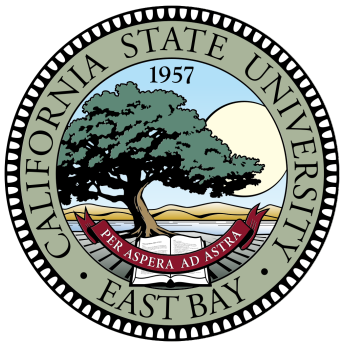


CS 6820 – Machine Learning

Lecture 2

Instructor: Eric S. Gayles, PhD.

Jan 7, 2018

Course Description

Lecture 1	Introduction, What is Machine Learning, Technical Enablers, Target Applications
Lecture 2	Linear Regression, Linear Classifiers, Logistic Regression, Multiclass Logistic Regression
Lecture 3	Decision Trees, Induction, Knowledge Representation, Machine Learning Tools
Lecture 4	Gradient Descent, Bias, Loss Functions, Perceptrons, Inductive Logic Programming
Lecture 5	KNN Classification, Multi-class Classification
Lecture 6	Bayesian Learning and Inference, Probabilistic Classification
	Mid-term
Lecture 7	Genetic Algorithms, Reinforcement Learning, and Adaptive Dynamic Programming
Lecture 8	Neural networks, Back propagation, Deep Learning
Lecture 9	Kernel Methods, SVMs, Overfitting, Underfitting
Lecture 10	Unsupervised Learning, Clustering, Principal Component Analysis (PCA)
Lecture 11	Markov decision processes, Gaussian Processes
	Final

Information about the Instructor

- Instructor:
  - Eric S. Gayles, PhD.
- Education:
  - Ph.D. Computer Science and Engineering - Pennsylvania State University
- Industrial Experience:
  - Google, Intel, Teradata, Institute for Defense Analyses, Startups & VC
- Email: eric.gayles@csueastbay.edu
- Office: TBD
  - Office Hours: After class on Mondays, 8:00-9:00pm

What is Machine Learning ?

"**Machine learning** is a field of computer science that gives computers the ability to learn without being explicitly programmed." - Arthur Samuel (1959)

What is Machine Learning ?

"A computer program is said to learn from experience (E) with some class of tasks (T) and a performance measure (P) if its performance at tasks in T as measured by P improves with E" - Tom Mitchell (1998)

What is Machine Learning?

- Optimize a performance criterion using example data or past experience.
- Role of Statistics: Inference from a sample
- Role of Computer science: Efficient algorithms to
  - Solve the optimization problem
  - Representing and evaluating the model for inference

Logistics

- Grading
  - 20% - Mid Term
  - 30% - Final
  - 5% - Project 1
  - 10% - Project 2
  - 10% - Project 3
  - 25% - Project 4 - Group

Source Material

- Required:
  - E. Alpaydin - Introduction to Machine Learning, Third Edition
  - H. Daumé - A Course in Machine Learning (online)

- Good optional references:
  - C. Bishop, Pattern Recognition and Machine Learning
  - K. Murphy, Machine Learning: a Probabilistic Perspective
  - P. Klein - Coding the Matrix: Linear Algebra through Applications to Computer Science

ML Tools

- Matlab
- GNU Octave
- SciPy, Numpy, SciKits
  - Recommend installing Jupyter from www.anaconda.com (evolved from iPython)
  - Good Read–Eval–Print Loop (REPL) environment
  - https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks
- Basic Linear Algebra Subprograms (BLAS)
- LAPACK — Linear Algebra PACKage
- Shogun-toolbox
- MLlib – Apache
- Deeplearn.js
- ConvnetJS
- MLPack
- TensorFlow

Some Intro Terminology

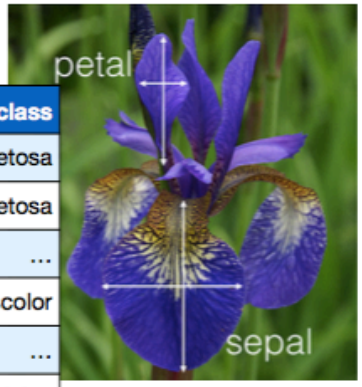
- Features - Distinct traits that can be used to describe each item within a set of samples in a quantitative manner.
- Feature vector - An n-dimensional vector of numerical quantized values that represent an object (sample).
- Feature extraction - Preparation of a feature vector that transforms the data from a high-dimensional space to a space of fewer dimensions.
- Training/Evolution set - Set of data input into our model to discover potentially predictive relationships.
- Training - Process of evaluation and optimizing the weights of a network.

Some Intro Terminology

IRIS

https://archive.ics.uci.edu/ml/datasets/Iris

Instances (samples, observations)



	sepal_length	sepal_width	petal_length	petal_width	class
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
…	…	…	…	…	…
50	6.4	3.2	4.5	1.5	veriscolor
…	…	…	…	…	…
150	5.9	3.0	5.1	1.8	virginica

Features (attributes, dimensions)

Classes (targets)

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ x_{31} & x_{32} & \cdots & x_{3D} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix}$$

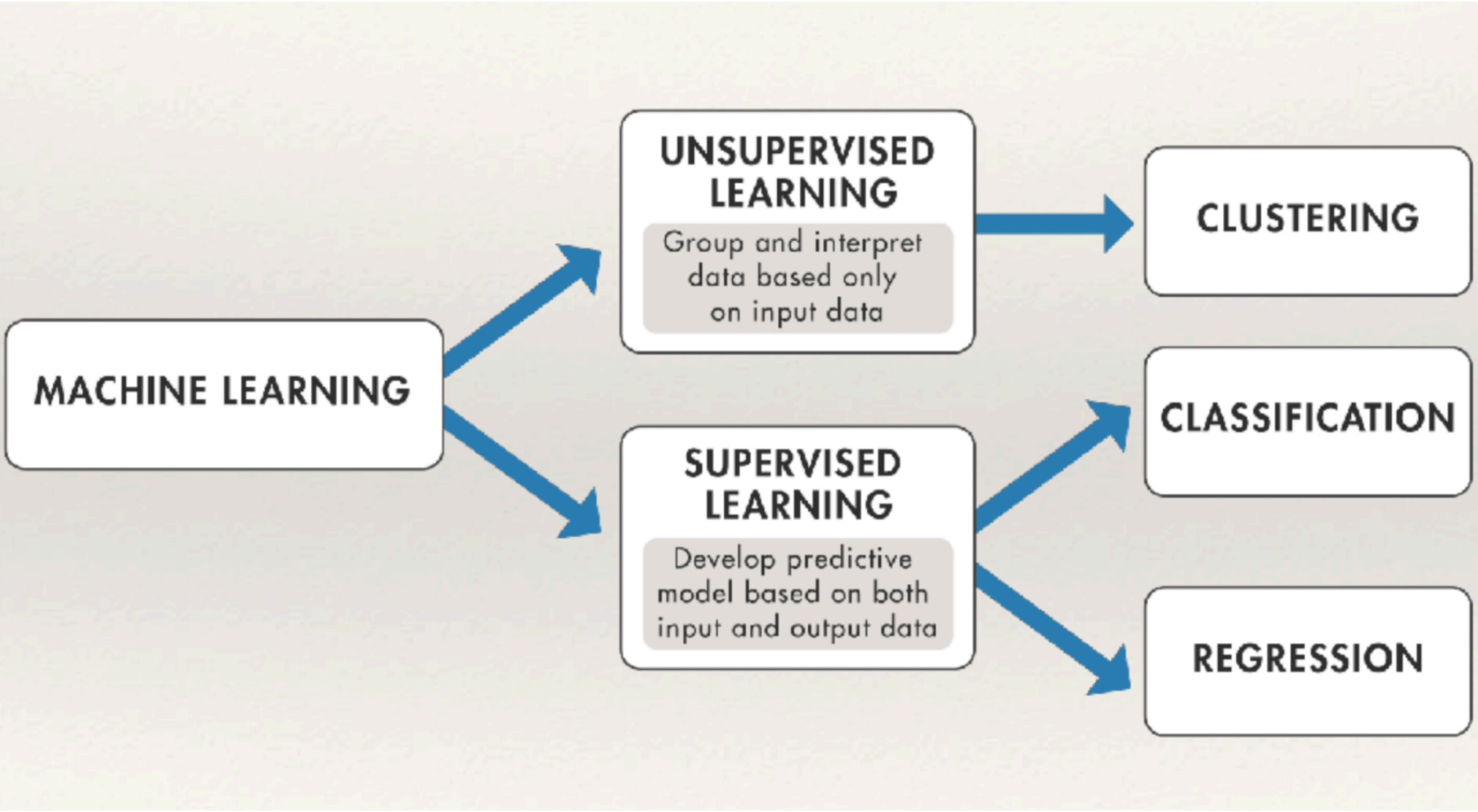
$$\mathbf{y} = [y_1, y_2, y_3, \cdots y_N]$$

*Raschka

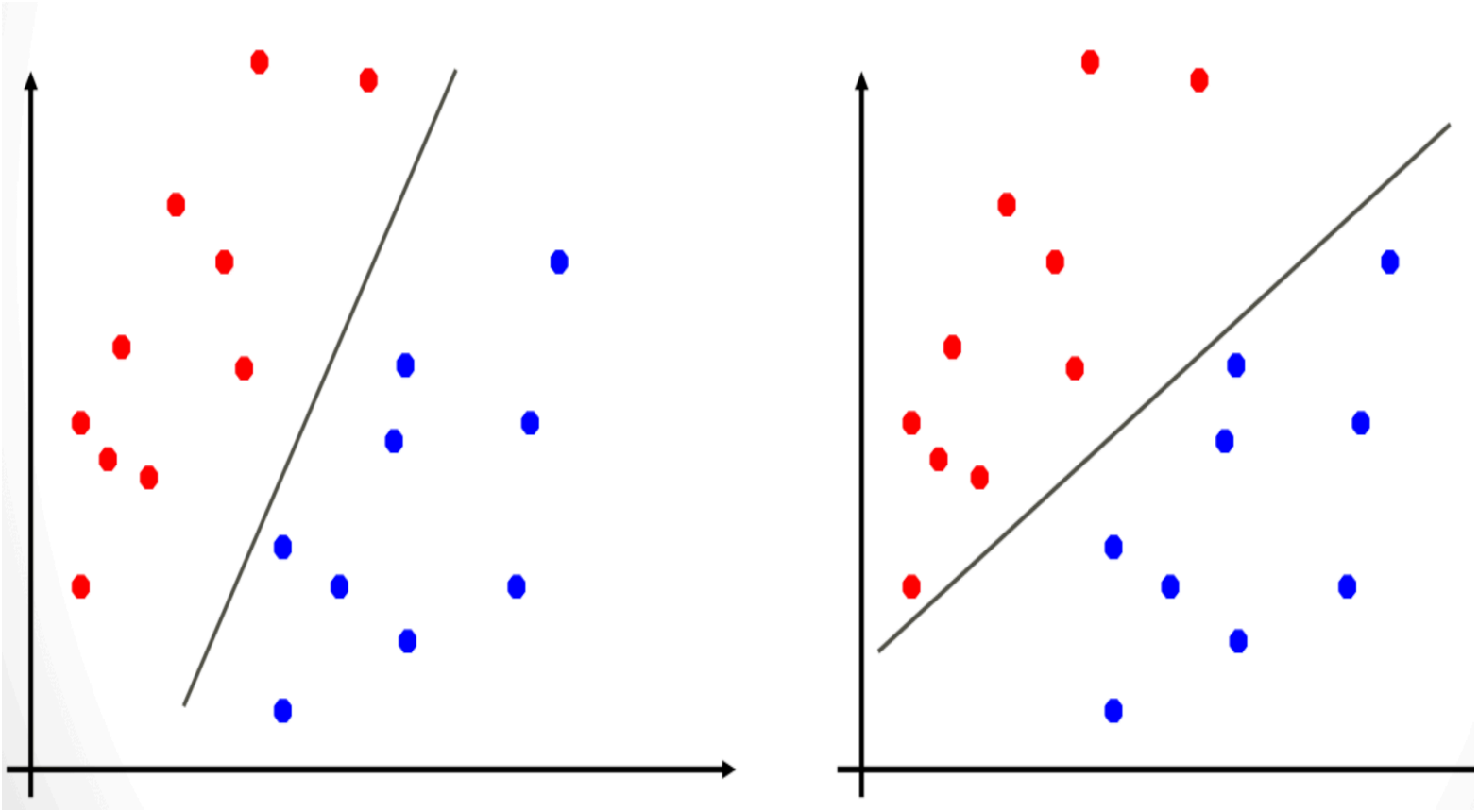
Some Intro Terminology

- Classification – prediction of a sample's class based on a priori observations (training data)
- Clustering - the assignment of a set of observations (samples) into subsets such that observations in the same cluster have meaningfully similar features that are statistically distinguishably from members of other subsets.
- Regression analysis – the statistical process to estimate relationships among variables.
- Regression - predicting the output value using training data.

