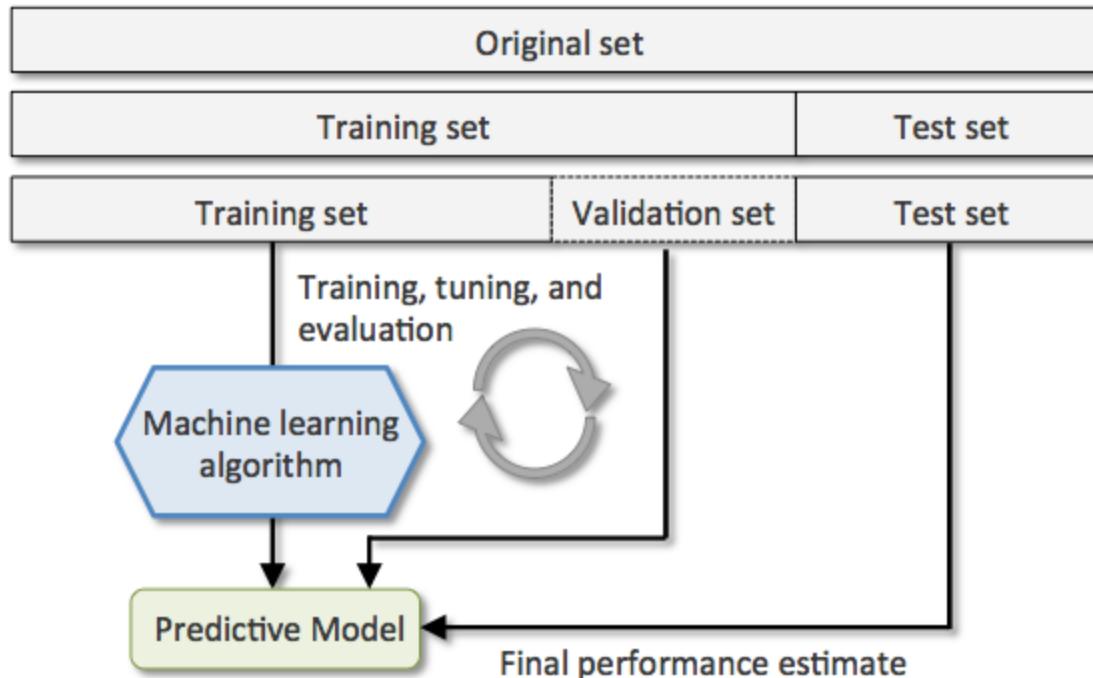
actual class



Courtesy: vision.jhu.edu

Avg. accuracy may not be very meaningful with imbalanced class label distribution

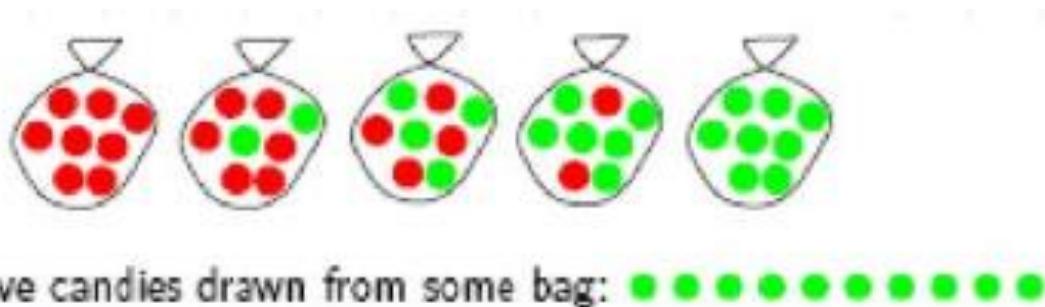
The Train-Validation-Test paradigm



- Decision Trees
- K-NN

Data – a probability-based perspective

- The basis for Statistical Learning Theory



- Domain described by random variables (r.v.)
 - $X = \{\text{apple, grape}\}$
 - $b_i \in [1,5]$
- **Data = Instantiation of some or all r.v.'s in the domain**

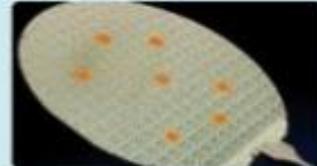
Uncertainty arises from many sources

Process Uncertainty

Processes contain
“randomness”



Uncertain travel times



Semiconductor yield

Data Uncertainty

Data input is uncertain



Intended Spelling → Text Entry → Actual Spelling



GPS Uncertainty



Testimony {Paris Airport} Ambiguity

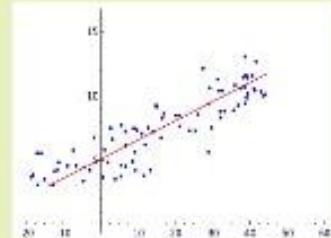


Contaminated? (John Smith, Dallas)
Rumors (John Smith, Kansas)

Conflicting Data

Model Uncertainty

All modeling is approximate



Fitting a curve to data



Forecasting a hurricane
(www.noaa.gov)

Independent vs. Dependent Events



Using the bag of marbles on the left, what is the probability of pulling a black marble two times in a row? $P(\text{black}, \text{black})$

When you put 1st marble back in
(Independent Events)

$$\frac{2}{10} * \frac{2}{10}$$

$$\frac{1}{5} * \frac{1}{5} = \frac{1}{25}$$

When you KEEP 1st marble
(Dependent Events)

$$\frac{2}{10} * \frac{1}{9}$$

$$\frac{1}{5} * \frac{1}{9}$$

$$P(A \text{ and } B) = P(A) \times P(B)$$

$$P(A \text{ and } B) = P(A) \times P(B|A)$$

Probability of B given A

Marginal Probabilities



$\begin{cases} x = 1 & (\text{Rains}) \\ x = 0 & (\text{Doesn't rain}) \end{cases}$



$\begin{cases} y = 1 & (\text{Have umbrella}) \\ y = 0 & (\text{Don't have umbrella}) \end{cases}$

$$\Pr(x = 1) = 0.6$$

$$\Pr(x = 0) = 0.4$$

$$\Pr(y = 1) = 0.3$$

$$\Pr(y = 0) = 0.7$$

Joint Probability



$$\begin{cases} x = 1 & (\text{Rains}) \\ x = 0 & (\text{Doesn't rain}) \end{cases}$$

$$\begin{cases} y = 1 & (\text{Have umbrella}) \\ y = 0 & (\text{Don't have umbrella}) \end{cases}$$

$$\begin{array}{ll} \Pr(x=1) = 0.6 & \\ \Pr(x=0) = 0.4 & \\ \Pr(y=1) = 0.3 & \\ \Pr(y=0) = 0.7 & \end{array}$$

$$\Pr(x=0) = \sum_{y=0}^1 \Pr(x=0, y)$$

$$\begin{aligned} &= \Pr(x=0, y=0) + \Pr(x=0, y=1) \\ &= 0.28 + 0.12 = 0.4 \end{aligned}$$

Case 1 : Rains but you have an umbrella

$$\begin{aligned} \Pr(x=1, y=1) &= \Pr(x=1) \times \Pr(y=1) \\ &= 0.6 \times 0.3 \\ &= 0.18 \end{aligned}$$

Case 2 : Rains but you DON'T have an umbrella

$$\begin{aligned} \Pr(x=1, y=0) &= \Pr(x=1) \times \Pr(y=0) \\ &= 0.6 \times 0.7 \\ &= 0.42 \end{aligned}$$

Conditional Probability



Given

$x = 1$ (Rains)



What's the Probability of
 $y = 1$ (Bring umbrella)



$\begin{cases} x = 1 & \text{(Rains)} \\ x = 0 & \text{(Doesn't rain)} \end{cases}$

$\begin{cases} y = 1 & \text{(Have umbrella)} \\ y = 0 & \text{(Don't have umbrella)} \end{cases}$

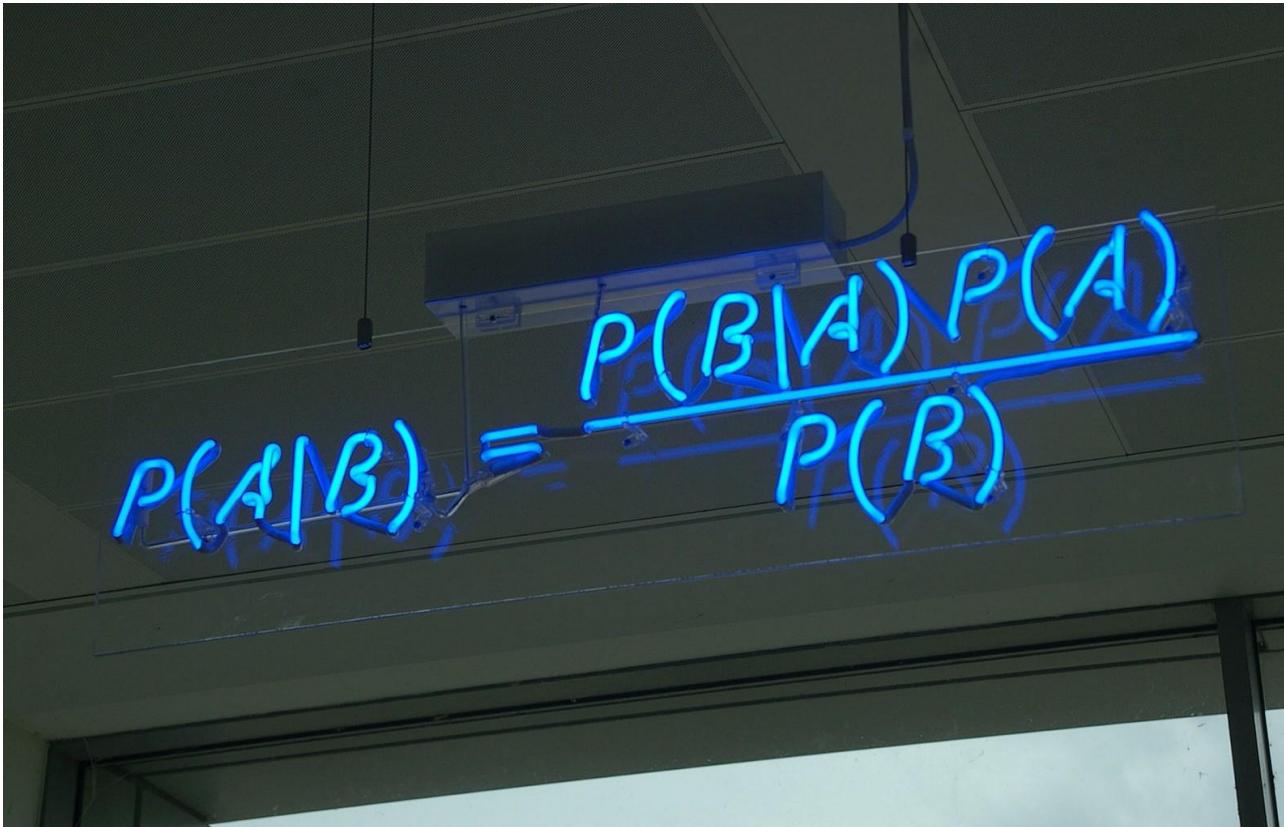
$\Pr(x = 1) = 0.6$

$\Pr(x = 0) = 0.4$

$\Pr(y = 1) = 0.3$

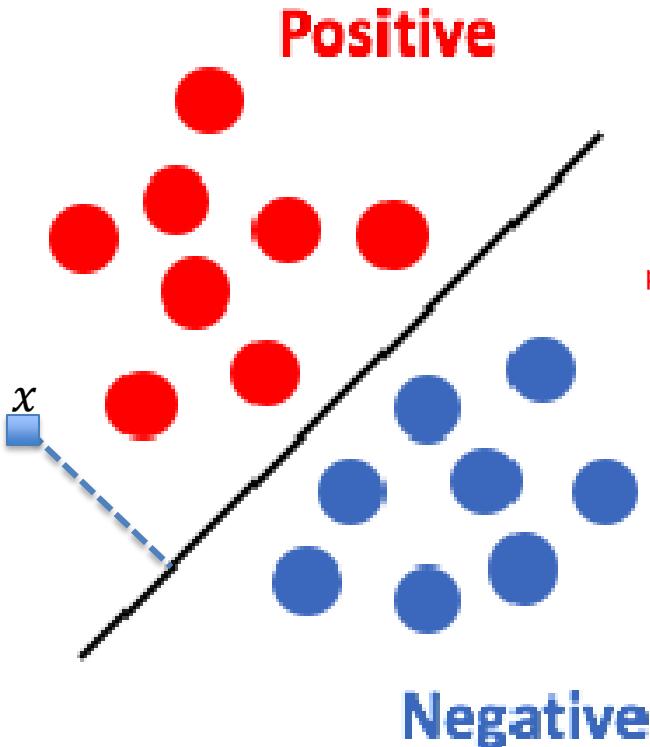
$\Pr(y = 0) = 0.7$

Bayes' Rule


$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

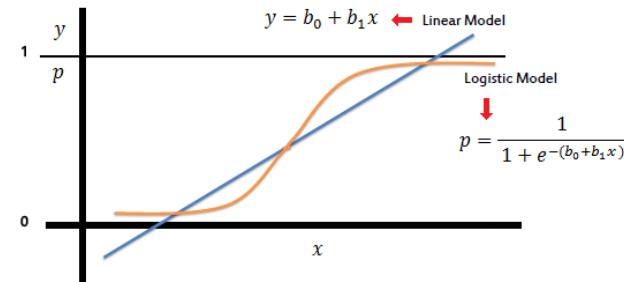
- Decision Trees
- K-NN
- Naïve Bayes

Logistic Regression - Intuition



$$p(X)$$

$p(X) = p(Y=1|x) = \text{probability that } x \text{ belongs to positive class}$



$$y = b_0 + b_1 x$$

Logistic Model

$$p = \frac{1}{1 + e^{-(b_0 + b_1 x)}}$$

$$\Rightarrow p(X) = \frac{e^{(\beta_0 + \beta_1 x)}}{e^{(\beta_0 + \beta_1 x)} + 1}$$

$$\Rightarrow p(e^{(\beta_0 + \beta_1 x)} + 1) = e^{(\beta_0 + \beta_1 x)}$$

$$\Rightarrow p \cdot e^{(\beta_0 + \beta_1 x)} + p = e^{(\beta_0 + \beta_1 x)}$$

$$\Rightarrow p = e^{(\beta_0 + \beta_1 x)} - p \cdot e^{(\beta_0 + \beta_1 x)}$$

$$\Rightarrow p = e^{(\beta_0 + \beta_1 x)}(1 - p)$$

$$\Rightarrow \frac{p}{1 - p} = e^{(\beta_0 + \beta_1 x)}$$

$$\Rightarrow \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x$$

Distance of x from boundary

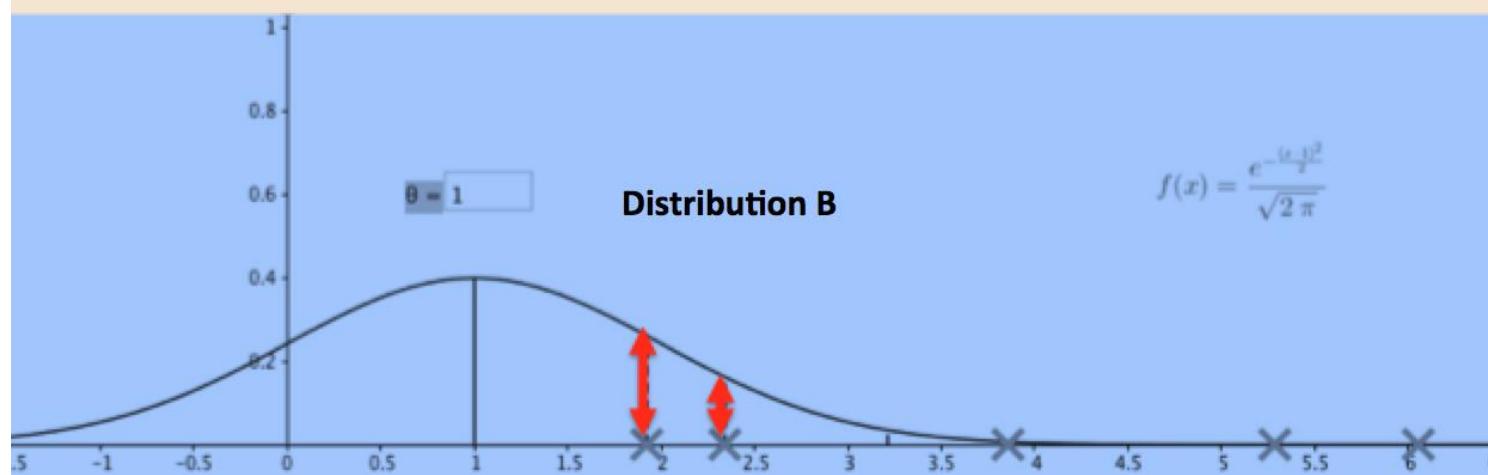
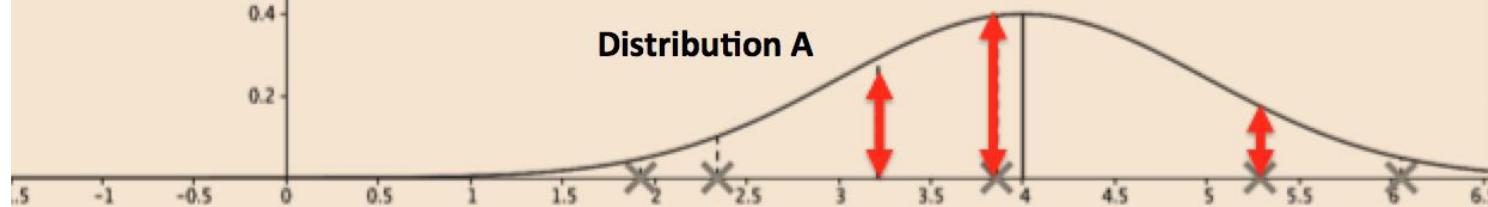
- Decision Trees
- K-NN
- Naïve Bayes
- Logistic Regression

Maximum Likelihood Estimator will maximize $p(x_1)p(x_2)p(x_3)\dots$
-> called the Likelihood function

Pick value of theta that results in a longer red arrows

$$\theta = 4$$

$$f(x) = \frac{e^{-\frac{(x-\mu)^2}{2}}}{\sqrt{2\pi}}$$



Taxonomy of classifiers : a modelling perspective

	Probabilistic	Non-Probabilistic
Discriminative	<ul style="list-style-type: none">• Logistic Regression• Probabilistic neural nets• <p>Model $p(\text{class} = C x)$</p>	<ul style="list-style-type: none">• K-nn• Linear classifier• SVM• Neural networks•
Generative	<ul style="list-style-type: none">• Naïve Bayes• Model-based (e.g., GMM)•	

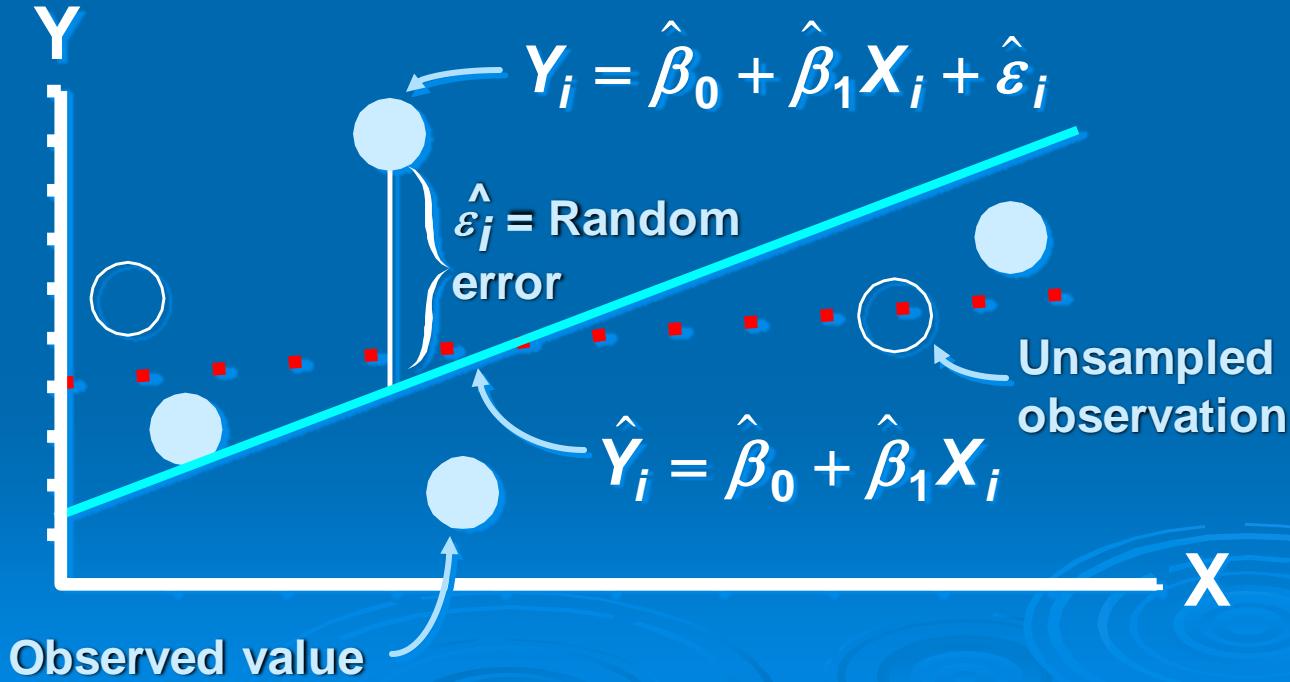
Model the classification rule directly

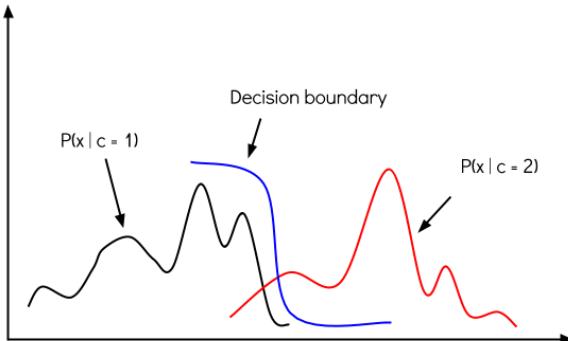
Model $p(x | \text{class} = C)$

Supervised Learning



Sample Linear Regression Model





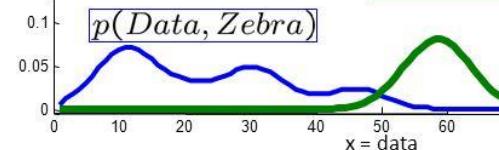
Discriminative vs. generative

- Generative model

(The artist)

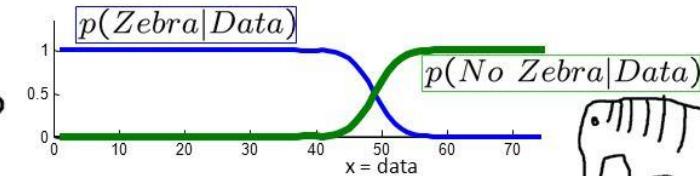
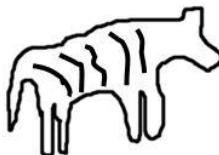


$p(\text{Data}, \text{No Zebra})$



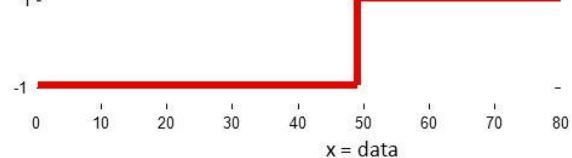
- Discriminative model

(The lousy painter)

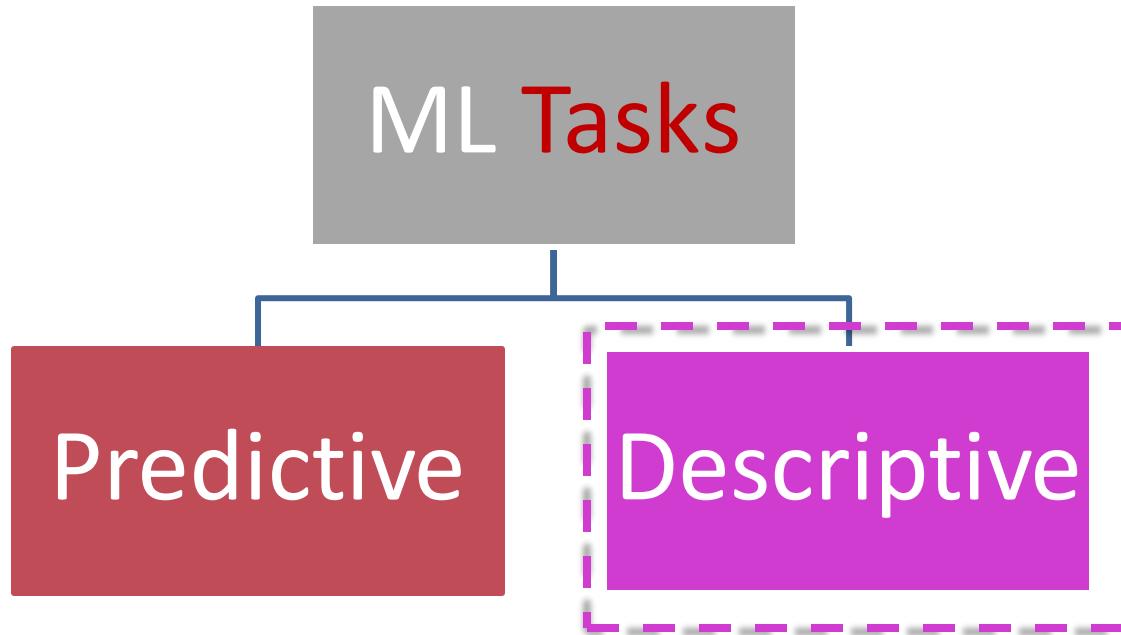


- Classification function

$\text{label} = F_{\text{Zebra}}(\text{Data})$



ML Tasks



ML::Tasks → Descriptive

- Study/Exploit the ‘structure’ of data
 - Density Estimation
 - Clustering
 - Dimensionality Reduction
- Also studied as ‘Unsupervised Learning’
 - ‘Input’ data without paired ‘Output’

$$\{x^{(1)}, \dots, x^{(m)}\} \quad x^{(i)} \in \mathbb{R}^n$$

The k -means clustering algorithm is as follows:

1. Initialize **cluster centroids** $\mu_1, \mu_2, \dots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {

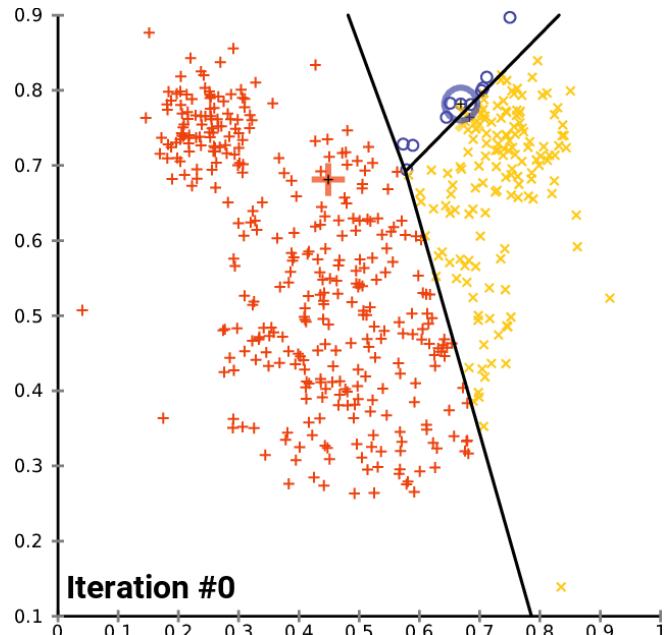
For every i , set

$$c^{(i)} := \arg \min_j \|x^{(i)} - \mu_j\|^2.$$

For each j , set

$$\mu_j := \frac{\sum_{i=1}^m \mathbf{1}\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m \mathbf{1}\{c^{(i)} = j\}}.$$

}



$$\{x^{(1)}, \dots, x^{(m)}\} \quad x^{(i)} \in \mathbb{R}^n$$

EXPECTATION MAXIMIZATION

The k -means clustering algorithm is as follows:

1. Initialize **cluster centroids** $\mu_1, \mu_2, \dots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {

For every i , set

$$c^{(i)} := \arg \min_j \|x^{(i)} - \mu_j\|^2.$$

E

Assignment step: Assign each data point to the closest cluster

For each j , set

$$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$

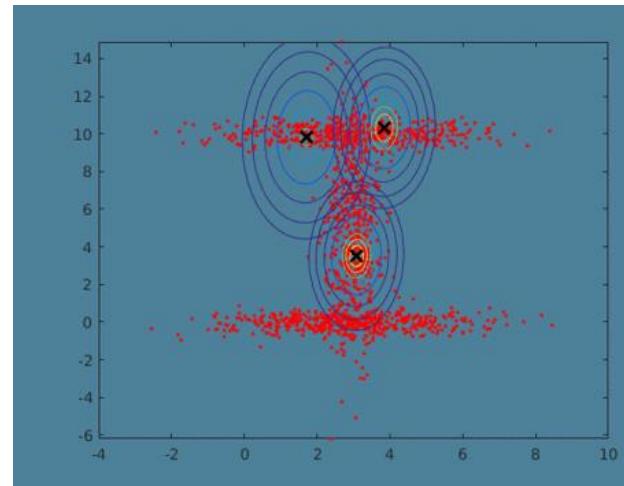
M

Refitting step: Move each cluster center to the center of the data assigned to it

}

GMM - Algorithm

- Initialize the means μ_k , covariances Σ_k and mixing coefficients π_k
- Iterate until convergence:
 - ▶ E-step: Evaluate the responsibilities given current parameters
$$\gamma_k^{(n)} = p(z^{(n)} | \mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)} | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}^{(n)} | \mu_j, \Sigma_j)}$$
 - ▶ M-step: Re-estimate the parameters given current responsibilities
$$\mu_k = \frac{1}{N_k} \sum_{n=1}^N \gamma_k^{(n)} \mathbf{x}^{(n)}$$
$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^N \gamma_k^{(n)} (\mathbf{x}^{(n)} - \mu_k)(\mathbf{x}^{(n)} - \mu_k)^T$$
$$\pi_k = \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{n=1}^N \gamma_k^{(n)}$$

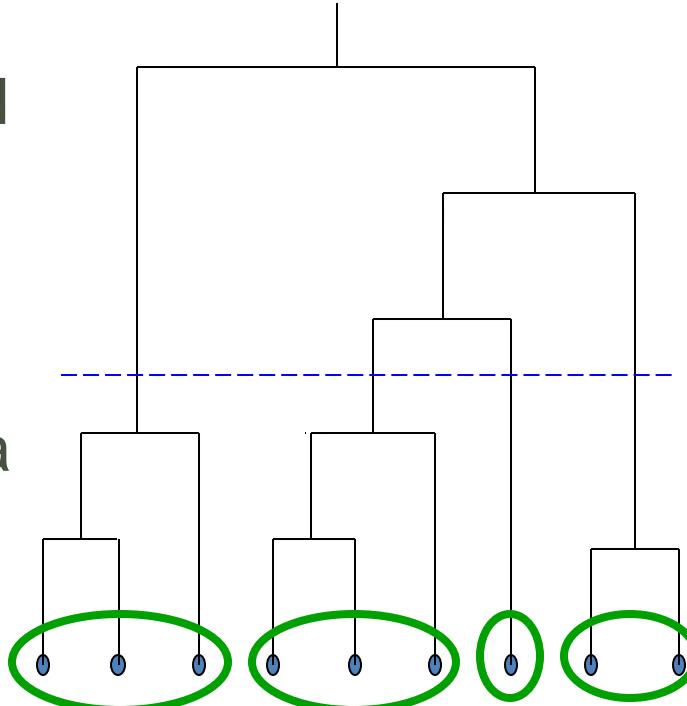


- ▶ Evaluate log likelihood and check for convergence

$$\ln p(\mathbf{X} | \pi, \mu, \Sigma) = \sum_{n=1}^N \ln \left(\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}^{(n)} | \mu_k, \Sigma_k) \right)$$

Dendrogram: Hierarchical Clustering

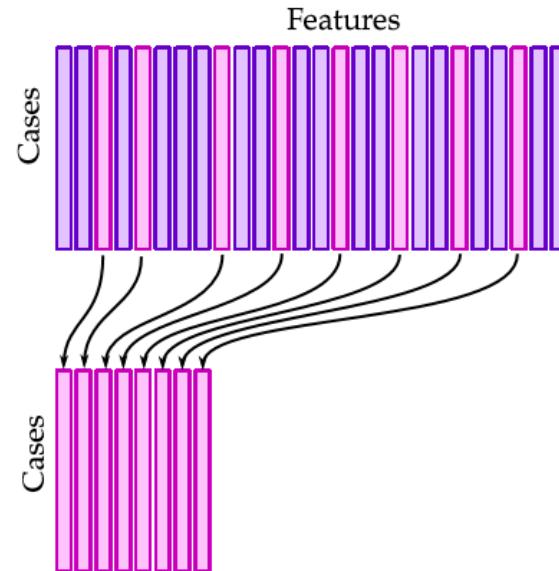
- Clustering obtained by cutting the dendrogram at a desired level: each **connected** component forms a cluster.



