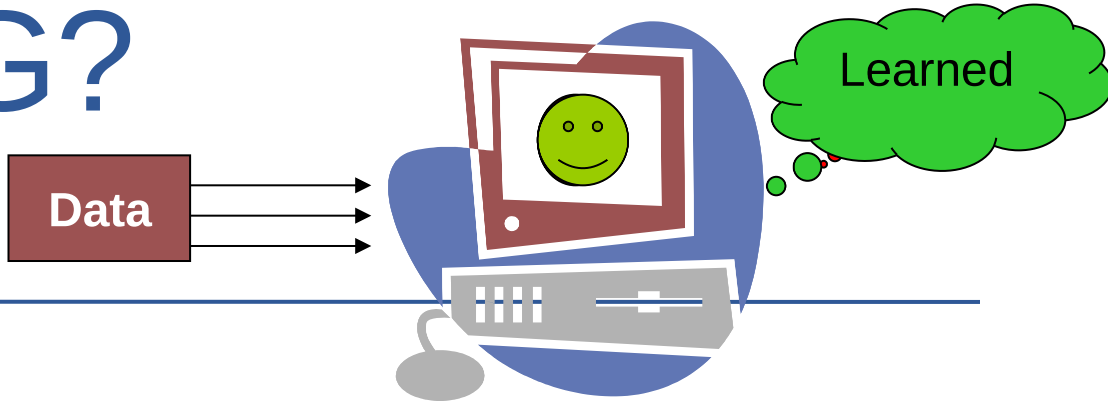
WHY MACHINE LEARNING?



Lecture-1

STTP on “Deep Learning, Computer Vision and Speech Processing”

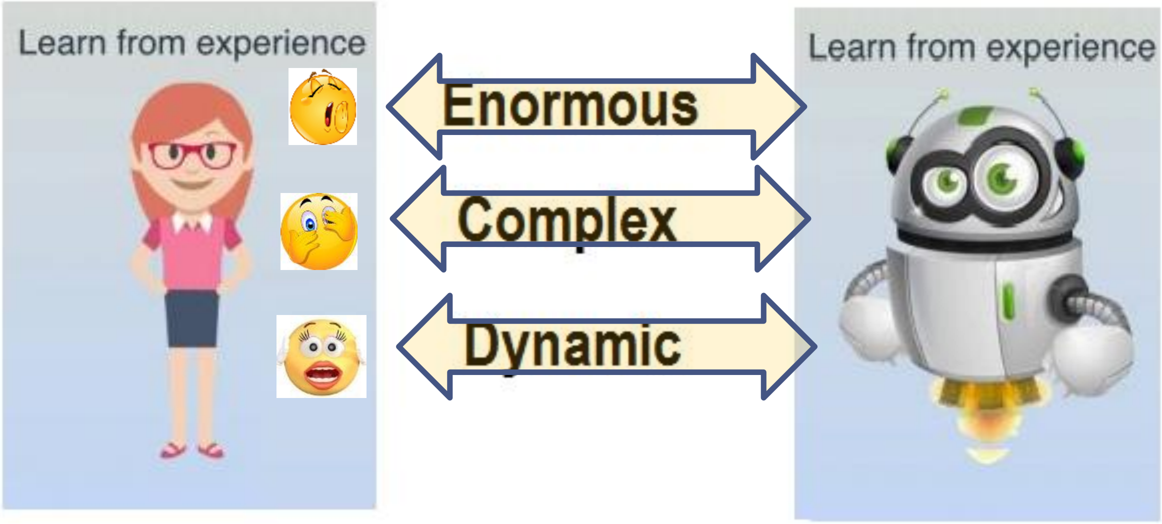
By: Suprava, Patnaik, Professor, ExTC, XIE, Mumbai

Outline

- Introduction to Machine Learning
- Classification, Regression, Clustering
- Performance Analysis
- Naïve Bays, NN, SVM
- Research conveniences

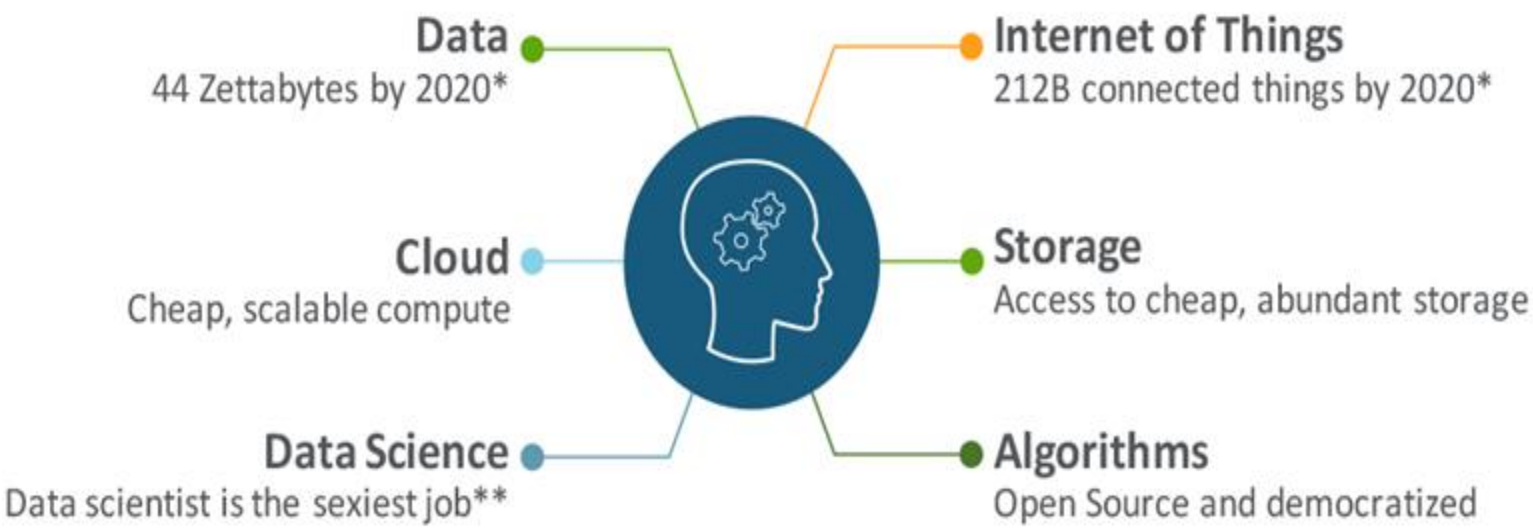
What is machine learning?

- Tools used for <u>large scale data</u> processing
- Techniques are suitable for <u>complex datasets</u> with huge number of variables and features, **<u>dynamic</u>** and ever changing data.
- Machine learning is the **brain of AI** that **extrapolates patterns**.