Some Intro Terminology

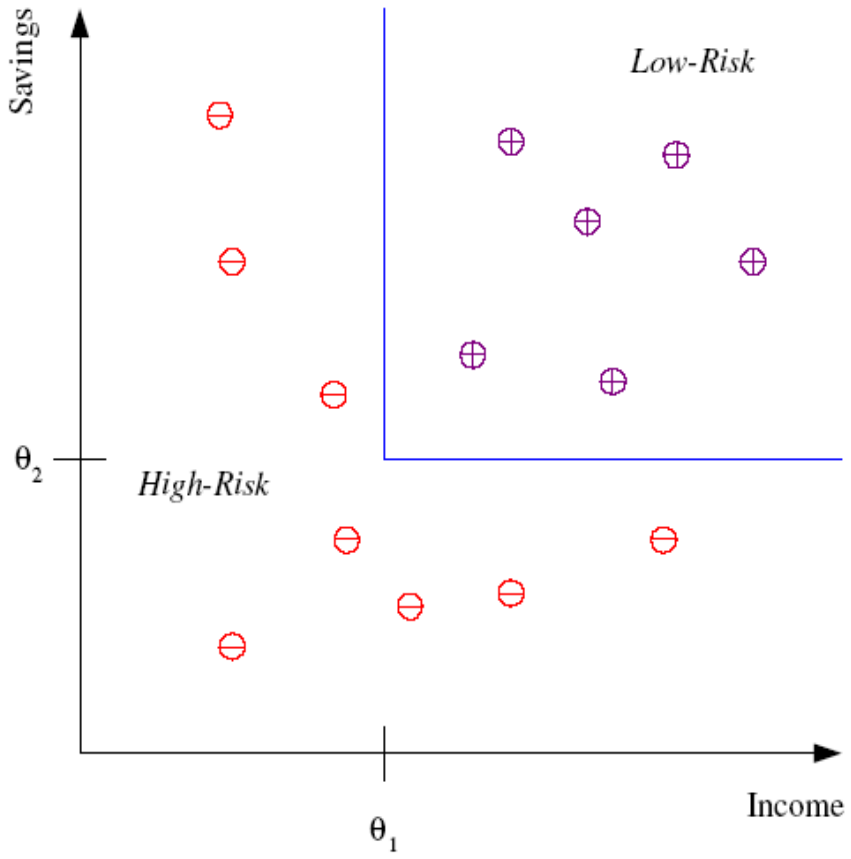


Linear Classifiers



Clas

- Example: Credit scoring
- Differentiating between low-risk and high-risk customers from their *income* and *savings*



Discriminant: IF *income* > θ_1 AND *savings* > θ_2
THEN low-risk ELSE high-risk

Classification: Applications

- Aka Pattern recognition
- Face recognition: Pose, lighting, occlusion (glasses, beard), make-up, hair style
- Character recognition: Different handwriting styles.
- Speech recognition: Temporal dependency.
- Medical diagnosis: From symptoms to illnesses
- Biometrics: Recognition/authentication using physical and/or behavioral characteristics: Face, iris, signature, etc
- Outlier/novelty detection:

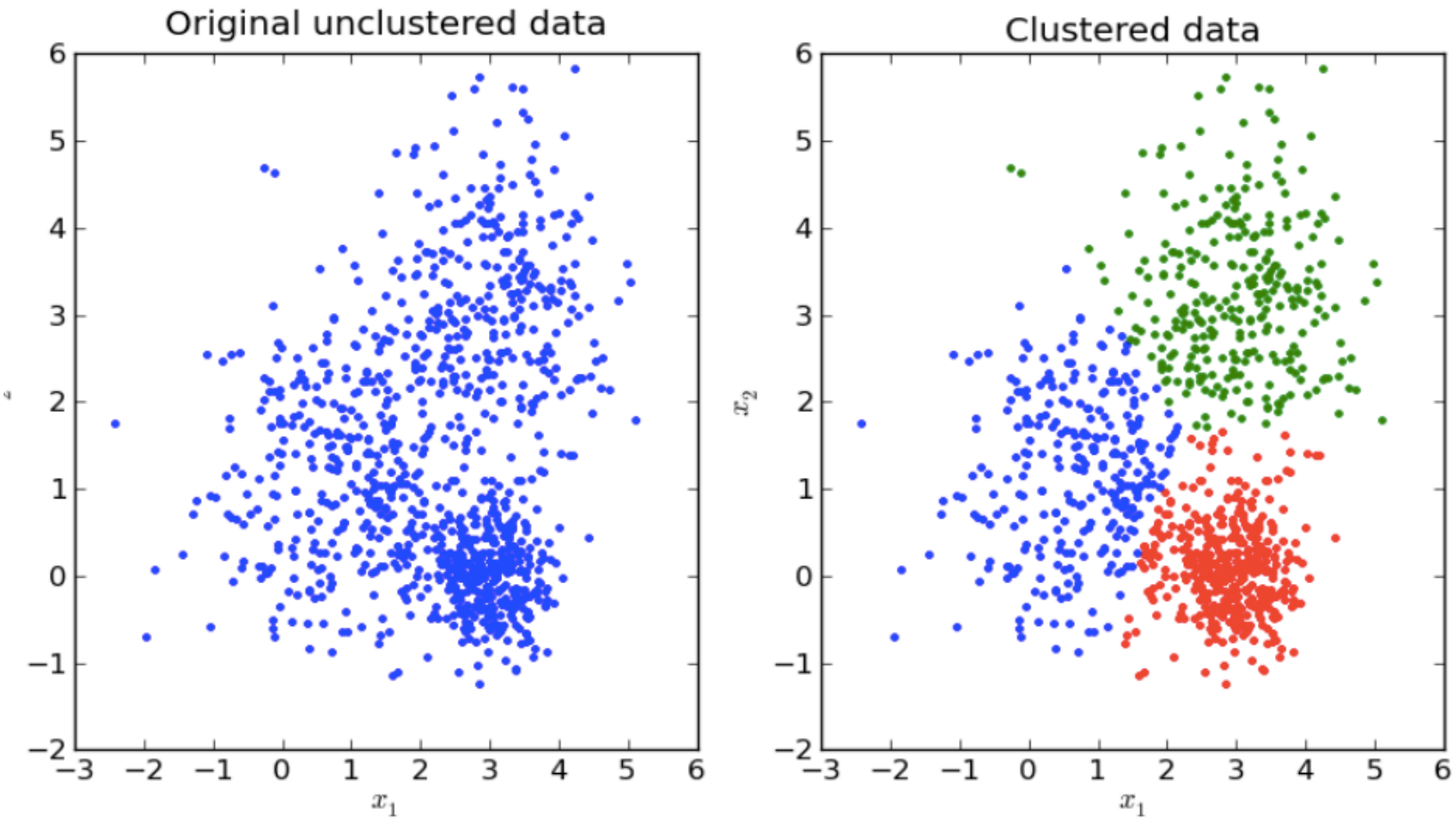
Face Recognition

Training examples of a person

Test images

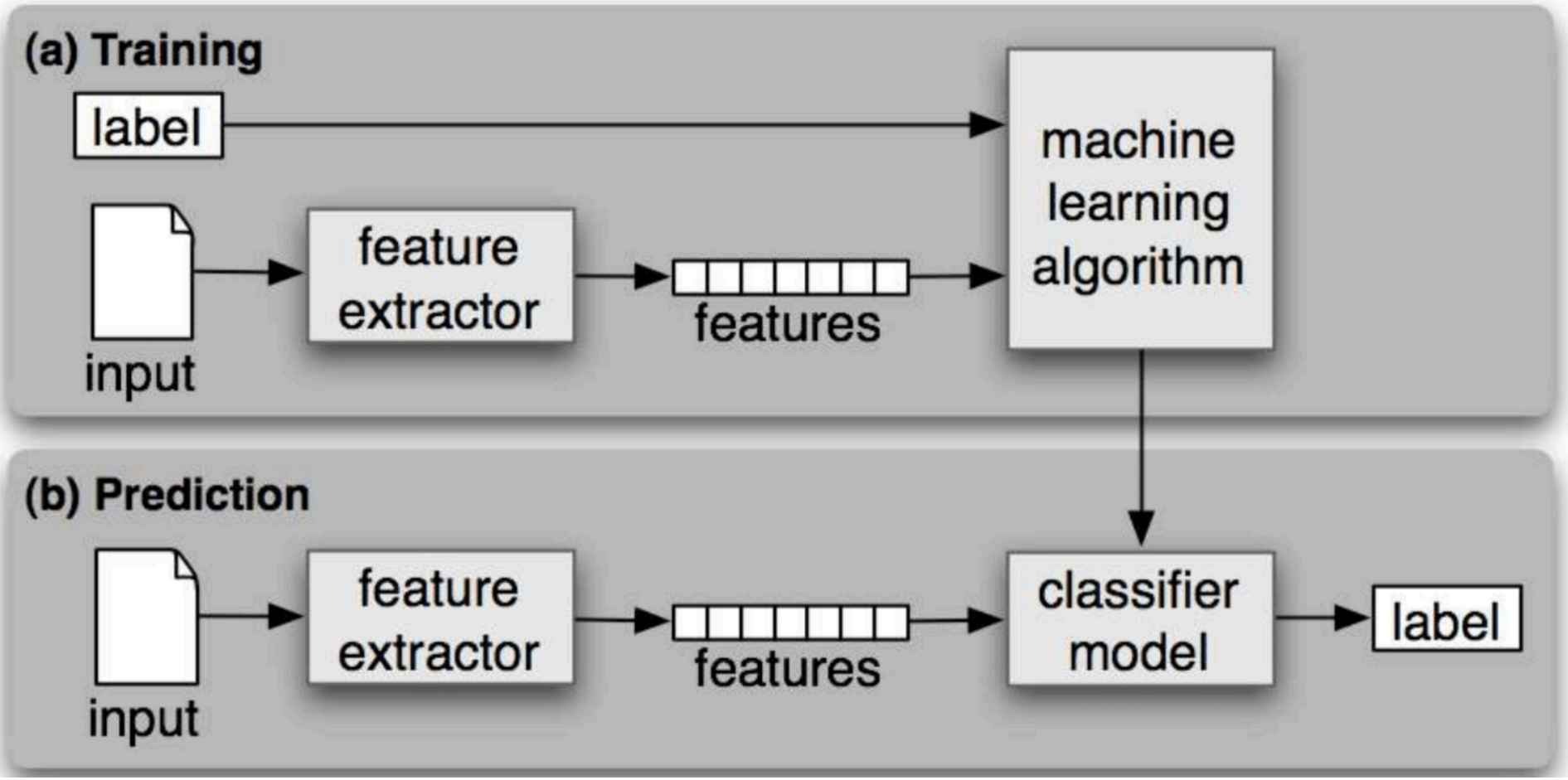
ORL dataset,
AT&T Laboratories, Cambridge UK