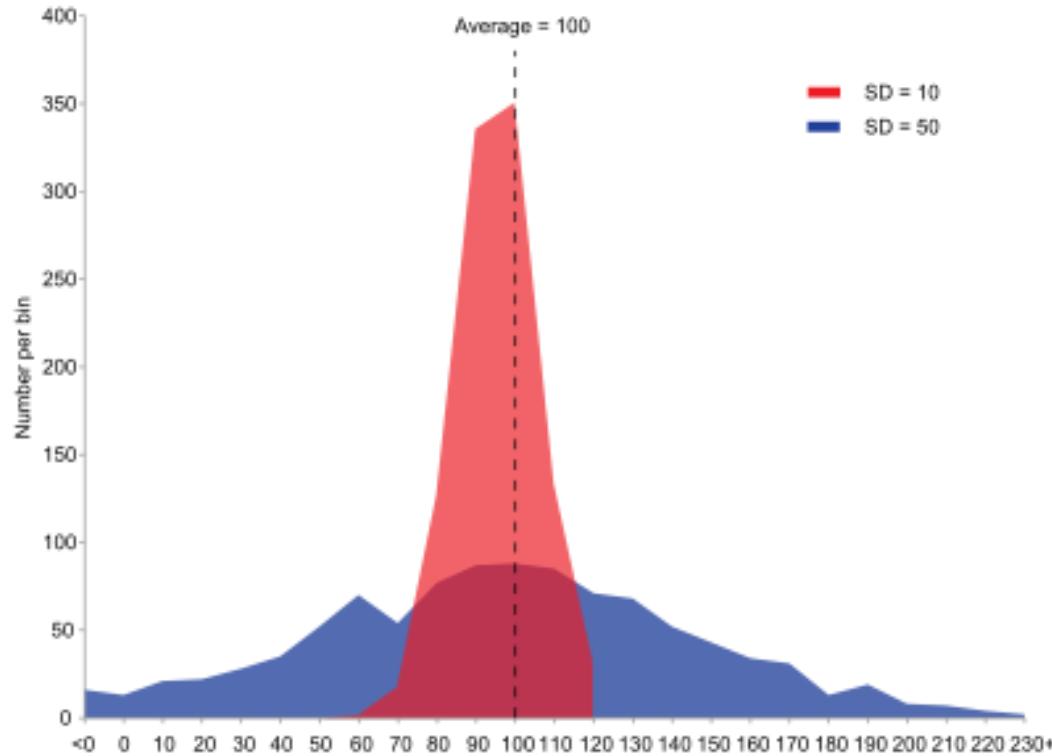
Feature Selection

- when we have a regular data set ($|T| > D$), but too flexible model, and/or
- when we have a high-dimensional data set with not enough data ($|T| < D$).

Data sets with thousands or millions of variables (features) are quite usual these days: we want to choose only those, which are needed to *construct simpler, faster, and more accurate models*.



Variance, Covariance and Correlation



Population

Variance:

$$s^2 = \frac{\sum (\bar{X} - X_i)^2}{N}$$

Standard Deviation:

$$s = \sqrt{\frac{\sum (\bar{X} - X_i)^2}{N}}$$

Sample Variance

$$s^2 = \frac{\sum (x - \bar{x})^2}{n - 1}$$

Sample Standard Deviation

$$s = \sqrt{\frac{\sum (x - \bar{x})^2}{n - 1}}$$

Covariance

Vectors 1 and 3

$$A = \begin{bmatrix} 1 & 3 & 5 \\ 5 & 4 & 1 \\ 3 & 8 & 6 \end{bmatrix}$$

Covariance

Cell (3, 1) or (1, 3)

$$\begin{bmatrix} 2.67 & 0.67 & -2.67 \\ 0.67 & 4.67 & 2.33 \\ -2.67 & 2.33 & 4.67 \end{bmatrix}$$

Covariance Matrix

$$\begin{bmatrix} 0.39701 & 0.51117 \\ 0.55582 & 0.93003 \\ 0.59403 & 0.96645 \\ 0.51544 & 0.29759 \\ 0.85313 & 0.18118 \\ 0.88564 & 0.69114 \end{bmatrix}$$

$$\left(\begin{array}{cccc} M_1 & M_2 & M_3 & \dots & M_n \\ S_1 & q_{1,1} & q_{1,2} & q_{1,3} & \dots & q_{1,n} \\ S_2 & q_{2,1} & q_{2,2} & q_{2,3} & \dots & q_{2,n} \\ S_3 & q_{3,1} & q_{3,2} & q_{3,3} & \dots & q_{3,n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ S_m & q_{m,1} & q_{m,2} & q_{m,3} & \dots & q_{m,n} \end{array} \right)$$

Variance:

$$s^2 = \frac{\sum(\bar{X} - X_i)^2}{N}$$

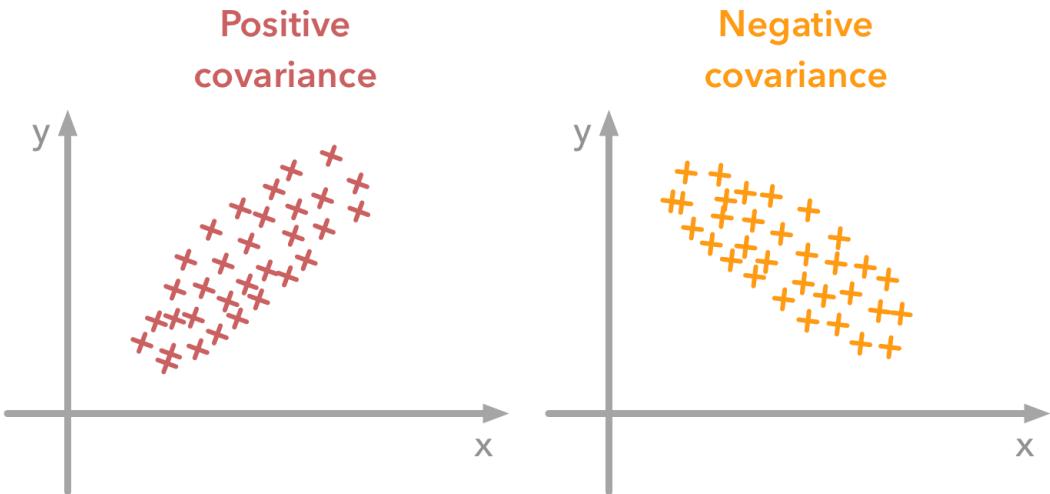
$$\text{cov}(X, Y) = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$$

$$\text{Cov}(M_a, M_b) = \frac{1}{m} \sum_{i=1}^m (q_{i,a} - \bar{q}_a)(q_{i,b} - \bar{q}_b)$$

$$C = \begin{pmatrix} \text{cov}(M_1, M_1) & \text{cov}(M_1, M_2) & \dots & \text{cov}(M_1, M_n) \\ \text{cov}(M_2, M_1) & \text{cov}(M_2, M_2) & \dots & \text{cov}(M_2, M_n) \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(M_n, M_1) & \text{cov}(M_n, M_2) & \dots & \text{cov}(M_n, M_n) \end{pmatrix}_{m \times m}$$

n-dimensional Covariance Matrix

Covariance

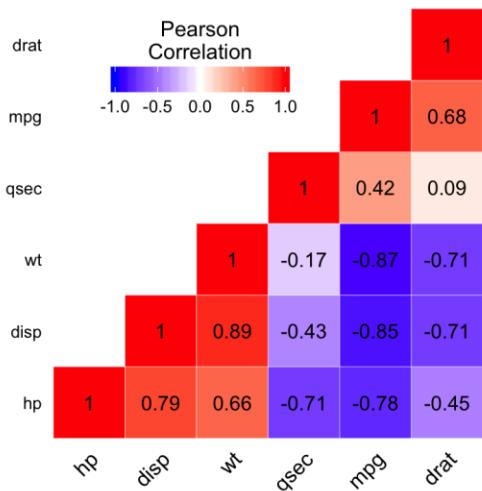


Variance:

$$s^2 = \frac{\sum (\bar{X} - X_i)^2}{N}$$

$$\text{cov}(X, Y) = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$$

Correlation



$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

$$\rho_{AB} = \frac{\text{cov}_{AB}}{\sigma_A \sigma_B}$$

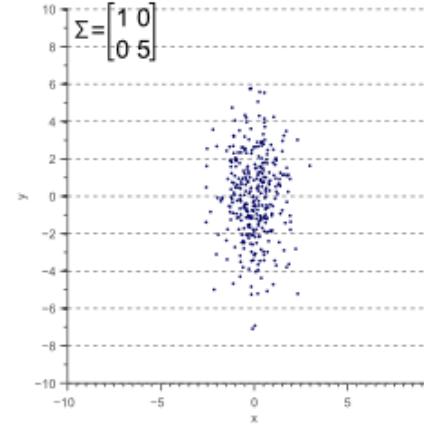
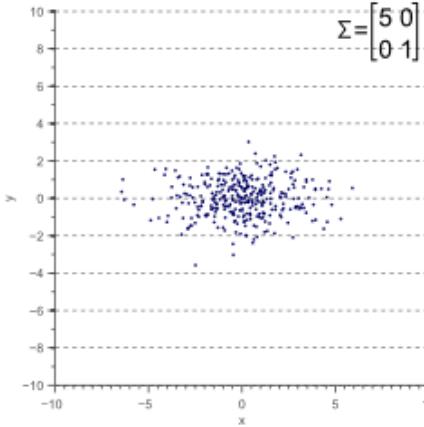
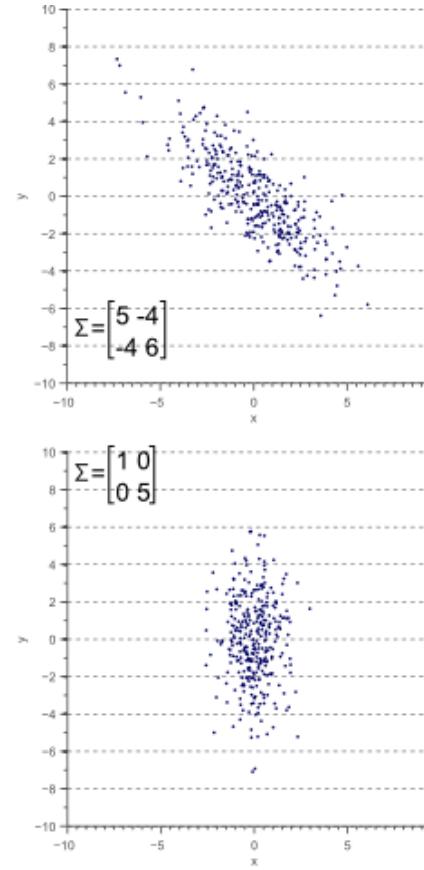
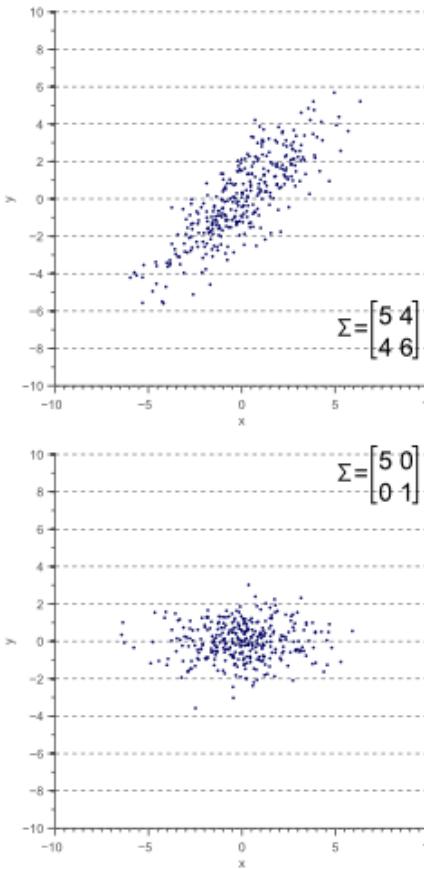
$$\text{cov}(\mathbf{X}, \mathbf{Y}) = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$$

$$r = \frac{1}{n-1} \sum_{i=1}^n \left(\frac{x_i - \bar{x}}{s_x} \right) \left(\frac{y_i - \bar{y}}{s_y} \right)$$

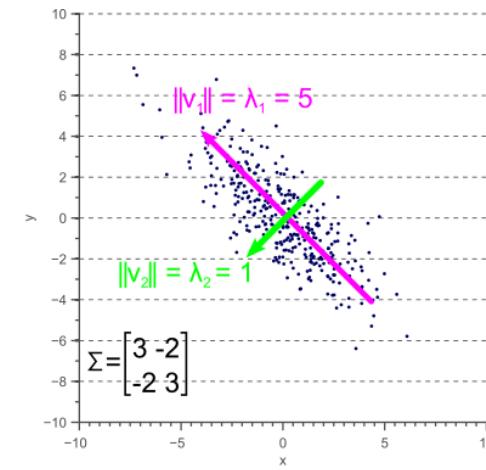
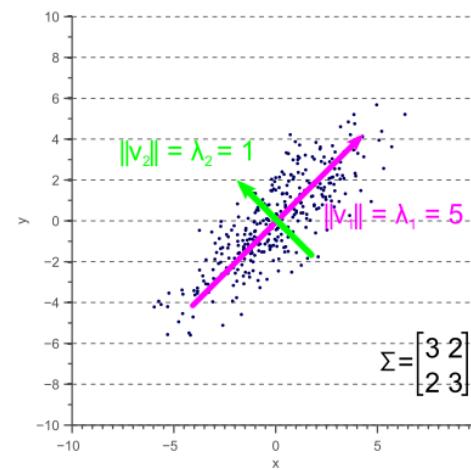
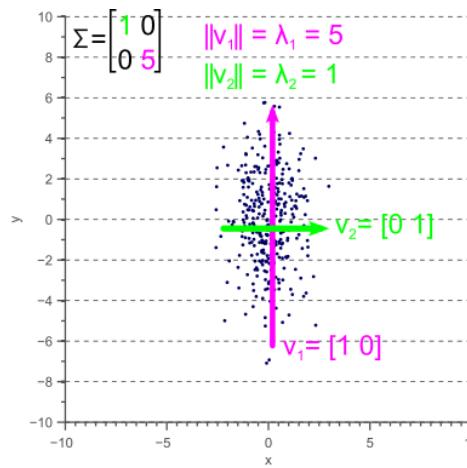
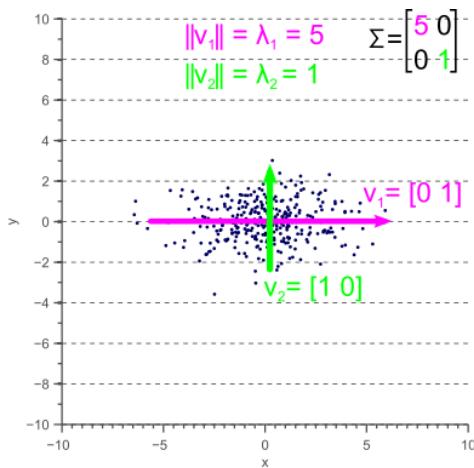


$$\mathbf{X}' = \frac{\mathbf{X} - \bar{\mathbf{x}}}{\sigma_{\mathbf{X}}}$$

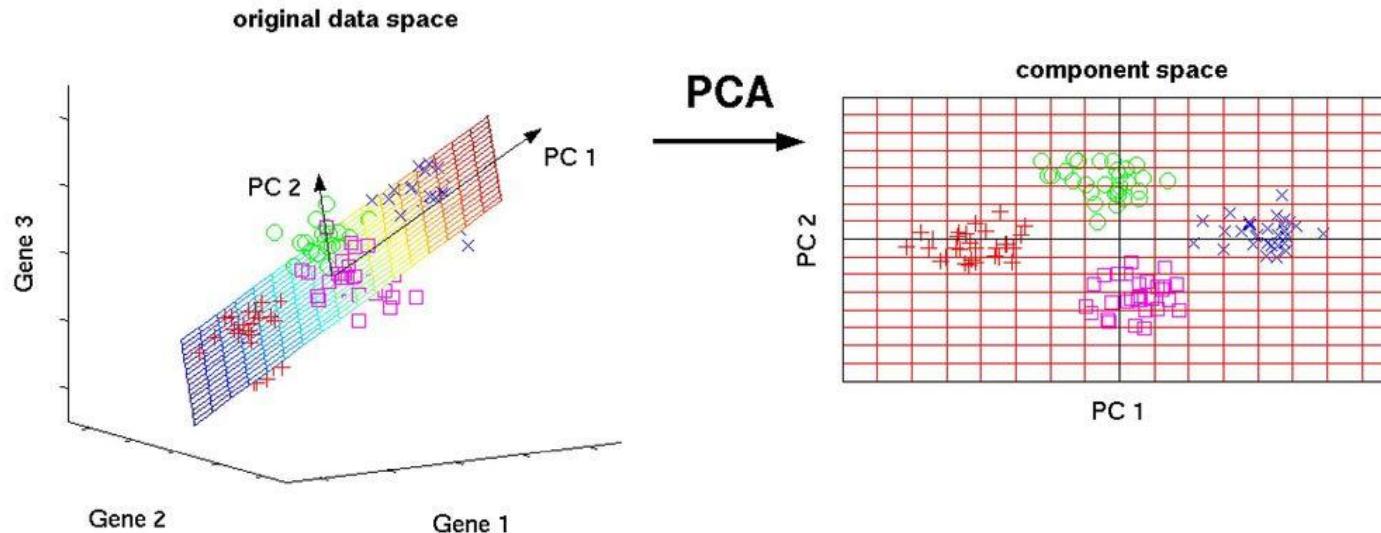
Covariance Matrix encodes spread and orientation of data



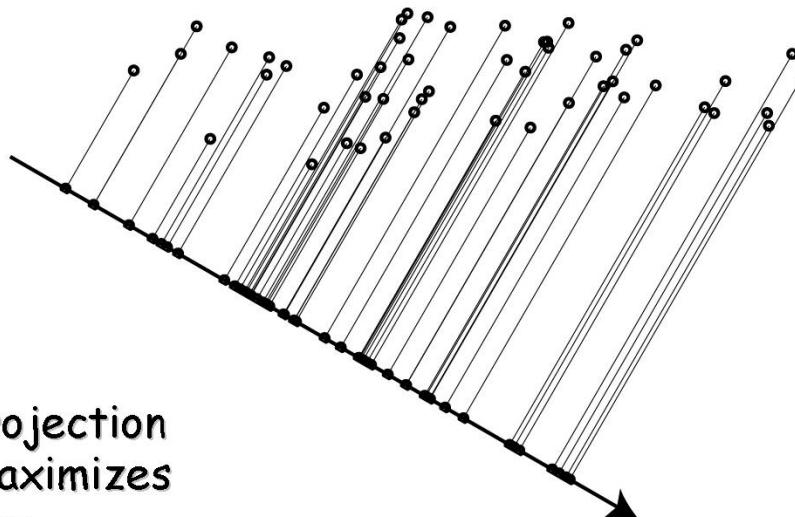
$$\Sigma \vec{v} = \lambda \vec{v}$$



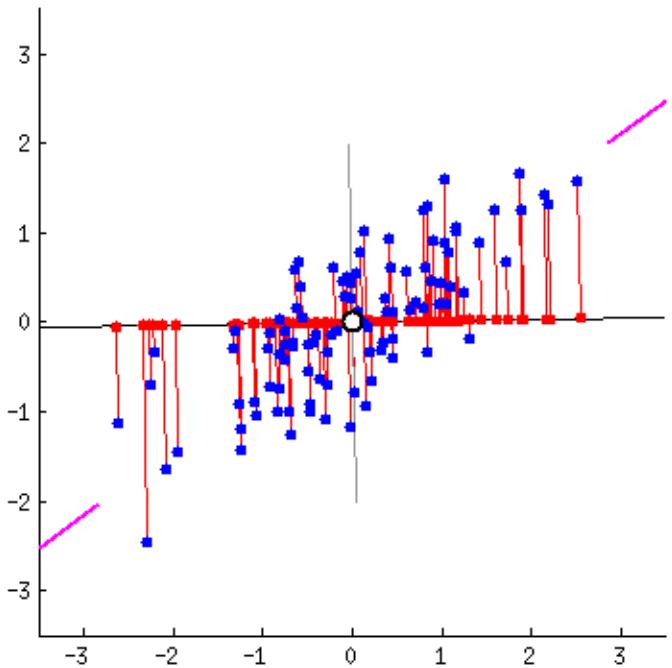
PCA finds new variables which are linear combinations of the original variables such that in the new space, the data has fewer dimensions.



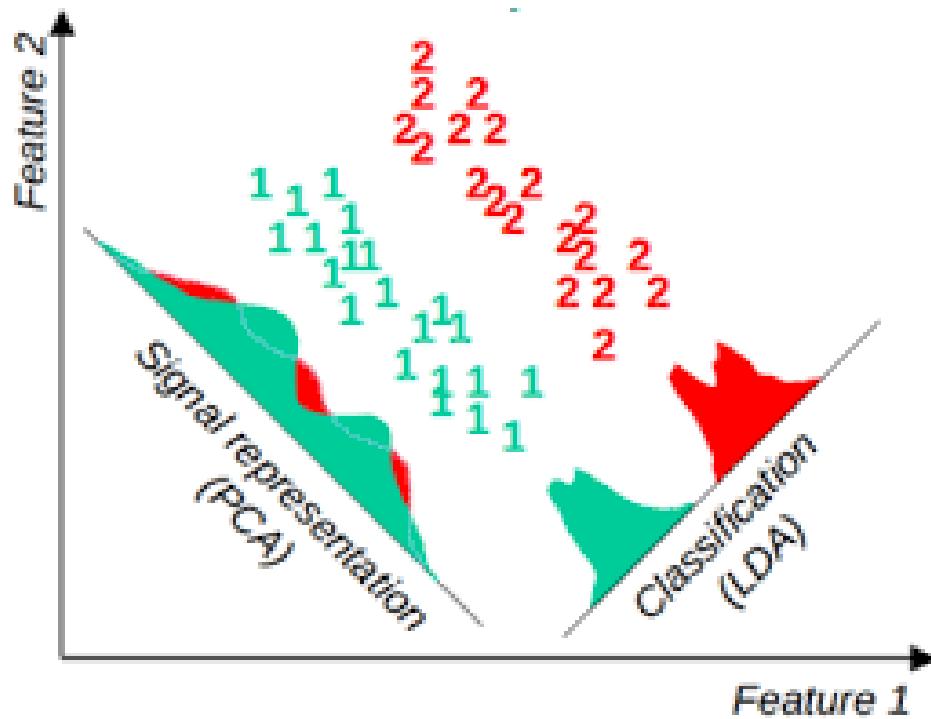
One-dimensional projection



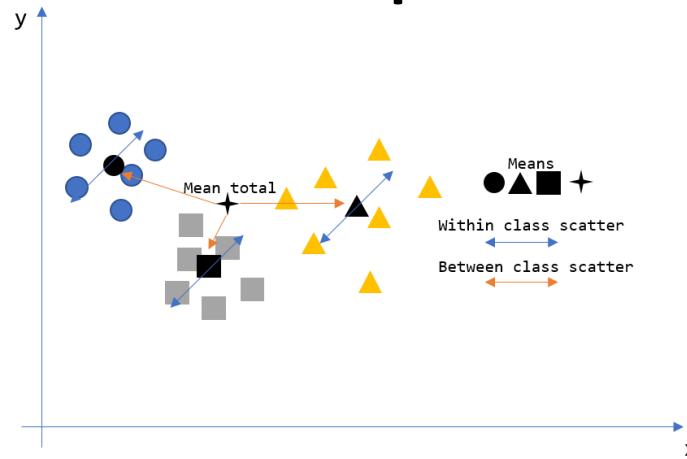
find projection
that maximizes
variance



Linear Discriminant Analysis (LDA)



Multiple Discriminant Analysis (MDA)



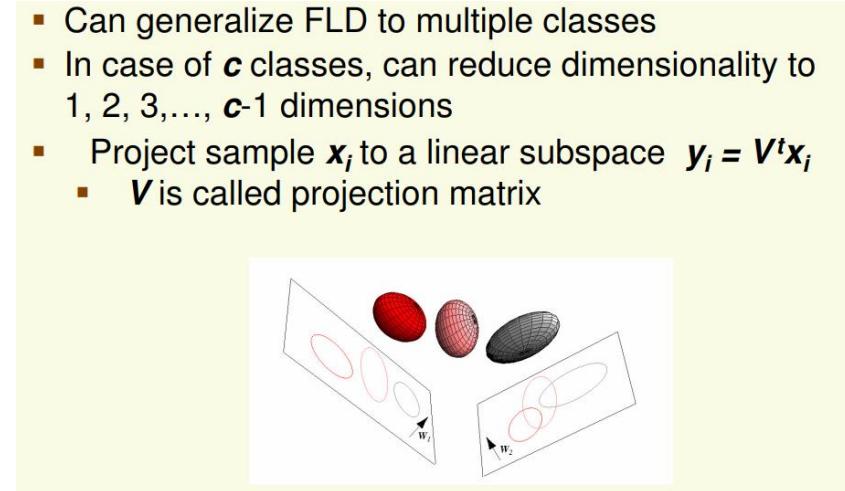
mean vector for each class: $\mathbf{m}_c = \frac{\sum_{i=1}^{N_c} \mathbf{x}_i}{N_c} \in R^{d \times 1}$

total mean vector: $\mathbf{m} = \frac{\sum_{i=1}^N \mathbf{x}_i}{N} \in R^{d \times 1}$

within-class scatter: $S_w = \sum_{c=1}^C \sum_{j=1}^{N_c} (\mathbf{x}_j - \mathbf{m}_c)(\mathbf{x}_j - \mathbf{m}_c)^T \in R^{d \times d}$

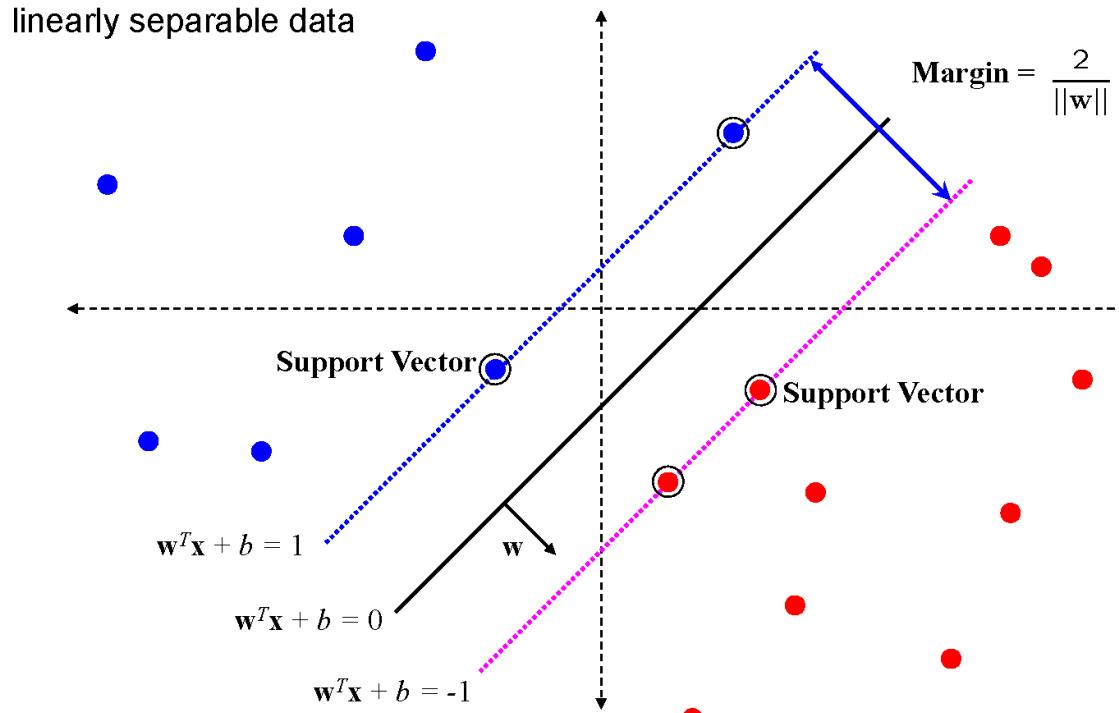
between-class scatter: $S_b = \sum_{c=1}^C N_c(\mathbf{m}_c - \mathbf{m})(\mathbf{m}_c - \mathbf{m})^T \in R^{d \times d}$

- Can generalize FLD to multiple classes
- In case of c classes, can reduce dimensionality to 1, 2, 3, ..., $c-1$ dimensions
- Project sample \mathbf{x}_i to a linear subspace $\mathbf{y}_i = \mathbf{V}^t \mathbf{x}_i$
 - \mathbf{V} is called projection matrix



How to find support vectors ?