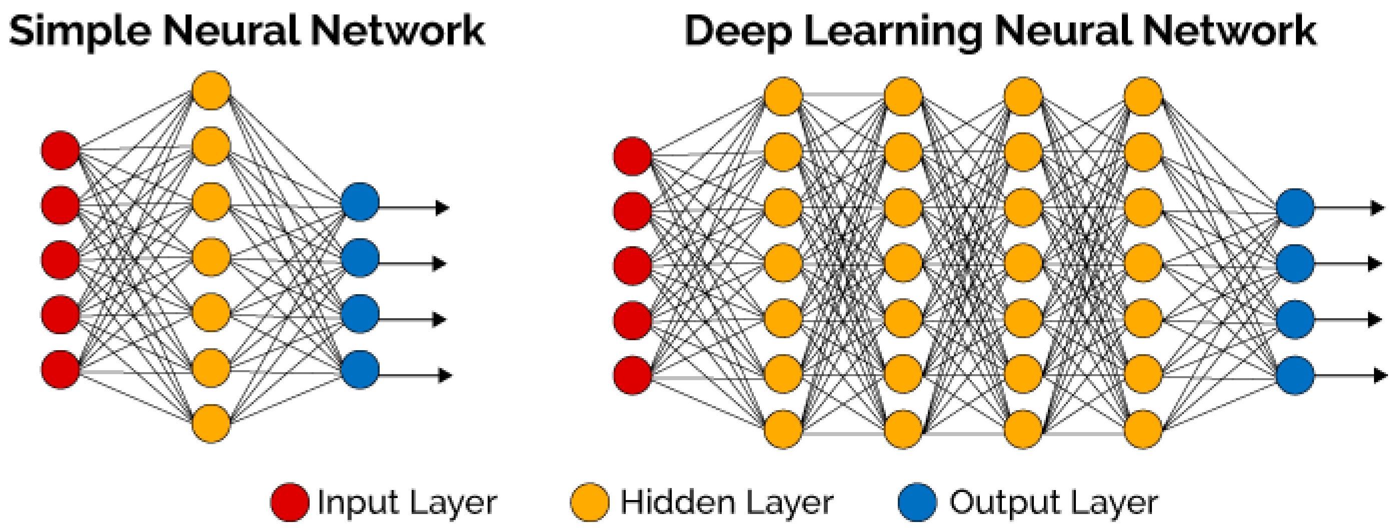


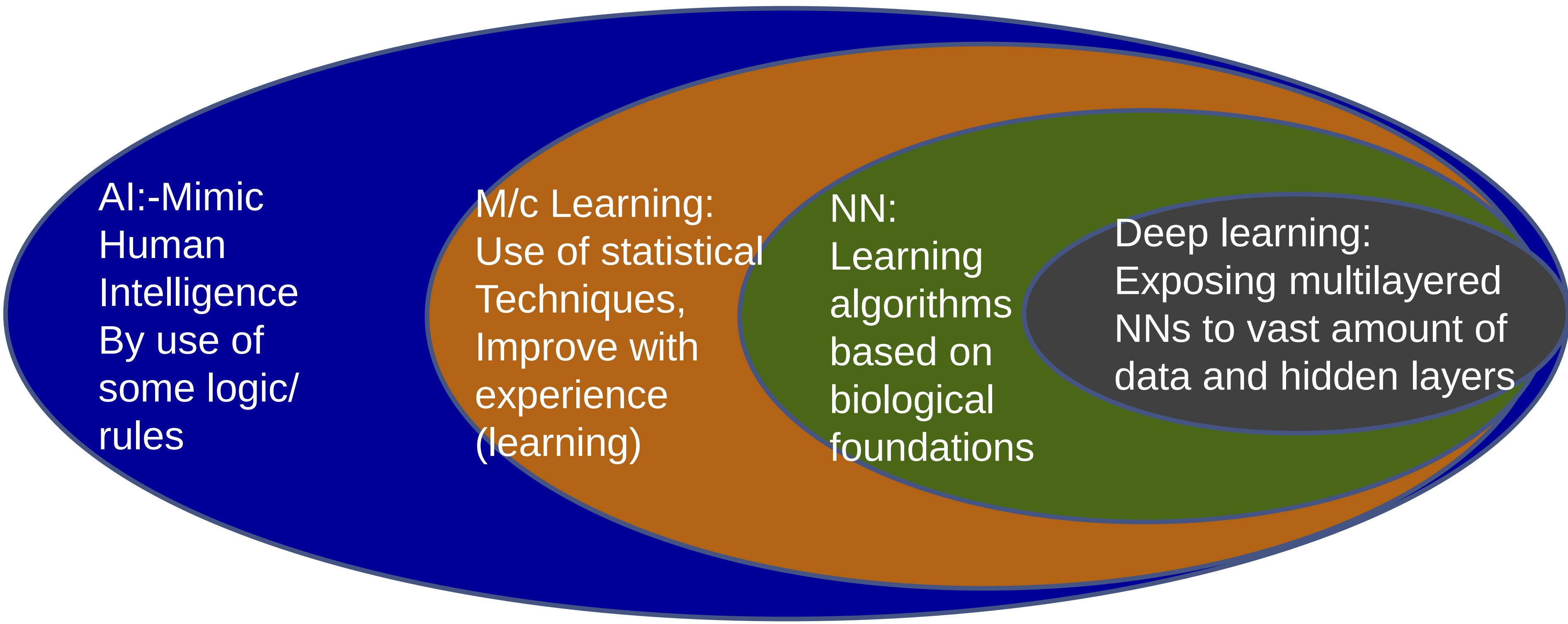
Why “Machine Learning”…?

- Human expertise does not exist (navigating on Mars),
- Humans are unable to explain their expertise (speech/ hand written text recognition)
- Solution changes in time (routing on a computer network)
- Solution needs to be adapted to particular cases (user biometrics)



Over-view


Simple Neural Network
Deep Learning Neural Network
Input Layer
Hidden Layer
Output Layer


AI:-Mimic Human Intelligence By use of some logic/ rules
M/c Learning: Use of statistical Techniques, Improve with experience (learning)
NN: Learning algorithms based on biological foundations
Deep learning: Exposing multilayered NNs to vast amount of data and hidden layers

CPU vs GPU

deep learning needs considerable hardware to run efficiently

- Extensive training done over large database with enormous samples
- Massive computation

Suppose you have to transfer goods from one place to the other.
Ferrari or a freight truck….?????

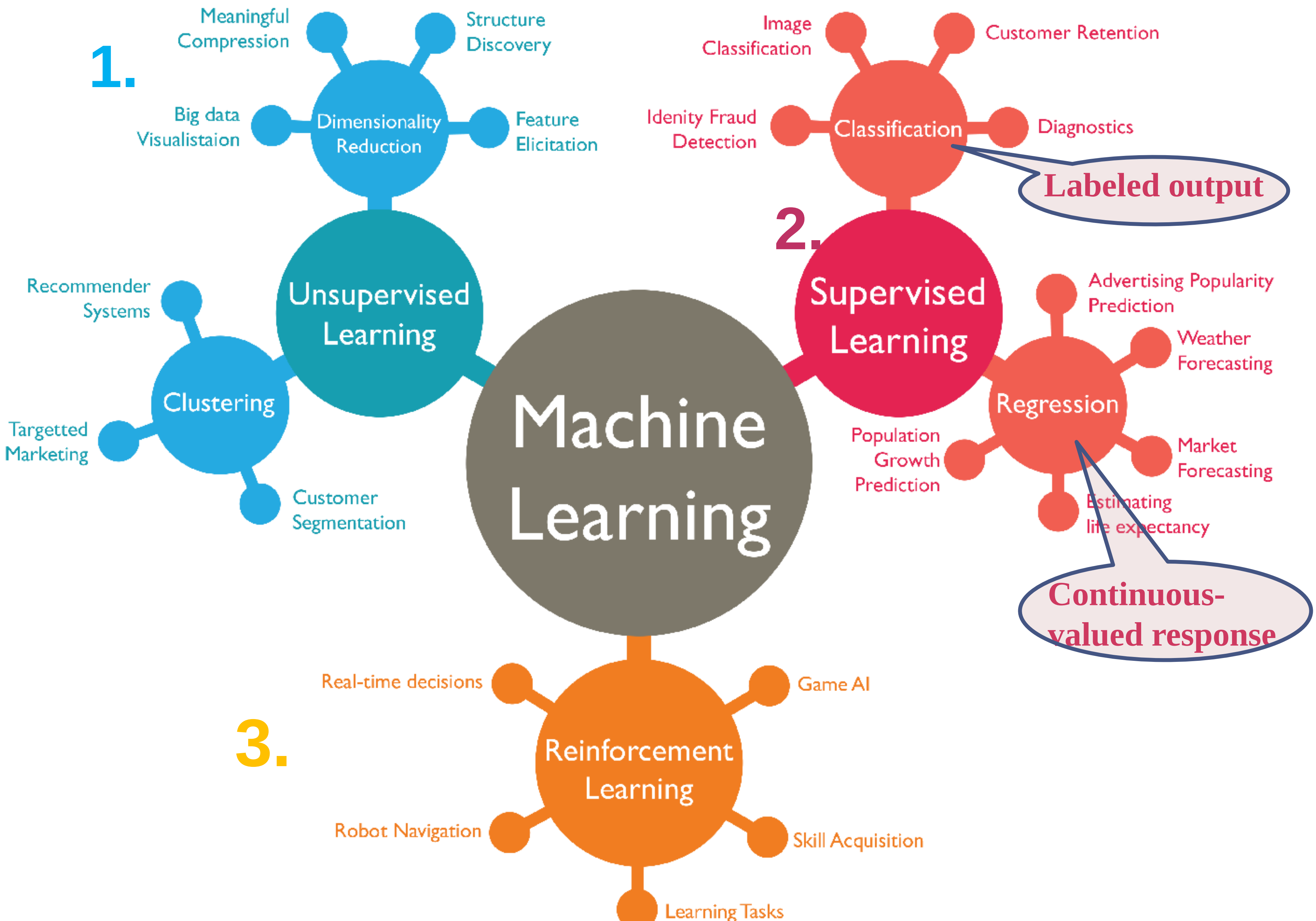
Evolution from NN to Deep-learning

- computationally intensive part of neural network is made up of multiple matrix multiplications.

- This is because GPUs were designed to handle these matrix operations in parallel, as this is essential for computer games for e.g. 3D effects, land rendering, etc. On the other hand, a single core CPU would take a matrix operation in serial, one element at a time.

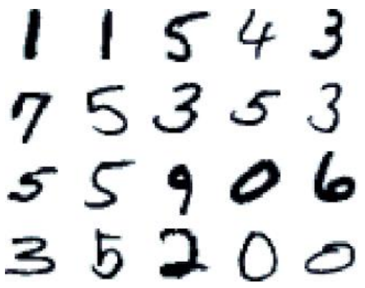
- A single GPU could have hundreds or thousands of cores, while a CPU typically has no more than a few cores (between two and eight).