Clustering Example



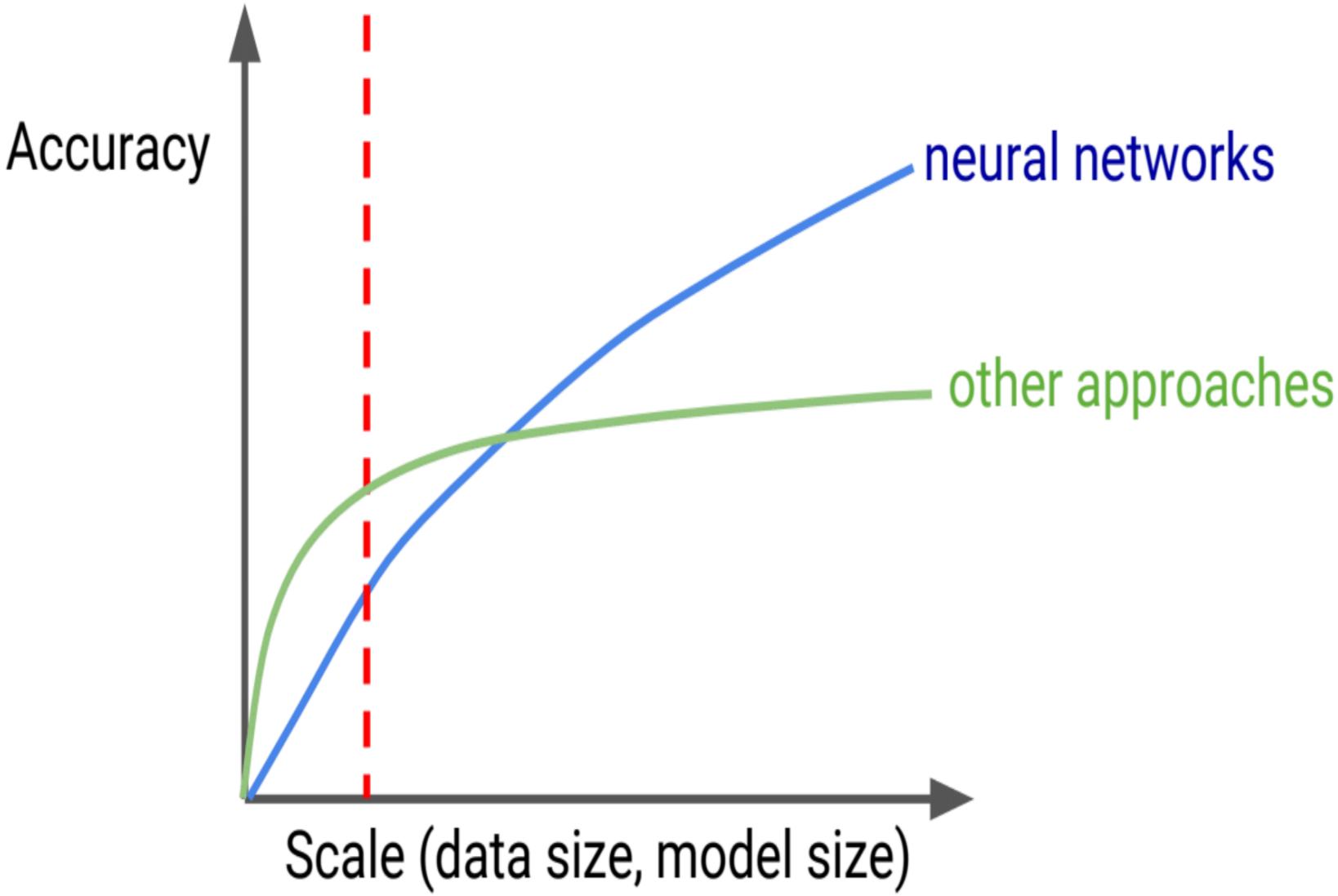
http://pypr.sourceforge.net/kmeans.html

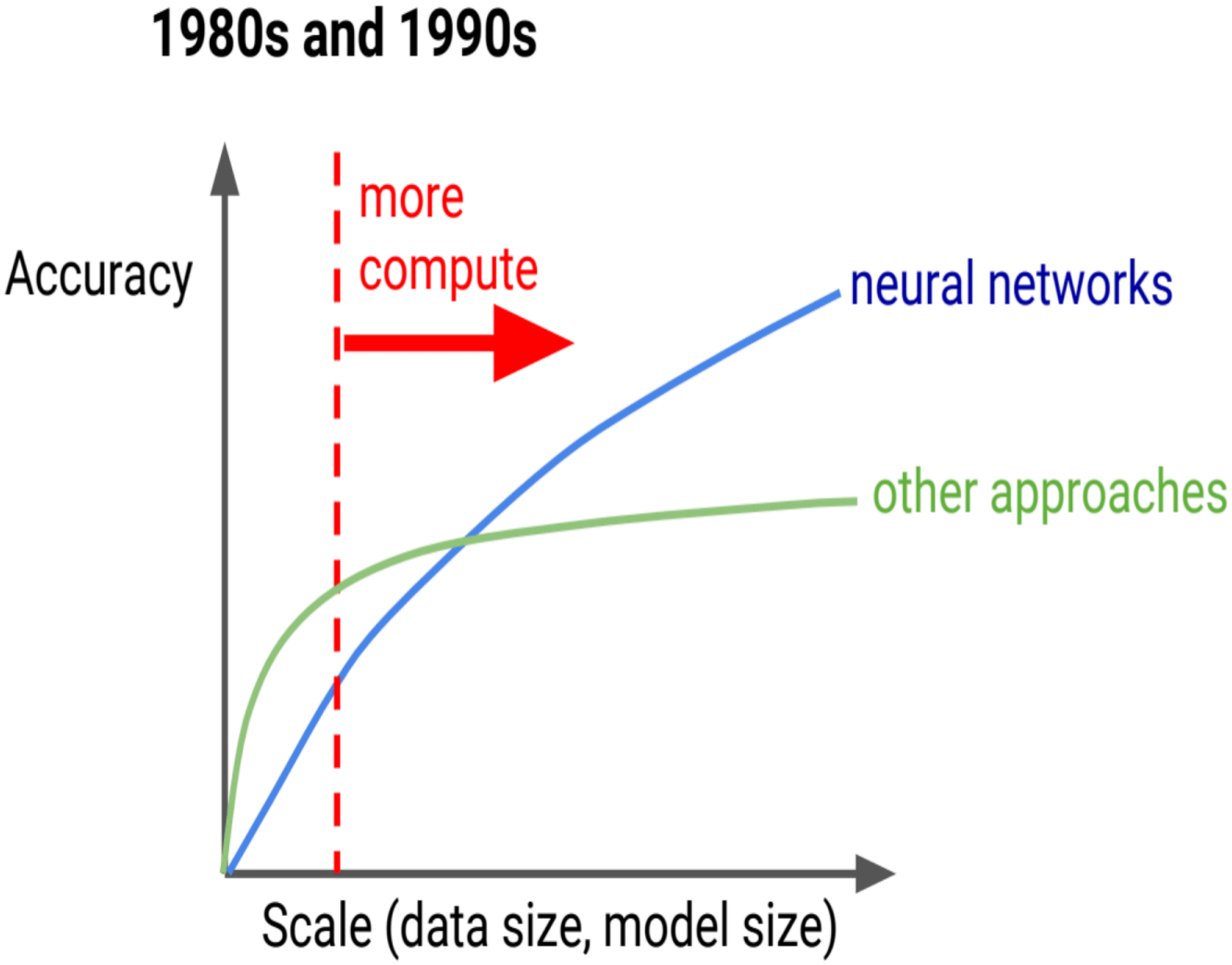
The Process



* Jain, Apache/Solr Data Analytics Group

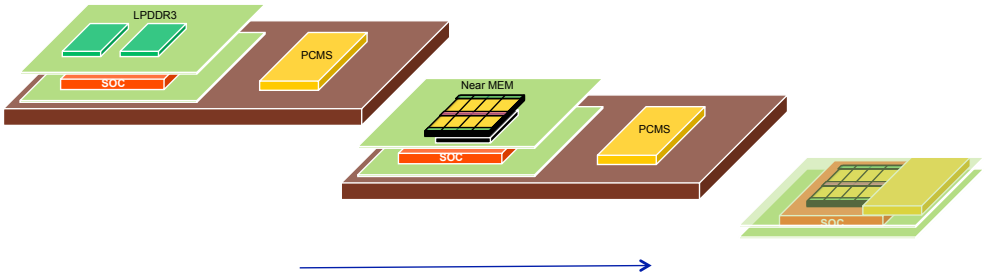
1980s and 1990s

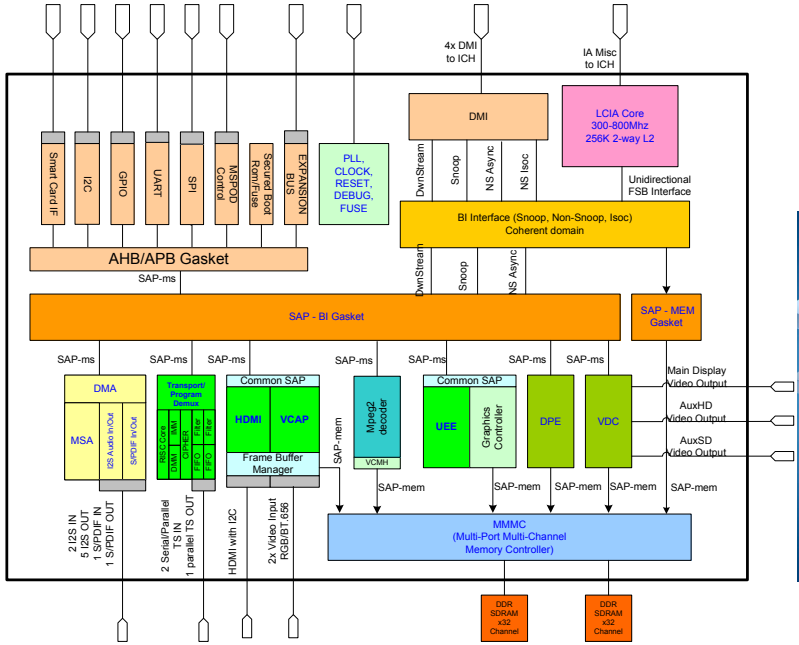


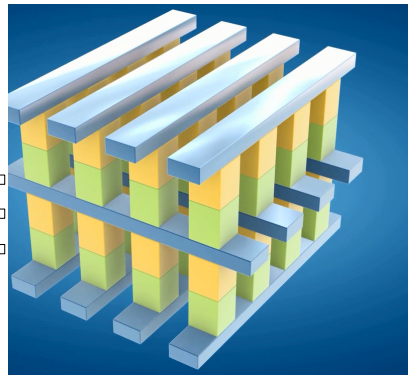

1980s and 1990s
Accuracy
more
compute
neural networks
other approaches
Scale (data size, model size)

Increasing levels of Integration

- Today's IC incorporate IP from across the globe leveraging varying degrees of partnerships
- Density gains are exponentially driving this trend – both in number of IP and complexity of each IP
- New usages are emerging

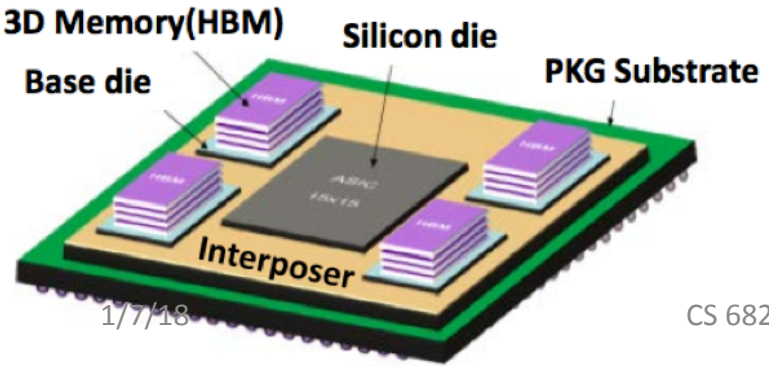




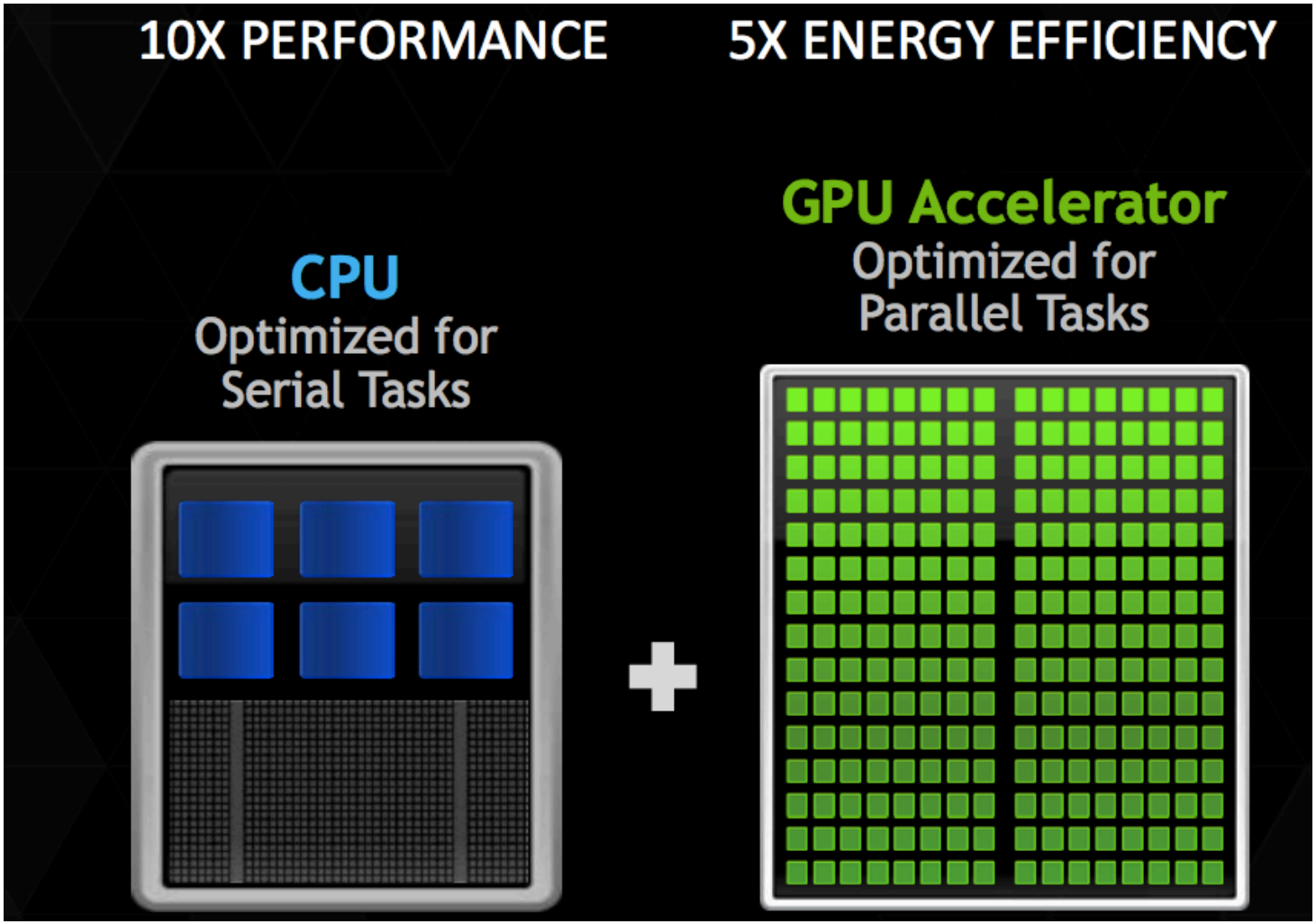


New Enabled Technologies

- MEMs
- Remote sensors
- "Smart Dust" & IoT
- Mobile applications reaching Terascale capabilities
- Increased dependency on automation
- Remote programmability of deeply embedded hardware components in the Data Center



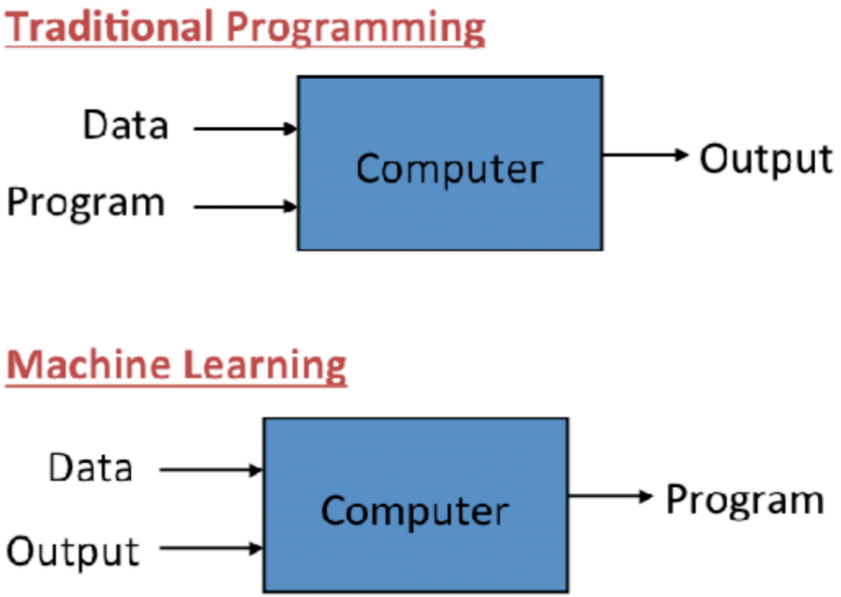
GPUs and Fined Grain Architectures



*Nvidia

Anatomy of a Machine Learning Solution

- Any Machine Learning algorithm has three parts
  - The Output (prediction)
  - The Objective Function or Performance Matrix
  - The Input (samples)



IS THIS A

CAT or DOG?