Support Vector Machine



SVM – Optimization

- Learning the SVM can be formulated as an optimization:

$$\max_{\mathbf{w}} \frac{2}{\|\mathbf{w}\|} \text{ subject to } \mathbf{w}^\top \mathbf{x}_i + b \begin{cases} \geq 1 & \text{if } y_i = +1 \\ \leq -1 & \text{if } y_i = -1 \end{cases} \text{ for } i = 1 \dots N$$

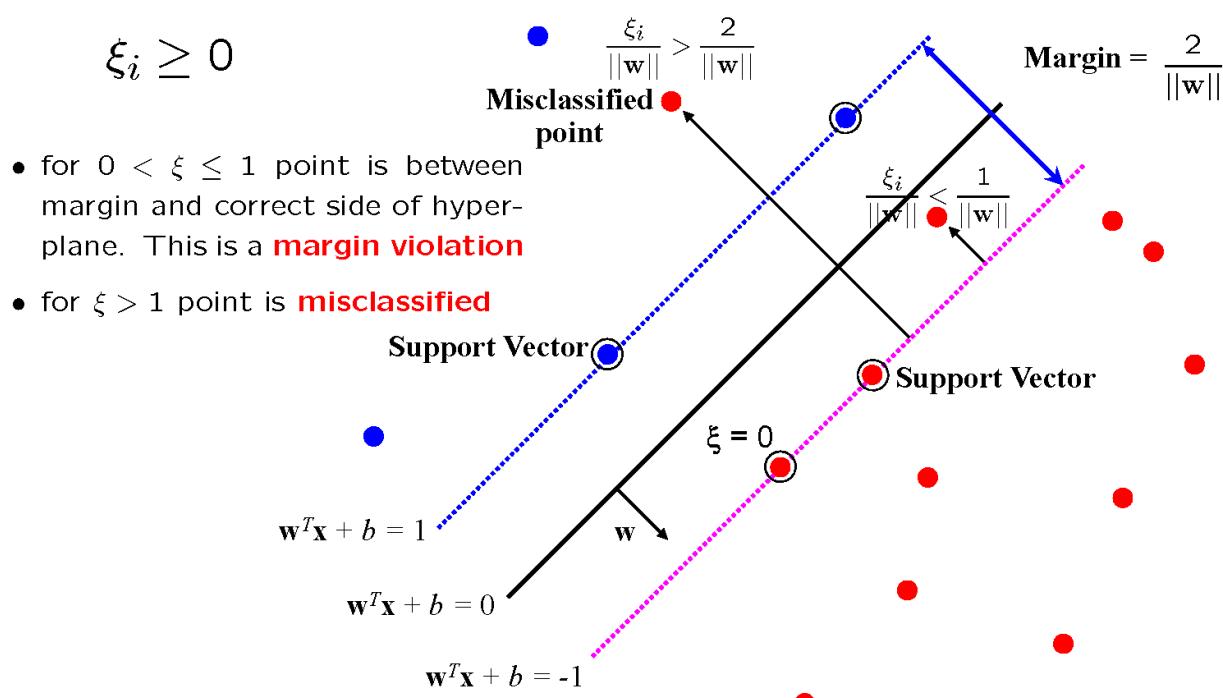
- Or equivalently

$$\min_{\mathbf{w}} \|\mathbf{w}\|^2 \text{ subject to } y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \text{ for } i = 1 \dots N$$

- This is a quadratic optimization problem subject to linear constraints and there is a unique minimum

Quadratic programming in python: <https://scaron.info/blog/quadratic-programming-in-python.html>

Introduce “slack” variables



“Soft” margin solution

The optimization problem becomes

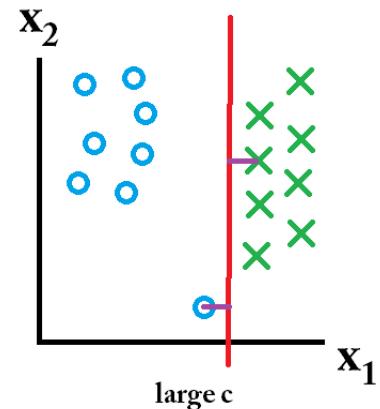
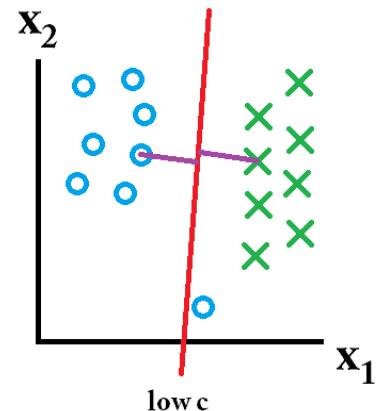
$$\min_{\mathbf{w} \in \mathbb{R}^d, \xi_i \in \mathbb{R}^+} \|\mathbf{w}\|^2 + C \sum_i^N \xi_i$$

subject to

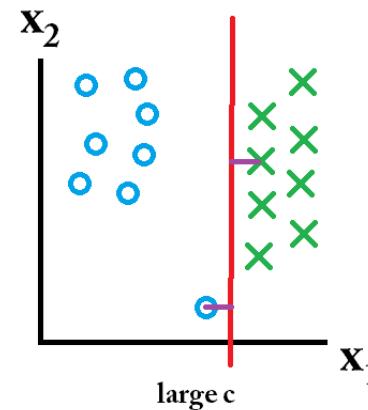
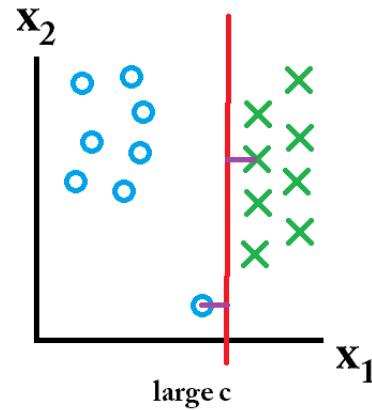
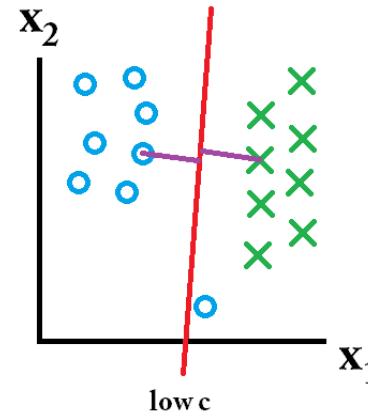
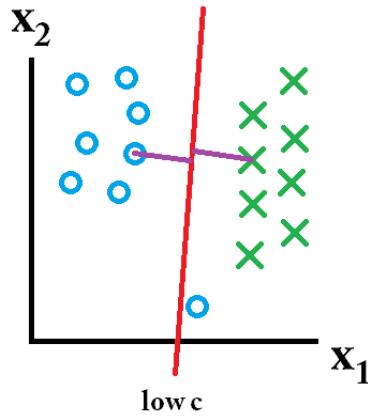
$$y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \text{ for } i = 1 \dots N$$

- Every constraint can be satisfied if ξ_i is sufficiently large
- C is a regularization parameter:
 - small C allows constraints to be easily ignored → large margin
 - large C makes constraints hard to ignore → narrow margin
 - $C = \infty$ enforces all constraints: hard margin
- This is still a quadratic optimization problem and there is a unique minimum. Note, there is only one parameter, C .

C = 0 ???



NOTE: Best margin = That which generalizes to ‘unseen’ data

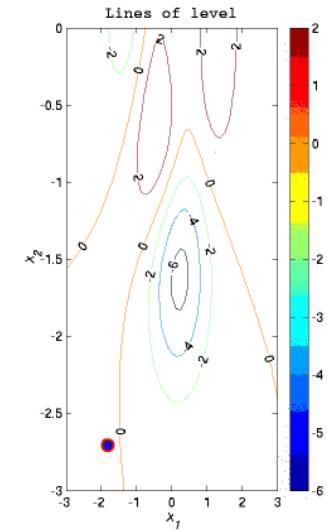
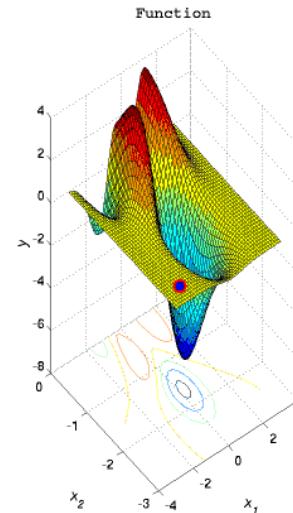
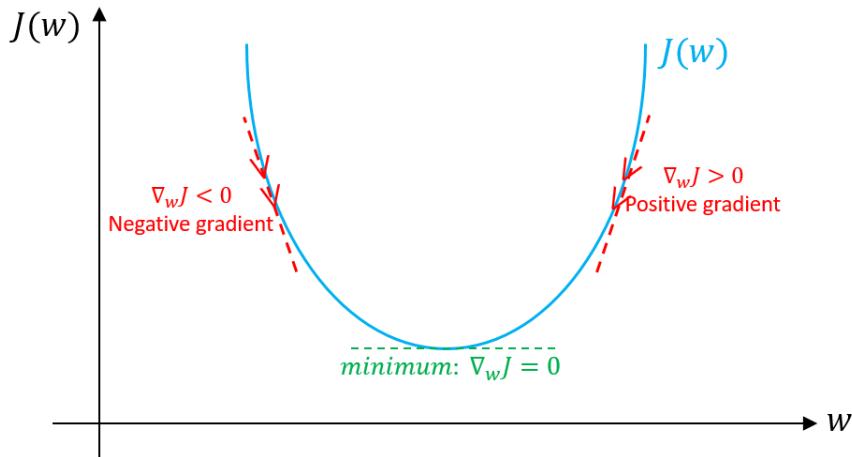


Gradient (or steepest) descent algorithm for SVM

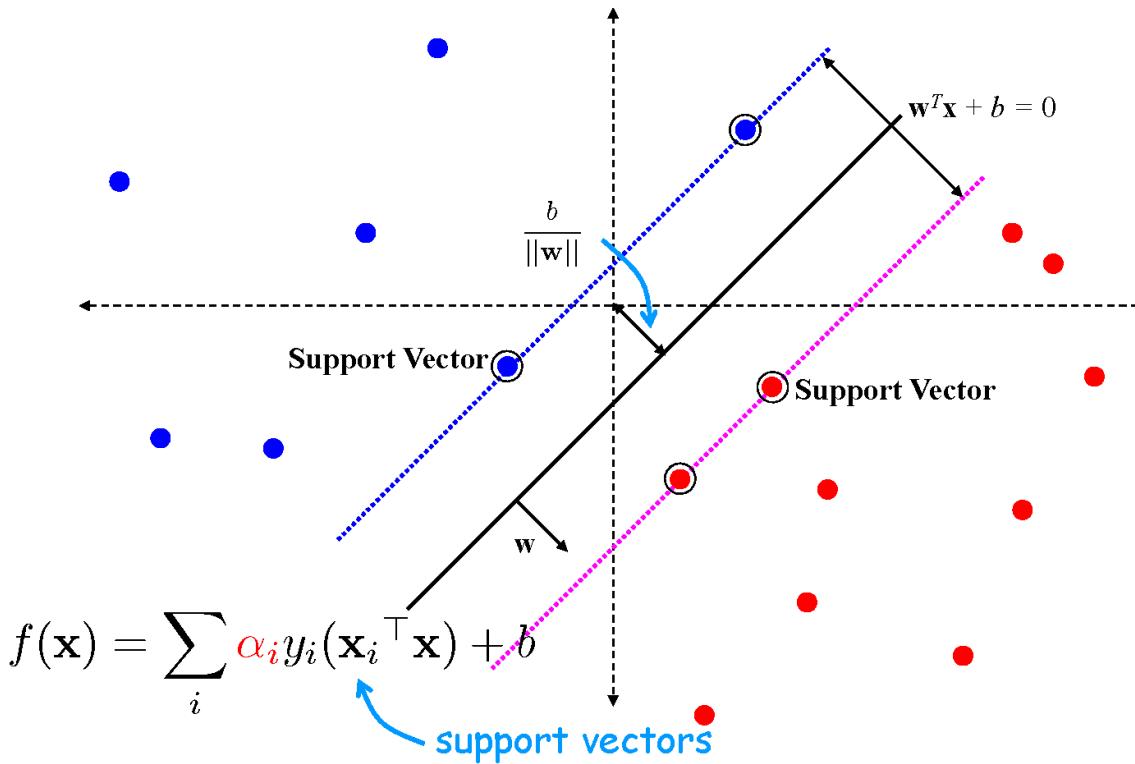
To minimize a cost function $\mathcal{C}(\mathbf{w})$ use the iterative update

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}} \mathcal{C}(\mathbf{w}_t)$$

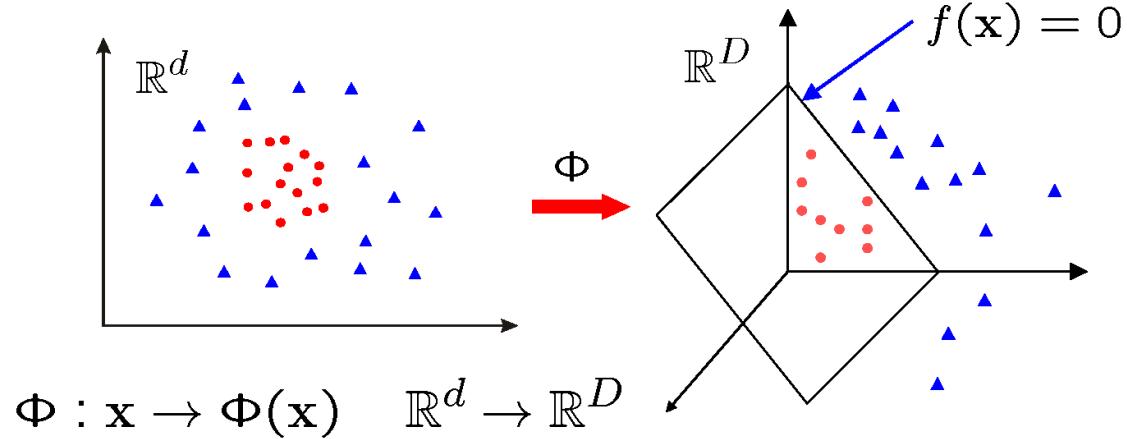
where η is the learning rate.



Support Vector Machine



SVM classifiers in a transformed feature space



Learn classifier linear in \mathbf{w} for \mathbb{R}^D :

$$f(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x}) + b$$

$\Phi(\mathbf{x})$ is a **feature map**

Dual Classifier in transformed feature space

Classifier:

$$\begin{aligned} f(\mathbf{x}) &= \sum_i^N \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b \\ \rightarrow f(\mathbf{x}) &= \sum_i^N \alpha_i y_i \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}) + b \end{aligned}$$

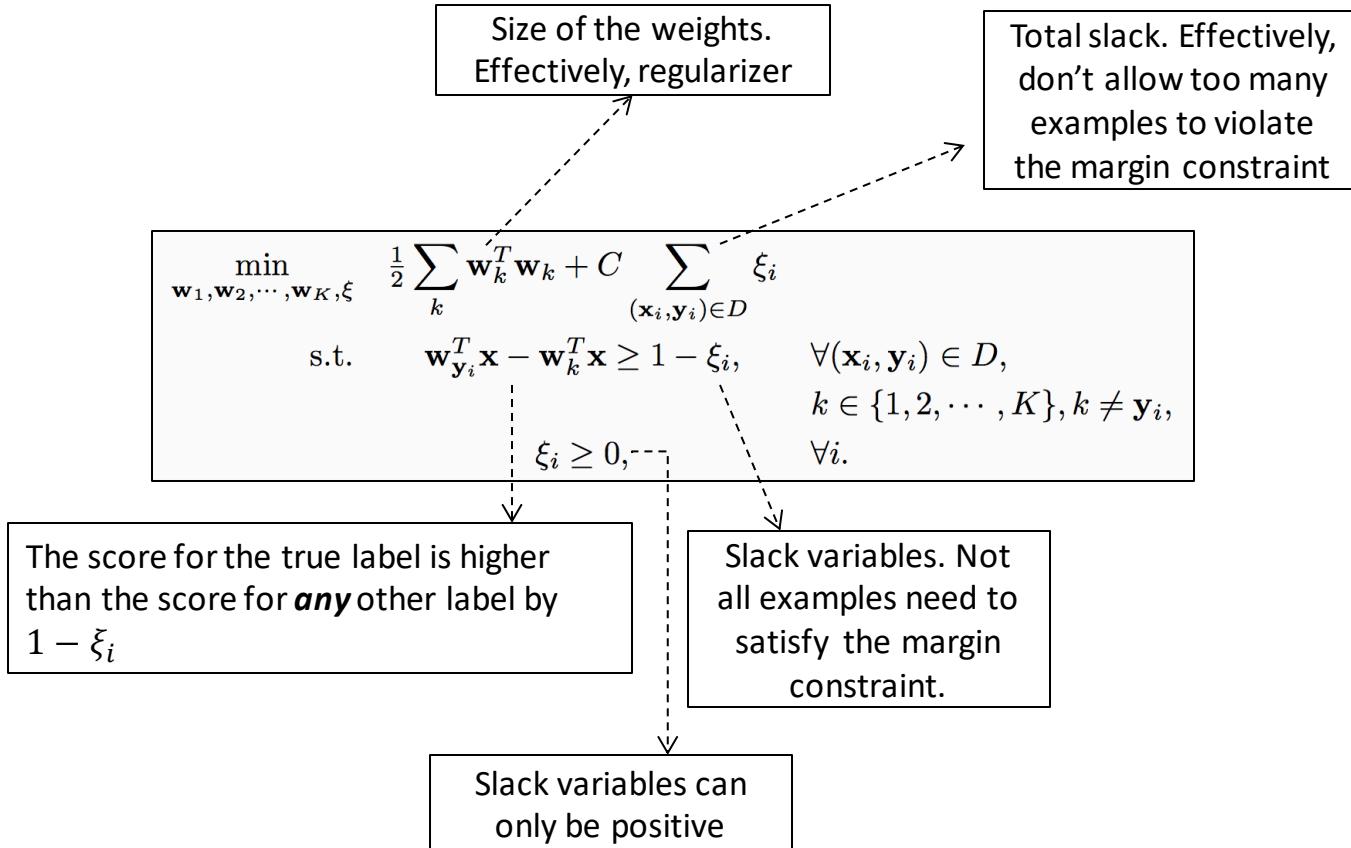
Learning:

$$\begin{aligned} &\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \mathbf{x}_j^\top \mathbf{x}_k \\ \rightarrow &\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \Phi(\mathbf{x}_j)^\top \Phi(\mathbf{x}_k) \end{aligned}$$

subject to

$$0 \leq \alpha_i \leq C \text{ for } \forall i, \text{ and } \sum_i \alpha_i y_i = 0$$

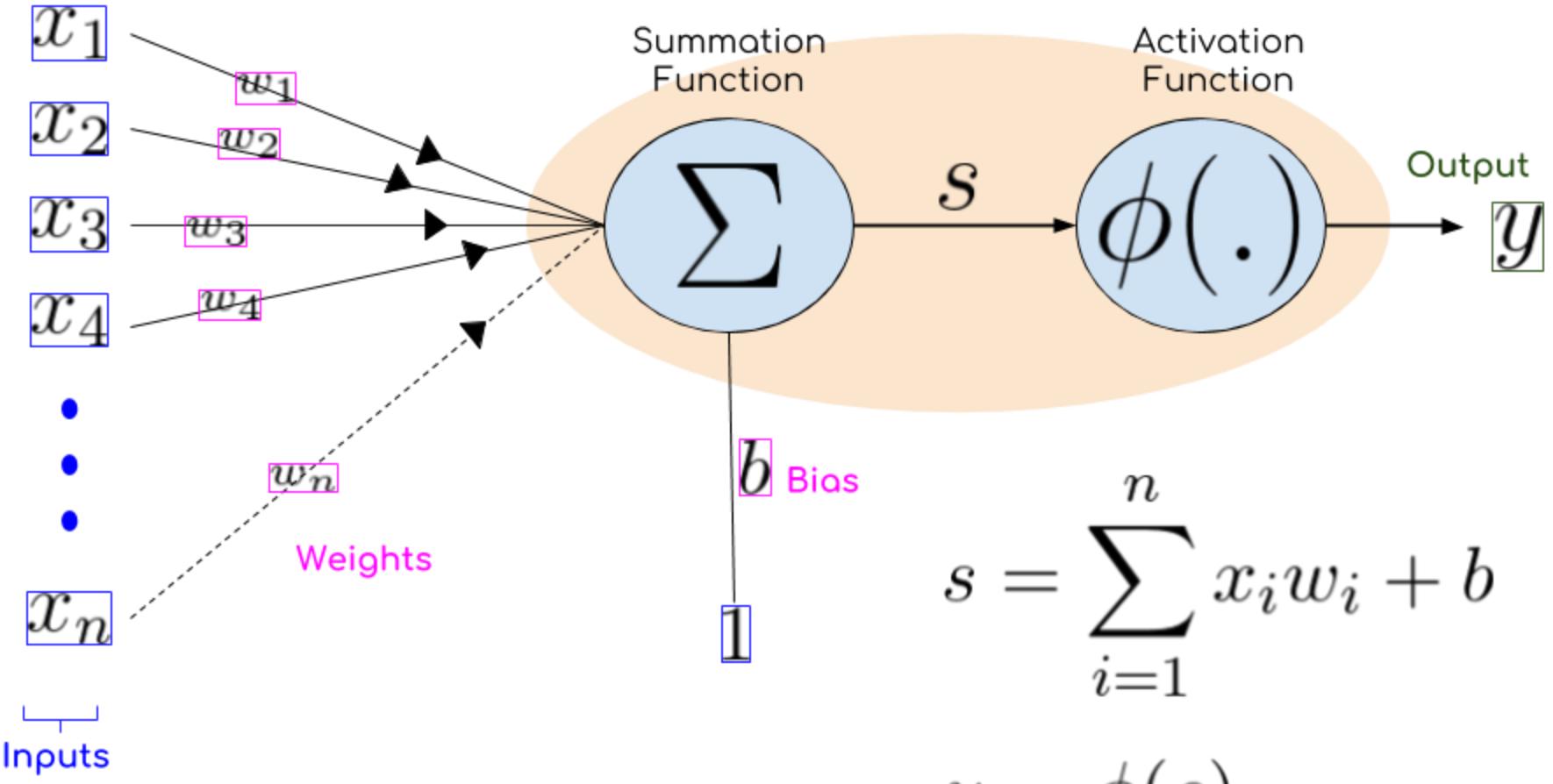
Multiclass SVM: General case



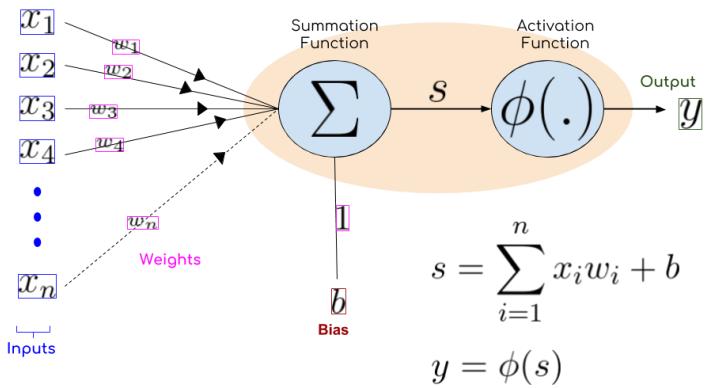
Multiclass SVM: Summary

- Training:
 - Optimize the SVM objective
- Prediction:
 - Winner takes all
 $\text{argmax}_i \mathbf{w}_i^T \mathbf{x}$

Neural Networks

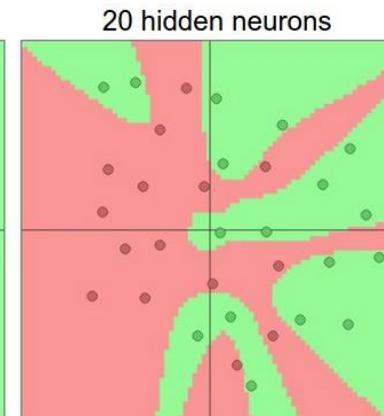
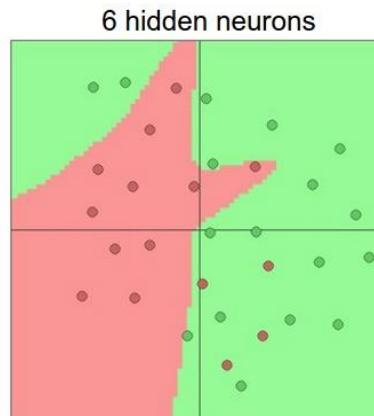
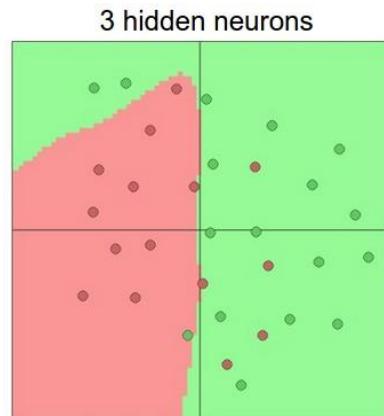
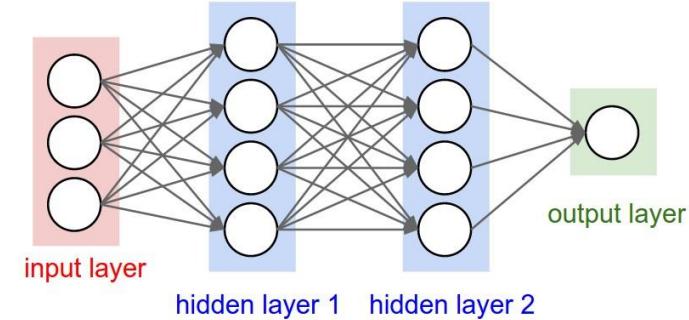
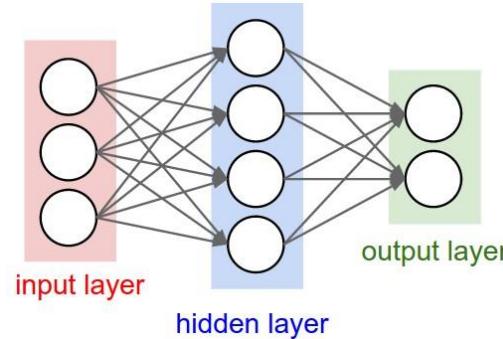


ACTIVATION FUNCTIONS



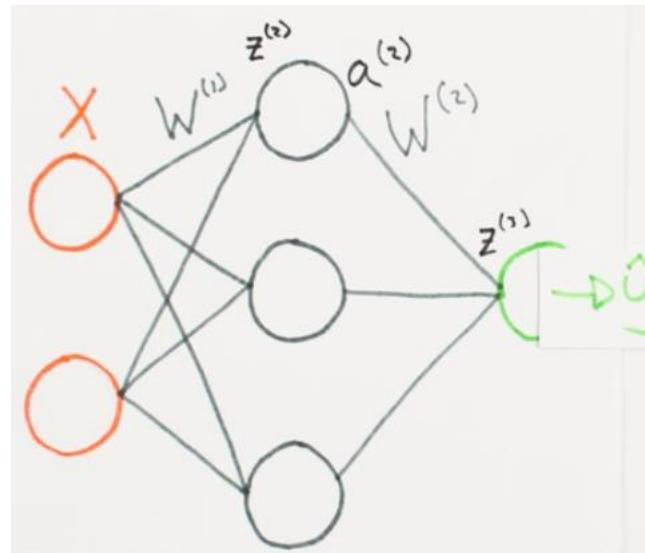
Activation Function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Unit Step (Heaviside Function)	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\phi(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Piece-wise Linear	$\phi(z) = \begin{cases} 0 & z \leq -\frac{1}{2} \\ z + \frac{1}{2} & -\frac{1}{2} \leq z \leq \frac{1}{2} \\ 1 & z \geq \frac{1}{2} \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression,	
Hyperbolic Tangent (tanh)	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$		
ReLU	$\phi(z) = \begin{cases} 0 & z < 0 \\ z & z > 0 \end{cases}$		

Why Use Only One Neuron ?