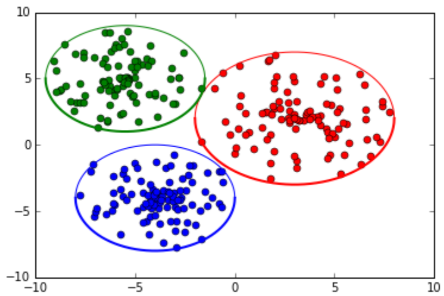
Learning


1.
Meaningful Compression
Structure Discovery
Big data Visualistaion
Dimensionality Reduction
Feature Elicitation
Recommender Systems
Unsupervised Learning
Clustering
Targetted Marketing
Customer Segmentation
2.
Image Classification
Customer Retention
Idenity Fraud Detection
Classification
Diagnostics
Labeled output
Supervised Learning
Advertising Popularity Prediction
Weather Forecasting
Regression
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Continuous-valued response
Machine Learning
3.
Real-time decisions
Game AI
Reinforcement Learning
Robot Navigation
Skill Acquisition
Learning Tasks

Learning

- Supervised:
  - uses set of labeled data
  - Classification, regression
  - hand written digits from MNIST (examples/ data)

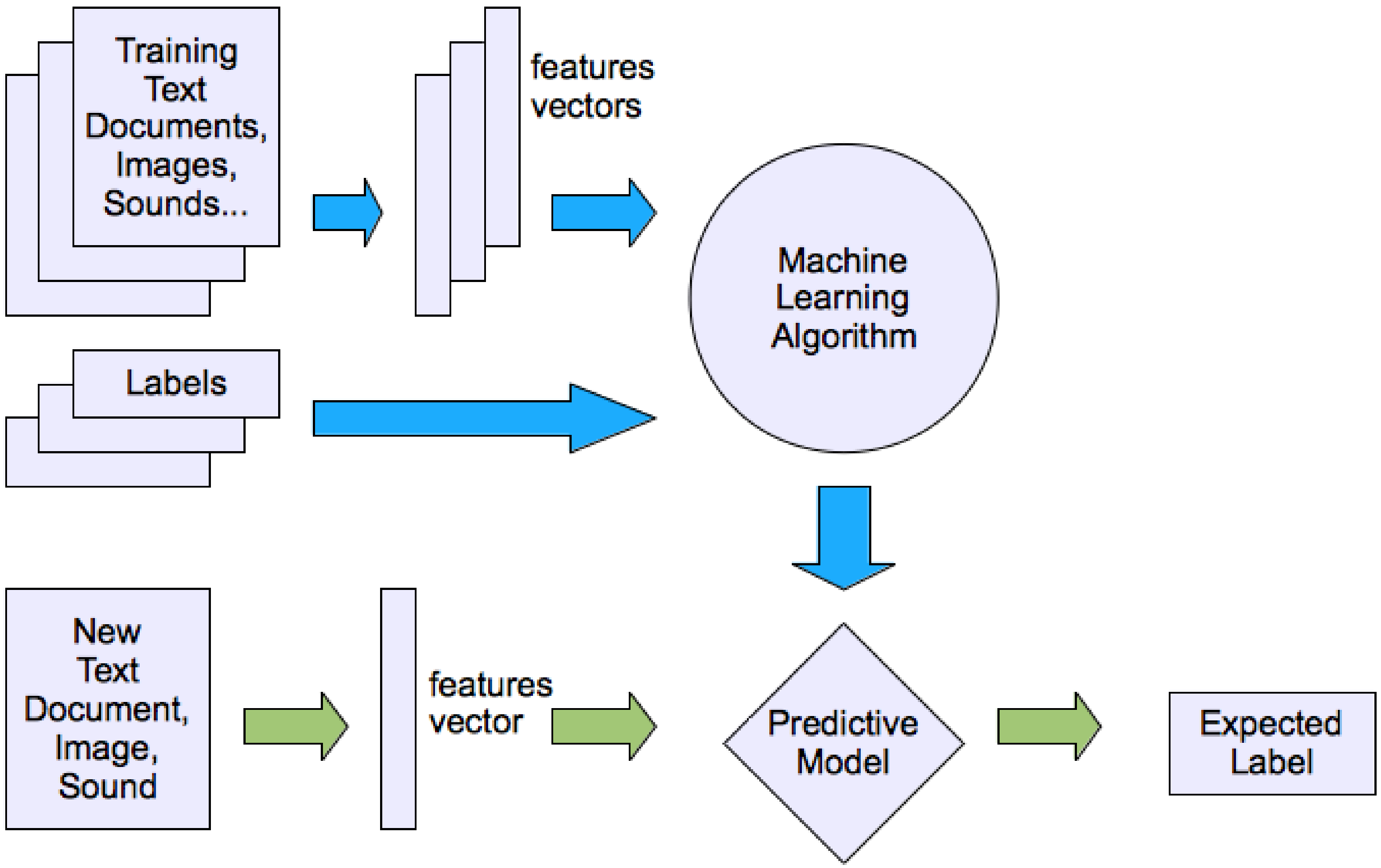




- Unsupervised:
  - Attempt to separate un-labeled data into subsets
  - Based on definition of similarity/ close-ness
  - Dimension reduction, finding association, pdf estimation

- Reinforced :
  - Uses feedback to improve on learning performance
  - Maximize the total reward
  - Computer games, robotics or automation, traffic monitoring

Supervised learning structure



Outline

- Introduction to Machine Learning
- Classification, Regression, Clustering
- Performance Analysis
- Naïve Bays, NN, SVM
- Research conveniences

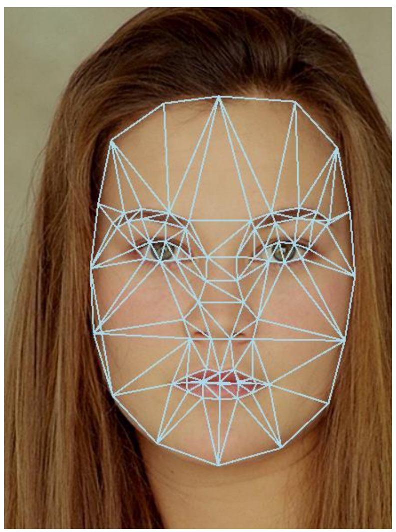
Classification

Given: training images and their categories

What are the categories of these test images?

Regression

- Learn: function to reproduce attractiveness ranking based on training inputs (feature geometry) and outputs



Vector of distances v

Attractiveness score f(v)

Regression vs Classification

Students Profile: Hard working, intelligent, sharp memory, etc.

CLASSIFICATION

PASS
Or
FAIL..???

categorical in nature, Max-likelihood / softmax

REGRESSION

How much mark…?

dependent variable is continuous (linear)

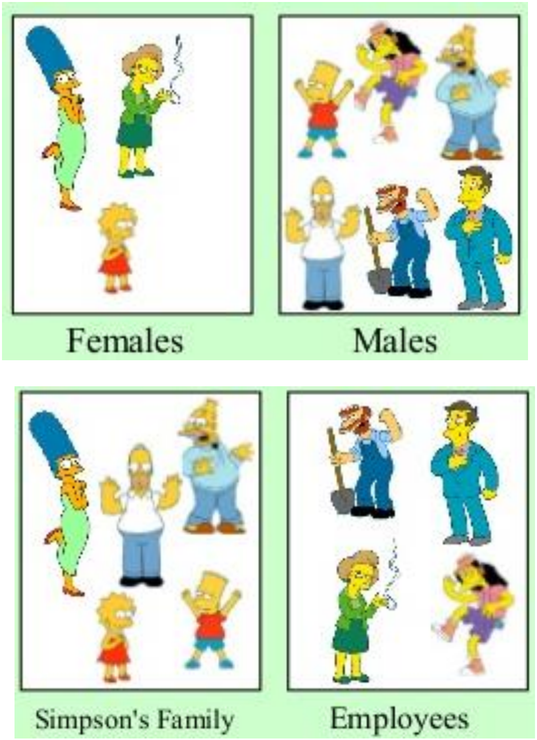
Both are supervised.

Linear classification is also called **logistic regression**.

Classification Challenges

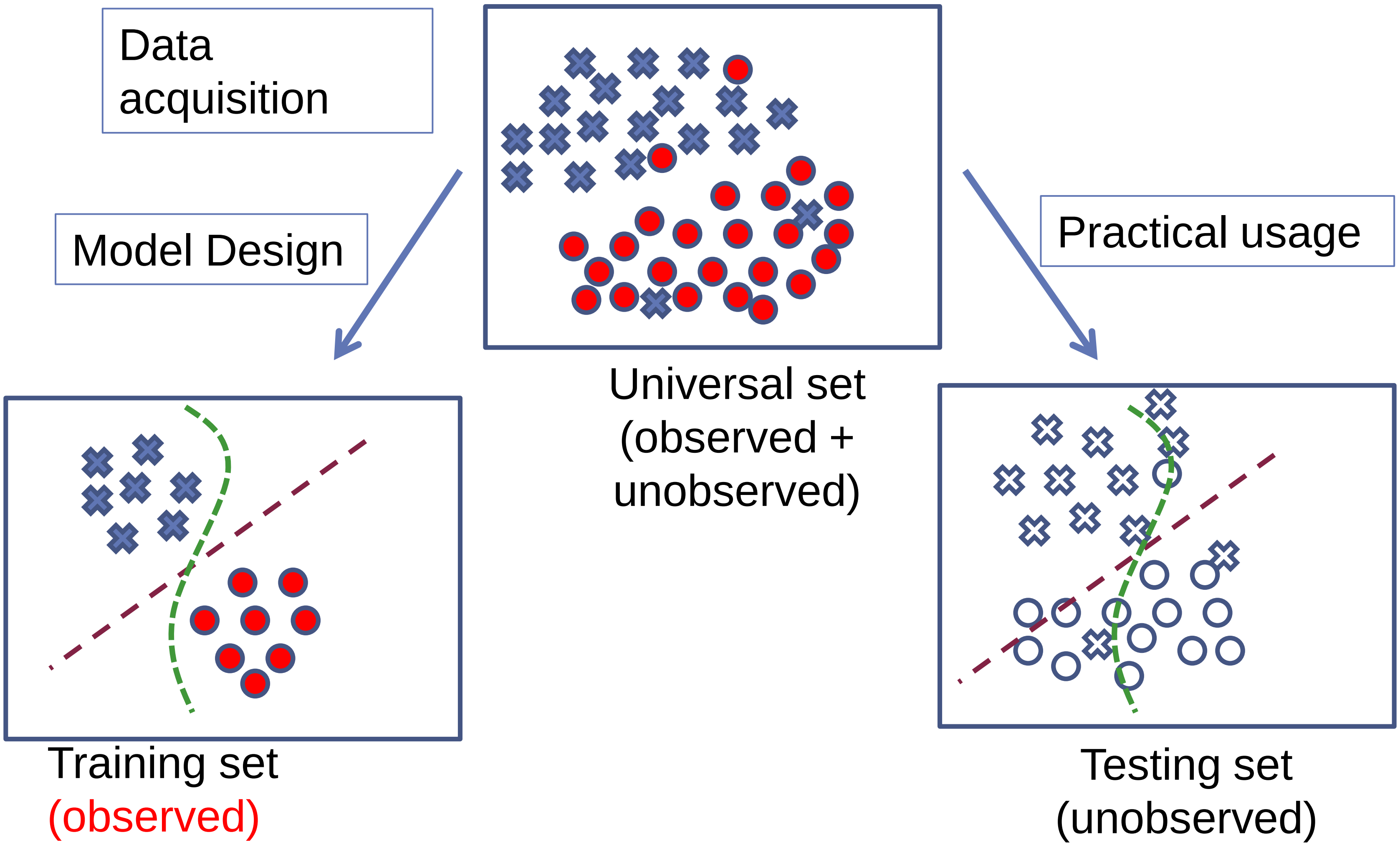
- Types of classification
  - Binary: putting the output into one of the two possible classes
    - Example: email spam filtering, medical diagnosis
  - Multi-class: More than two possible classes
    - Example handwritten digit recognition (MNIST: 0-9)
  - Multi-label classification: target classes are disjointed.
    - Example: scene category