CAT DOG

OUTPUT LAYER

ACTIVATED NEURONS

INPUT LAYER

DEEP neural NETWORK

Categories of Machine Learning

- Supervised Learning – Given examples of inputs and the desired categories, predict the outputs of future samples
  - Classification
  - Regression
  - Time series prediction

- Unsupervised Learning: Given only inputs (no categories), automatically discover representations, correlations, features, structure, etc.
  - Clustering
  - Outlier detection
  - Compression

- Reinforcement Learning
  - Allows machines and software agents to automatically determine the ideal behavior within a specific context, in order to maximize its performance.

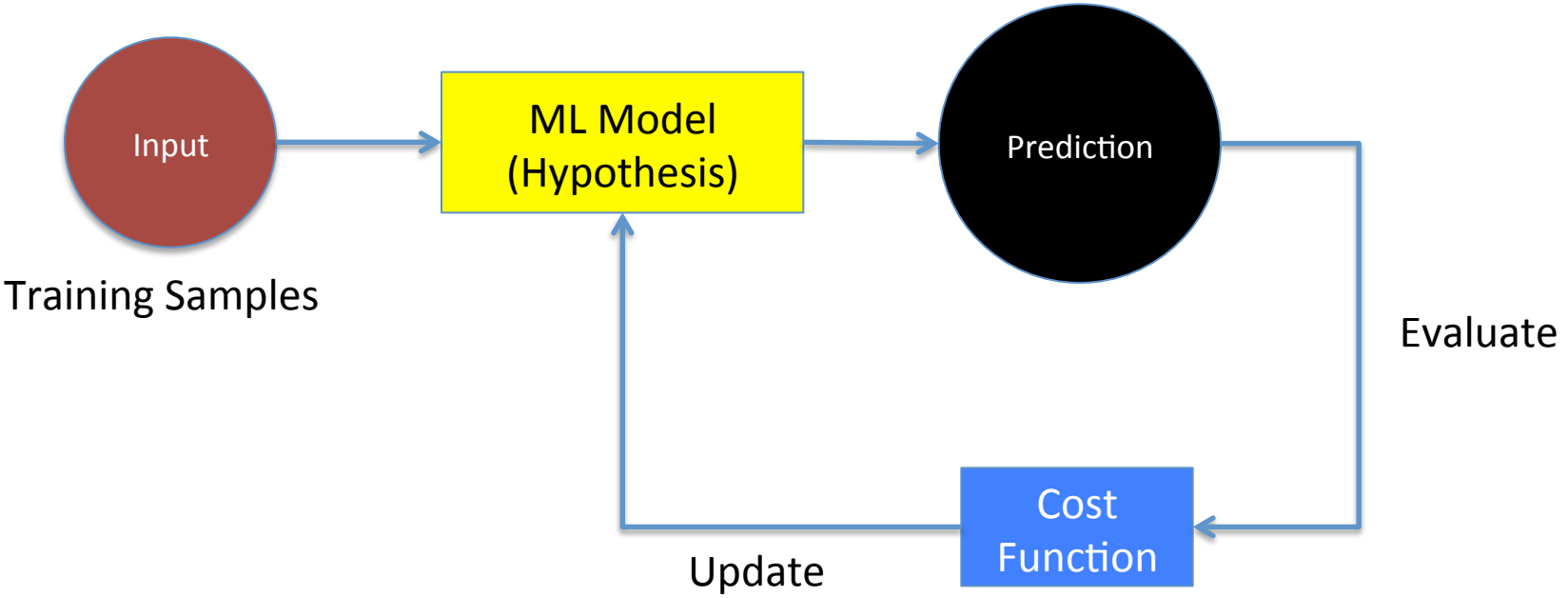
Supervised Learning

- Given: Training set $\{(x_i, y_i) \mid i = 1 \ldots N\}$
- Find: A good approximation to $f : X \rightarrow Y$

example									label
				train					
aardvark	→	1	1	18	4	22	1	18	11	−
cow	→	3	15	23						+
giraffe	→	7	9	18	1	6	6	5		−
termite	→	20	5	18	13	9	20	5		−
oyster	→	15	25	19	20	5	18			+
dove	→	4	15	22	5					−
spider	→	19	16	9	4	5	18			+
dog	→	4	15	7						+
elephant	→	5	12	5	16	8	1	14	20	+
				test					
rabbit	→	18	1	2	2	9	20			
frog	→	6	18	15	7					
kangaroo	→	11	1	14	7	1	18	15	15	

*Schapire

The Training Process



Optimization Problems

- Some well defined problem
- Representation of an element in the solution space
- A quantifiable metric of the goodness of any and all proposed solutions
- A method of ordering the proposed solutions
  - $S_i > S_j$

Optimization Problems

- Formally …

  "An optimization problem is the problem of finding the best solution from all feasible solutions. Optimization problems can be divided into two categories depending on whether the variables are continuous or discrete."

Sample Data Set

Data:

Patient103 time=1	*Patient103* time=2	*Patient103* time=n
Age: 23	Age: 23	Age: 23
FirstPregnancy: no	FirstPregnancy: no	FirstPregnancy: no
Anemia: no	Anemia: no	Anemia: no
Diabetes: no	Diabetes: YES	Diabetes: no
PreviousPrematureBirth: no	PreviousPrematureBirth: no	PreviousPrematureBirth: no
Ultrasound: ?	Ultrasound: abnormal	Ultrasound: ?
Elective C–Section: ?	Elective C–Section: no	Elective C–Section: no
Emergency C–Section: ?	Emergency C–Section: ?	**Emergency C–Section: Yes**
...	...	...

Optimization Problems

- An example representation for a learned function

$w_0 + w_1 \cdot bp(b) + w_2 \cdot rp(b) + w_3 \cdot bk(b) + w_4 \cdot rk(b) + w_5 \cdot bt(b) + w_6 \cdot rt(b$

- $bp(b)$: number of black pieces on board b
- $rp(b)$: number of red pieces on b
- $bk(b)$: number of black kings on b
- $rk(b)$: number of red kings on b
- $bt(b)$: number of red pieces threatened by black (i.e., which can be taken on black's next turn)
- $rt(b)$: number of black pieces threatened by red

Representing Data

- Have m observations (data points)

$$\left\{x^{(1)} \ldots, x^{(m)}\right\}$$

- Each observation is a vector consisting of n features

$$x^{(j)} = [x_1^{(j)} x_2^{(j)} \ldots x_n^{(j)}]$$

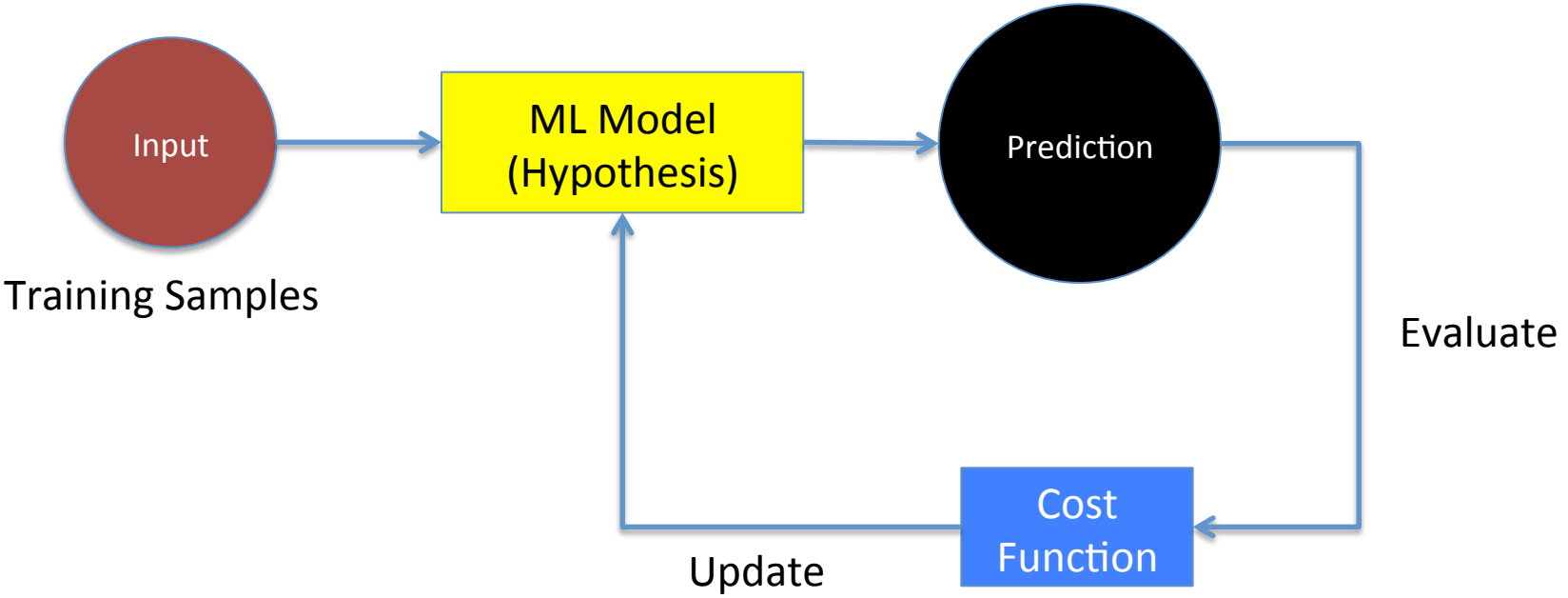
- Often, represent this as a "data matrix"

$$\underline{X} = \begin{bmatrix} x_1^{(1)} & \ldots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(m)} & \ldots & x_n^{(m)} \end{bmatrix}$$

Machine Learning

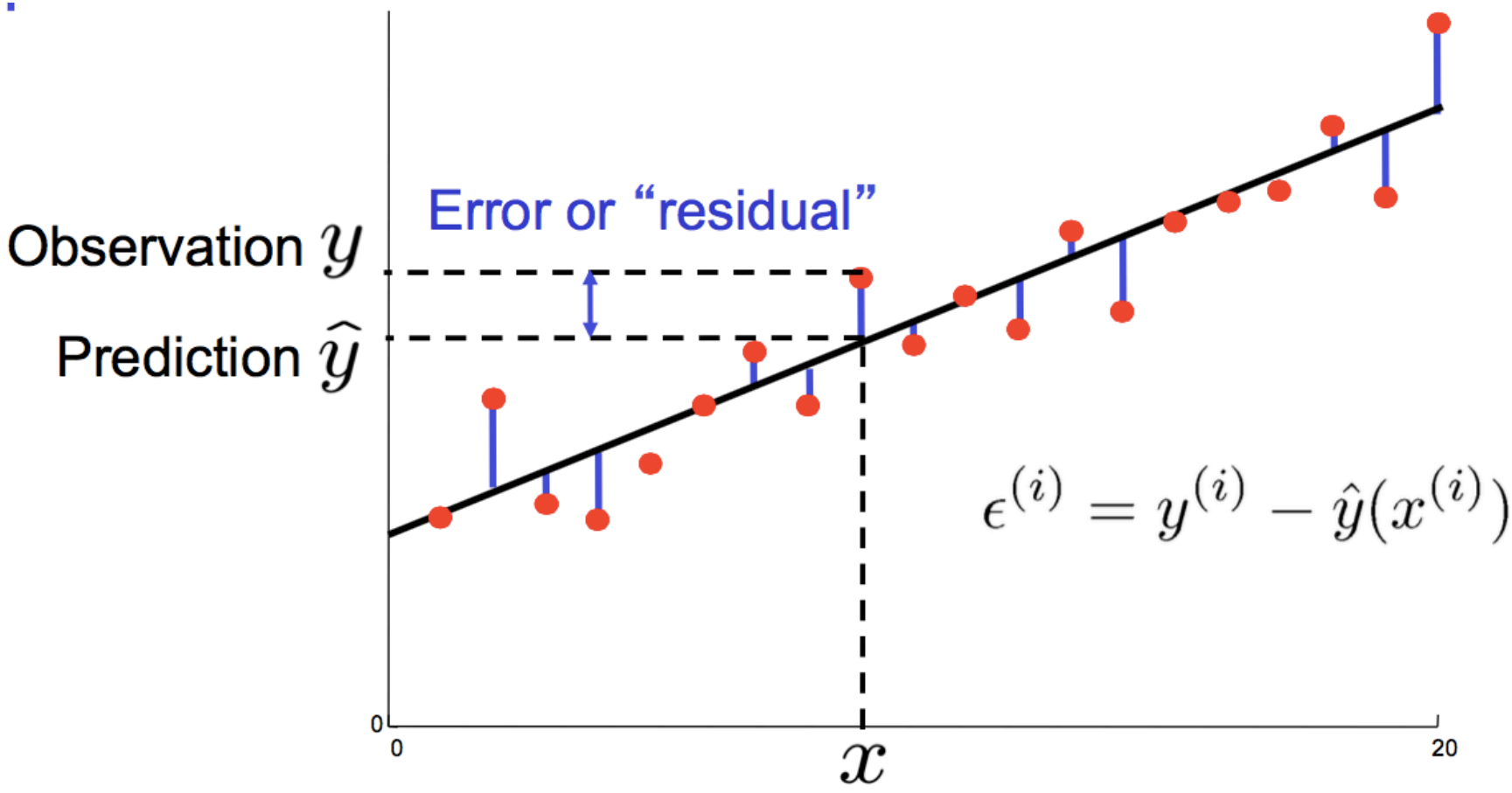
- **Premise:** A hypothesis (i.e. ML Model or classifier) that is consistent with a sufficiently large number of representative training examples is likely to accurately classify novel instances drawn from the same universe