Multi-Neuron Networks :: FORWARD PROPAGATION

```
In [7]: NN = Neural_Network()  
In [8]: yHat = NN.forward(X)  
  
In [9]: yHat  
Out[9]: array([[ 0.59470263],  
   [ 0.58177822],  
   [ 0.50641742]])  
  
In [10]: y  
Out[10]: array([[ 0.75],  
   [ 0.82],  
   [ 0.93]])
```



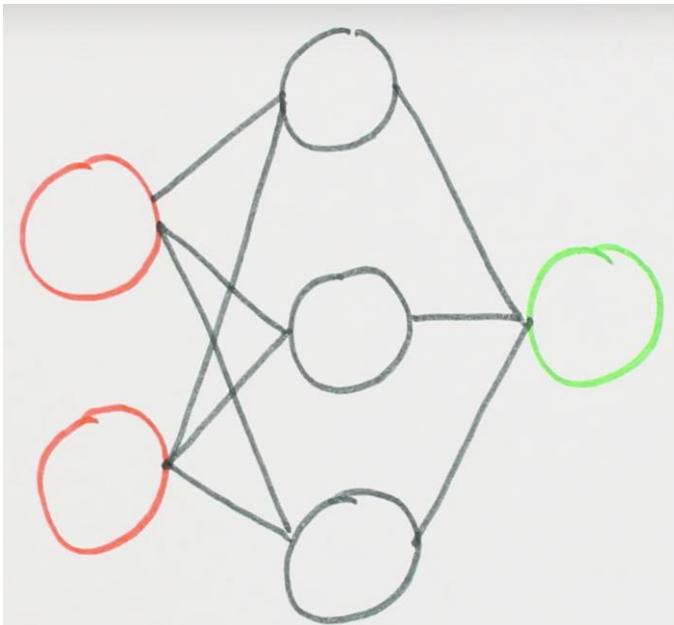
$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

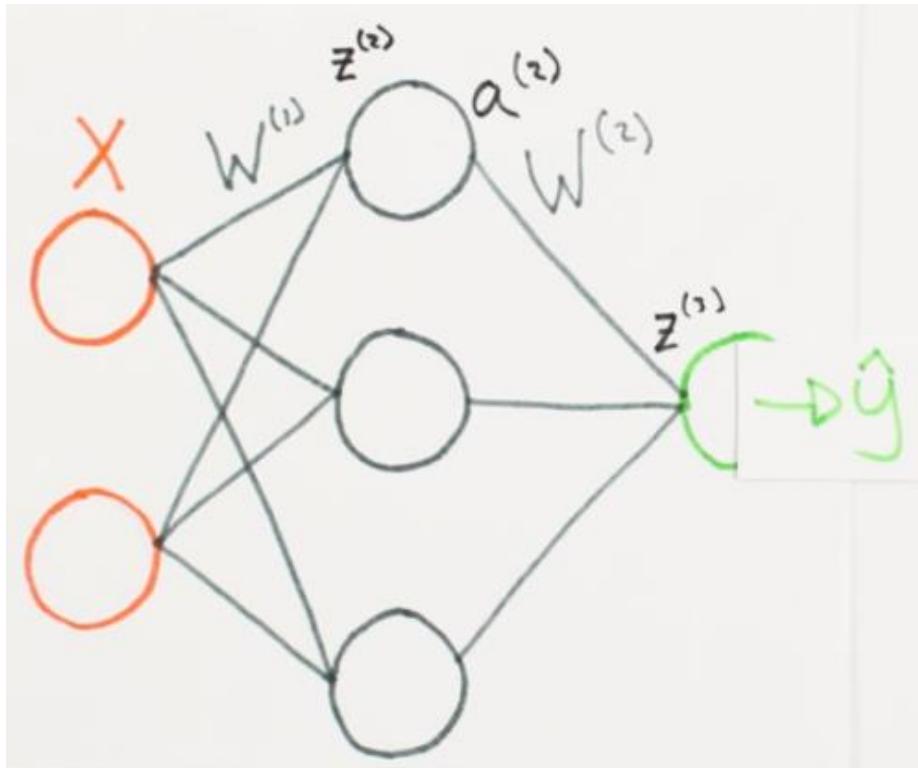
$$\hat{y} = f(z^{(3)}) \quad (4)$$

Multi-Neuron Networks :: Gradient Descent



Training a Network
=
Minimizing a Cost
Function

Multi-Neuron Networks :: Gradient Descent



$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

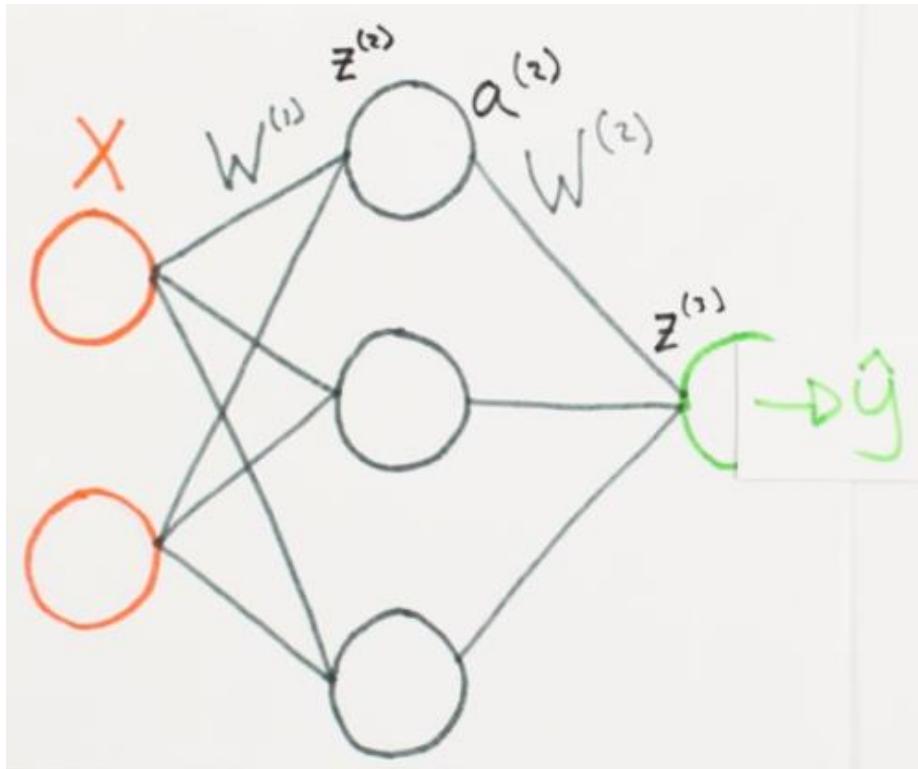
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2} (y - \hat{y})^2 \quad (5)$$

$$J = \sum \frac{1}{2} (y - f(f(XW^{(1)}|W^{(2)}|))^2$$

Multi-Neuron Networks :: Gradient Descent

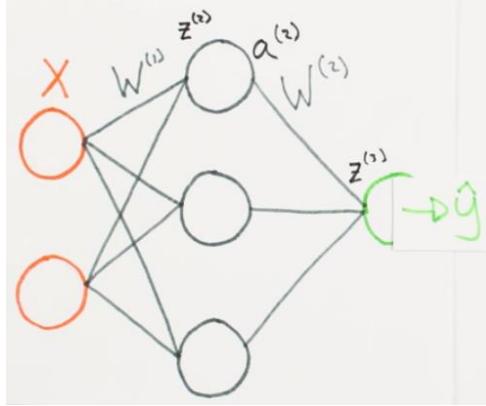


$$J = \sum \frac{1}{2} (y - f(f(XW^{(1)}))W^{(2)})^2$$

↑ HOW DOES THIS CHANGE
IF I CHANGE THESE? ↑

$$\frac{\partial J}{\partial W}$$

Multi-Neuron Networks :: Backpropagation



$$J = \sum \frac{1}{2} (y - f(f(XW^{(1)})W^{(2)}))^2$$

↑ HOW DOES THIS CHANGE
IF I CHANGE THESE?

$$\frac{\partial J}{\partial W}$$

$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix}$$
$$\frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(1)}} & \frac{\partial J}{\partial W_{12}^{(1)}} & \frac{\partial J}{\partial W_{13}^{(1)}} \\ \frac{\partial J}{\partial W_{21}^{(1)}} & \frac{\partial J}{\partial W_{22}^{(1)}} & \frac{\partial J}{\partial W_{23}^{(1)}} \end{bmatrix}$$
$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{31}^{(2)} \end{bmatrix}$$
$$\frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(2)}} \\ \frac{\partial J}{\partial W_{21}^{(2)}} \\ \frac{\partial J}{\partial W_{31}^{(2)}} \end{bmatrix}$$

Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2} (y - \hat{y})^2$$

$$\frac{\partial J}{\partial W^{(2)}} = (a^{(2)})^T \delta^{(3)}$$

$$\delta^{(3)} = -(y - \hat{y})f'(z^{(3)})$$

$$\frac{\partial J}{\partial W^{(1)}} = X^T \delta^{(2)}$$

$$\delta^{(2)} = \delta^{(3)} (W^{(2)})^T f'(z^{(2)})$$

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

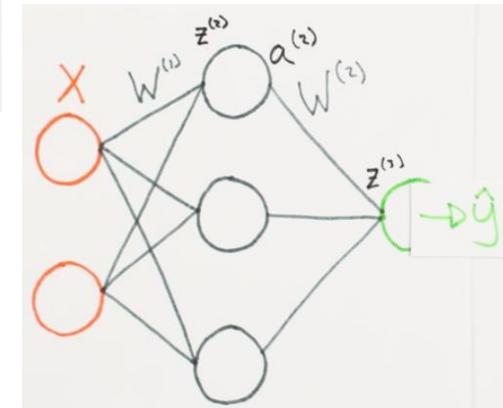
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2} (y - f(f(XW^{(1)}))W^{(2)})^2$$

HOW DOES THIS CHANGE
IF I CHANGE THESE?

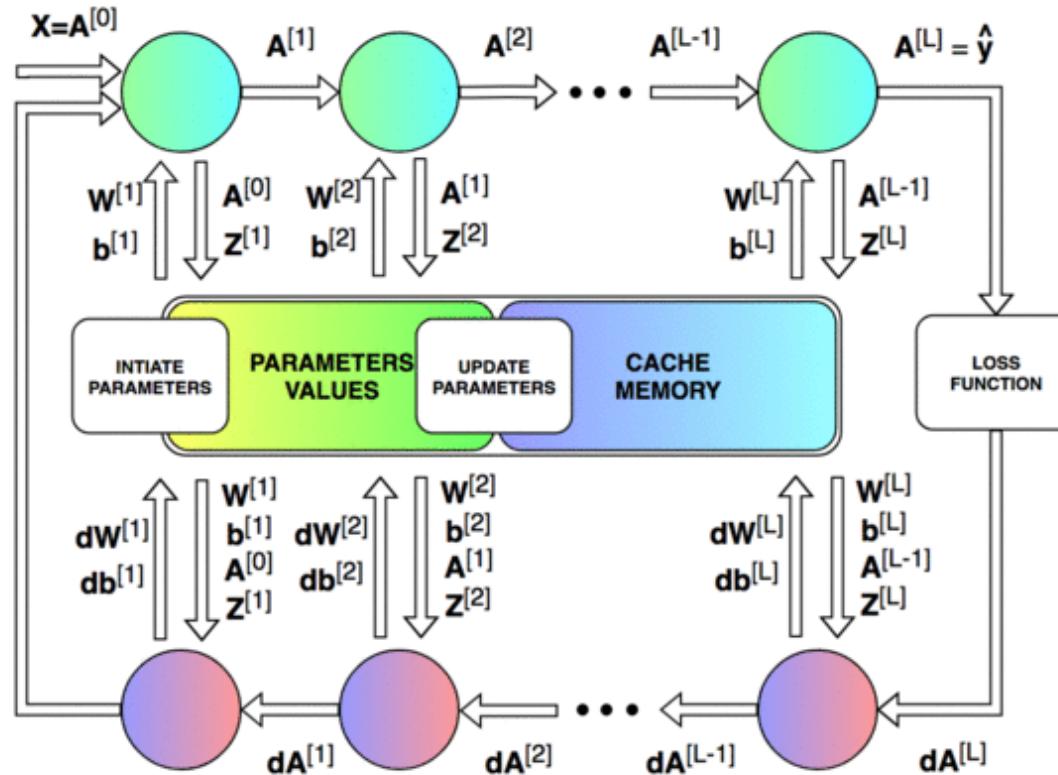
$$\frac{\partial J}{\partial W}$$



$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \quad \frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(1)}} & \frac{\partial J}{\partial W_{12}^{(1)}} & \frac{\partial J}{\partial W_{13}^{(1)}} \\ \frac{\partial J}{\partial W_{21}^{(1)}} & \frac{\partial J}{\partial W_{22}^{(1)}} & \frac{\partial J}{\partial W_{23}^{(1)}} \end{bmatrix}$$

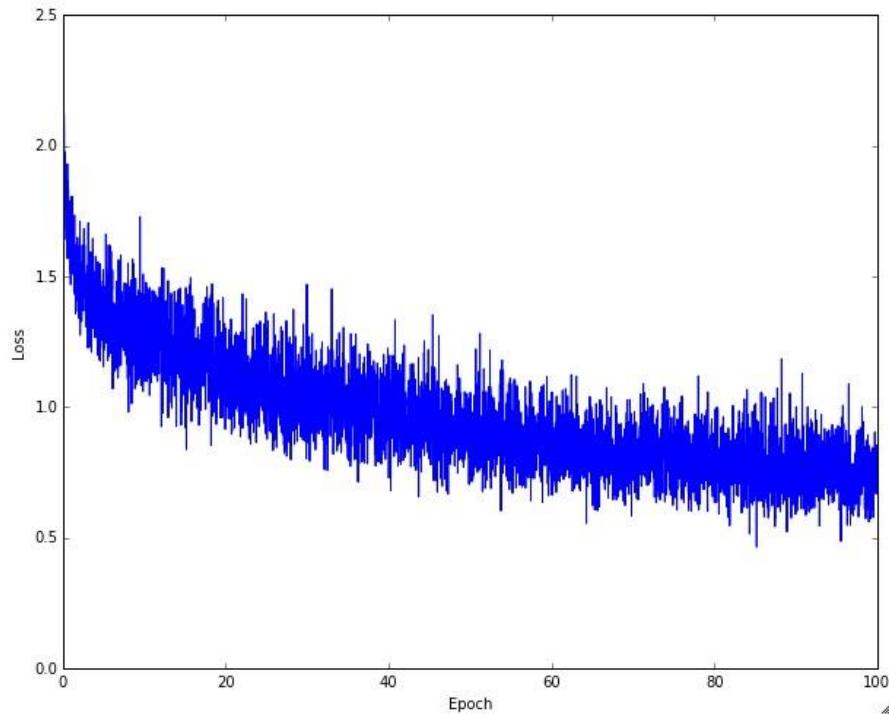
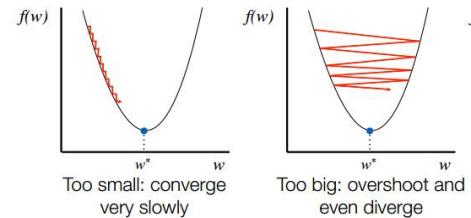
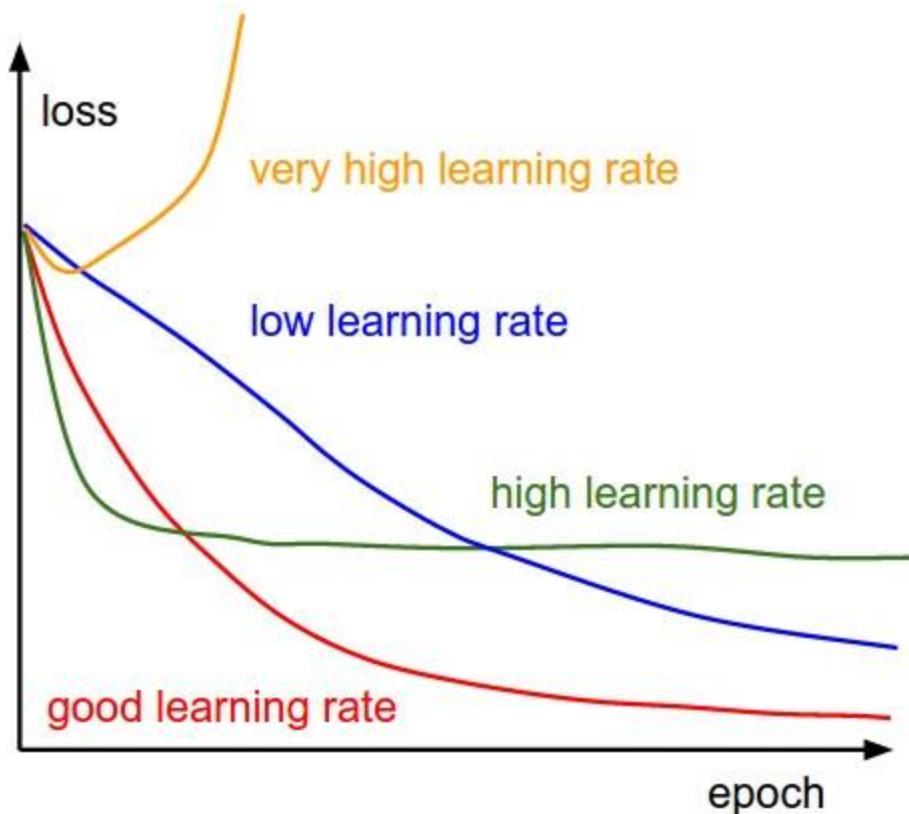
$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{31}^{(2)} \end{bmatrix} \quad \frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(2)}} \\ \frac{\partial J}{\partial W_{21}^{(2)}} \\ \frac{\partial J}{\partial W_{31}^{(2)}} \end{bmatrix}$$

FORWARD PROPAGATION

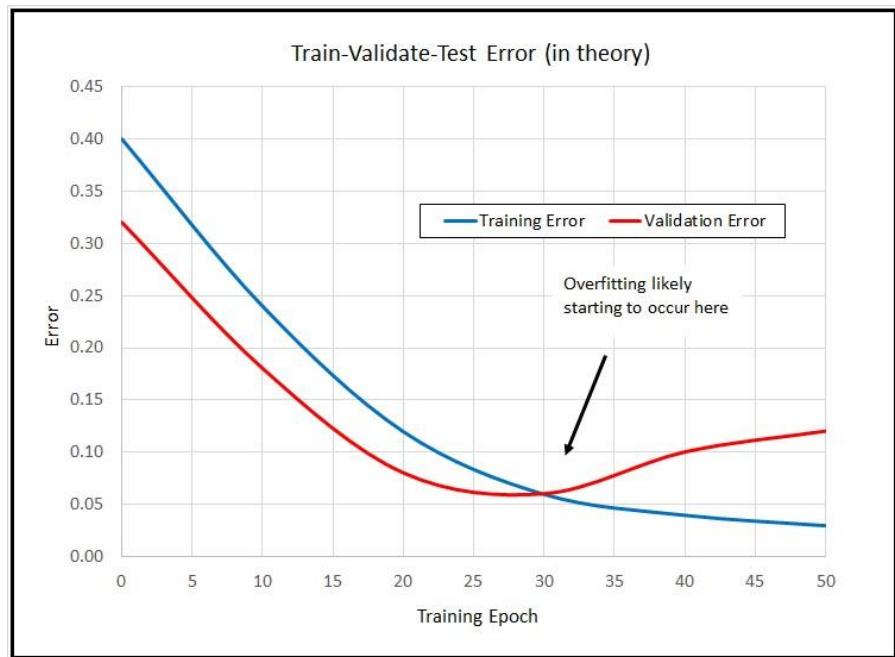
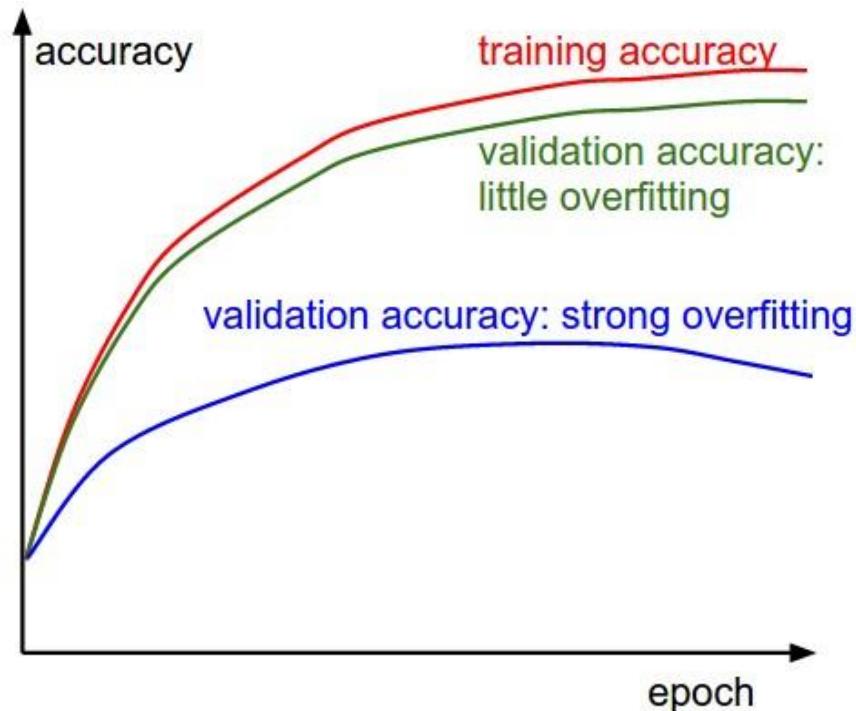


BACKWARD PROPAGATION

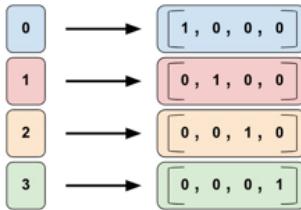
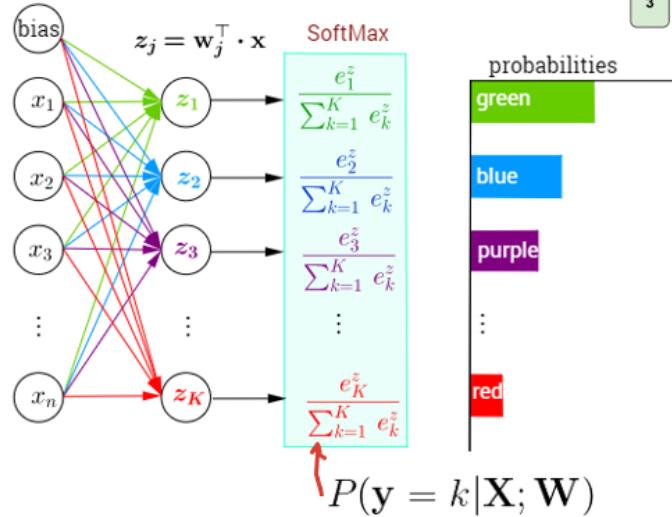
Training – setting learning rate



When to stop training



Classification Loss



CROSS-ENTROPY

$$D(S, L) = - \sum_i L_i \log(S_i)$$

Diagram illustrating the Cross-Entropy loss function:

- S (Y):** A vector of predicted probabilities $[0.7, 0.2, 0.1]$.
- L:** A vector of target probabilities $[1.0, 0.0, 0.0]$.
- The formula shows the calculation of the cross-entropy loss between S and L .

$$D(S, L) \neq D(L, S)$$

$$\begin{aligned} D_{\text{KL}}(p|q) &= \sum_i p_i \log \frac{p_i}{q_i} \\ &= \sum_i (-p_i \log q_i + p_i \log p_i) \\ &= -\sum_i p_i \log q_i + \sum_i p_i \log p_i \\ &= -\sum_i p_i \log q_i - \sum_i p_i \log \frac{1}{p_i} \\ &= -\sum_i p_i \log q_i - H(p) \\ &= \sum_i p_i \log \frac{1}{q_i} - H(p) \end{aligned}$$



G. Hinton Yann LeCun Y. Bengio
(U.Toronto, (Facebook AI) (MILA, Montreal)
Google DeepMind)



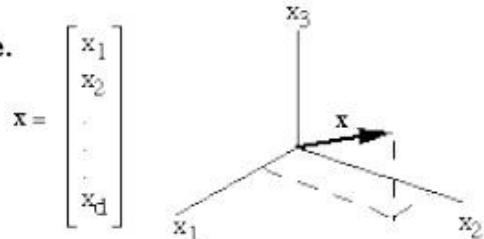
Feature Vectors

- Usually a single object can be represented using several features, e.g.

- x_1 = shape (e.g. nr of sides)
- x_2 = size (e.g. some numeric value)
- x_3 = color (e.g. rgb values)
- ...
- x_d = some other (numeric) feature.

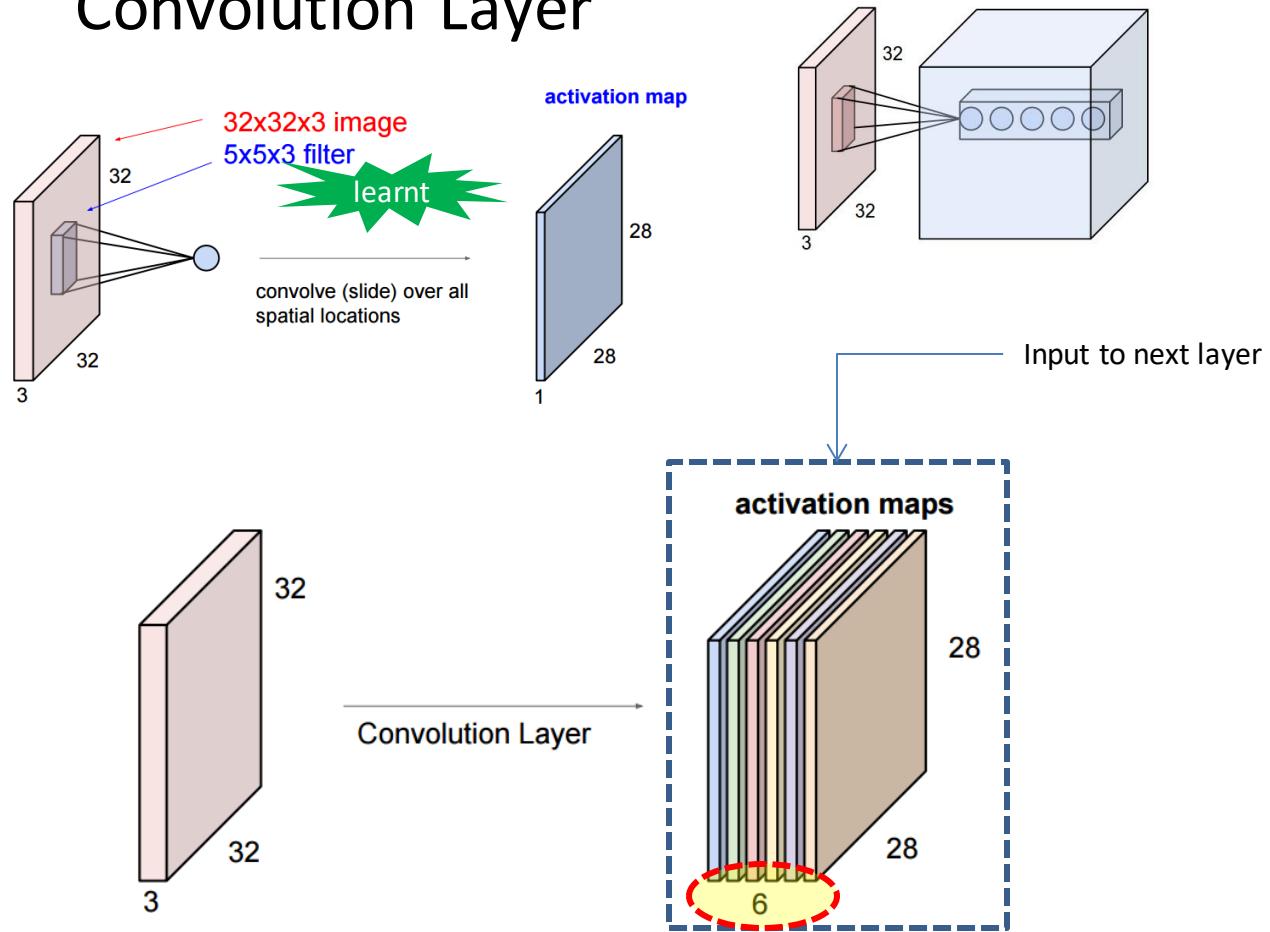


- **X** becomes a feature vector
 - x is a point in a d-dimensional **feature space**.