Clustering: Grouping Objects (similarity analysis)



- ➢Subjective analysis
- ➢Unsupervised

Training and testing

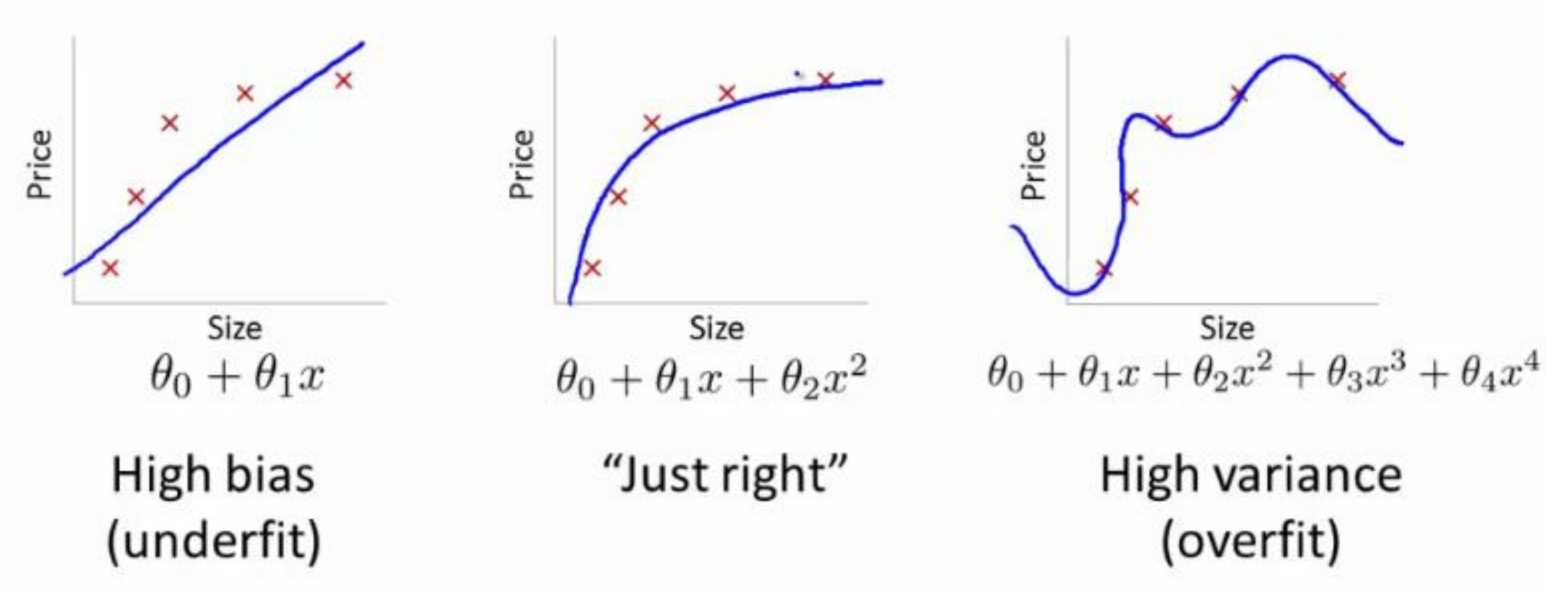


Outline

- Introduction to Machine Learning
- Classification, Regression, Clustering
- Performance Analysis
- Naïve Bays, NN, SVM
- Research conveniences

Performance

- There are several factors affecting the performance
  - Modeling- Architecture, back ground knowledge, feedback
  - Optimization Algorithm- how to extract useful information from the data
- Generalizing data
- Overfitting, underfitting and bias-variance tradeoff



Achieving good generalization

- Consideration 1: **Bias**
  - How well does your model fit the observed data (training set)?
  - It may be a good idea to accept some fitting error, because it may be due to noise or other “accidental” characteristics of one particular training set

- Consideration 2: **Variance**
  - How robust is the model to the selection of a particular training set?
  - To put it differently, if we learn models on two different training sets, how consistent will the models be?
  - How well does your model fit the test set?

Under-fitting & over-fitting

- How to recognize underfitting?
  - High training error and high test error
- How to deal with underfitting?
  - Find a more complex model, with more parameters..!!

- How to recognize overfitting?
  - Low training error, but high test error
- How to deal with overfitting?
  - Get more training data
  - Decrease the number of parameters in your model
  - Regularization: penalize certain parts of the parameter space or introduce additional constraints to deal with a potentially ill-posed problem

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

A\P	C(+ ve)	¬C(- ve)	
C (+ ve)	**True Positive: TP**	**False Negative: FN**	**P**
¬C (- ve)	**False Positive: FP**	**True Negative: TN**	**N**
	P’	**N’**	

- **Classifier Accuracy,** or recognition rate: percentage of test set tuples that are correctly classified

  Accuracy = (TP + TN)/All

- **Error rate:** *1 – accuracy*, or

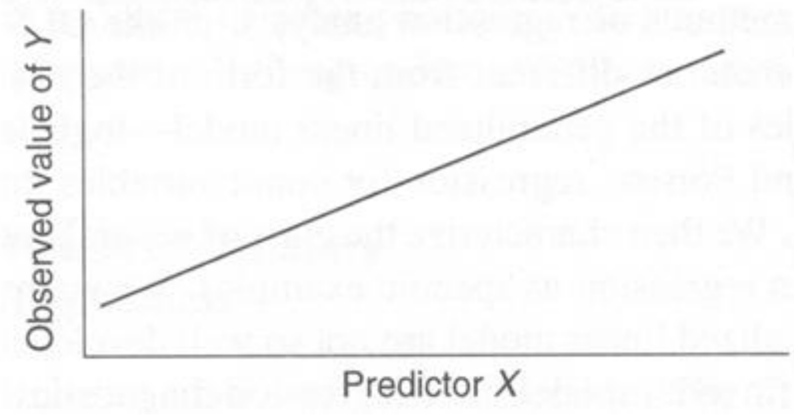
  Error rate = (FP + FN)/All

Class Imbalance Problem:

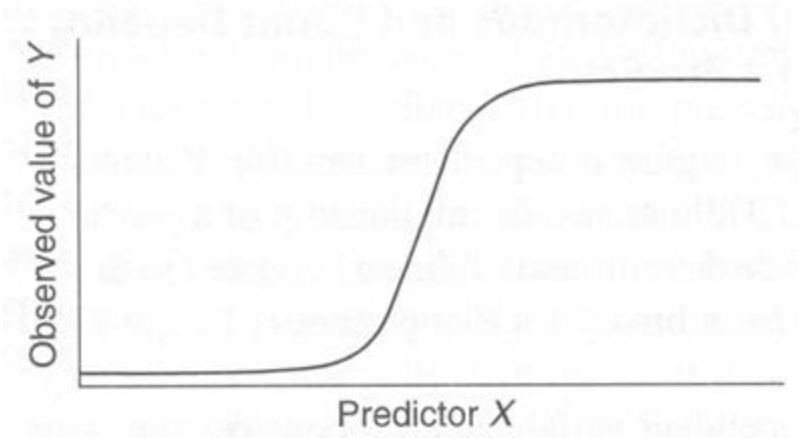
- One class may be *rare*, e.g. fraud, or HIV-positive
- Significant *majority of the negative class* and minority of the positive class
- **Sensitivity**: True Positive recognition rate
  - **Sensitivity = TP/P**
- **Specificity**: True Negative recognition rate
  - **Specificity = TN/N**

Prediction with Logistic Regression

(A) For a continuous outcome variable Y



(B) For a binary outcome variable



$$u = B_0 + B_1 X_1 + B_2 X_2 + \cdots + B_K X_K$$

$$\widehat{Y}_i = \frac{e^u}{1 + e^u}$$

- Where u is the regular linear regression equation
- Y-hat is the estimated probability of the i^{th} category
- Output **prediction is always between zero and one**, and is interpreted as a probability.