The Training Process



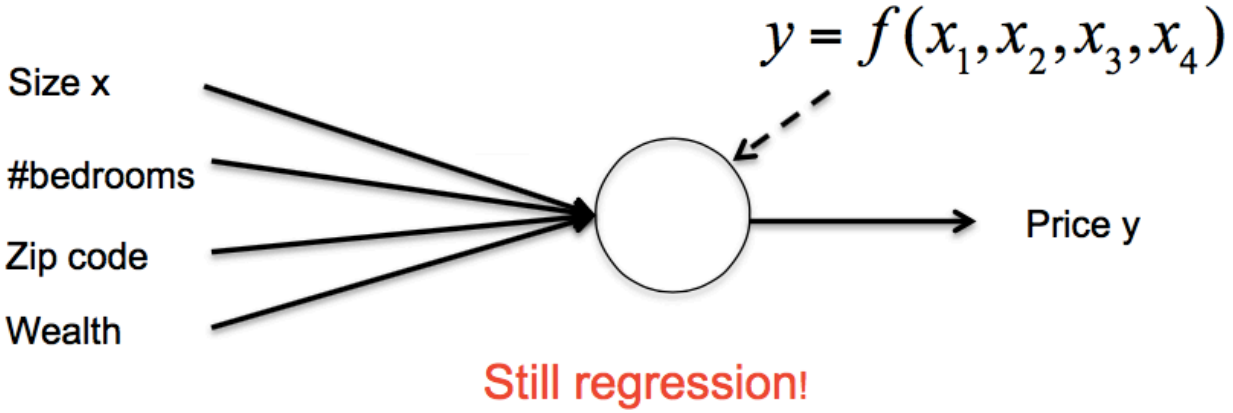
- Predict – apply rules to test data (samples)
- Score – get feedback on performance of our model
- Learn (use feedback) – change predictor to do better

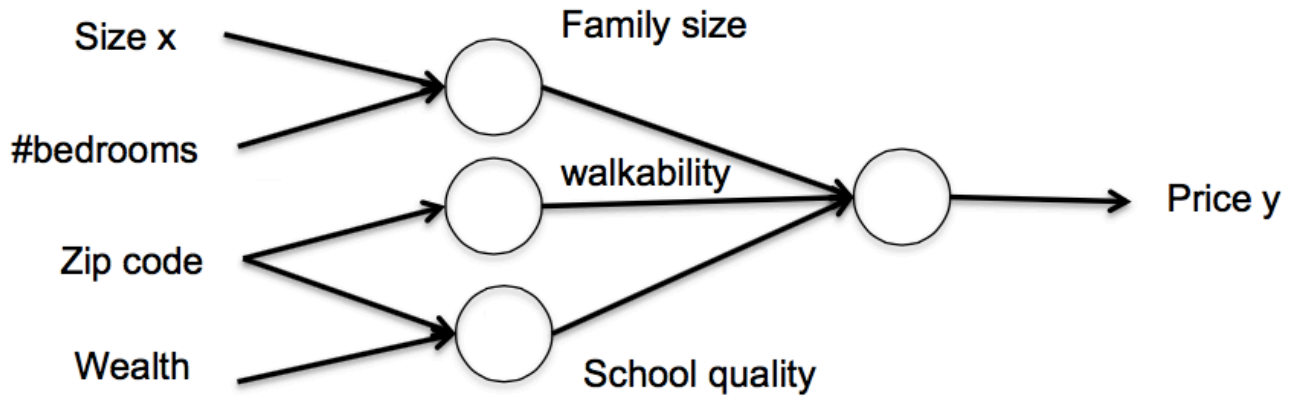
Linear Regression



$$\text{MSE} = \frac{1}{m} \sum_i \left(y^{(i)} - \hat{y}(x^{(i)}) \right)^2$$

Going from Regression to NN





Loss or Cost Function

- For the purposed of ML, we need to move beyond just the prediction capability of linear regression. We need is a cost function so we can tune our model.

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

Linear Regression

- Regression analysis is used to:
  - Predict the value of a dependent variable based on the value of at least one independent variable
  - Explain the impact of changes in an independent variable on the dependent variable

Dependent variable: the variable we wish to explain
(also called the endogenous variable)

Independent variable: the variable used to explain the dependent variable
(also called the exogenous variable)

* Basic Business Statistic, Berenson et. al.

Linear Regression

- An equation can be fit to show the best linear relationship between two variables:

$$Y = \beta_0 + \beta_1 X$$

Where Y is the dependent variable and
X is the independent variable
β_0 is the Y-intercept
β_1 is the slope

* Basic Business Statistic, Berenson et. al.

Linear Regression

- The relationship between X and Y is described by a linear function
- Changes in Y are assumed to be caused by changes in X
- Linear regression population equation model

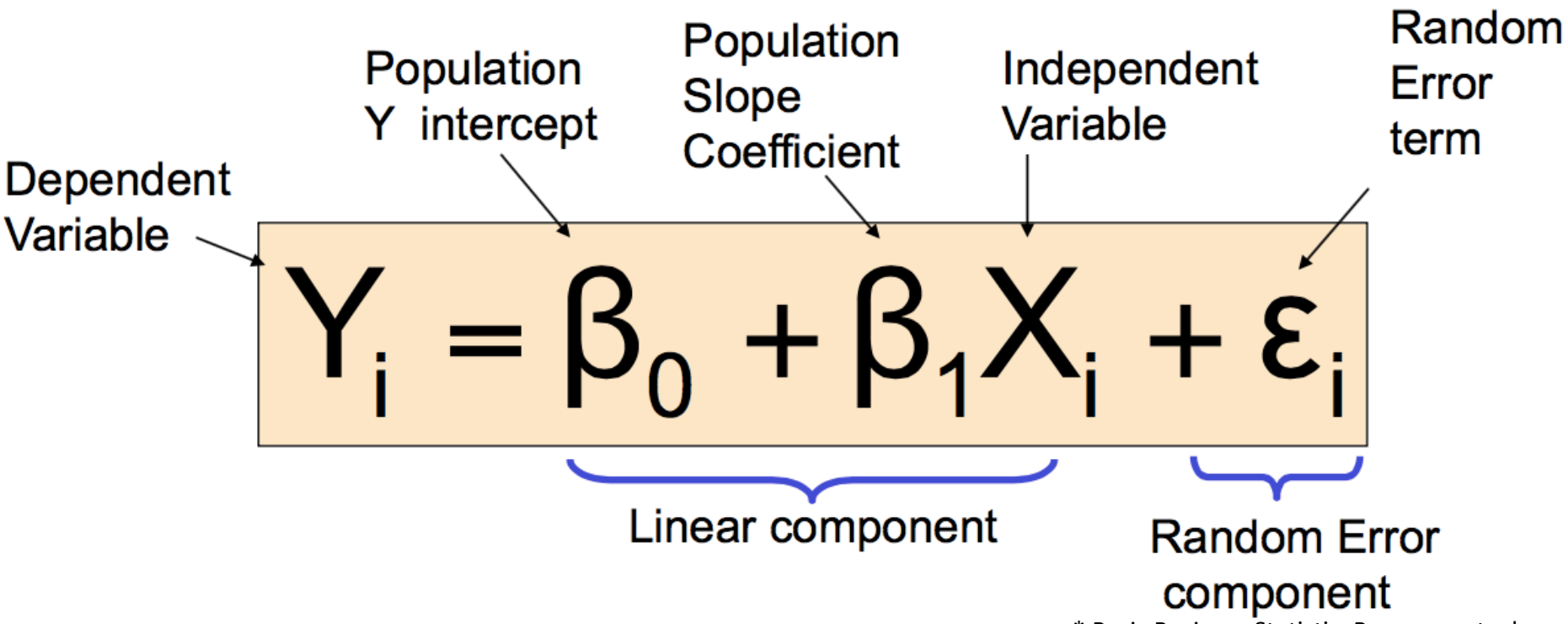
$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

- Where β_0 and β_1 are the population model coefficients and ε is a random error term.

* Basic Business Statistic, Berenson et. al.

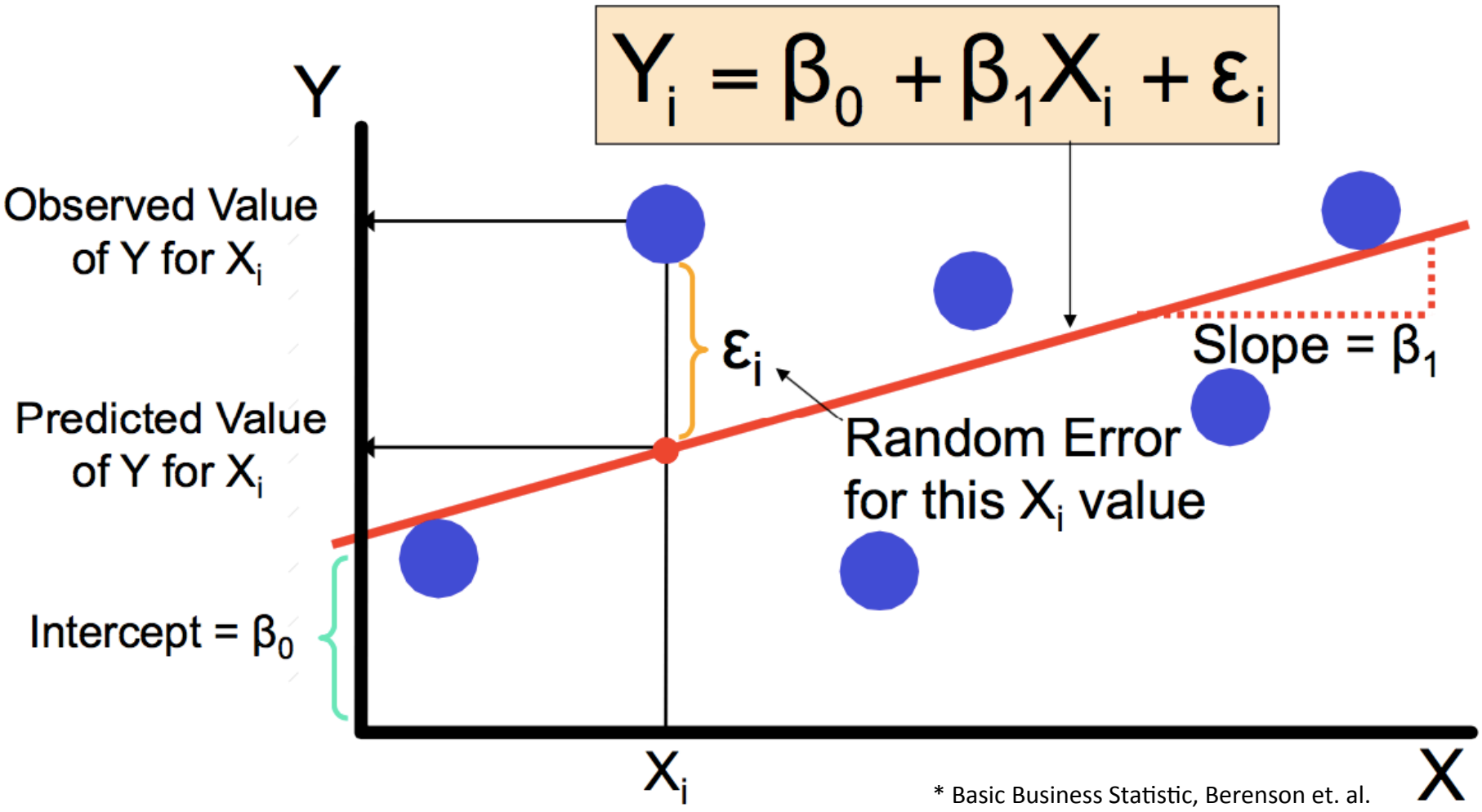
Linear Regression

The population regression model:



* Basic Business Statistic, Berenson et. al.