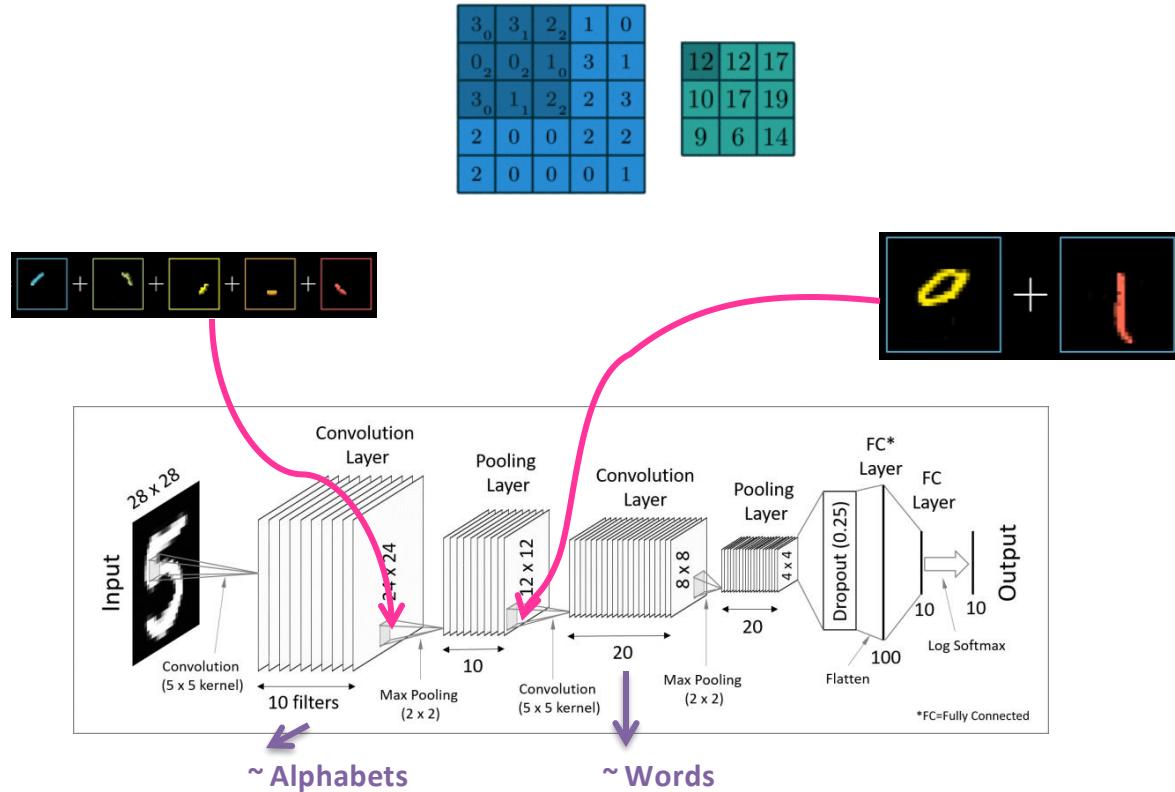


Convolution Layer

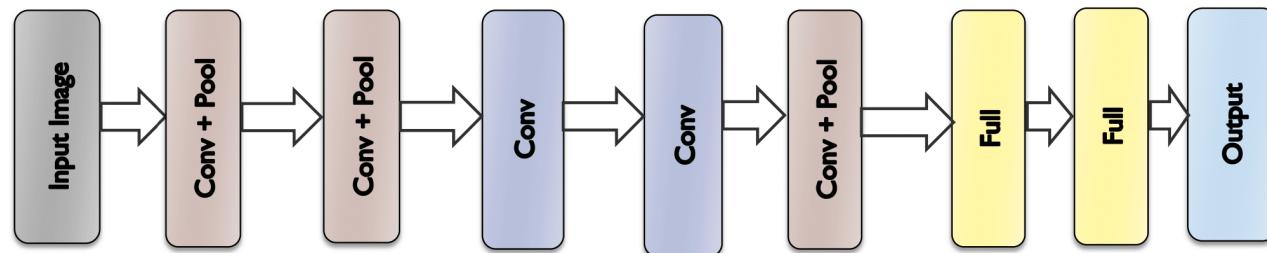
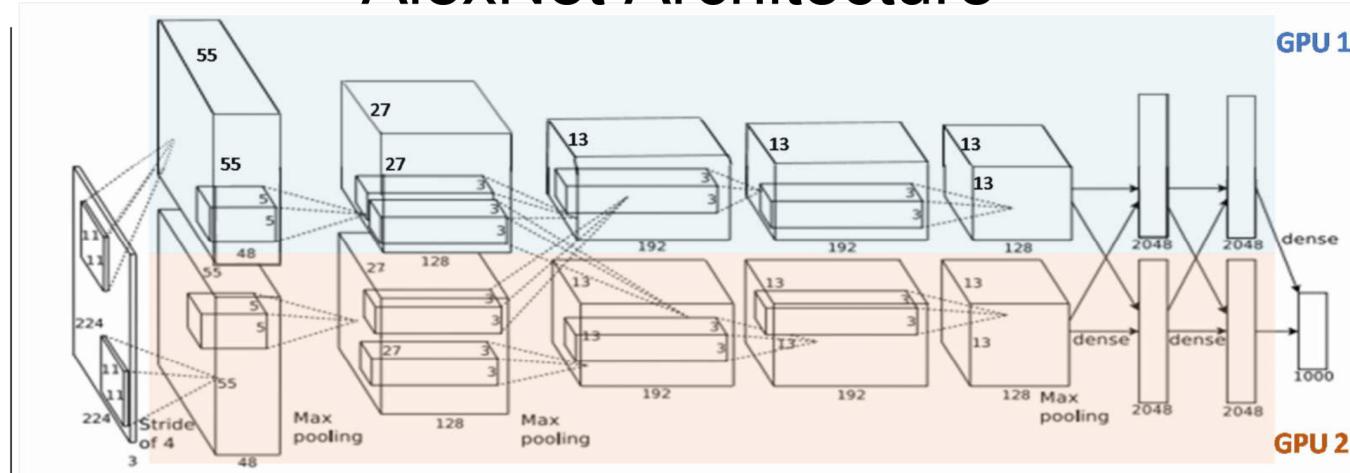


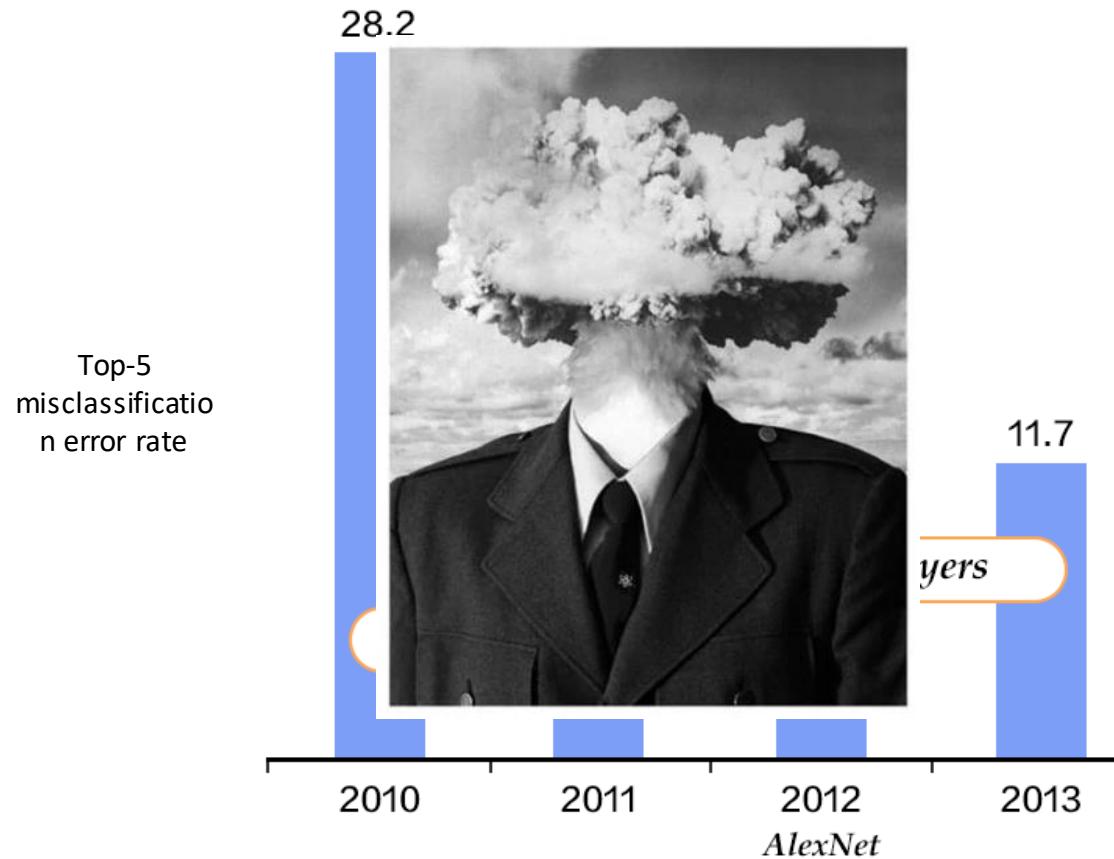


Design choices : Filter size, # of filters



AlexNet Architecture

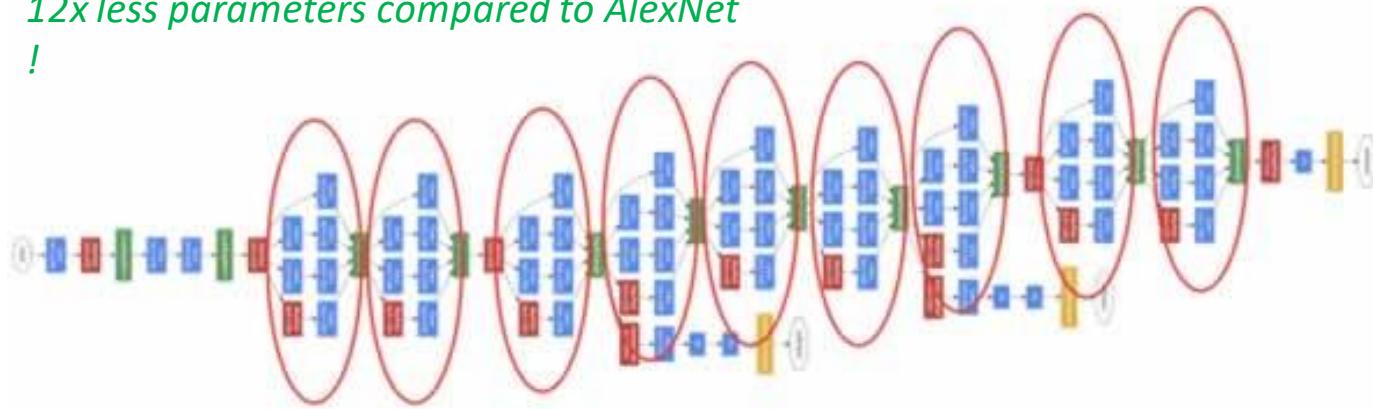






GoogLeNet

12x less parameters compared to AlexNet
!



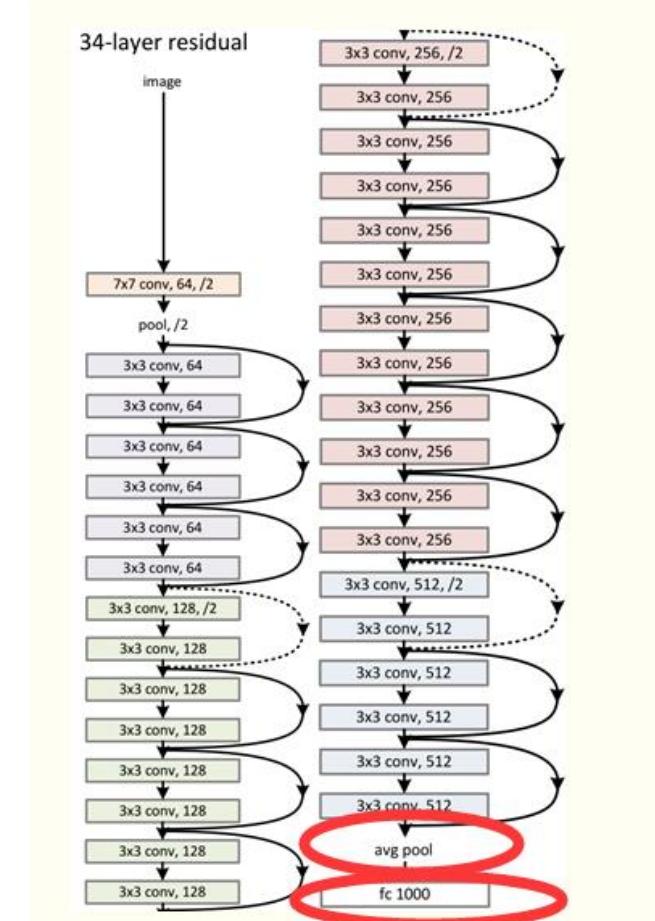
9 Inception modules

Network in a network in a network...

Convolution
Pooling
Softmax
Other



Residual Networks



Gradients can be supplied to shallower layers directly if needed (via skips)

Transfer Learning



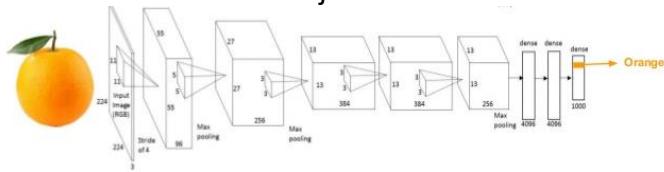


Landscape of CNNs : Architectures

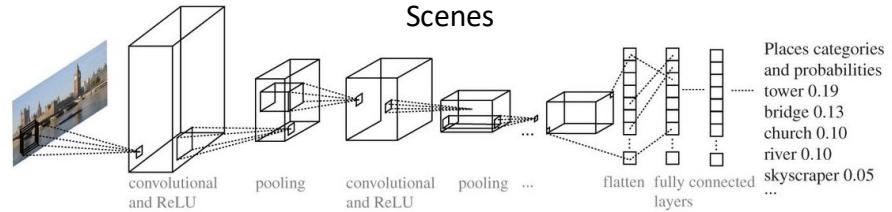


Image → Label

Objects



Scenes



Faces

DeepFace Architecture

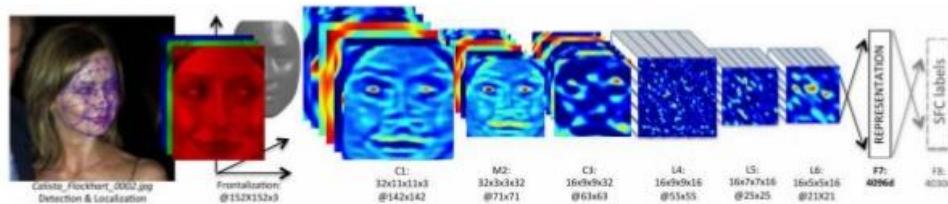
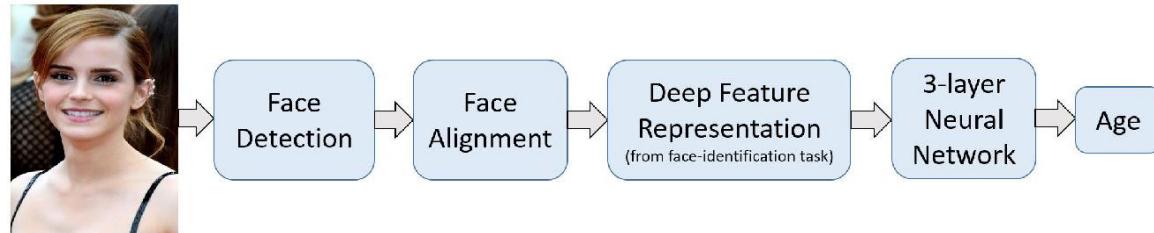
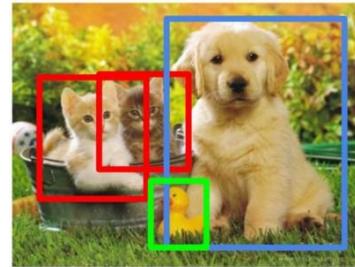




Image → Number



Object Detection

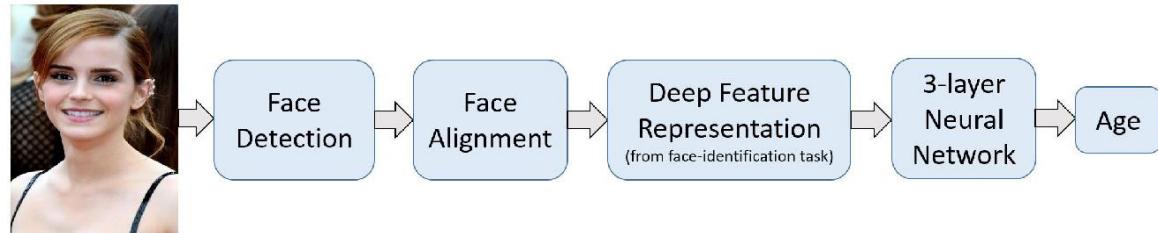


CAT, DOG, DUCK

Image → Number



Age Estimation



Object
detection

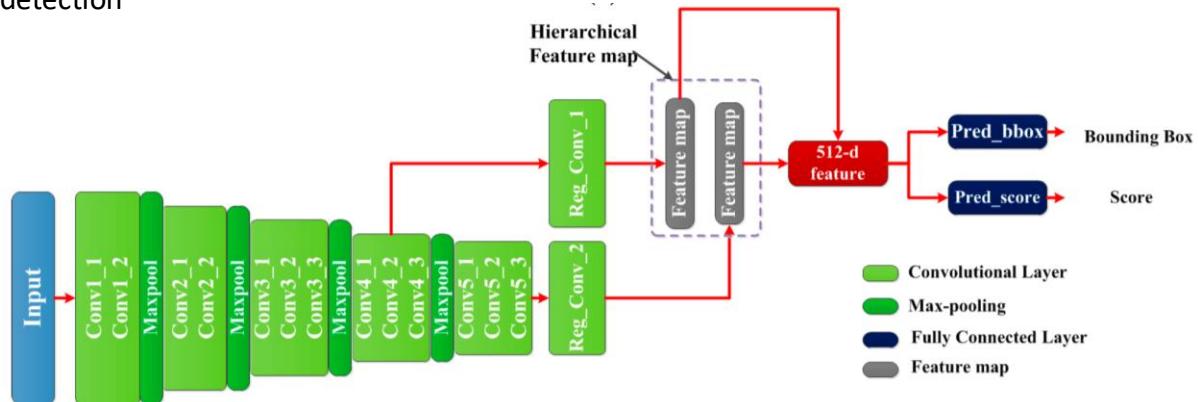
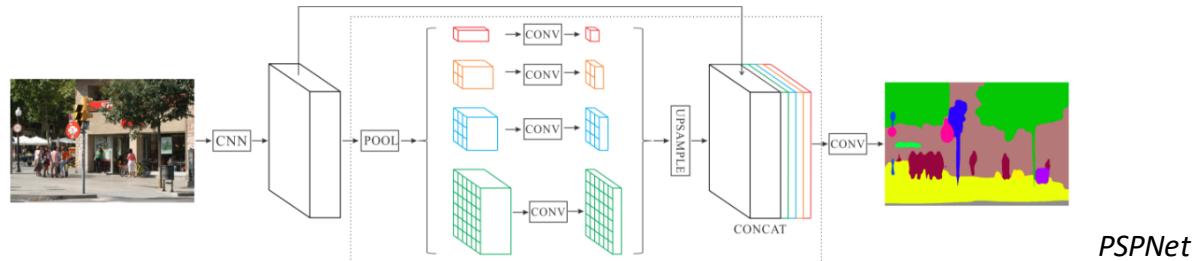


Image → Label Image

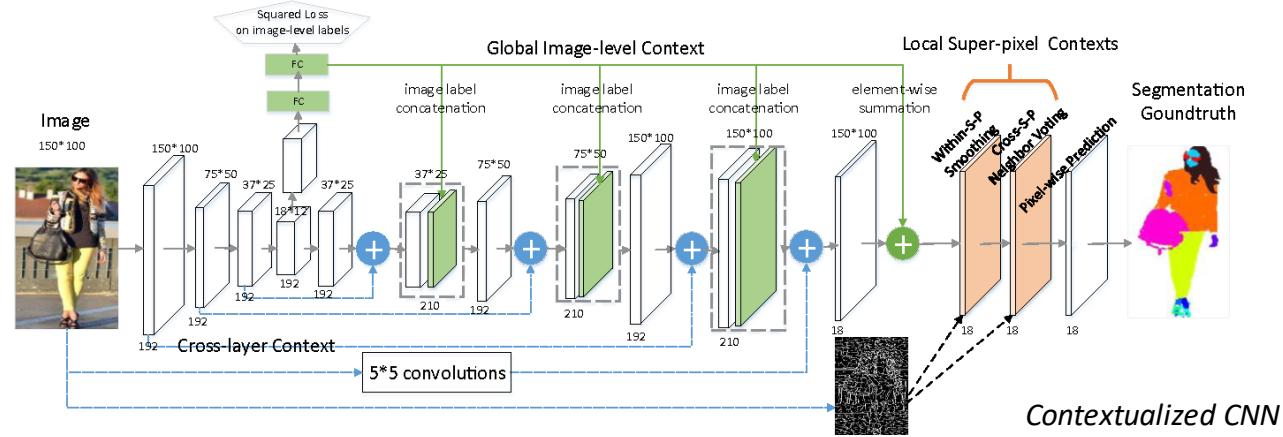


Scene Parsing



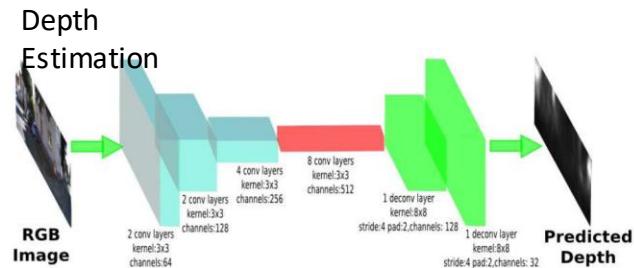
PSPNet

Object Parsing

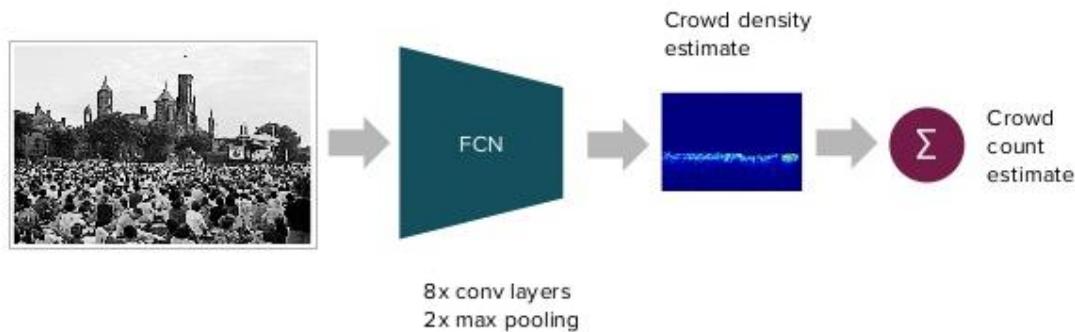


Contextualized CNN

Image → Image



Crowd Counting

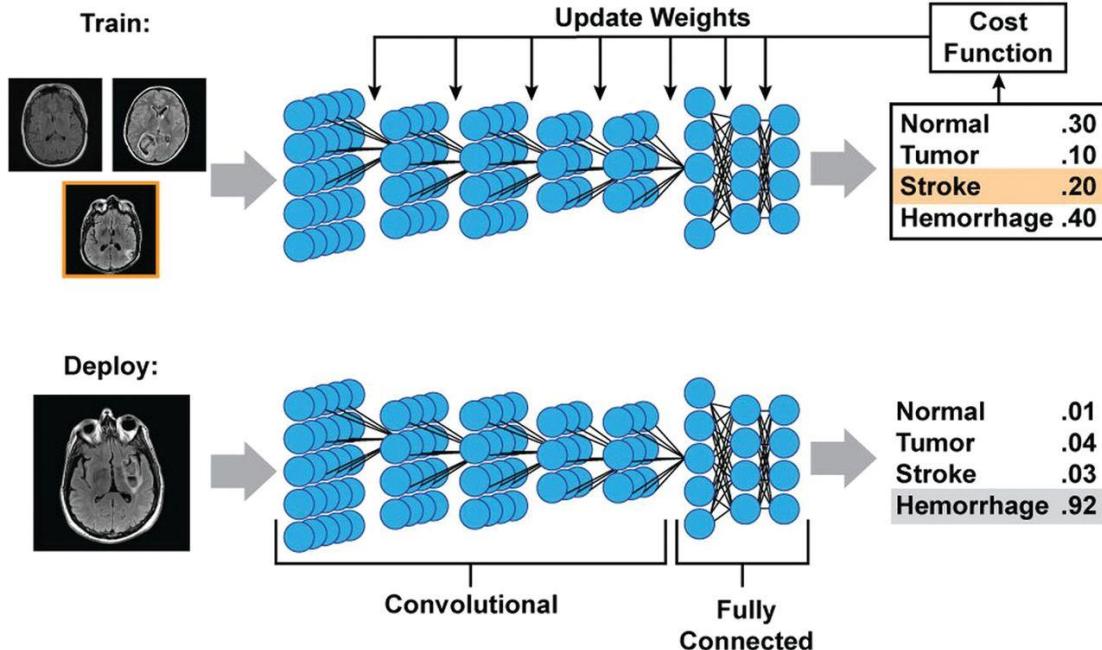




Multi-channel input

NeuroRadiology (fMRI)

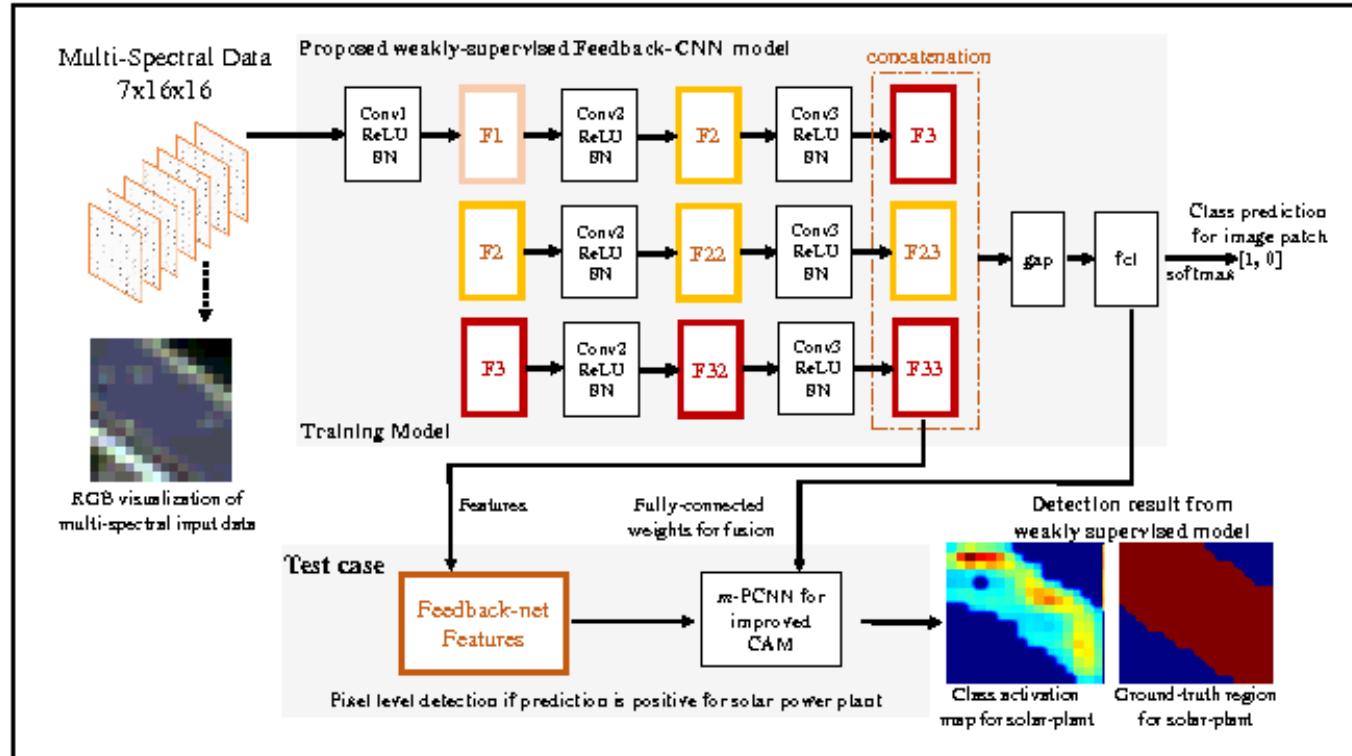
Convolutional Neural Networks





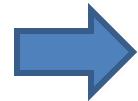
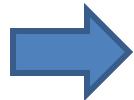
Multi-channel input