Logistic regression

- Outcome is categorical
- Binomial: win or loss, dead or alive, yes or no
- Multinomial: disease ‘A’ or disease ‘B’ or disease ’ c’. Not ordered.
- Predicts probability of a particular outcome
- Output is restricted to (0,1), interpretable as probability.

- Logistic function is defined as: $y(t) = \frac{1}{1+e^{-t}}$

Maximum likelihood Estimation

- Let y is a linear function:
- By applying logistic model:

$$y = b_0 + b_1 x_1 + .. b_k x_k = \sum_{i=0,1,..k} b_i x_i$$

$$P(y) = \frac{1}{1 + e^{-\sum_i b_i x_i}} \Rightarrow \frac{P(y)}{1 - P(y)} = e^{\sum_i b_i x_i} = \prod_i e^{b_i x_i}$$

Logistic models are multiplicative in terms of input parameters and associated interpretation.

- The value $\exp(b_j)$ tells us how the odds of the response being “true” increase (or decrease) as x_j increases by one unit, all other things being equal.

Maximum Likelihood Estimation

- The solution to a Logistic Regression problem is the set of parameters **b_i s** that maximizes the likelihood of the data.

$$P(z) = \frac{\exp(z)}{1+\exp(z)} \quad \Rightarrow P'(z) = P(z)(1-P(z))$$

- Product of the predicted probabilities of the k- individual observations.
- (*X*, *y*) is the set of observations; *X* is a K+1 dimensional input vector and y is the observation.

$$L\left(X \mid y\right) = \prod_{i=1, y=1}^{k} P\left(x_i\right) \quad \prod_{i=1, y=0}^{k} \left(1-P\left(x_i\right)\right)$$

Log Likelihood function

- Max-likelihood

$$L(X \mid y) = \prod_{i=1, y=1}^{k} P(x_i) \prod_{i=1, y=0}^{k} \left(1 - P(x_i)\right)$$

- Taking log on both sides: Log-likelihood

$$\Im(X \mid y) = \sum_{i=0, y=1}^{k} \log\left(P(x_i)\right) + \sum_{i=0, y=0}^{k} \log\left(1 - P(x_i)\right)$$

- Analogous to residual error or Sum-Square-Error

Benefits of logistic regression

- Logistic regression models are multiplicative in their inputs.

- The exponent of each coefficient tells you how a unit change in that input variable affects the response being true.
- Logistic regression preserves the marginal probabilities of the training data.

- Overly large coefficient magnitudes, overly large error bars on the coefficient estimates, and the wrong sign on a coefficient could be indications of correlated inputs.

- Coefficients that tend to infinity could be a sign that an input is perfectly correlated with a subset of your responses.

Outline

- Introduction to Machine Learning
- Classification, Regression, Clustering
- Performance Analysis
- Naïve Bays, NN, SVM
- Research conveniences

Popular techniques

- Linear regression
- Decision Tree
- K-means
- Naïve Bayes
- Support Vector Machines

- Neural Networks (1980s)
- Deep Learning (2010)

Integration and utilization of graphical processing units

Naïve Bayes (Probabilistic Model)

- Used as a Classification Algorithm
- Assumption: features are independent (It is simple hence Naïve)
- Maximum likelihood estimation

Bayes Theorem:-Probability of observing event A given B is true is equal to probability of event B given A multiplied by probability of A upon probability of B.

likelihood

posterior

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

Prior

- P(A|B) : Probability (conditional probability) of occurrence of event A given the event B is true
- P(A) and P(B): Probabilities of the occurrence of event A and B respectively
- P(B|A): Probability of the occurrence of event B given the event A is true

Example:

Given two coins, one is unfair with 90% of flips getting a head and 10% getting a tail, another one is fair. Randomly pick one coin and flip it. If we get a head, what is the probability that this coin is the unfair one.

$$P(U \mid H) = \frac{P(H \mid U)P(U)}{P(H)} = \frac{P(H \mid U)P(U)}{P(H \mid U)P(U) + P(H \mid F)P(F)}$$

$$= \frac{0.9 \times 0.5}{0.9 \times 0.5 + 0.5 \times 0.5} = 0.64$$

Given a data sample with n features $x = (x1, x2 \ldots xn)$

Probability that the sample belongs to each of K possible classes..?=

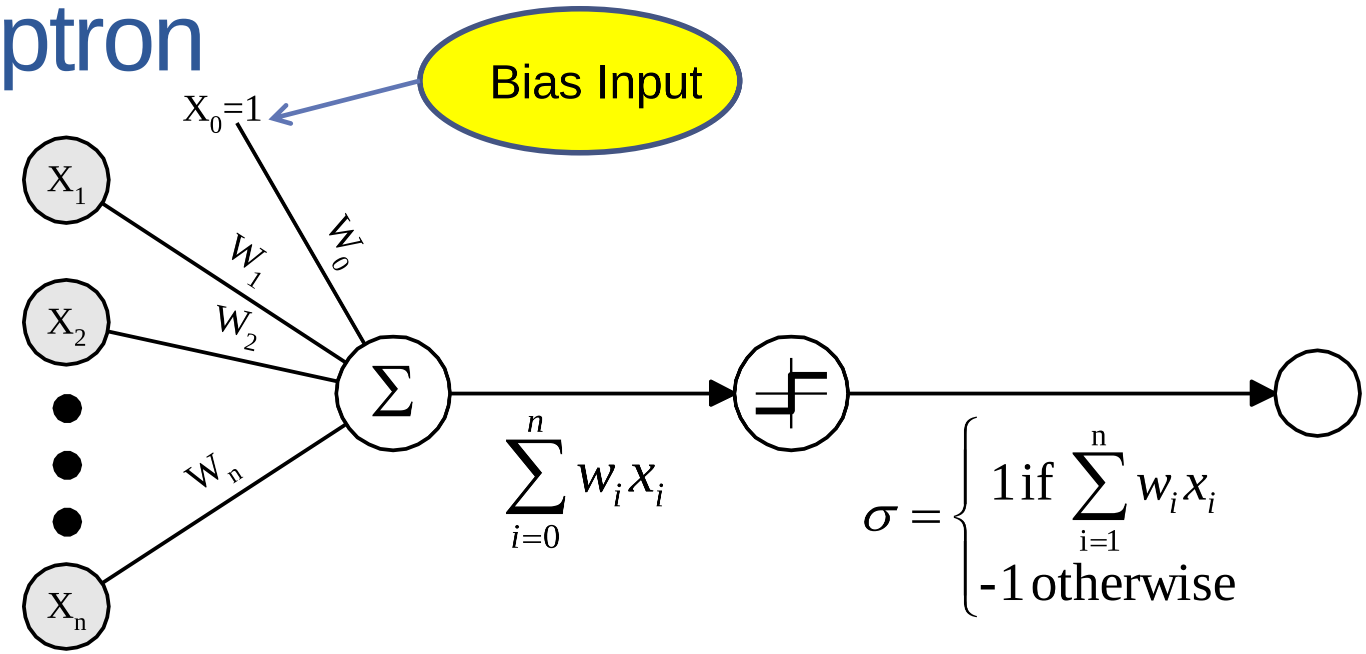
$$P(y_k \mid x) = \frac{P(x \mid y_k)P(y_k)}{P(x)}$$

Example

- Suppose that a test for using a particular drug is 99% sensitive and 99% specific. That is, the test will produce 99% true positive results for drug users and 99% true negative results for non-drug users. Suppose that 0.5% of people are users of the drug. What is the probability that a randomly selected individual with a positive test is a user?

$$
\begin{aligned}
P(\text{User} \mid +) &= \frac{P(+ \mid \text{User})P(\text{User})}{P(+)} \\
&= \frac{P(+ \mid \text{User})P(\text{User})}{P(+ \mid \text{User})P(\text{User}) + P(+ \mid \text{Non-user})P(\text{Non-user})} \\
&= \frac{0.99 \times 0.005}{0.99 \times 0.005 + 0.01 \times 0.995} \\
&\approx 33.2\%
\end{aligned}
$$

Perceptron

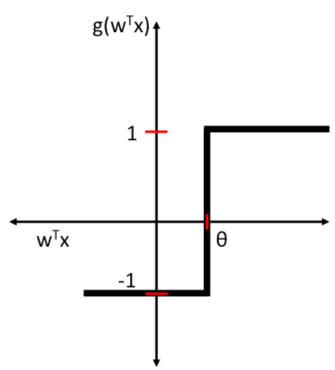


$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \ldots + w_n x_n > 0 \\ -1 & \text{otherwise} \end{cases}$$

Sometimes we will use simpler vector notation:

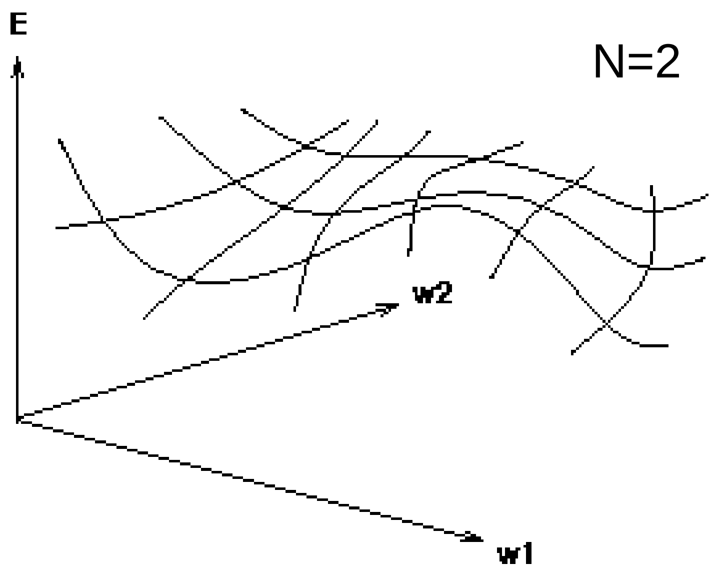
$$o(\vec{x}) = \begin{cases} 1 \text{ if } \vec{w} \cdot \vec{x} > 0 \\ -1 \text{ otherwise} \end{cases}$$

$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i w_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^{n} x_i w_i < \theta \end{cases}$$



Error Surface

- Think of the N weights as a point in an N-dimensional space
- Add a dimension for the observed error
- Try to minimize your position on the "error surface"



x_1	x_2	t	**k**
-1	-1	-1	1
-1	1	-1	2
1	-1	-1	3
1	1	1	4

$$e_k = t^k - o^k = t^k - \left(w_1 x_1^k + w_2 x_2^k \right)$$

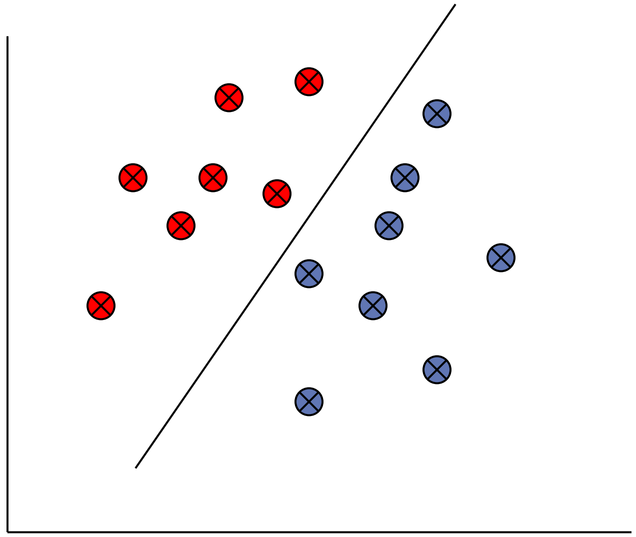
$$E_k(w_1, w_2) = \frac{1}{2} \sum_k e_k^2$$

Perceptron learning:

$$\Delta w_i = \eta(t - o)\, x_i$$

- Where w_i is the weight from input i to perceptron node, η is the learning rate, t_j is the target for the current instance, o is the current output, and x_i is i^{th} input
- Least perturbation principle
  - Only change weights if there is an error
  - small η rather than changing weights abruptly but sufficient to make current pattern correct
  - Scale by x_i
- Each iteration through the training set is an ***epoch***
- Continue training until total training set error ceases to improve
- Treat threshold like any other weight. Call it a *bias* since it biases the output up or down.

Linear Separability



$$W_1X_1 + W_2X_2 > \theta \quad (Z=1)$$

$$W_1X_1 + W_2X_2 < \theta \quad (Z=0)$$

So, what is decision boundary?

$$W_1X_1 + W_2X_2 = \theta$$

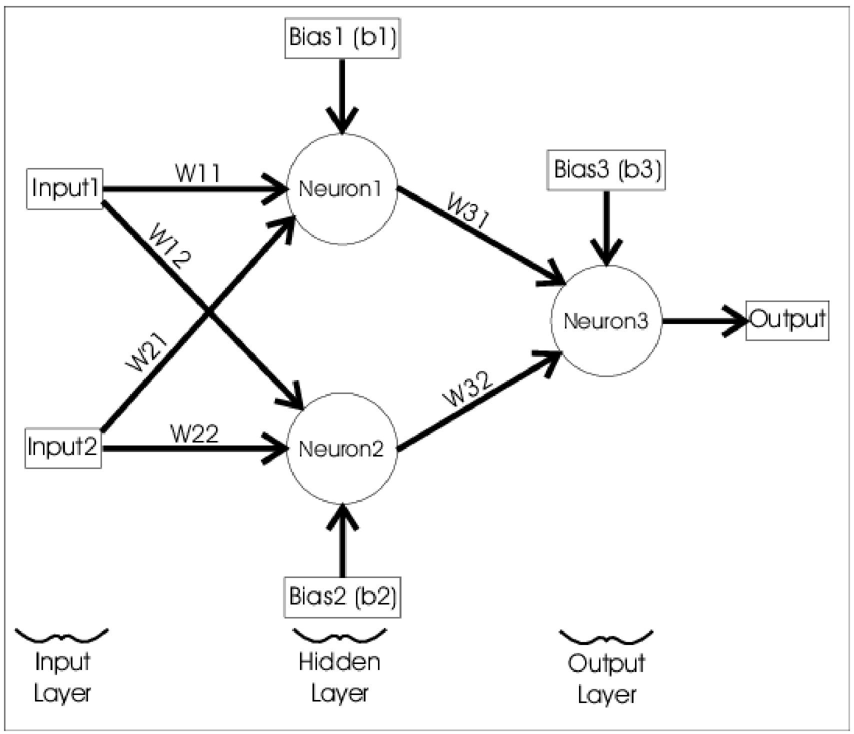
$$X_2 + W_1X_1/W_2 = \theta/W_2$$

$$X_2 = (-W_1/W_2)X_1 + \theta/W_2$$

$$Y = MX + B$$

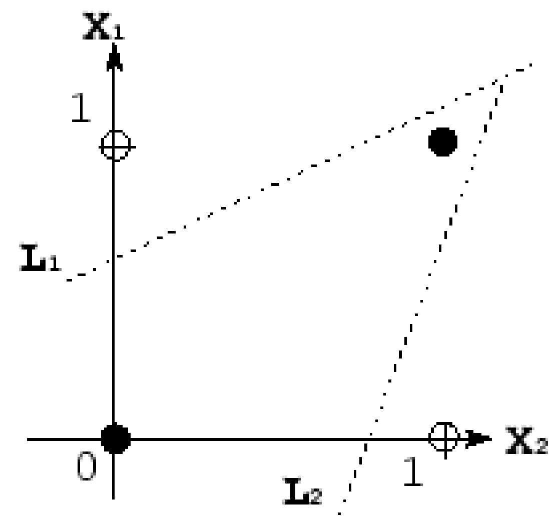
Beyond Linear Separation

- More perceptron in parallel more layers to combine the outputs.



X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$$Y = X_1 \oplus X_2$$



Gradient

To understand, consider simple *linear unit*, where

$$o = w_0 + w_1 x_1 + \ldots + w_n x_n$$

Idea : learn w_i's that minimize the squared error

$$E[\vec{w}] \equiv \tfrac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

Where D is the set of training examples

- To make error decrease the fastest

Gradient $\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n} \right]$

Training rule: $\Delta w_i = -\eta \, \nabla E[\vec{w}]$

i.e., $\Delta w_i = -\eta \, \frac{\partial E}{\partial w_i}$

Derivatives of error wrt weights

Compute deltas

Gradient Descent

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_d (t_d - o_d)^2$$

$$= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2$$

$$= \frac{1}{2} \sum_d 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d)$$

$$= \sum_d (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x}_d)$$

$$\frac{\partial E}{\partial w_i} = \sum_d (t_d - o_d)(-x_{i,d})$$

Gradient Descent

Each training examples is a pair of the form $<\vec{x}, t>$, *where* $\vec{x}$ *is the vecto*

input values and t is the target output value.

η *is the learning rate (e.g., .05).* $0 < \eta < 1$

- Initialize each w_i to some small random value
- Until the termination condition is met, do

  - Initialize each Δw_i to zero.