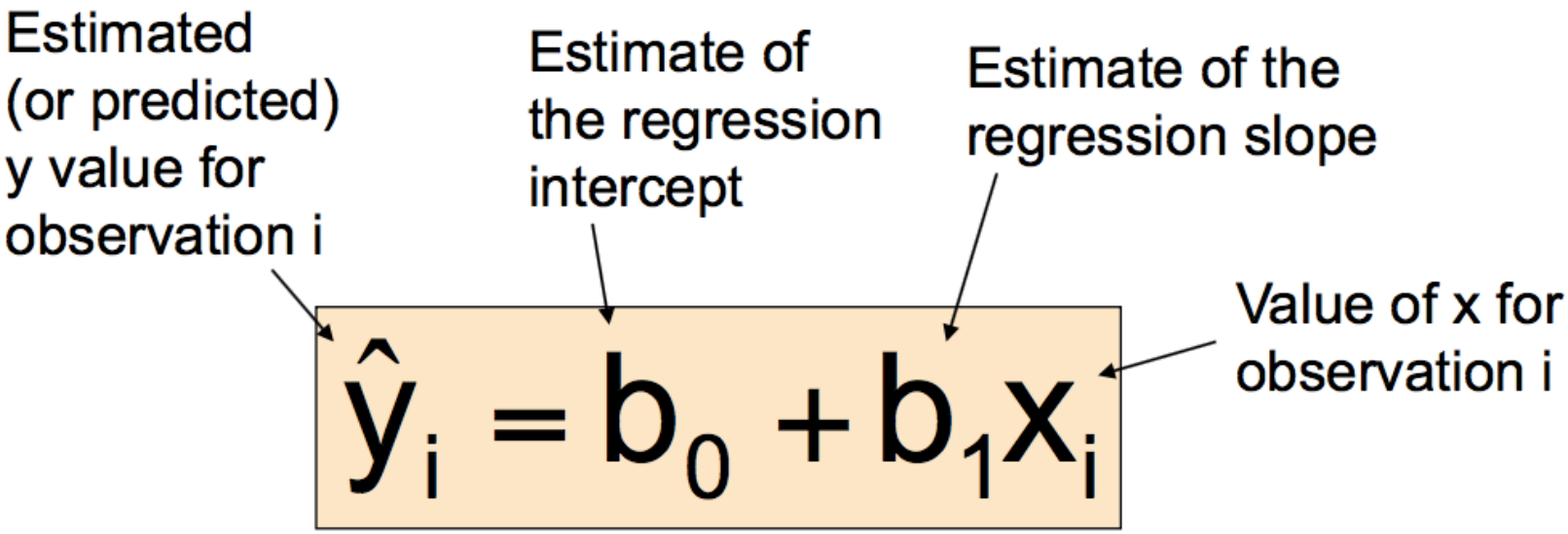
Linear Regression



* Basic Business Statistic, Berenson et. al.

Linear Regression

The simple linear regression equation provides an estimate of the population regression line



The individual random error terms e_i have a mean of zero

$$e_i = (y_i - \hat{y}_i) = y_i - (b_0 + b_1 x_i)$$

* Basic Business Statistic, Berenson et. al.

Linear Regression

- b_0 and b_1 are obtained by finding the values of b_0 and b_1 that minimize the sum of the squared differences between y and $\hat{y}$:

$$\min \text{ SSE} = \min \sum e_i^2$$

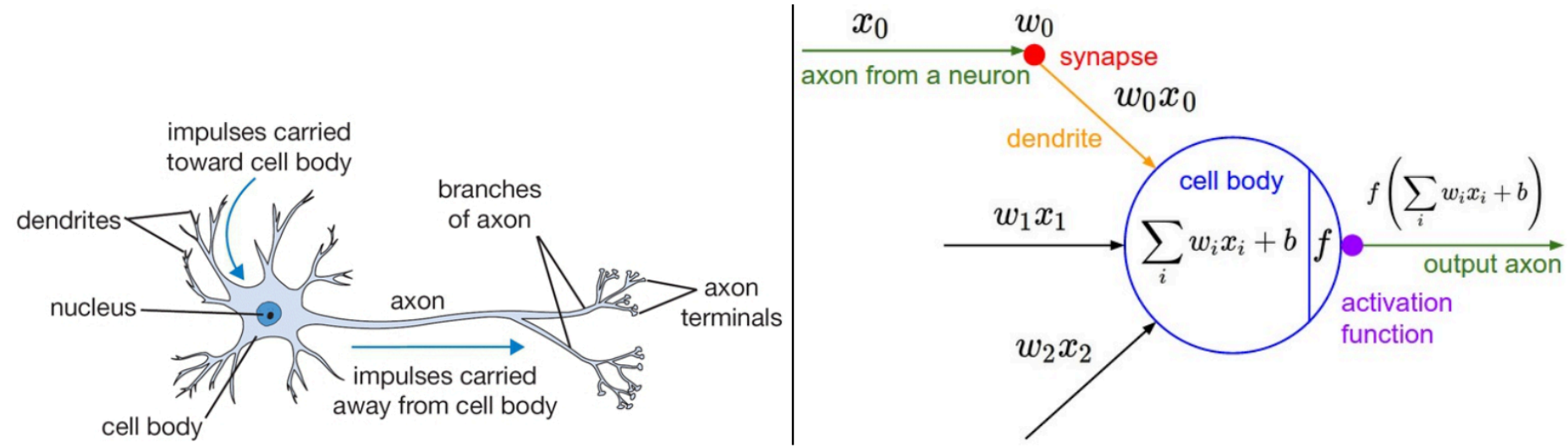
$$= \min \sum (y_i - \hat{y}_i)^2$$

$$= \min \sum [y_i - (b_0 + b_1 x_i)]^2$$

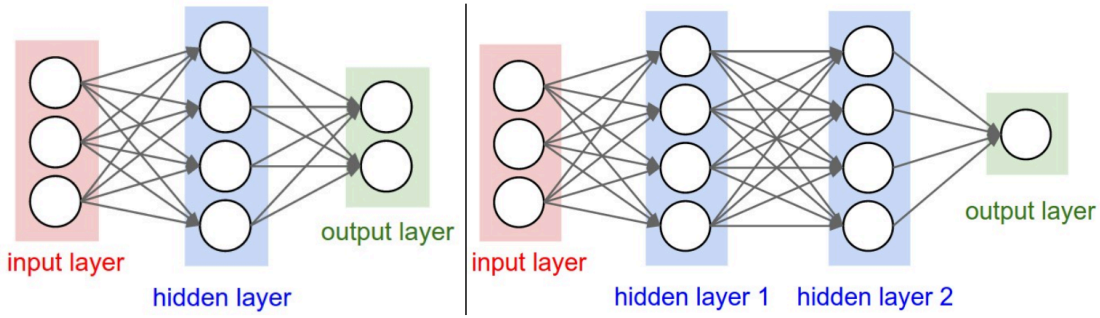
Differential calculus is used to obtain the coefficient estimators b_0 and b_1 that minimize SSE

* Basic Business Statistic, Berenson et. al.

Perceptron

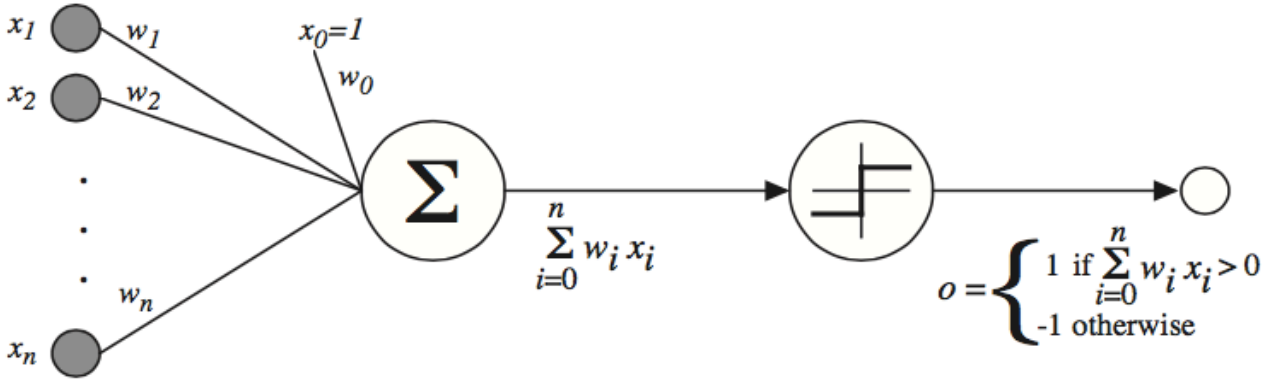


A cartoon drawing of a biological neuron (left) and its mathematical model (right).



Left: A 2-layer Neural Network (one hidden layer of 4 neurons (or units) and one output layer with 2 neurons), and three inputs. Right: A 3-layer neural network with three inputs, two hidden layers of 4 neurons each and one output layer. Notice that in both cases there are connections (synapses) between neurons across layers, but not within a layer.

Perceptron



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

Sometimes we'll use simpler vector notation:

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

* T. Mitchell

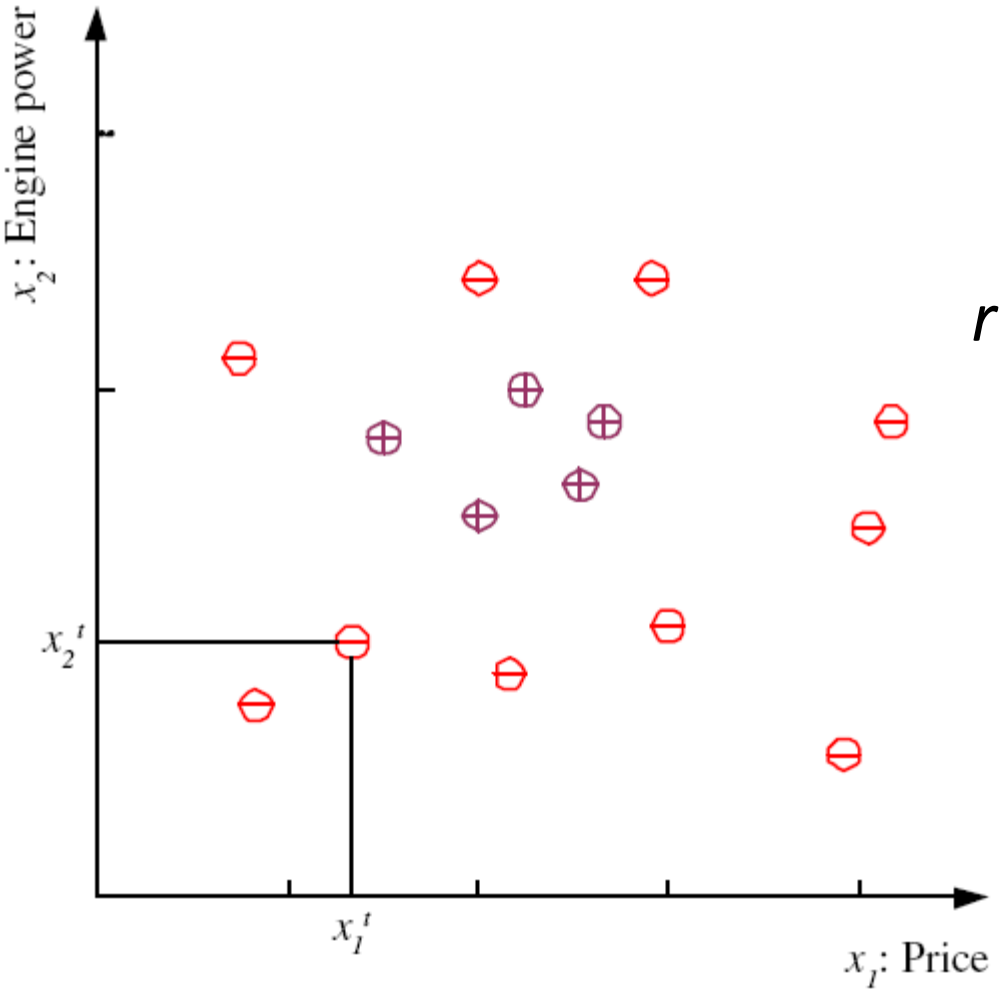
Learning a Class from Examples

- Class C of a "family car"
  - Prediction: Is car *x* a family car?
  - Knowledge extraction: What do people expect from a family car?
- Output:

  Positive (+) and negative (–) examples
- Input representation:

  x_1: price, x_2 : engine power

Training set $\mathcal{X}$



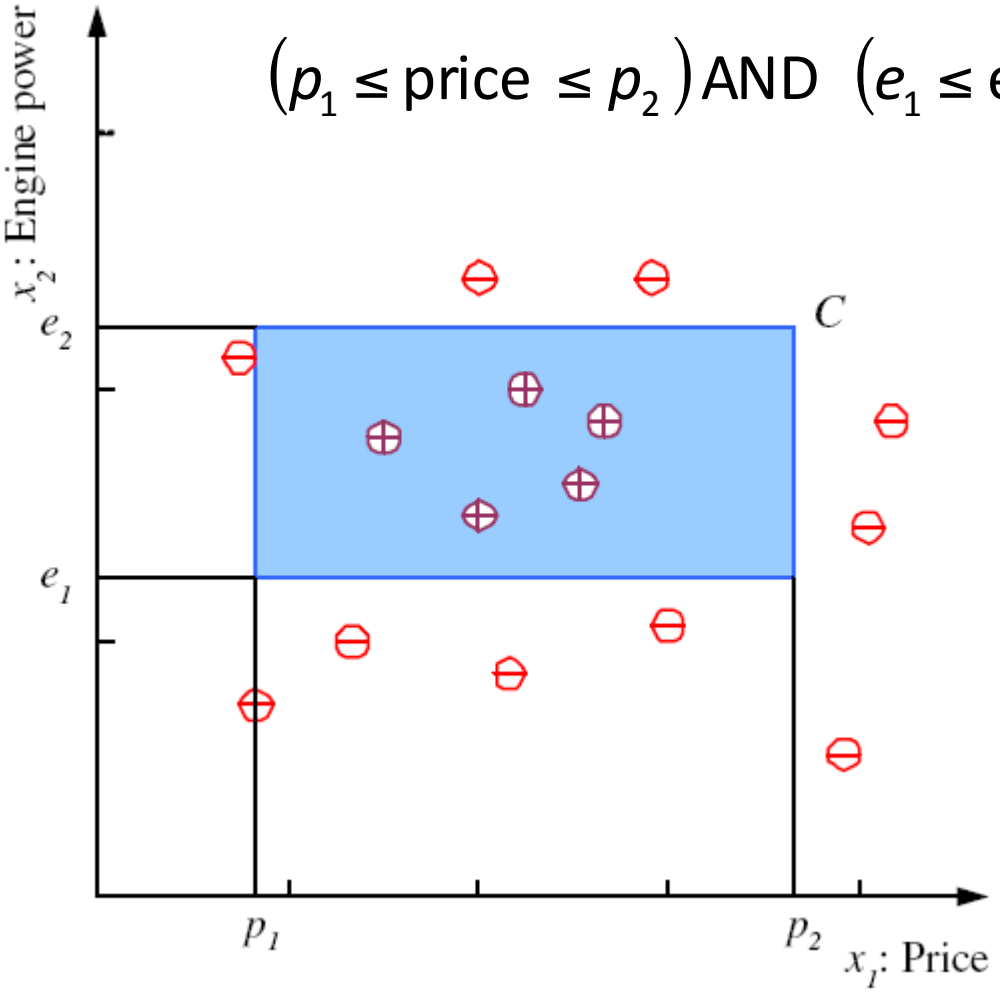
$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N$$

$$r = \begin{cases} 1 \text{ if } \mathbf{x} \text{ is positive} \\ 0 \text{ if } \mathbf{x} \text{ is negative} \end{cases}$$