Solar-power plant detection from multi-spectral data



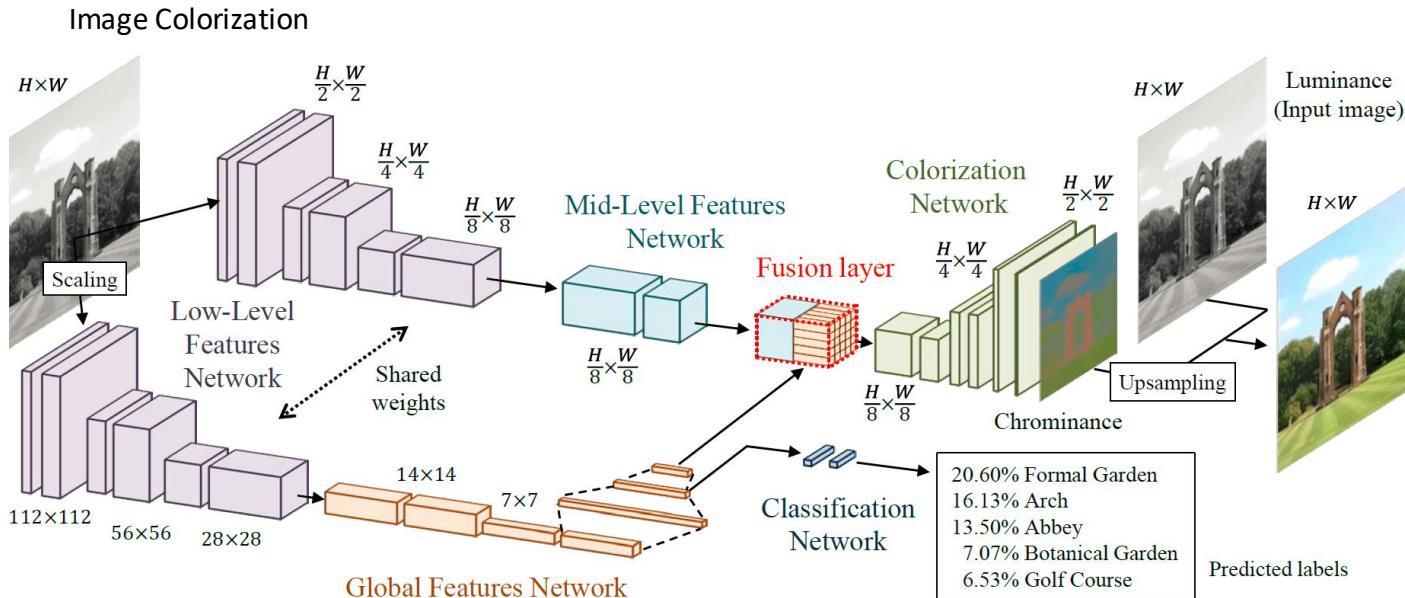
Multi-branch input

Image Colorization





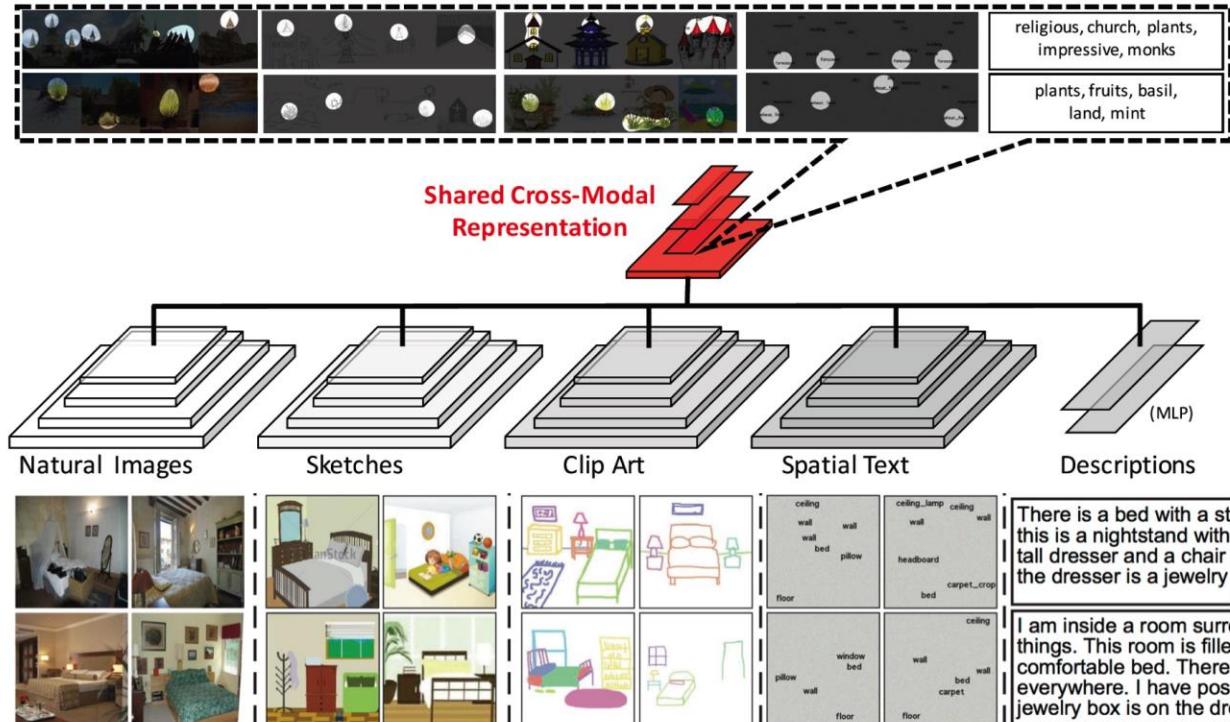
Multi-branch input





Multi-branch input

Cross-modal Scene Classification



<http://projects.csail.mit.edu/cmplaces/>



Multi-branch input

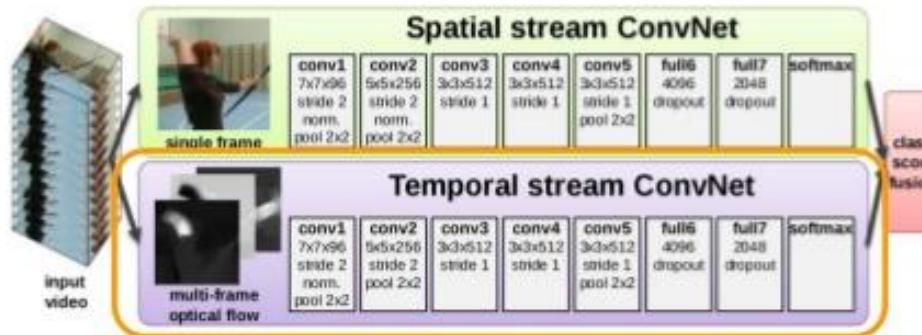
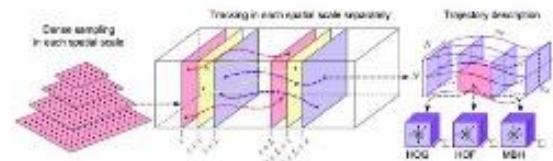
Activity Recognition

Recognition: Two stream

Two CNNs in parallel:

- One for RGB images
- One for Optical flow (hand-crafted features)

Fusion after the softmax layer

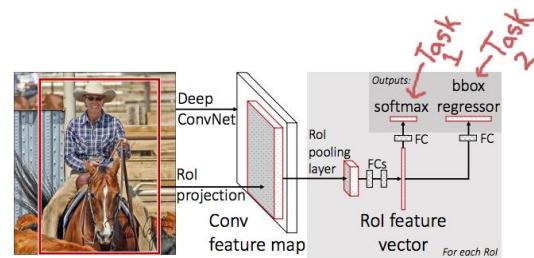


Simonyan, Karen, and Andrew Zisserman. ["Two-stream convolutional networks for action recognition in videos."](#) NIPS 2014.



Multi-branch output

Object Detection, Classification



Scene Parsing

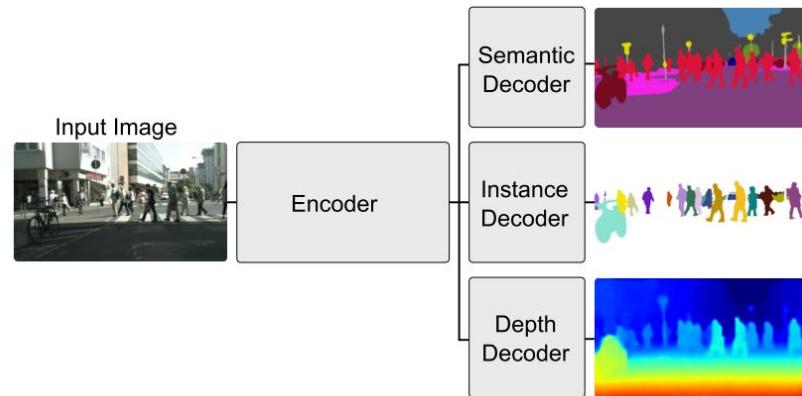
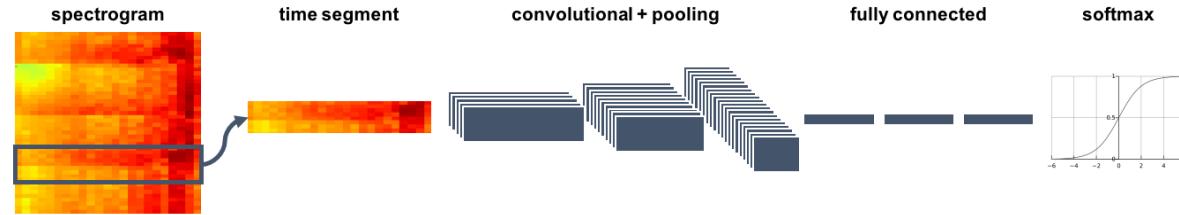




Image CNNs for non-image data

Audio Beat Detection



$$\equiv \mathbb{E}_{\mathcal{D}} \left[(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x}))^2 \right] = \underbrace{\mathbb{E}_{\mathcal{D}} \left[(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x}))^2 \right]}_{\text{var}(\mathbf{x})} + \underbrace{\left(\bar{g}(\mathbf{x}) - f(\mathbf{x}) \right)^2}_{\text{bias}(\mathbf{x})}$$

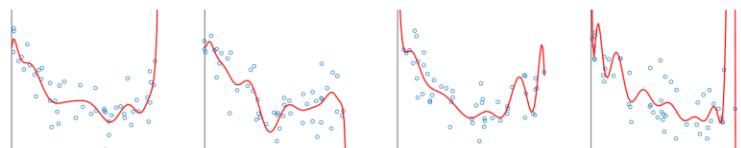
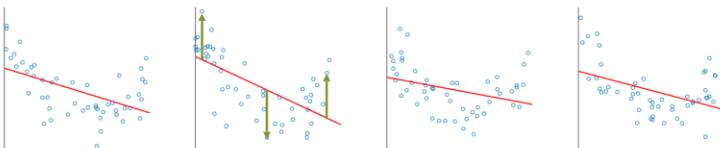
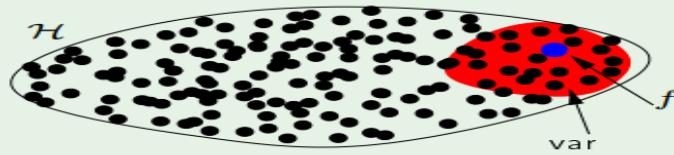
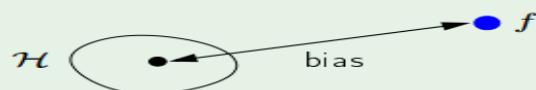


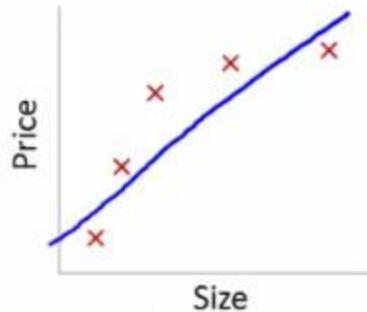
$$\mathbb{E}_{\mathbf{x}}[\text{bias}(\mathbf{x}) + \text{var}(\mathbf{x})]$$

The tradeoff

$$\text{bias} = \mathbb{E}_{\mathbf{x}} \left[(\bar{g}(\mathbf{x}) - f(\mathbf{x}))^2 \right]$$

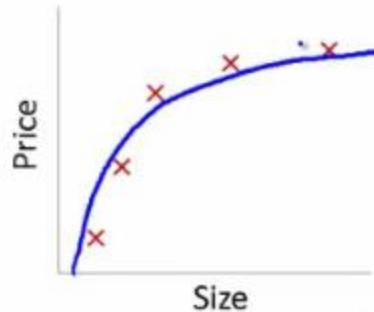
$$\text{var} = \mathbb{E}_{\mathbf{x}} \left[\mathbb{E}_{\mathcal{D}} \left[(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x}))^2 \right] \right]$$





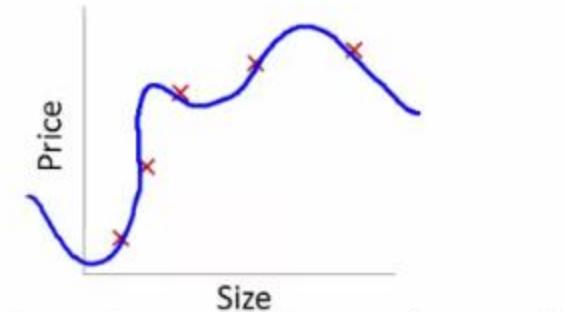
$$\theta_0 + \theta_1 x$$

High bias
(underfit)



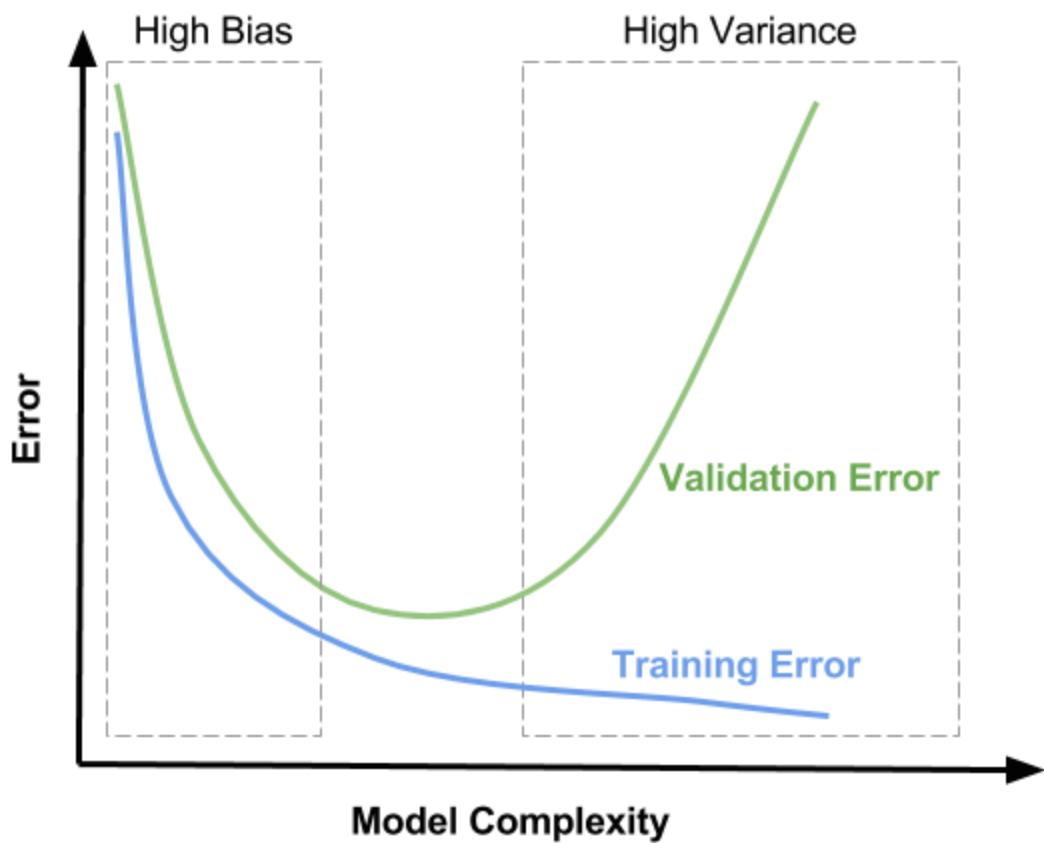
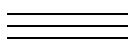
$$\theta_0 + \theta_1 x + \theta_2 x^2$$

"Just right"



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

High variance
(overfit)





Regularization

- Remember the intuition: complicated hypotheses lead to overfitting
- Idea: change the error function to *penalize hypothesis complexity*:

$$J(\mathbf{w}) = J_D(\mathbf{w}) + \lambda J_{pen}(\mathbf{w})$$

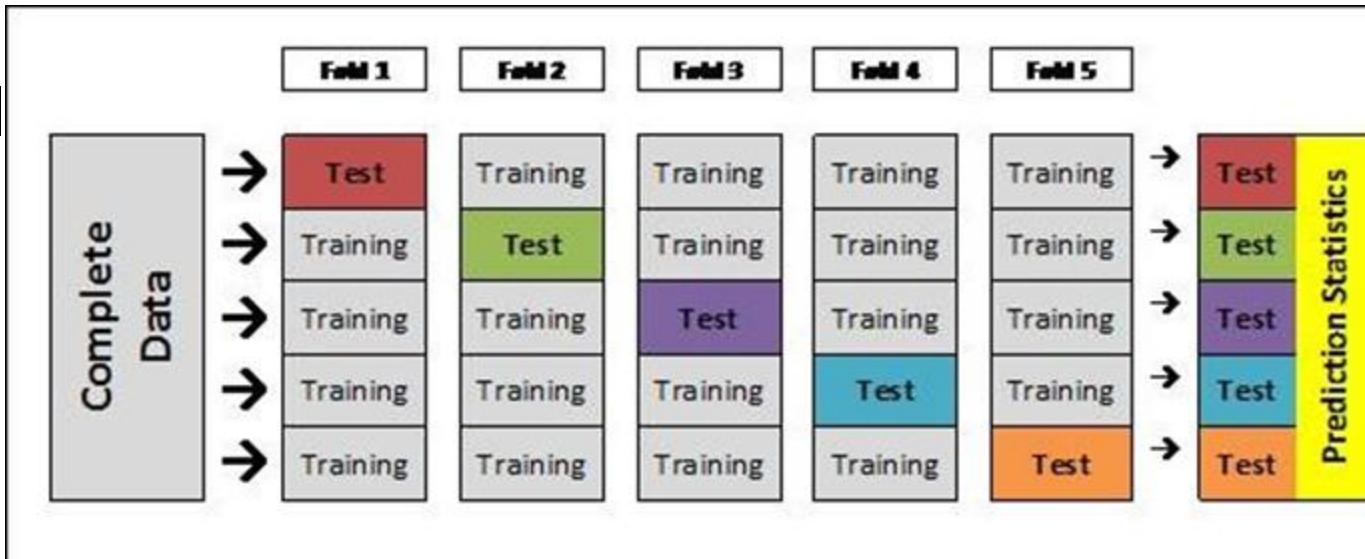
This is called *regularization* in machine learning and *shrinkage* in statistics

- λ is called *regularization coefficient* and controls how much we value fitting the data well, vs. a simple hypothesis



Fold → bias

K-fold Cross Validation



Assess model's bias-variance characteristics using k-fold Cross Validation?

- With k -fold CV , you get k different estimates of your model's error- $e_1, e_2, e_3, \dots, e_k$
- $\text{Mean(errors)} \sim 0 \rightarrow$ low bias
- $\text{Variance(errors)} \sim 0 \rightarrow$ low variance

Input:

Dataset L
Ensemble size T

```
1. for t=1 to T:  
2.   sample = BootstrapSample(L)  
3.   ht = TrainClassifier(sample)
```

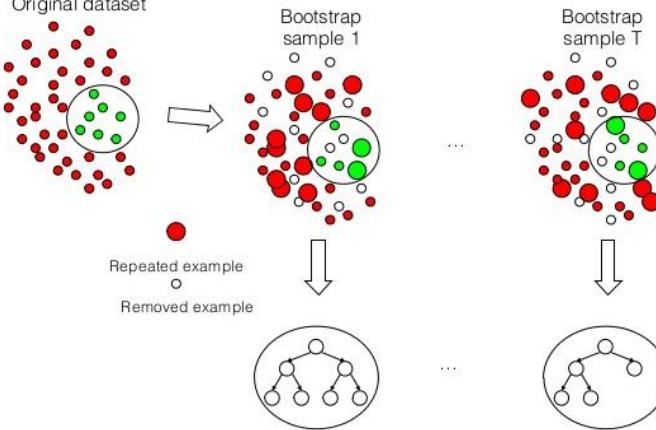
} *Bootstrap*

Output:

$$H(x) = \operatorname{argmax}_j \left(\sum_{t=1}^T I(h_t(x) = j) \right)$$

} *Aggregation*

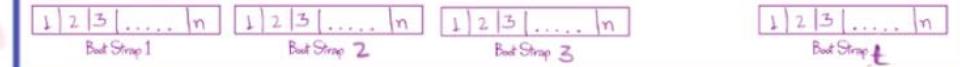
+

Original dataset

Bagging

Training Set

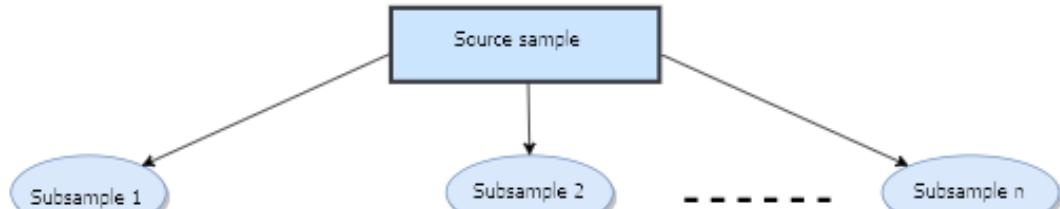
Draw n
With Replacement

**Hypotheses****Predictions****Voting****Final Prediction**

Random Forests

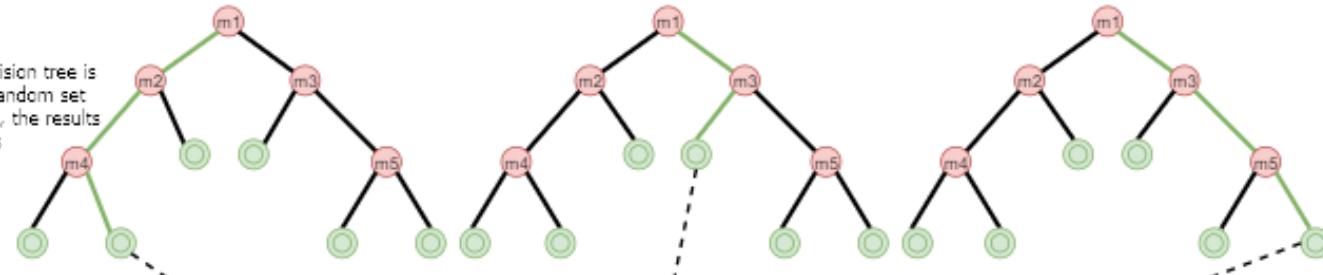
Bootstrap sampling

r (percentage) examples are selected
(0.63 in classical implementation)
in n random subsamples



Building the models

for each subsample, a decision tree is constructed based on a random set of m features (covariants), the results fall into leaves



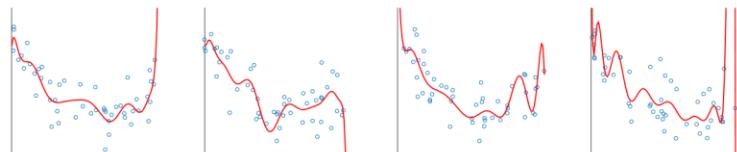
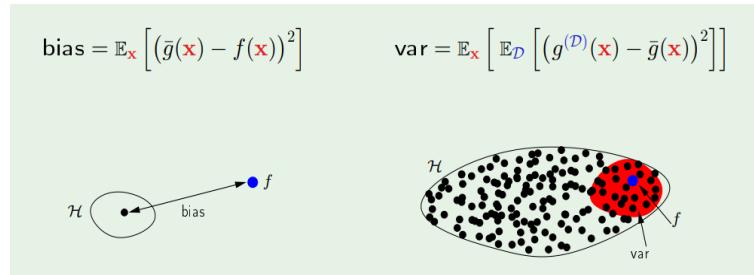
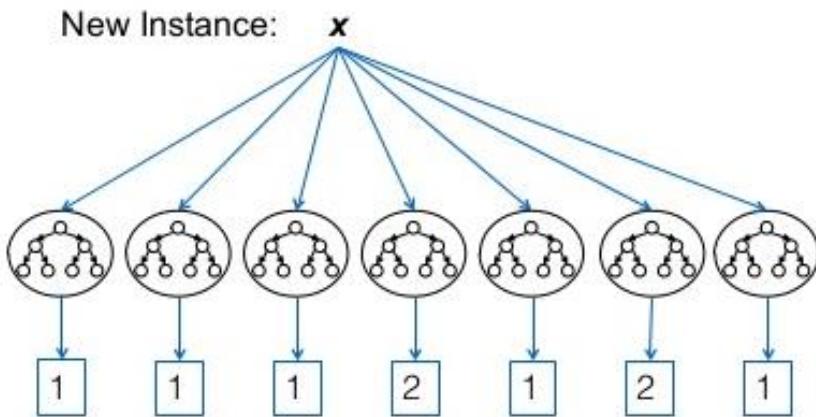
Bootstrap aggregating

results from all constructed trees are gathered and averaged

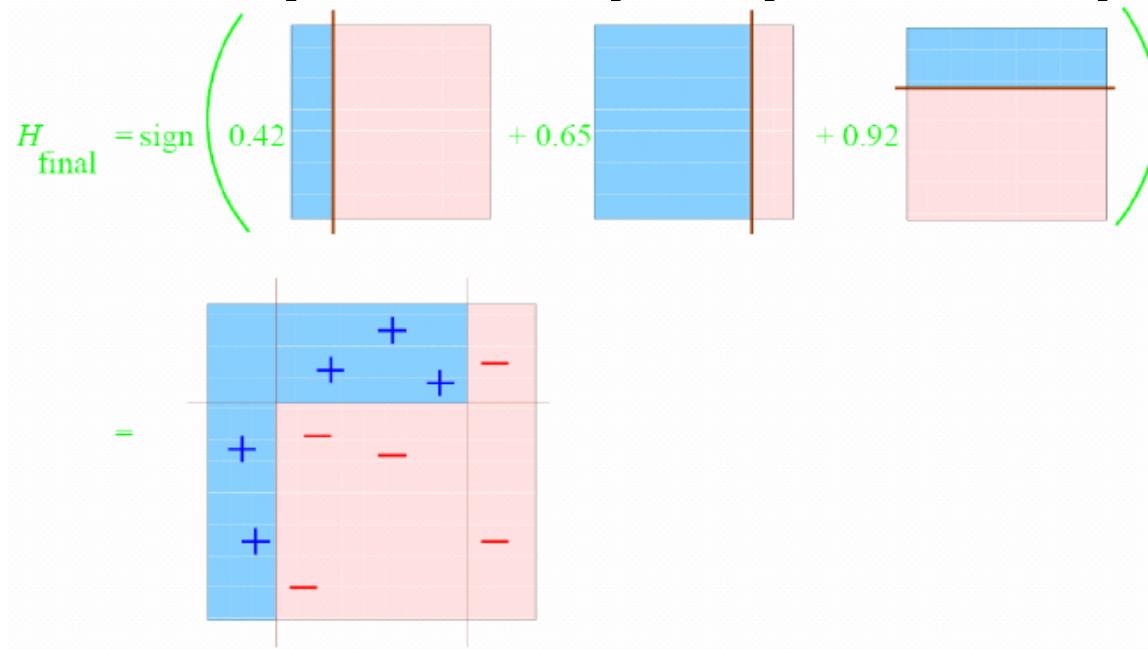


- Ensemble methods that minimize variance
 - Classifier Bagging
 - Classifier + Feature Bagging (e.g. Random Forests)

$$\mathbb{E}_{\mathcal{D}} \left[(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x}))^2 \right] = \underbrace{\mathbb{E}_{\mathcal{D}} \left[(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x}))^2 \right]}_{\text{var}(\mathbf{x})} + \underbrace{(\bar{g}(\mathbf{x}) - f(\mathbf{x}))^2}_{\text{bias}(\mathbf{x})}$$



A toy example(cont'd)



Final Classifier: integrate the three “weak” classifiers and obtain a final strong classifier.

Given: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.

Initialize $D_1(i) = 1/m$ for $i = 1, \dots, m$.

For $t = 1, \dots, T$:

- Train weak learner using distribution D_t .
- Get weak hypothesis $h_t : \mathcal{X} \rightarrow \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i].$$

Error of model

- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$.
- Update, for $i = 1, \dots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Coefficient of model

Update Distribution

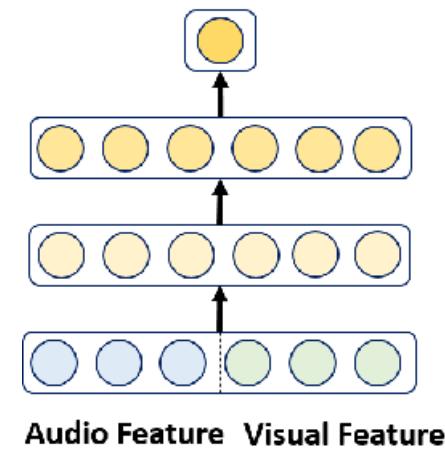
where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

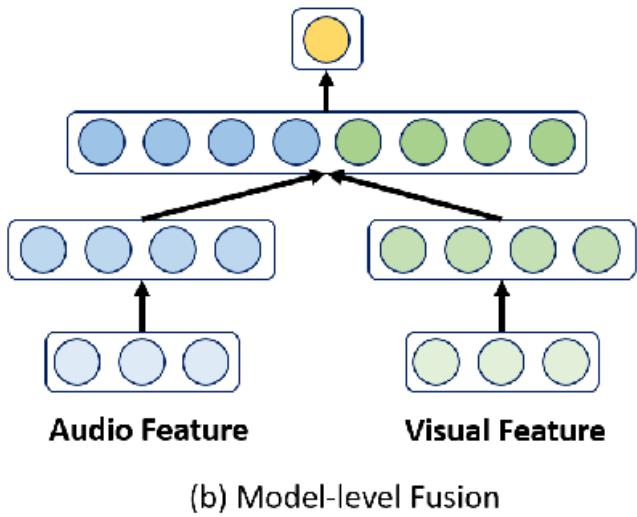
$$H(x) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(x) \right).$$

Final average

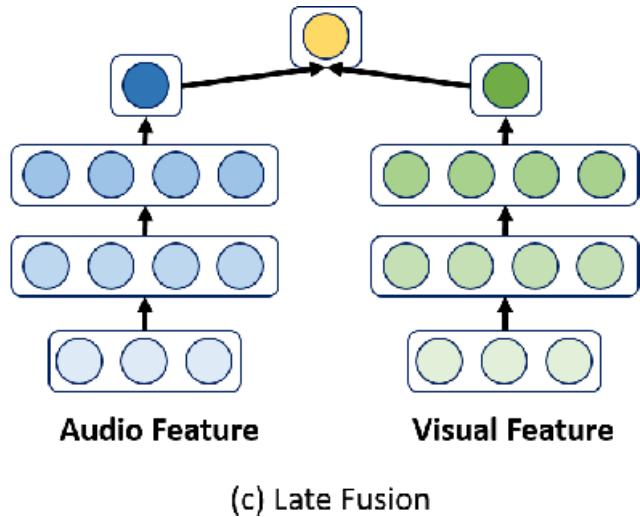
Theorem: training error drops exponentially fast



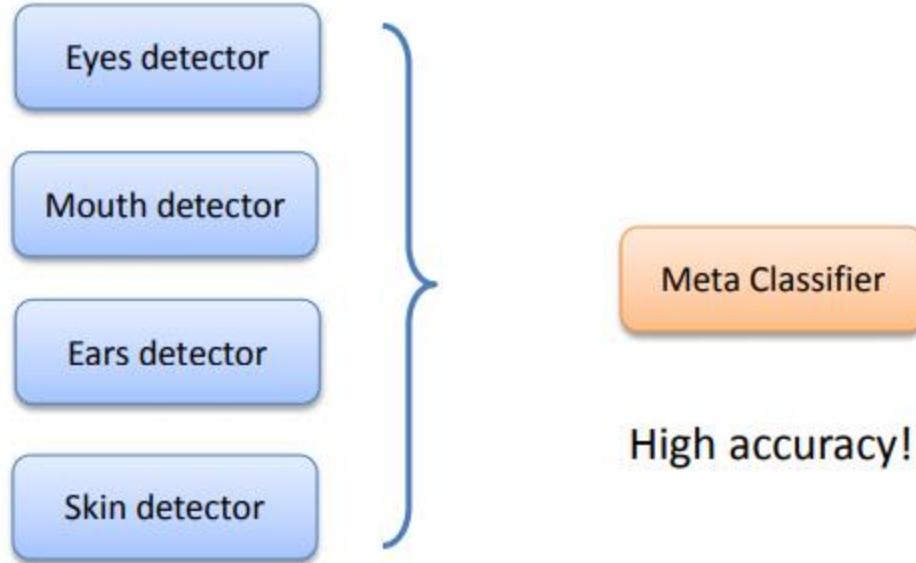
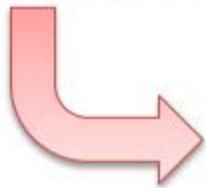
(a) Early Fusion



(b) Model-level Fusion



(c) Late Fusion

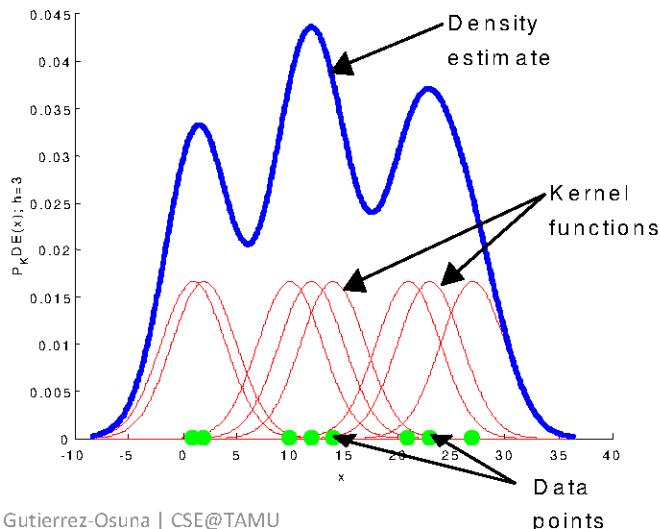


High accuracy!