  - For each $<\vec{x}, t>$ in *training_examples*, do

    * Input the instance $\vec{x}$ and compute output o

    * For each linear unit weight w_i, do

$$\Delta w_i \leftarrow \Delta w_i + \eta\ (t - o) x_i$$

  - For each linear unit weight w, do

$$w_i \leftarrow w_i + \Delta w_i$$

Incremental (Stochastic) Gradient Descent

Batch mode Gradient Descent:
Do until satisfied:

$$1.\ \text{Compute the gradient } \nabla E_D[\vec{w}]$$

$$2.\ \vec{w} \leftarrow \vec{w} - \eta \nabla E_D[\vec{w}]$$

$$E_D[\vec{w}] \equiv \tfrac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

Incremental mode Gradient Descent:
Do until satisfied:
- For each training example *d* in *D*

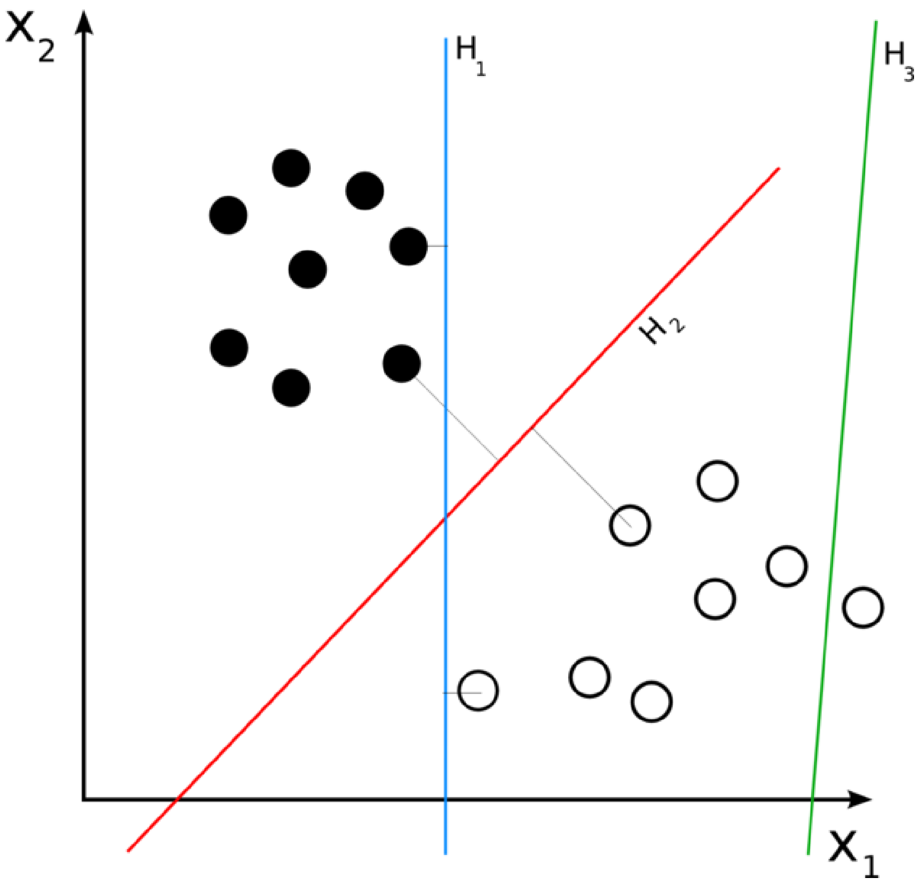
$$1.\ \text{Compute the gradient } \nabla E_d[\vec{w}]$$

$$2.\ \vec{w} \leftarrow \vec{w} - \eta \nabla E_d[\vec{w}]$$

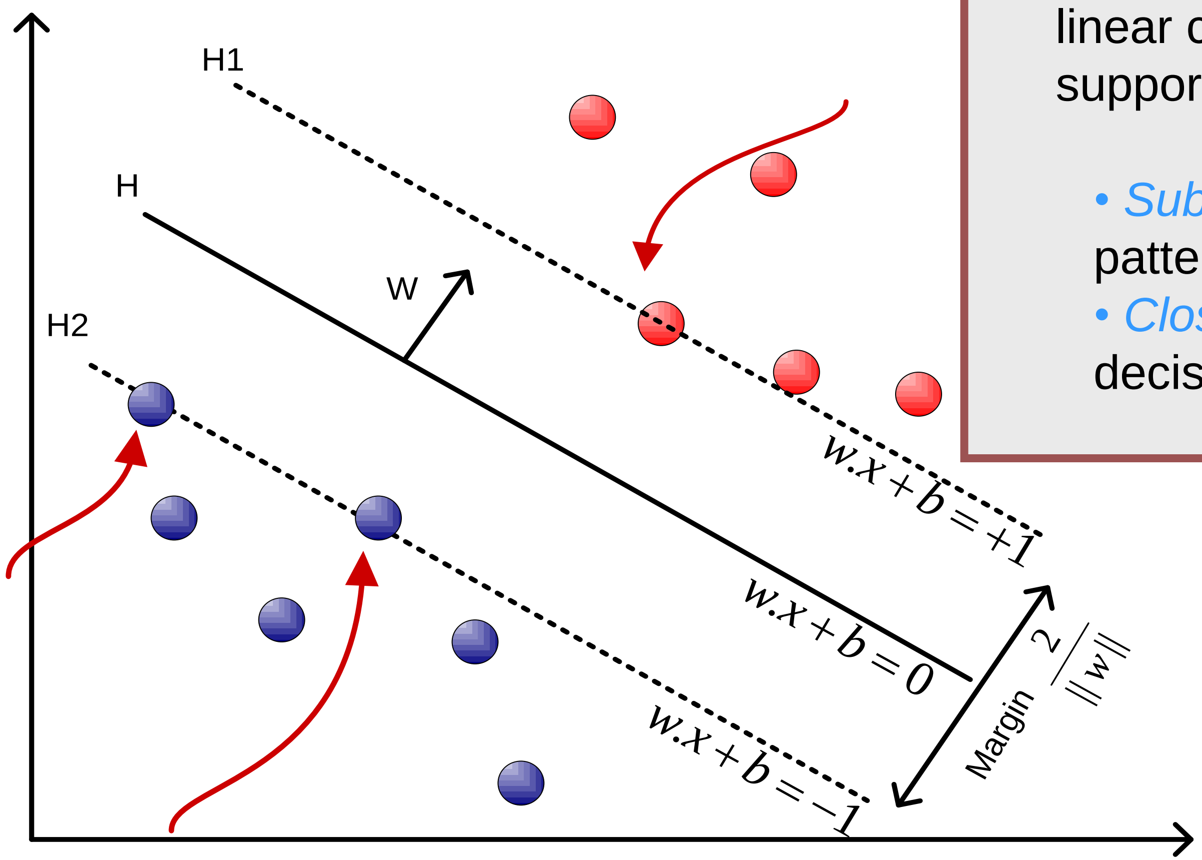
$$E_d[\vec{w}] \equiv \tfrac{1}{2} (t_d - o_d)^2$$

Incremental Gradient Descent can approximate *Batch Gradient Descent* arbitrarily closely if η made small enough

Support Vector Machine – (SVM)



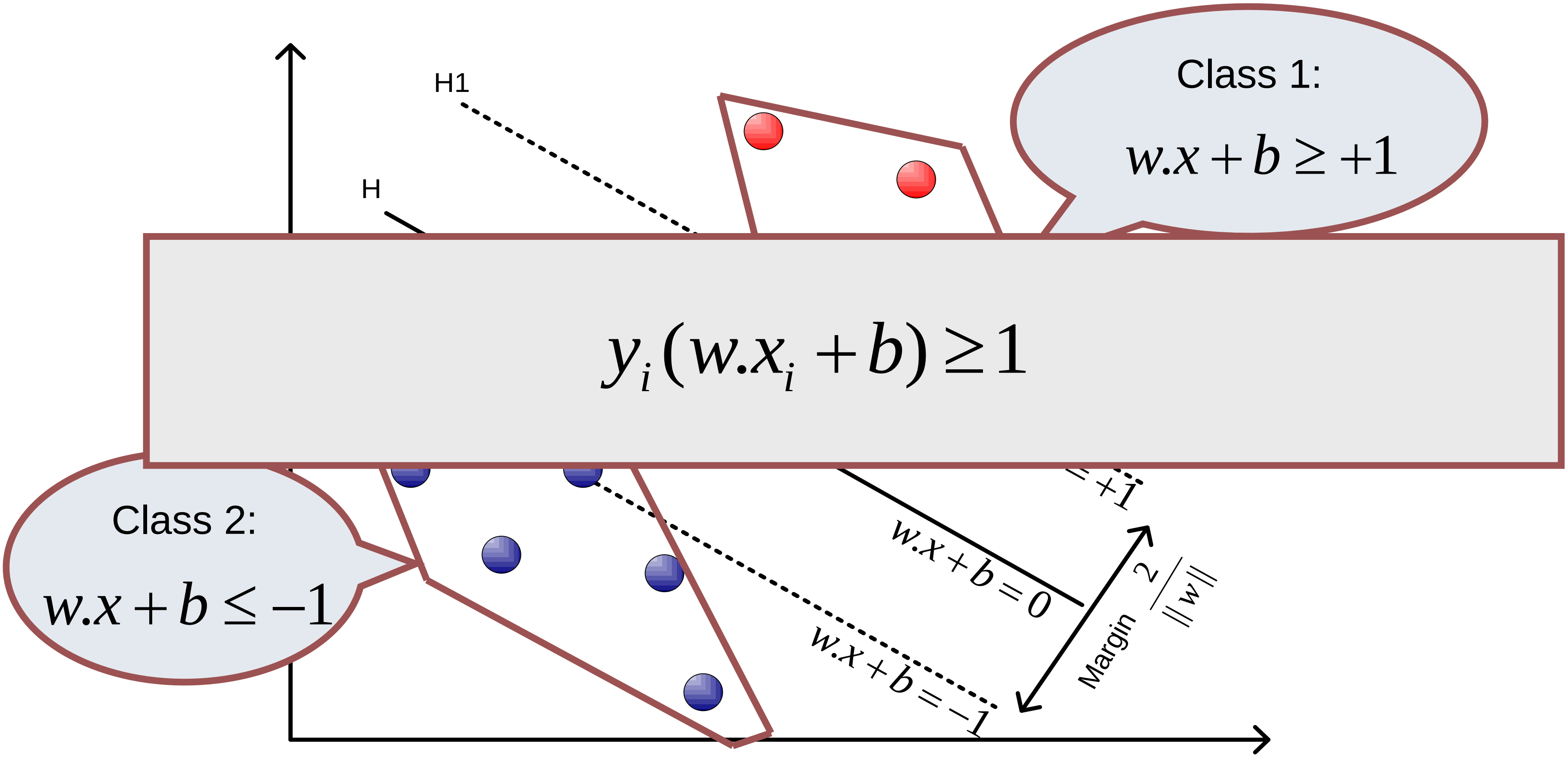
Support Vectors



Solution: expressed as a linear combination of support vectors:

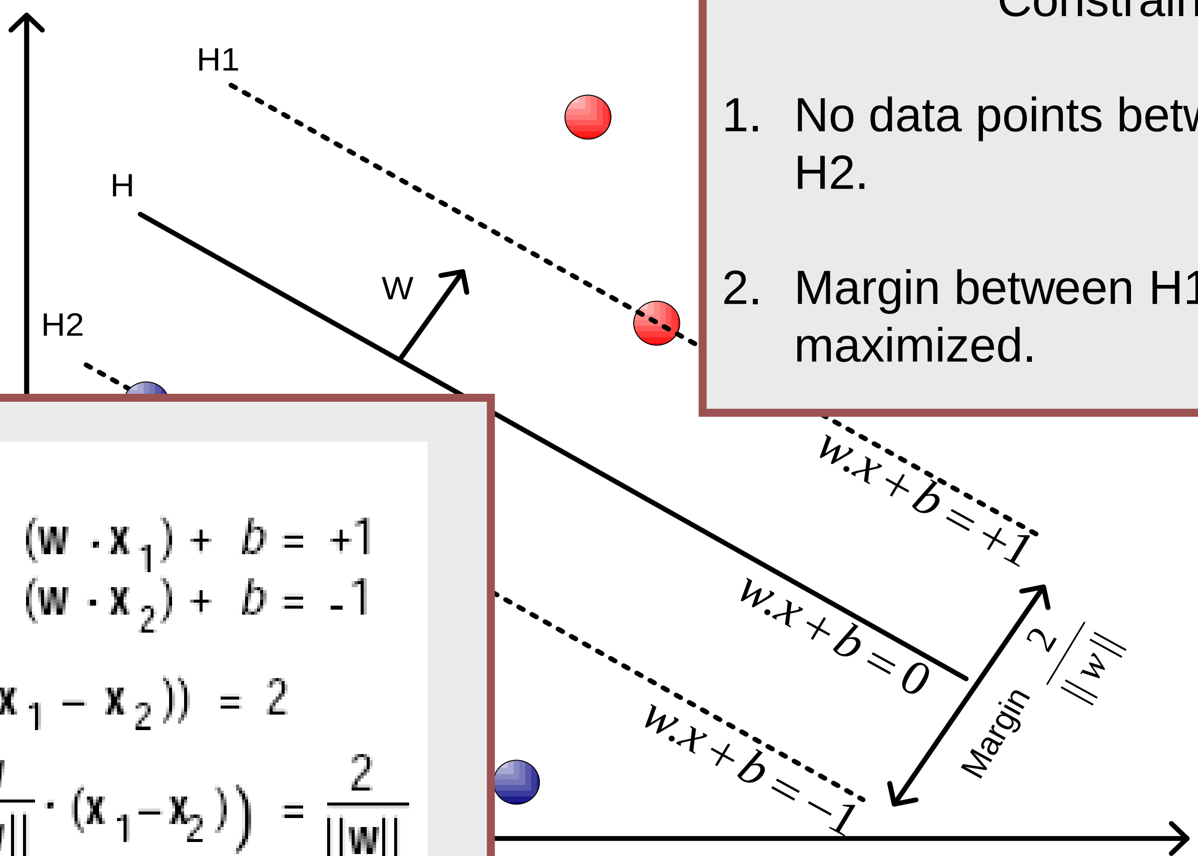
- *Subset* of training patterns
- *Close* to the decision boundary

Theory of SVM



$$y_i(w.x_i + b) \geq 1$$

Theory of SVM



Constraints

1. No data points between H1 and H2.
2. Margin between H1 and H2 is maximized.

Note:

$$(\mathbf{w} \cdot \mathbf{x}_1) + b = +1$$
$$(\mathbf{w} \cdot \mathbf{x}_2) + b = -1$$

$$\Rightarrow \quad (\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2)) = 2$$

$$\Rightarrow \quad \left(\frac{\mathbf{w}}{||\mathbf{w}||} \cdot (\mathbf{x}_1 - \mathbf{x}_2) \right) = \frac{2}{||\mathbf{w}||}$$

Quadratic Programming

- to maximize the margin $\frac{1}{2}\|w\|^2$; we need to minimize: $\frac{2}{\|w\|}$
- Quadratic Programming solved by introducing Lagrange multipliers

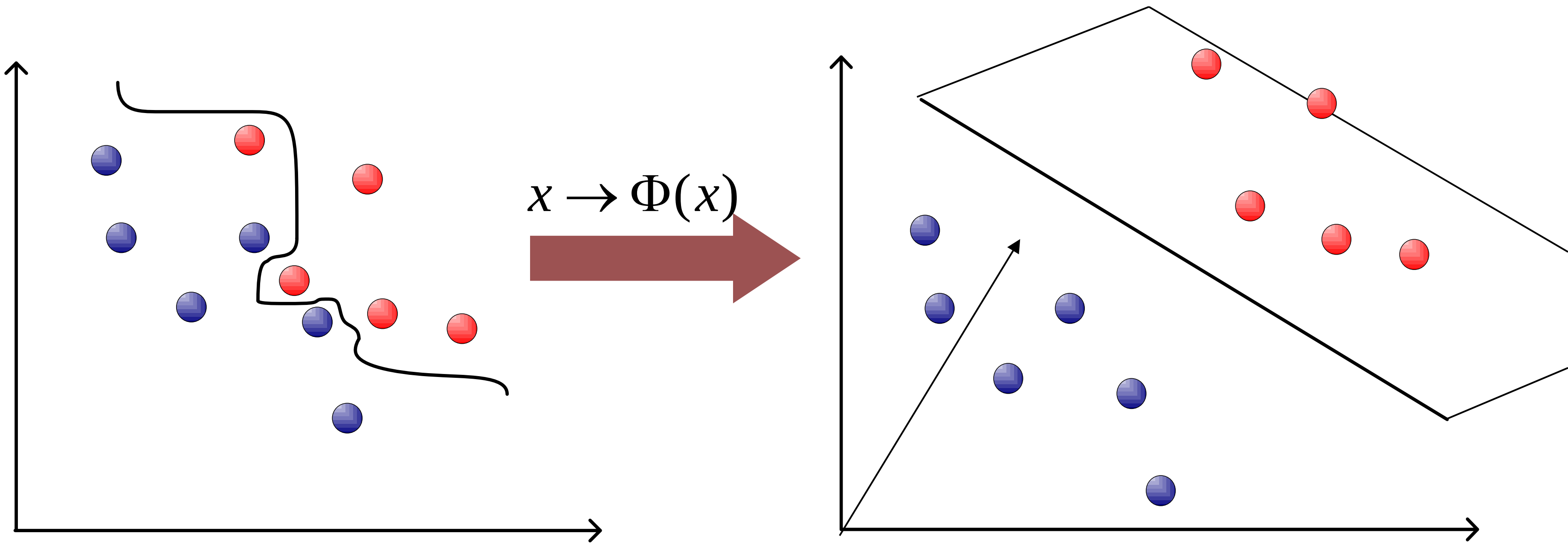
$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i y_i (x_i . w + b) + \sum_{i=1}^{N} \alpha_i$$

Non-linear SVM

1. SVM mapped the data sets of input space into a higher dimensional feature space

2. Linear and the large-margin learning algorithm is then applied.

Non-linear SVM



Input (data) space
(Non- linear)

Feature space
(Linear)

SVM vs. ANN

SVM	ANN
▪ Global minimum.	▪ Local minimum
▪ SVM does not overfit data (Structural Risk Minimization).	▪ANN is known to overfit data unless cross-validation is applied. Doesn’t have mathematical foundation.
▪ Multi-class implementation needs to be performed. Kernel functions are used to map non-linear pattern onto high dimensional space for feasibility of linear classification	▪ Naturally handles multi-class classification (softmax).
▪ Hard to learn- learned in batch modes using QP techniques.	▪ Can easily be learnt in increamental fashion.

Outline

- Introduction to Machine Learning
- Classification, Regression, Clustering
- Performance Analysis
- Naïve Bays, NN, SVM
- Research conveniences

Resources: Datasets

- UCI Repository: http://www.ics.uci.edu/~mlearn/MLRepository.html
- UCI KDD Archive: http://kdd.ics.uci.edu/summary.data.application.html
- Statlib: http://lib.stat.cmu.edu/
- Delve: http://www.cs.utoronto.ca/~delve/

Popular Open Source Package

- **Scikit-learn –**
  - a widely used open source library in python with packages like *sklearn.linear_model*, *sklearn.naive_bayes*, *sklearn.neural_network*, *sklearn.svm*, *sklearn.tree, etc.*
  - *It can load impotant classic datasets easily*

- **Theona-** http://deeplearning.net/software/theona/
- **TensorFlow-** http://www.tensorflow.org
- **Keras-** http://keras.io
  - Python library that can run on top of either Theona or TensorFlow, on a CPU or GPU