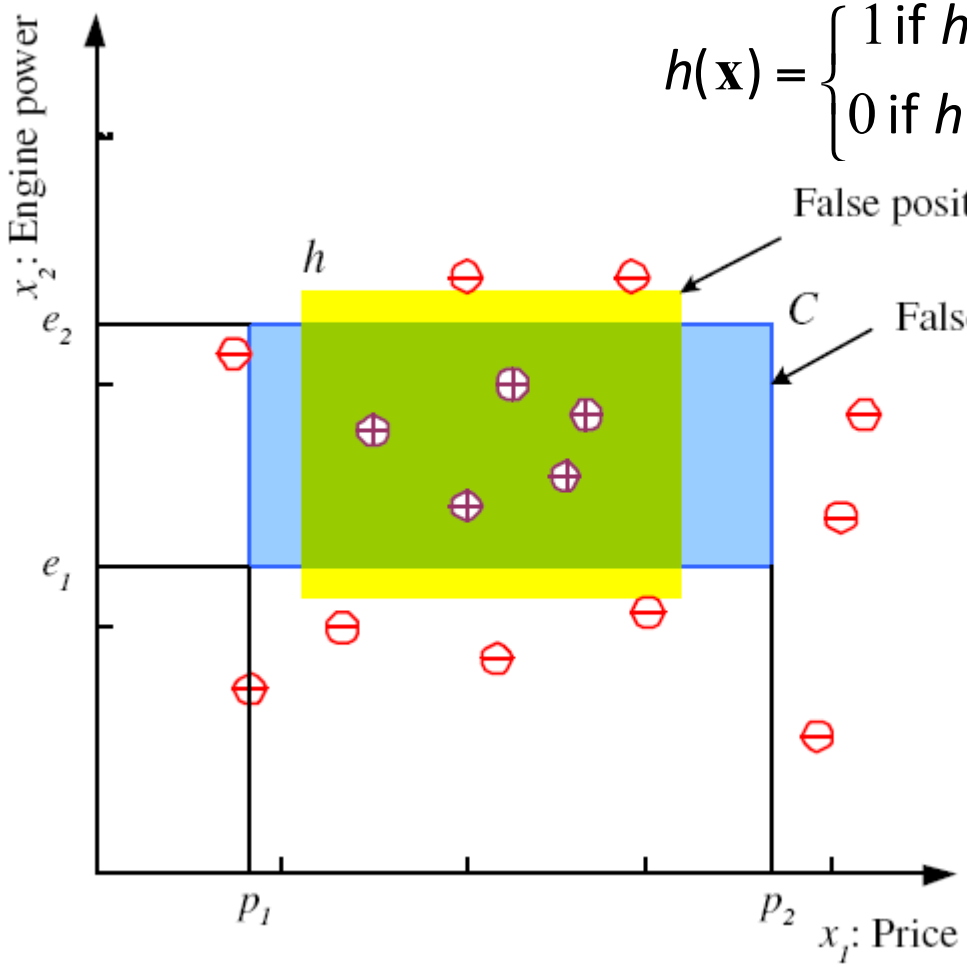
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Class C

$$(p_1 \leq \text{price} \leq p_2) \text{ AND } (e_1 \leq \text{engine power} \leq e_2)$$



Hypothesis class $\mathcal{H}$

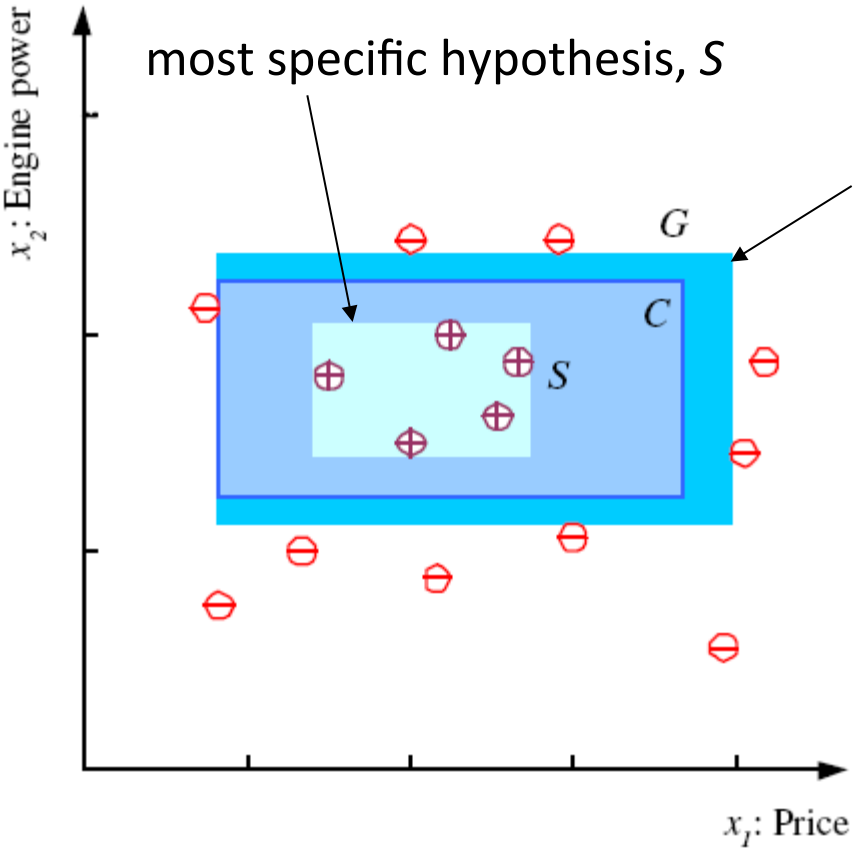


$$h(\mathbf{x}) = \begin{cases} 1 \text{ if } h \text{ says } \mathbf{x} \text{ is positive} \\ 0 \text{ if } h \text{ says } \mathbf{x} \text{ is negative} \end{cases}$$

Error of h on $\mathcal{H}$

$$E(h \mid \mathcal{X}) = \sum_{t=1}^{N} 1\left(h\left(\mathbf{x}^t\right) \neq r^t\right)$$

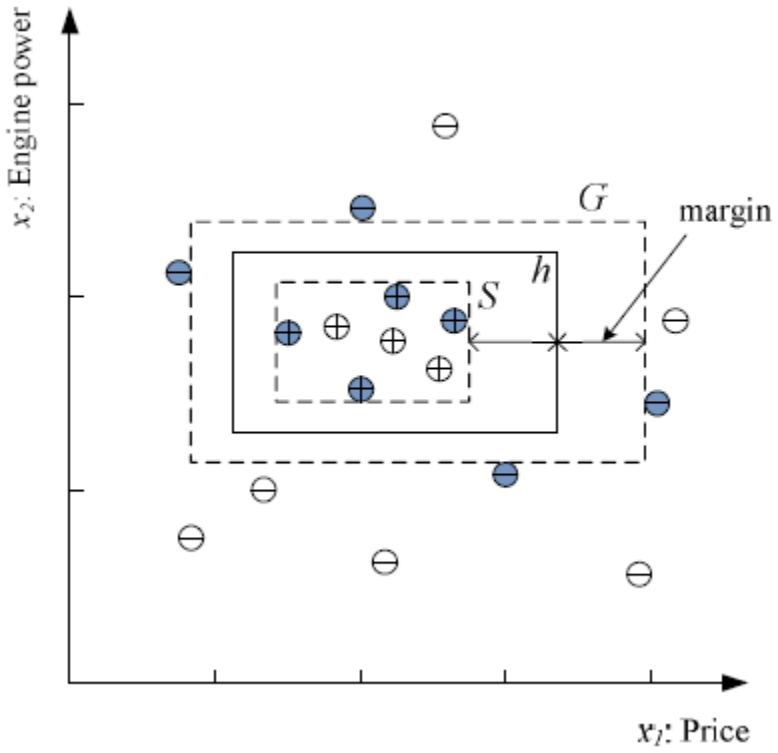
S, G, and the Version Space



$h \in H$, between S and G is consistent and make up the version space
(Mitchell, 1997)

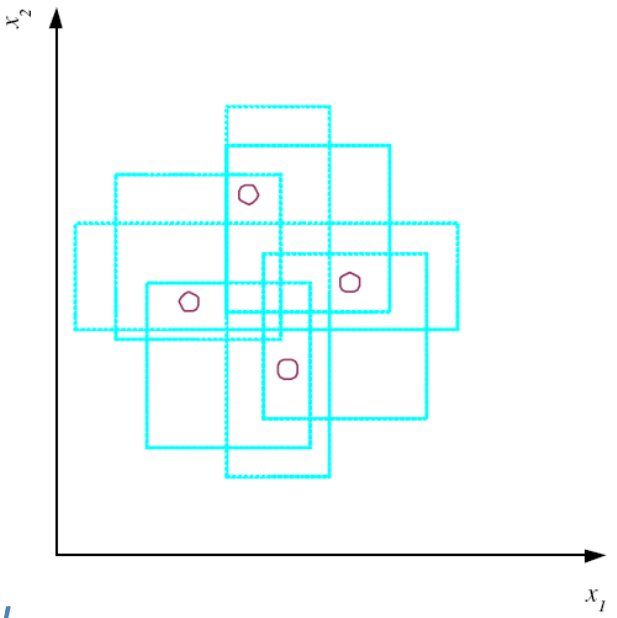
Margin

- Choose *h* with largest margin



VC Dimension

- N points can be labeled in 2^N ways as +/–
- $\mathcal{H}$ shatters N if there exists $h \in \mathcal{H}$ consistent for any of these:
  VC($\mathcal{H}$) = N



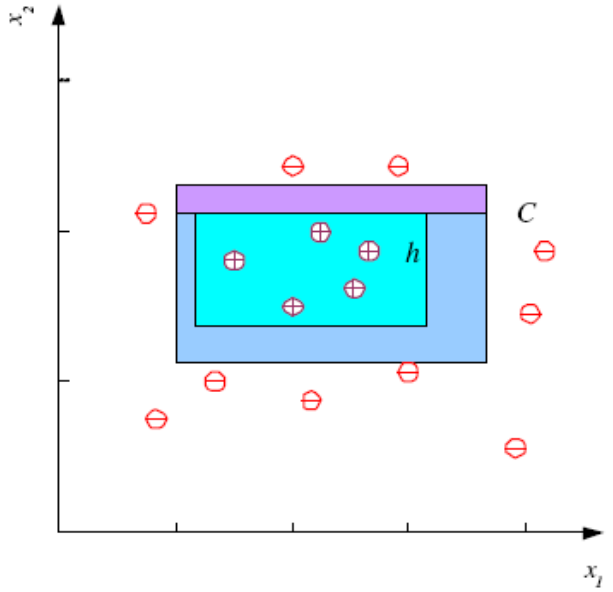
An axis-aligned rectangle shatters 4 points only !

Probably Approximately Correct (PAC) Learning

- How many training examples N should we have, such that with probability at least $1 - \delta$, h has error at most ε ?