Low individual accuracy
Computationally efficient

Interpretation

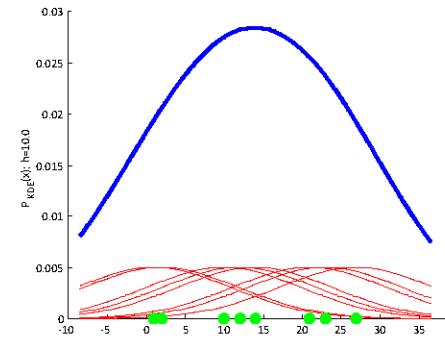
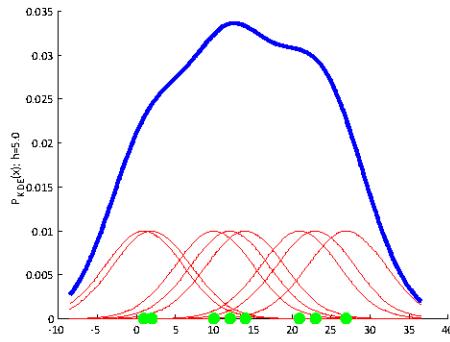
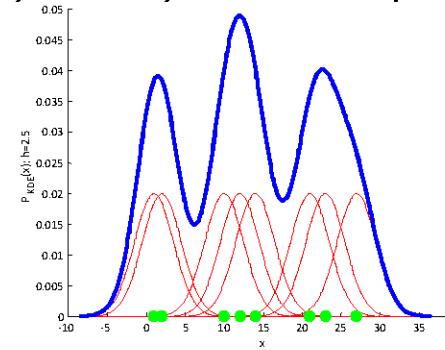
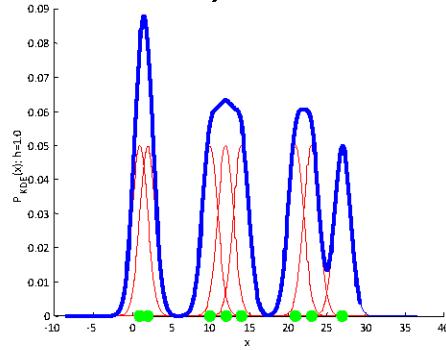
- Just as the Parzen window estimate can be seen as a sum of boxes centered at the data, the smooth kernel estimate is a sum of “bumps”
- The kernel function determines the shape of the bumps
- The parameter h , also called the smoothing parameter or bandwidth, determines their width



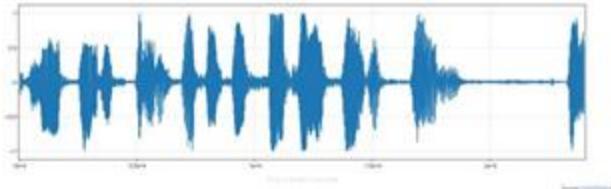
Bandwidth selection

The problem of choosing h is crucial in density estimation

- A large h will over-smooth the DE and mask the structure of the data
- A small h will yield a DE that is spiky and very hard to interpret



Sequential data



Audio

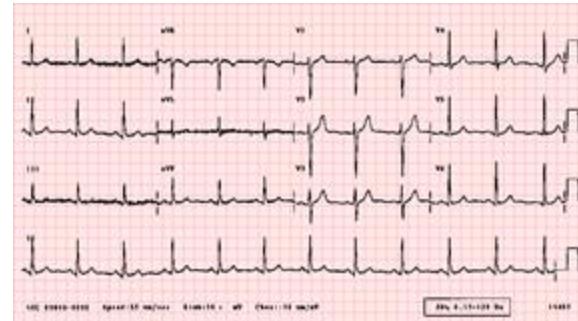
O Nachiketa, after pondering well the pleasures that are or seem to be delightful, you have renounced them all. You have not taken the road abounding in wealth, where many men sink.
(Kathopanishad, II:3)

Text

many more



Video



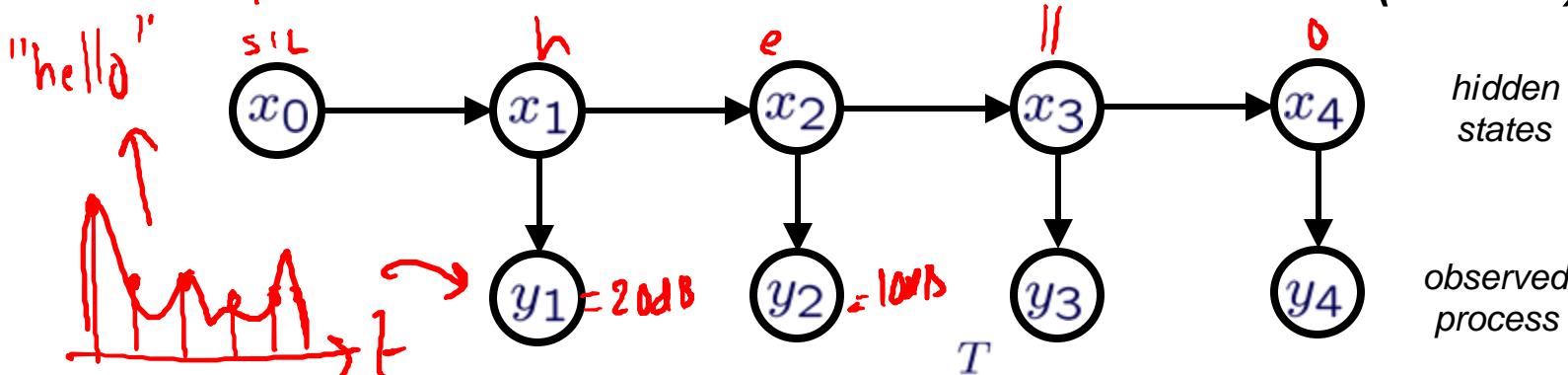
Biological

Hidden Markov Models

- Few realistic time series directly satisfy the assumptions of Markov processes:

*“Conditioned on the present,
the past & future are independent”*

- Motivates *hidden Markov models (HMM)*:



$$p(x_0, x_1, \dots, x_T) = p(x_0) \prod_{t=1}^T p(x_t | x_{t-1}) p(y_t | x_t)$$

Discrete HMMs: Observations

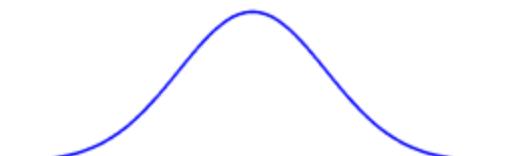
Discrete Observations

$$p(y_t | x_t = 1) = \begin{bmatrix} 0.3 \\ 0.1 \\ 0.5 \\ 0.1 \end{bmatrix}$$

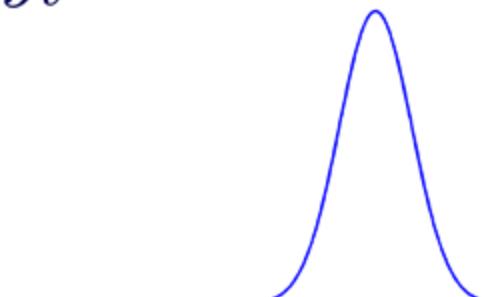
$$y_t \in \{1, 2, \dots, M\}$$

$$p(y_t | x_t = 2) = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.1 \\ 0.5 \end{bmatrix}$$

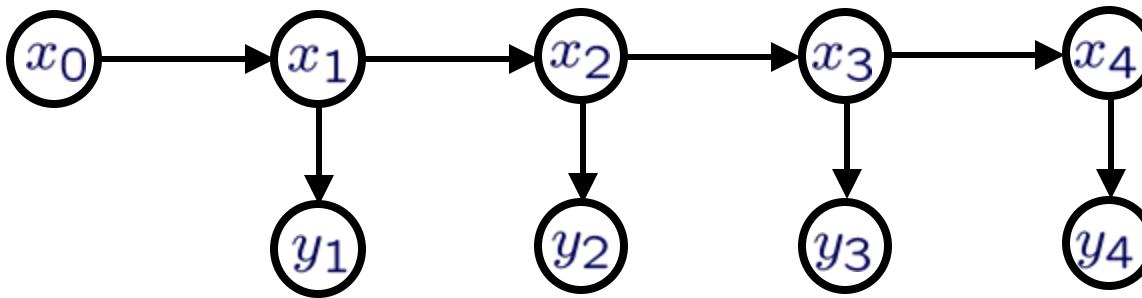
**Continuous
Observations**


$$\overline{p(y_t | x_t = 1)}$$

$$y_t \in \mathbb{R}^k$$

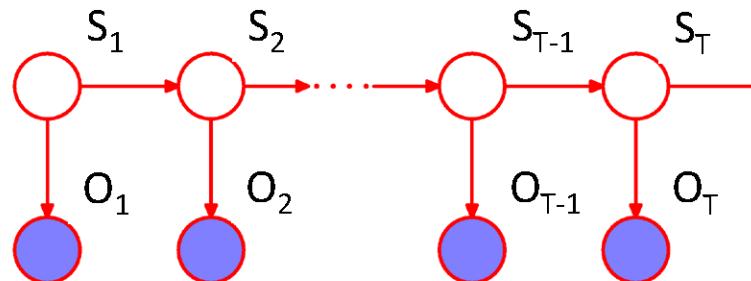

$$\overline{p(y_t | x_t = 2)}$$

Specifying an HMM



- Observation model: $P(y_i|x_i)$
- Transition model: $P(x_i|x_{i-1})$
- Initial state distribution: $P(x_0)$

Hidden Markov Models



$$p(S_1, \dots, S_T, O_1, \dots, O_T) = \prod_{t=1}^T p(O_t | S_t) \prod_{t=1}^T p(S_t | S_{t-1})$$

$$\begin{aligned} p(A, B) \\ p(A|B) &= \frac{p(A, B)}{p(B)} \\ &= \frac{p(A, B)}{\sum_A p(A, B)} \end{aligned}$$

1st order Markov assumption on hidden states $\{S_t\}$ $t = 1, \dots, T$
(can be extended to higher order).

Note: O_t depends on all previous observations $\{O_{t-1}, \dots, O_1\}$

Hidden Markov Models

- Parameters – stationary/homogeneous markov model
(independent of time t)

Initial probabilities

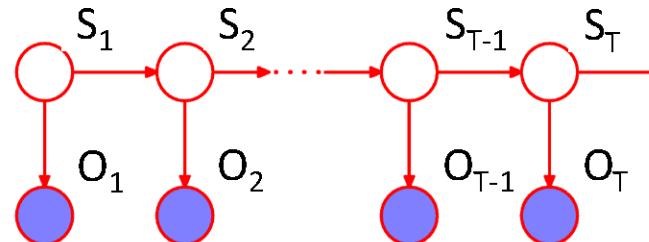
$$p(S_1 = i) = \pi_i$$

Transition probabilities

$$p(S_t = j | S_{t-1} = i) = p_{ij}$$

Emission probabilities

$$p(O_t = y | S_t = i) = q_i^y$$



$$p(\{S_t\}_{t=1}^T, \{O_t\}_{t=1}^T) =$$

$$p(S_1) \prod_{t=2}^T p(S_t | S_{t-1}) \prod_{t=1}^T p(O_t | S_t)$$

Three main problems in HMMs

$O_1 \rightarrow 1245526462146146136136661664661636616366163616515615115146123562344 \leftarrow O_T$

- **Evaluation** – Given HMM parameters & observation seqn $\{O_t\}_{t=1}^T$

find $p(\{O_t\}_{t=1}^T | \theta)$ prob of observed sequence

- How likely is this sequence, given our model of how the casino works?

- **Decoding** – Given HMM parameters & observation seqn $\{O_t\}_{t=1}^T$

find $\arg \max_{s_1, \dots, s_T} p(\{S_t\}_{t=1}^T | \{O_t\}_{t=1}^T, \theta)$ most probable

sequence of hidden states

- What portion of the sequence was generated with the fair die, and what portion with the loaded die?

- **Learning** – Given HMM with unknown parameters and $\{O_t\}_{t=1}^T$ observation sequence

find $\arg \max_{\theta} p(\{O_t\}_{t=1}^T | \theta)$ parameters that maximize

likelihood of observed data

- How “loaded” is the loaded die? How “fair” is the fair die? How often does the casino player change from fair to loaded, and back?

HMM Algorithms

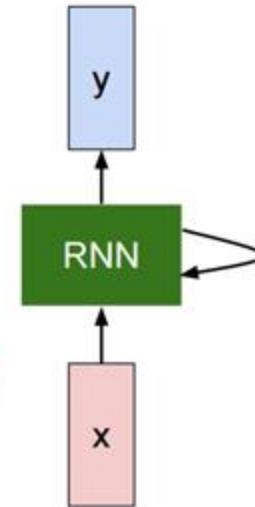
- **Evaluation** – What is the probability of the observed sequence? [Forward Algorithm](#)
- **Decoding** – What is the probability that the third roll was loaded given the observed sequence? [Forward-Backward Algorithm](#)
 - What is the most likely die sequence given the observed sequence? [Viterbi Algorithm](#)
- **Learning** – Under what parameterization is the observed sequence most probable? [Baum-Welch Algorithm \(EM\)](#)

Recurrent Neural Network

We can process a sequence of vectors x by applying a recurrence formula at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

new state / old state input vector at some time step
some function with parameters W



Task-specific abstraction for portion of sequence seen up to this point
 $(x_1, x_2 \dots x_{t-1})$

Training RNNs - Backpropagation through time

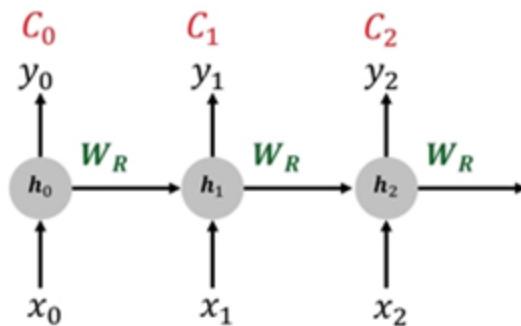
$$\mathbf{h}^{(t)} = g_h(W_I \mathbf{x}^{(t)} + W_R \mathbf{h}^{(t-1)} + \mathbf{b}_h)$$

$$\mathbf{y}^{(t)} = g_y(W_y \mathbf{h}^{(t)} + \mathbf{b}_y)$$

$$\frac{\partial C}{\partial W_R} = \sum_{t=1}^T \frac{\partial C_t}{\partial W_R}$$

$$\frac{\partial C_t}{\partial W_R} = \sum_{k=1}^t \frac{\partial C_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W_R}$$

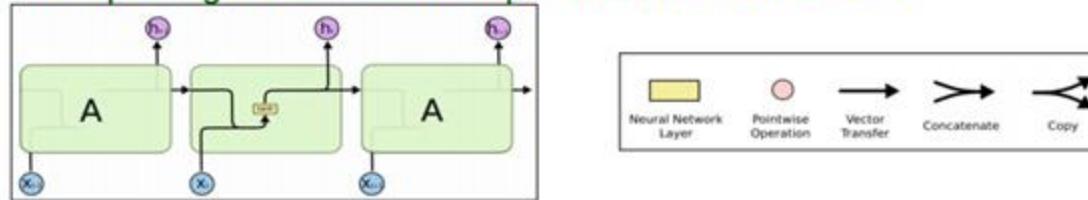
$$\frac{\partial h_t}{\partial h_k} = \prod_{i=k+1}^t \frac{\partial h_i}{\partial h_{i-1}} = \prod_{i=k+1}^t W_R^T \text{diag}(g'_h(W_I x^{(i)} + W_R h^{(i-1)} + b_h))$$



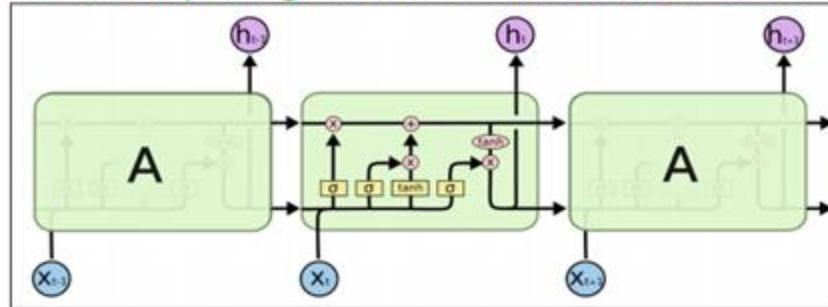
- (Potentially) multiple, intermediate costs C_0, C_1, C_2
- Shared weights \mathbf{W}_R

Long Short Term Memory (LSTM)

- Explicitly designed to avoid the long-term dependency problem
- RNNs have the form of a repeating chain structure
 - The repeating module has a simple structure such as tanh



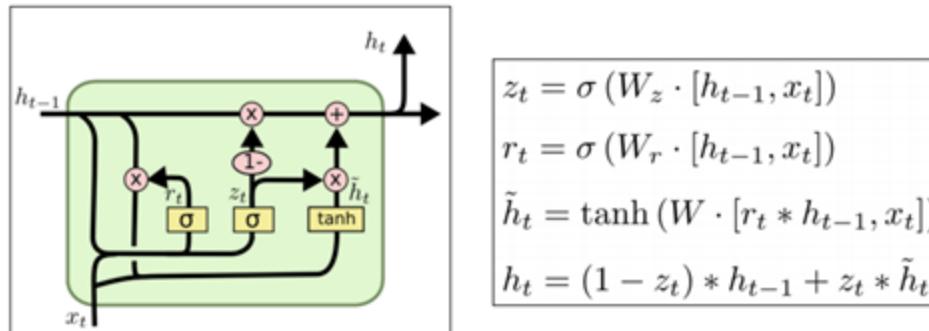
- LSTMs also have a chain structure
 - but the repeating module has a different structure



GRU: a LSTM variant

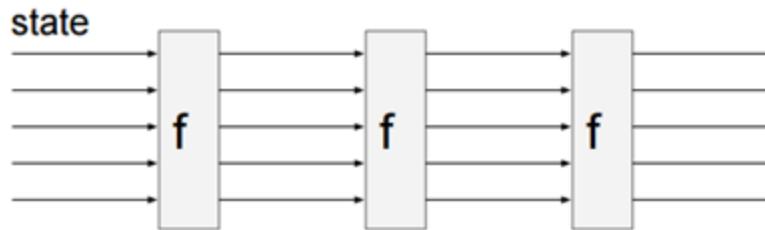
Gated Recurrent Unit (GRU)

- A dramatic variant of LSTM
 - It combines the forget and input gates into a single update gate
 - It also merges the cell state and hidden state, and makes some other changes
 - The resulting model is simpler than LSTM models
 - Has become increasingly popular



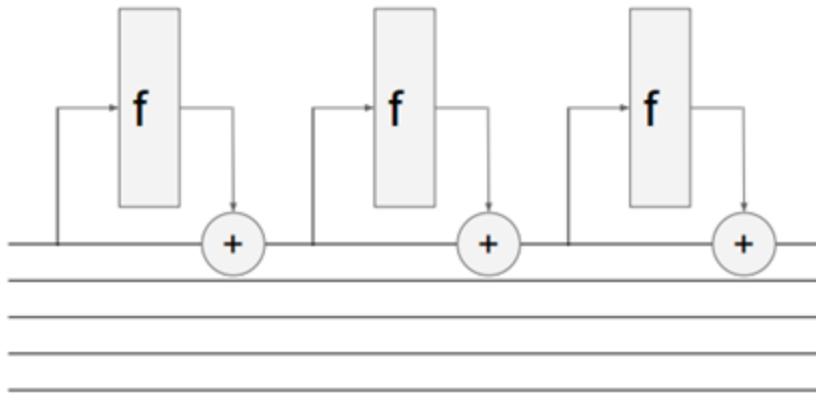
Long Short Term Memory (LSTM)

RNN

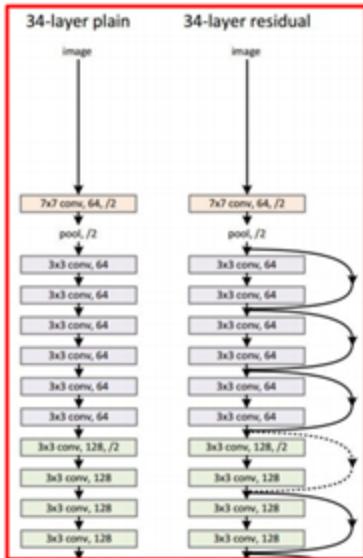


LSTM

(ignoring
forget gates)

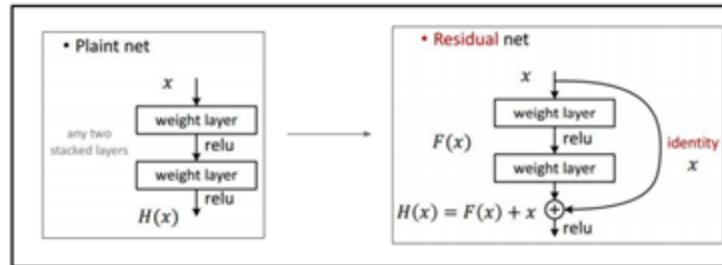


Long Short Term Memory (LSTM)



Recall:
“PlainNets” vs. ResNets

ResNet is to PlainNet what LSTM is to RNN, kind of.

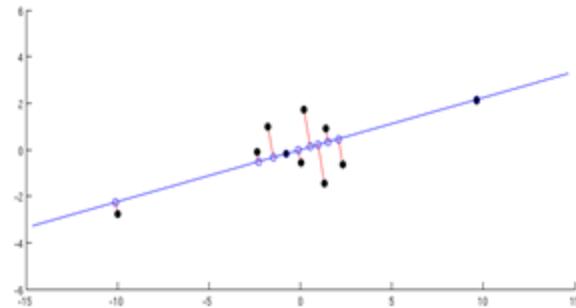
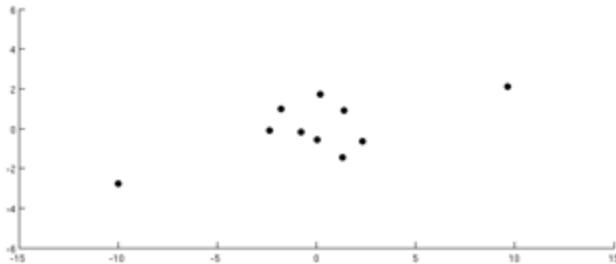


Getting targets when modeling sequences

- When applying machine learning to sequences, we often want to turn an input sequence into an output sequence that lives in a different domain.
 - *E. g. turn a sequence of sound pressures into a sequence of word identities.*
- When there is no separate target sequence, we can get a teaching signal by trying to predict the next term in the input sequence.
 - **The target output sequence is the input sequence with an advance of 1 step.**
 - This seems much more natural than trying to predict one pixel in an image from the other pixels, or one patch of an image from the rest of the image.
 - For temporal sequences there is a natural order for the predictions.
- Predicting the next term in a sequence blurs the distinction between supervised and unsupervised learning.
 - It uses methods designed for supervised learning, but it doesn't require a separate teaching signal.

Is PCA minimizing the right objective function ?

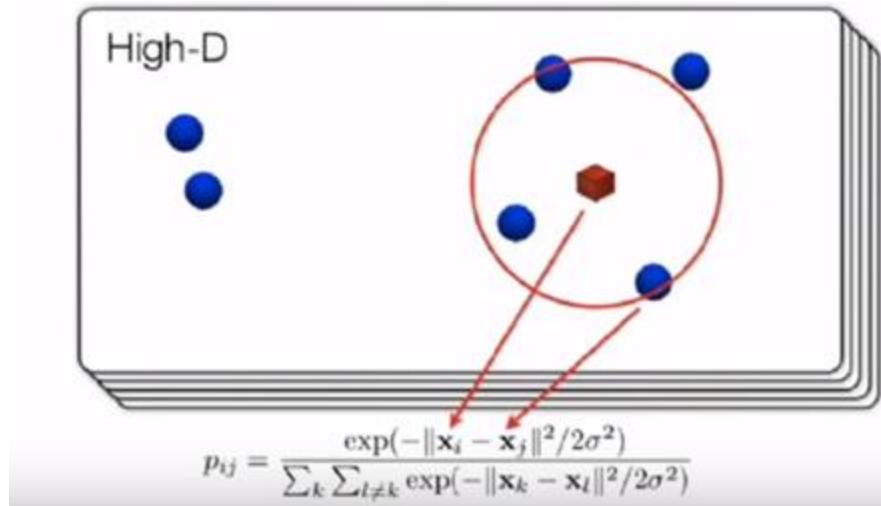
- PCA is mainly concerned with preserving **large** pairwise distances in the map



- “crowding” problem for smaller pairwise distances

t-SNE : Modeling Similarities in High-D

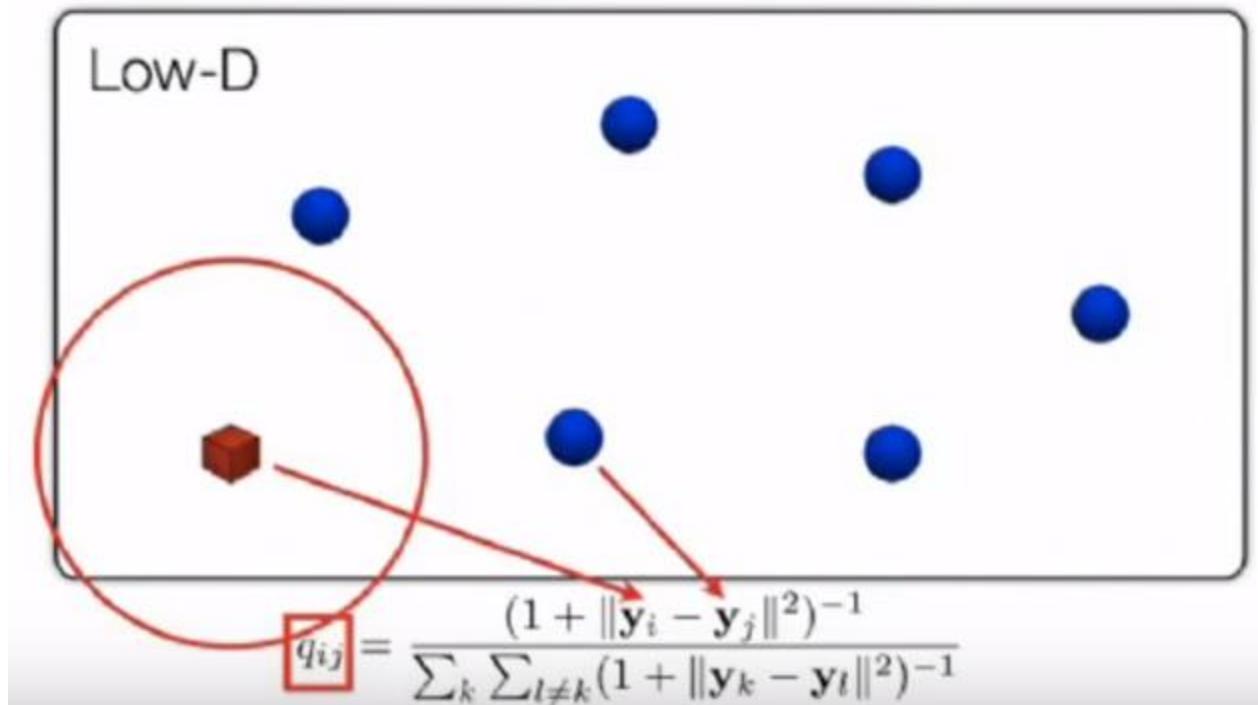
Center a gaussian under each point i



Prob of picking point pair (i,j) p_{ij} [their similarity]

Nearby points=>Large p_{ij} , Far points => Infinitesimal p_{ij}

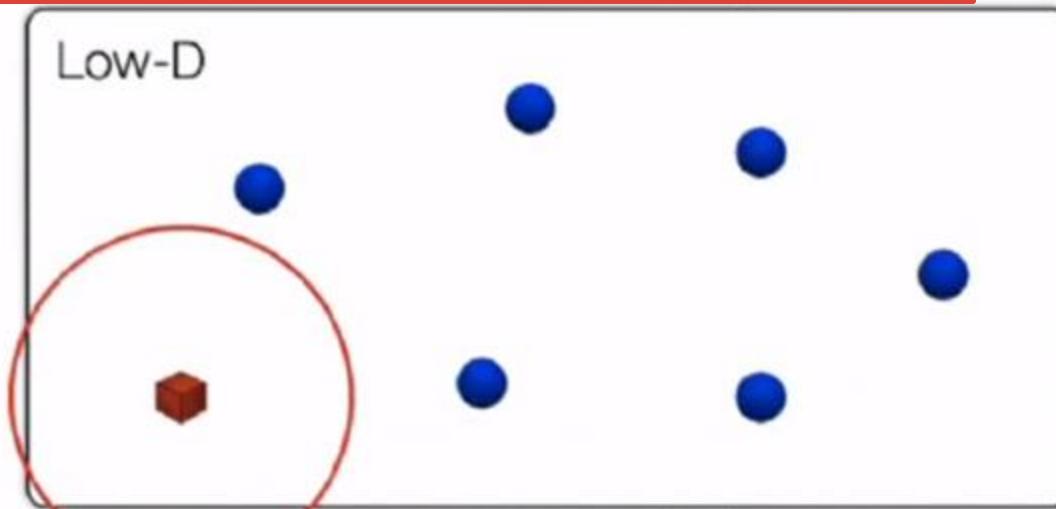
t-SNE : Modeling Similarities in Low-D



Lay out points in 2D/3D such that $q_{ij} \sim p_{ij}$

t-SNE : Modeling Similarities in Low-D

- Move points around to minimize: $KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}$



$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

Lay out points in 2D/3D such that $q_{ij} \sim p_{ij}$

t-SNE: Optimization

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{j' \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_{j'}\|^2/2\sigma_i^2)}$$

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

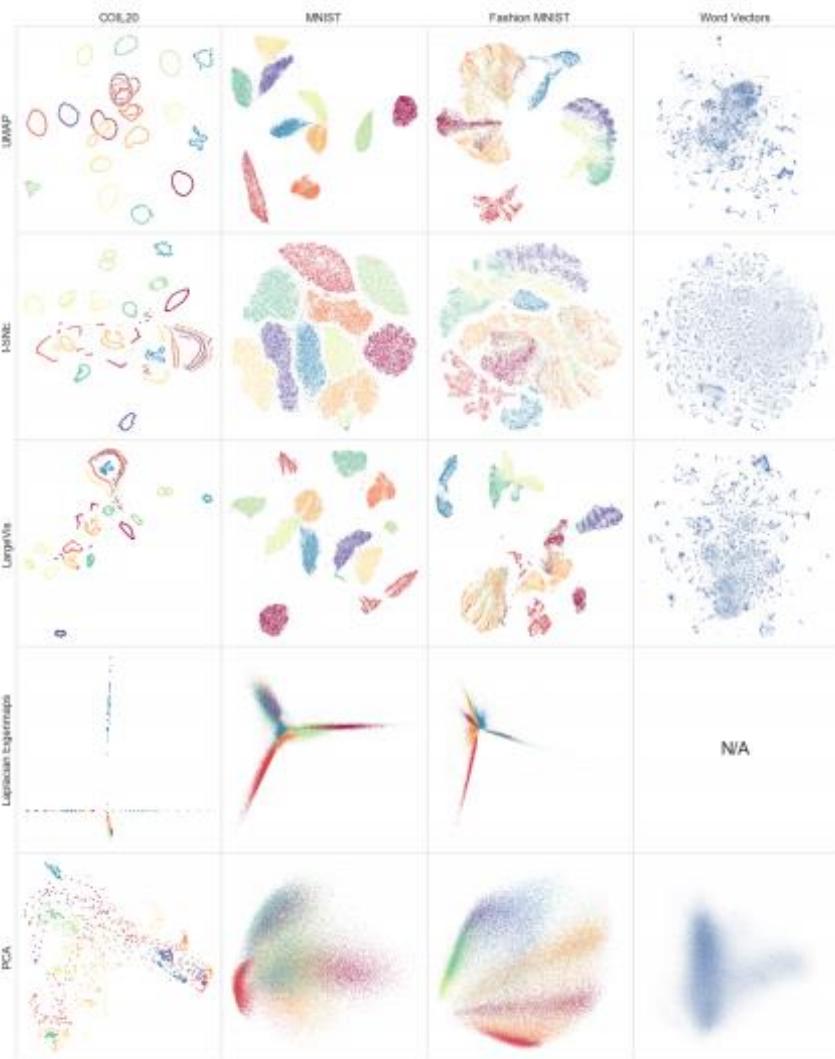
$$C = \sum_i KL(P_i || Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

$$\frac{\delta C}{\delta y_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

$$\gamma^{(t)} = \gamma^{(t-1)} + \eta \frac{\delta C}{\delta \gamma} + \alpha(t) \left(\gamma^{(t-1)} - \gamma^{(t-2)} \right)$$

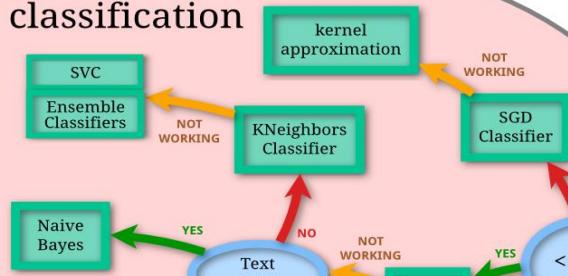
UMAP

(Universal Manifold approximation)

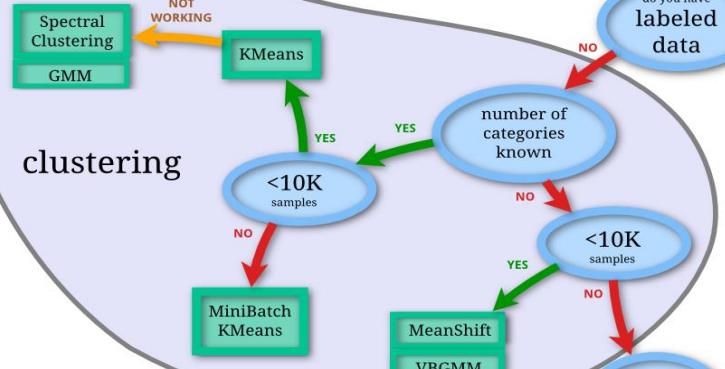


scikit-learn algorithm cheat-sheet

classification



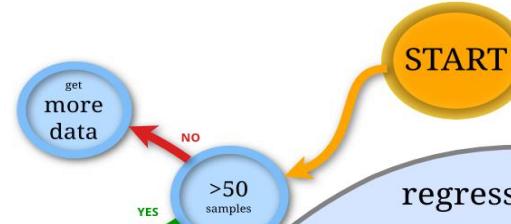
clustering



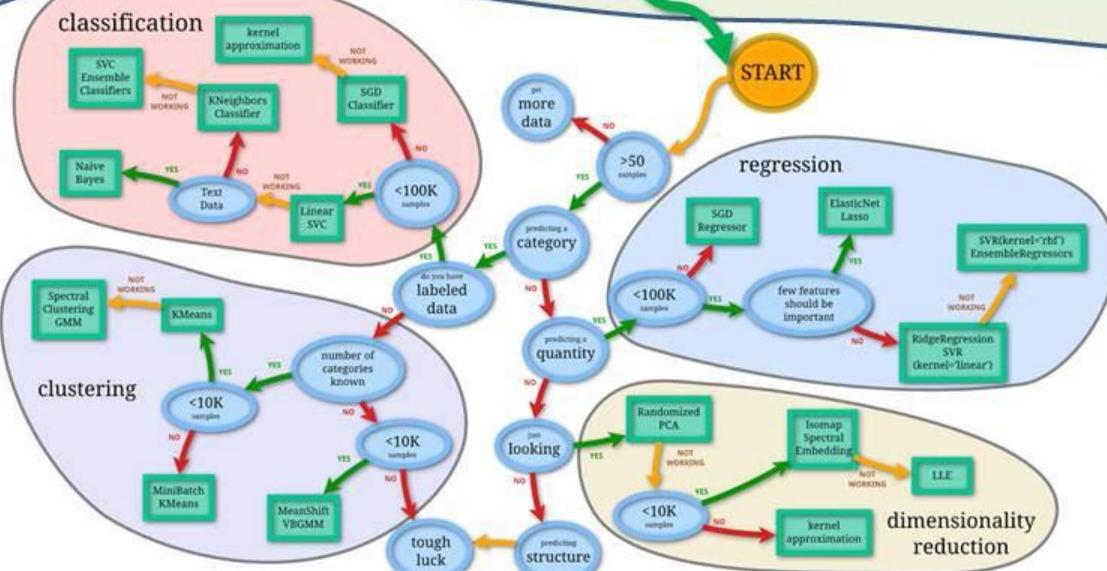
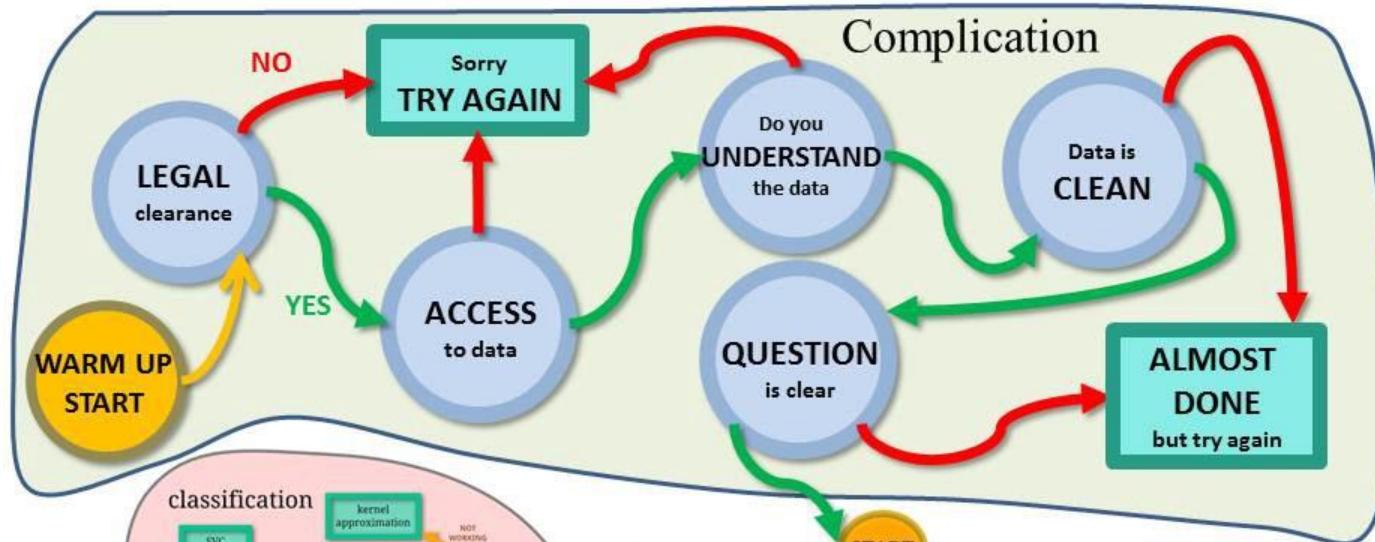
Back

scikit
learn

regression

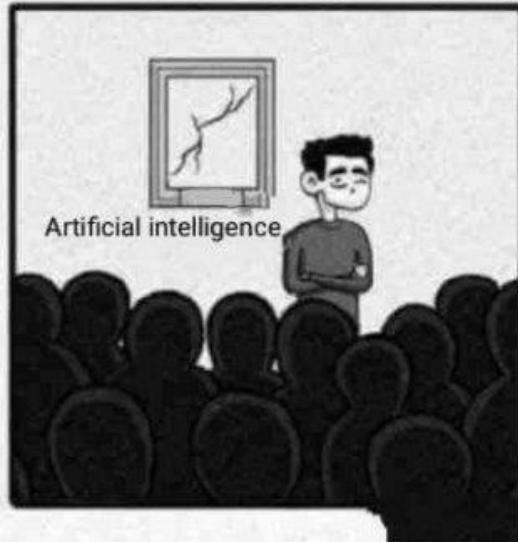
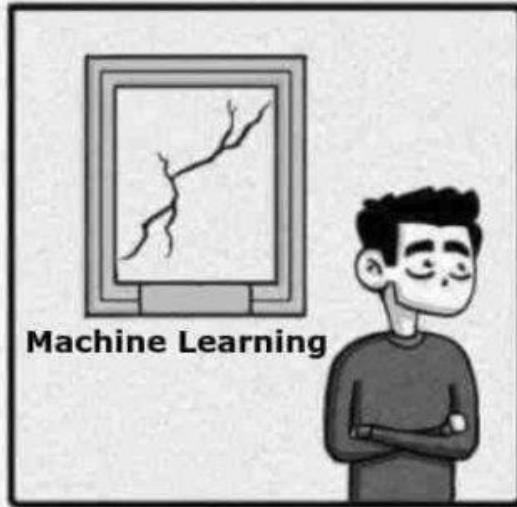
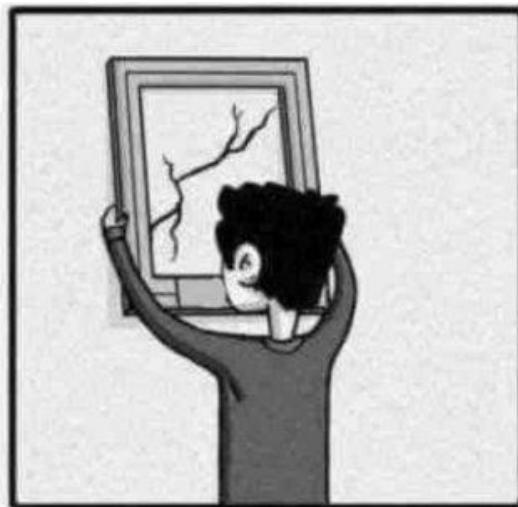
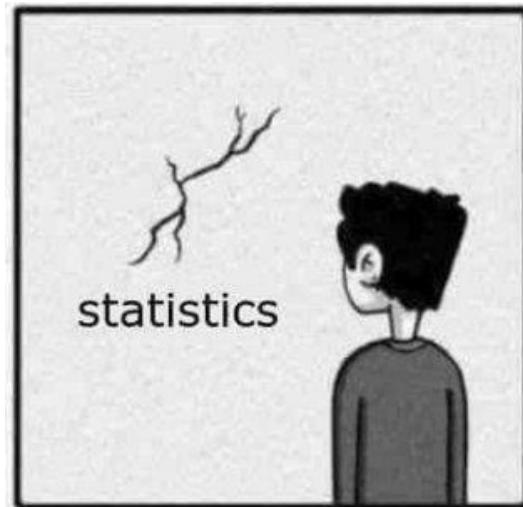


dimensionality reduction

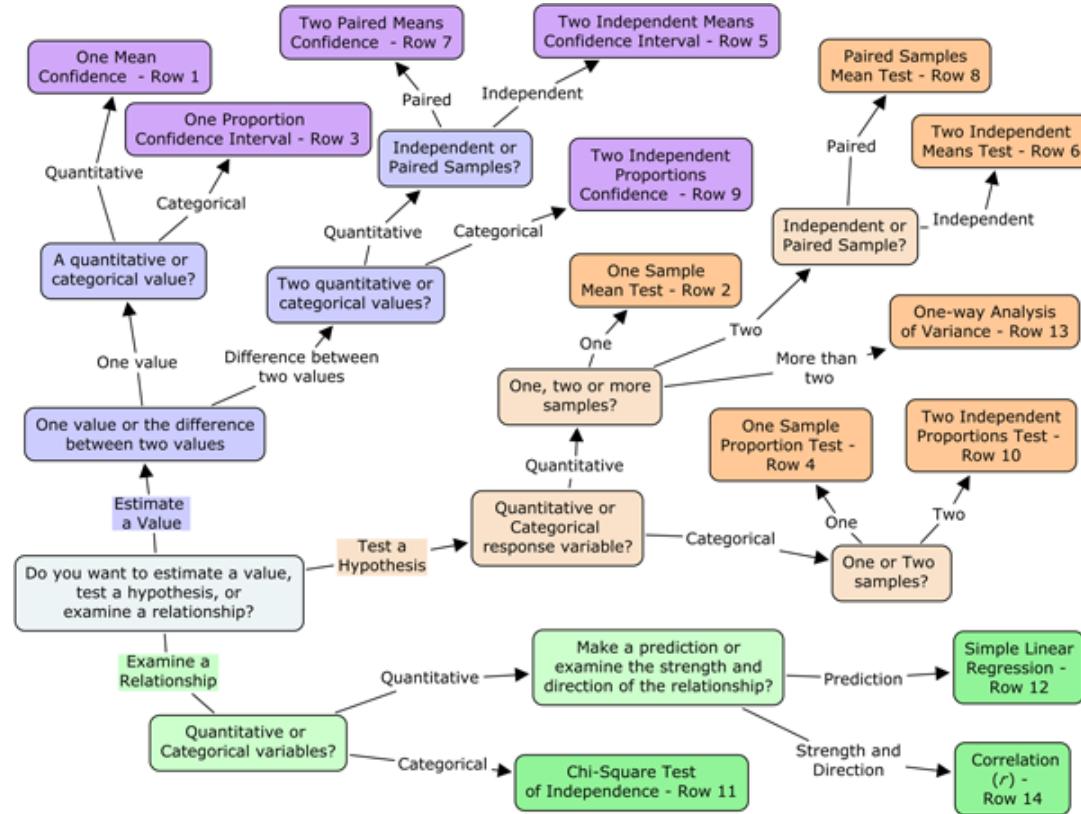


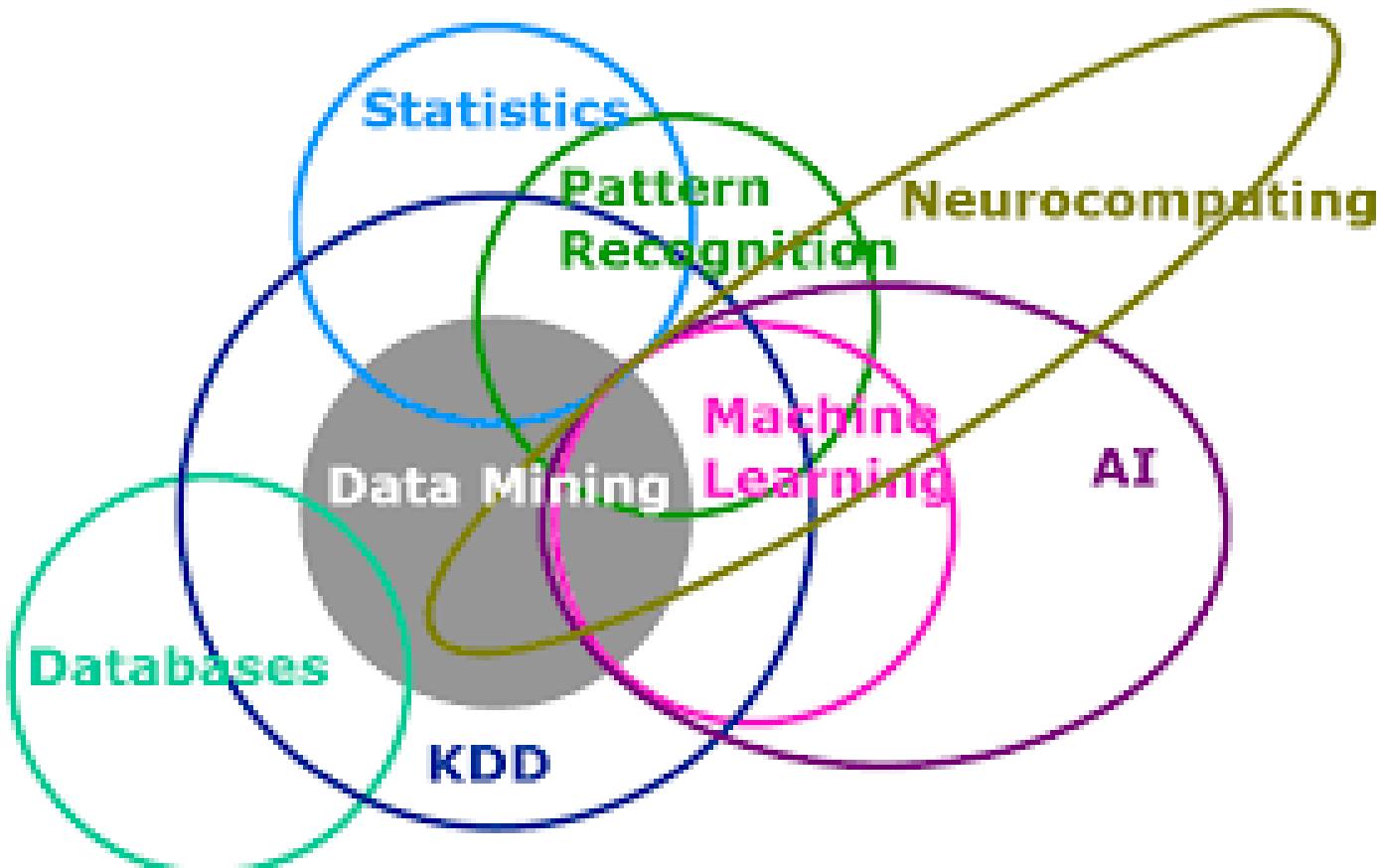
SMAI (Statistical Methods in AI)

- SMAI ~ Introduction to Machine Learning
- Good news: One half is already familiar to you [Machine !]
- Other half = What this course is about !



Statistical Methods - cheatsheet





Know the practical limits of ML

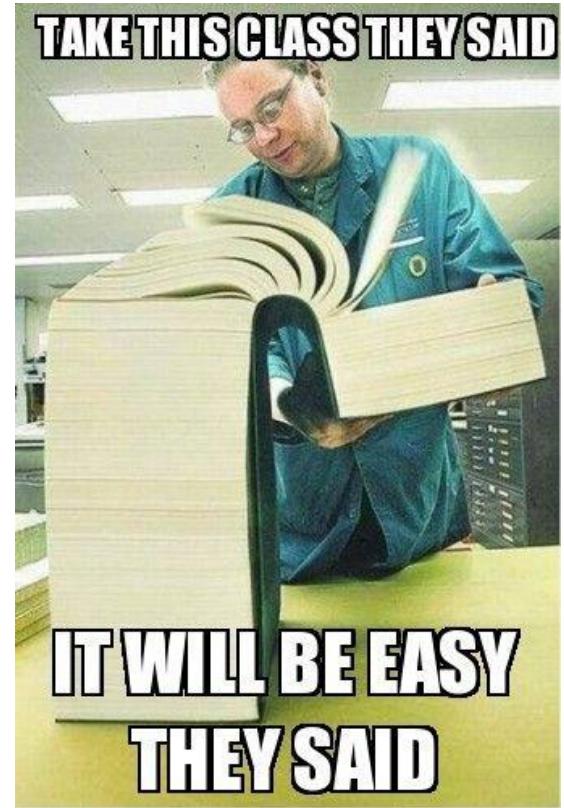


Course Objectives

- Determine whether ML is suitable for a problem
- Formulate a problem as a ML problem (data ,representations, tasks, algorithms)
- Understand and apply ML method(s)
- Be aware of ML pitfalls, follow best practices
- Be ready to dive deeper (into ML theory or applied areas)

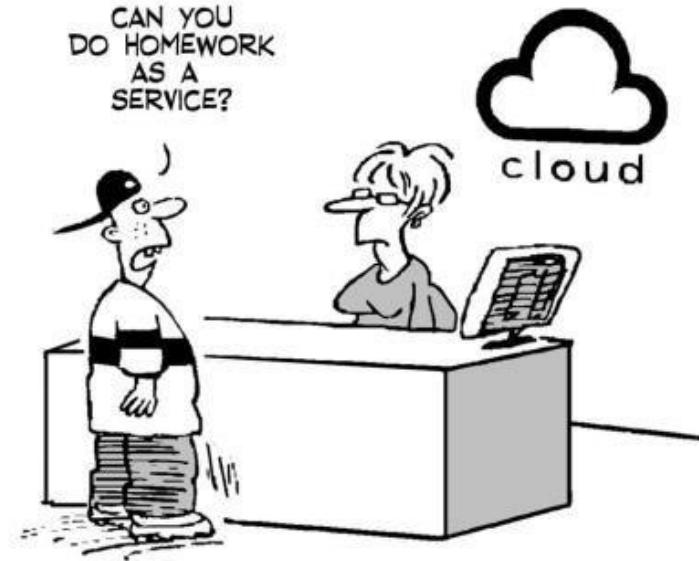
About the course – Grading Policy

- Assessment
 - 2 mid semester exams ($2 \times 15\% = 30\%$)
 - 1 Final Exam (30%)
 - 13 Assignments ($13 \times 2\% = 26\%$)
 - 1 Project (14%)



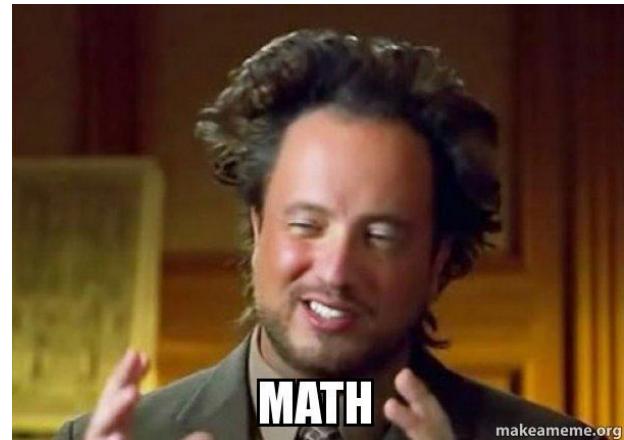
About the course – collaboration policy

- OK to discuss assignment questions and approaches
- But work must be your own (no copying – partially or fully)
- If you worked with someone, mention their name(s)
- We will be checking for copying/plagiarism
- Better to own up than be caught !

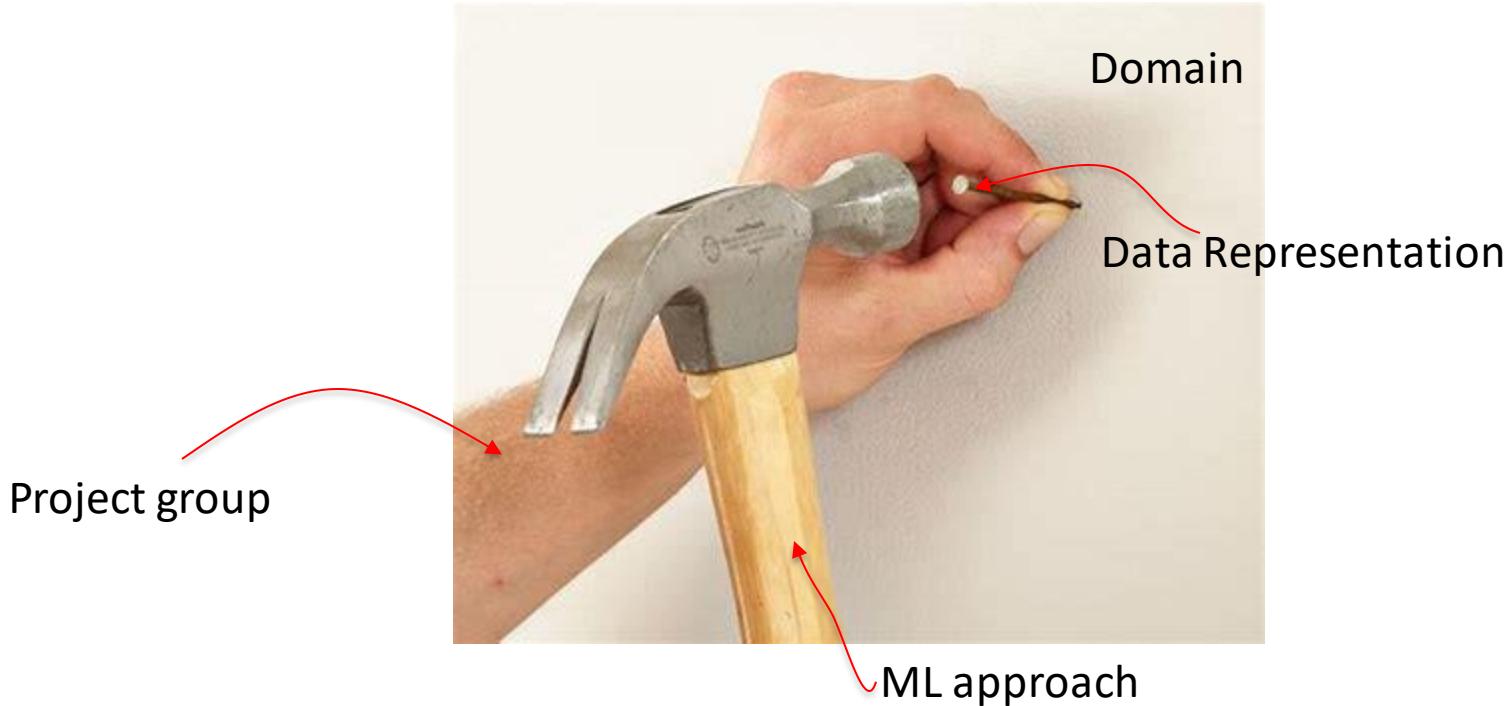


Additionally ...

- **Understand**, don't just memorize
- Love the math, not the toolbox !
- Capture the broad ideas and insights (useful years down the line)
- Implement ! No substitute for experience.
- Just the beginning



Projects



Projects

- Problem/Task-first thinking
- Come up with a top-3 list and why the problem is interesting / novel to study
- Be creative / original