  (Blumer et al., 1989)

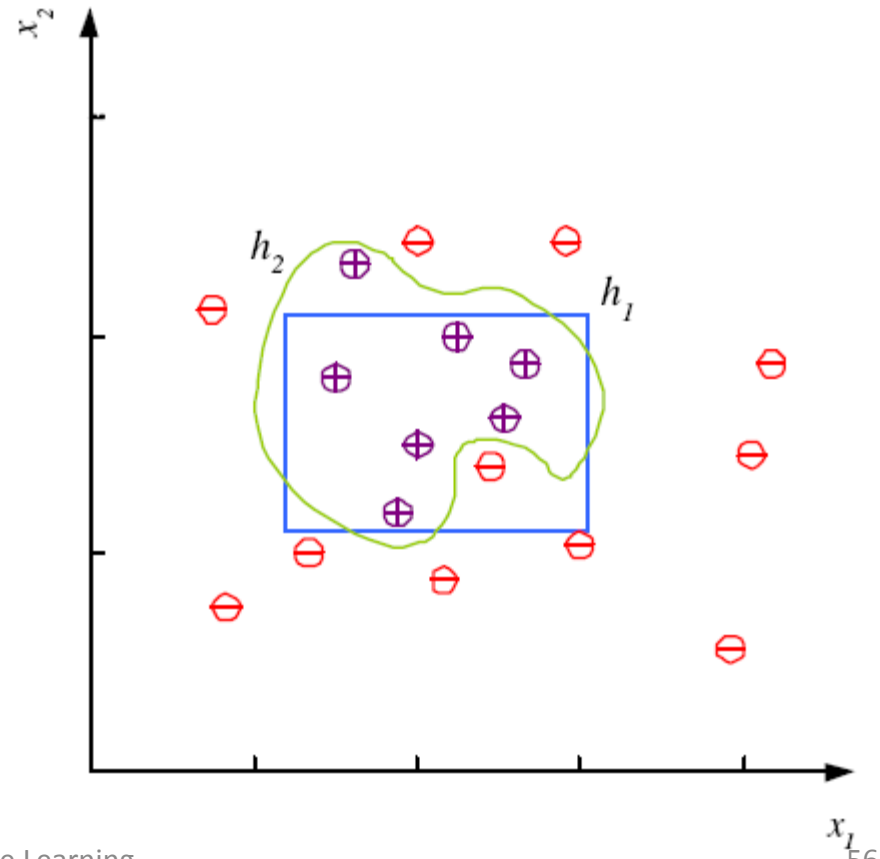
- Each strip is at most $\varepsilon/4$
- Pr that we miss a strip $1 - \varepsilon/4$
- Pr that N instances miss a strip $(1 - \varepsilon/4)^N$
- Pr that N instances miss 4 strips $4(1 - \varepsilon/4)^N$
- $4(1 - \varepsilon/4)^N \leq \delta$ and $(1 - x) \leq \exp(-x)$
- $4\exp(-\varepsilon N/4) \leq \delta$ and $N \geq (4/\varepsilon)\log(4/\delta)$



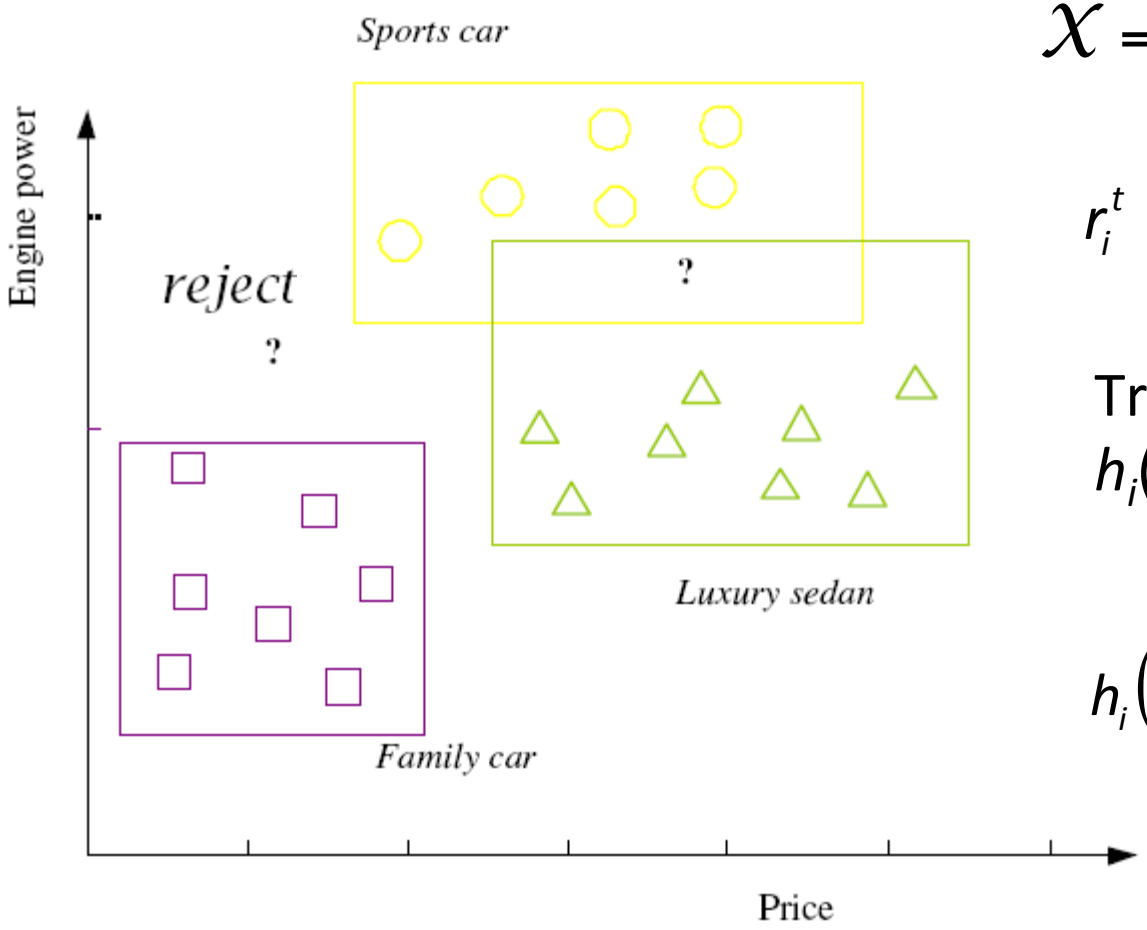
Noise and Model Complexity

Use the simpler one because

- Simpler to use (lower computational complexity)
- Easier to train (lower space complexity)
- Easier to explain (more interpretable)
- Generalizes better (lower variance - Occam's razor)



Multiple Classes, C_i i=1,...,K



$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N$$

$$r_i^t = \begin{cases} 1 \text{ if } \mathbf{x}^t \in C_i \\ 0 \text{ if } \mathbf{x}^t \in C_j, j \neq i \end{cases}$$

Train hypotheses $h_i(\boldsymbol{x})$, $i = 1,...,K$:

$$h_i\left(\mathbf{x}^t\right) = \begin{cases} 1 \text{ if } \mathbf{x}^t \in C_i \\ 0 \text{ if } \mathbf{x}^t \in C_j, j \neq i \end{cases}$$

Model Selection & Generalization

- Learning is an ill-posed problem; data is not sufficient to find a unique solution
- The need for inductive bias, assumptions about $\mathcal{H}$
- Generalization: How well a model performs on new data
- Overfitting: $\mathcal{H}$ more complex than C or f
- Underfitting: $\mathcal{H}$ less complex than C or f

Triple Trade-Off

- There is a trade-off between three factors (Dietterich, 2003):
  1. Complexity of $\mathcal{H}$, $c(\mathcal{H})$,
  2. Training set size, N,
  3. Generalization error, E, on new data

☐ As $N \uparrow$, $E \downarrow$

☐ As $c(\mathcal{H}) \uparrow$, first $E \downarrow$ and then $E \uparrow$

Cross-Validation

- To estimate generalization error, we need data unseen during training. We split the data as
  - Training set (50%)
  - Validation set (25%)
  - Test (publication) set (25%)
- Resampling when there is few data

Dimensions of a Supervised Learner

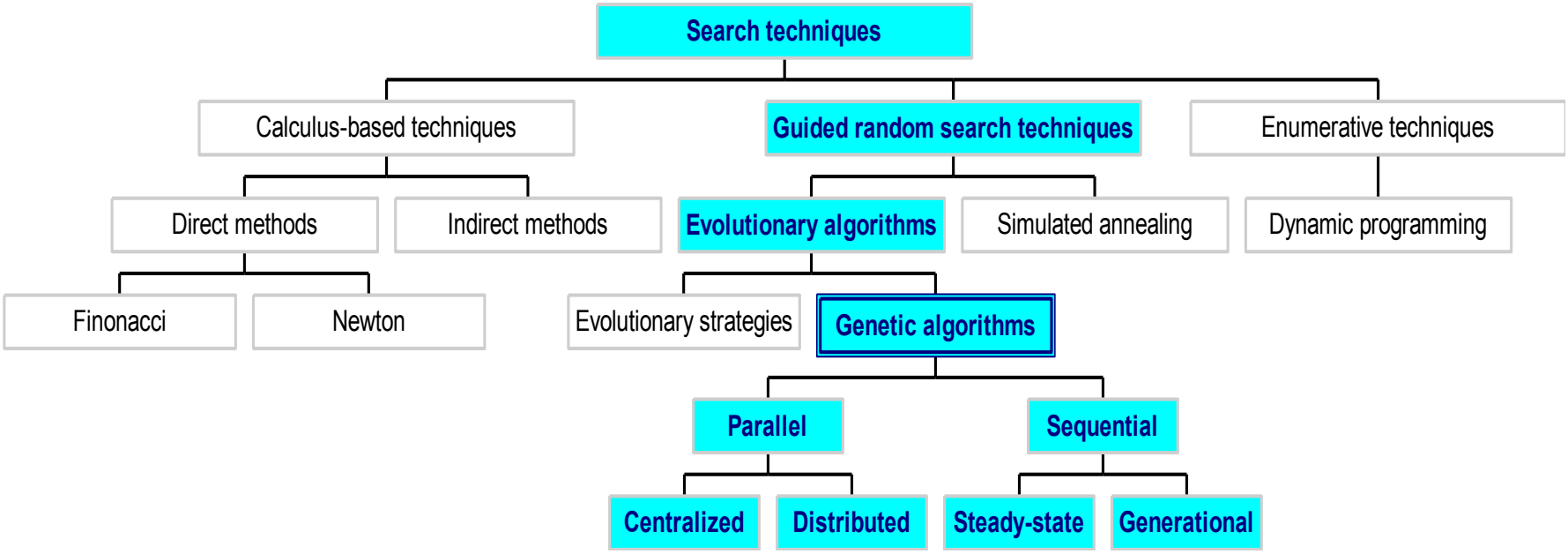
1. Model: $g(\mathbf{x} \mid \theta)$

2. Loss function: $E(\theta \mid \mathcal{X}) = \sum_t L(r^t, g(\mathbf{x}^t \mid \theta))$

3. Optimization procedure:

$$\theta^* = \arg\min_\theta E(\theta \mid \mathcal{X})$$

Other Approaches



A Simple Example

The Traveling Salesman Problem:

Find a tour of a given set of cities so that

- each city is visited only once
- the total distance traveled is minimized

Representation

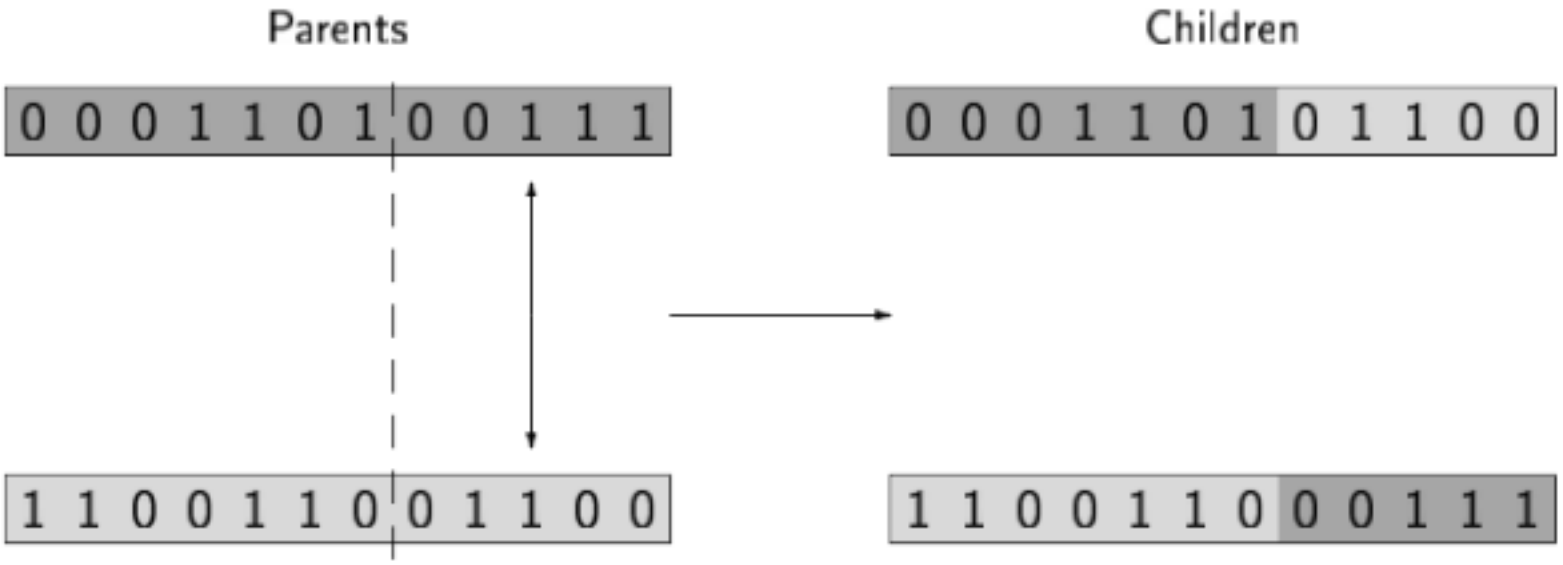
Representation is an ordered list of city numbers known as an *order-based* GA.

1) London	3) Dunedin	5) Beijing	7) Tokyo
2) Venice	4) Singapore	6) Phoenix	8) Victoria

CityList1 (3 5 7 2 1 6 4 8)

CityList2 (2 5 7 6 8 1 3 4)

Crossover



* Bodenhofer

Crossover

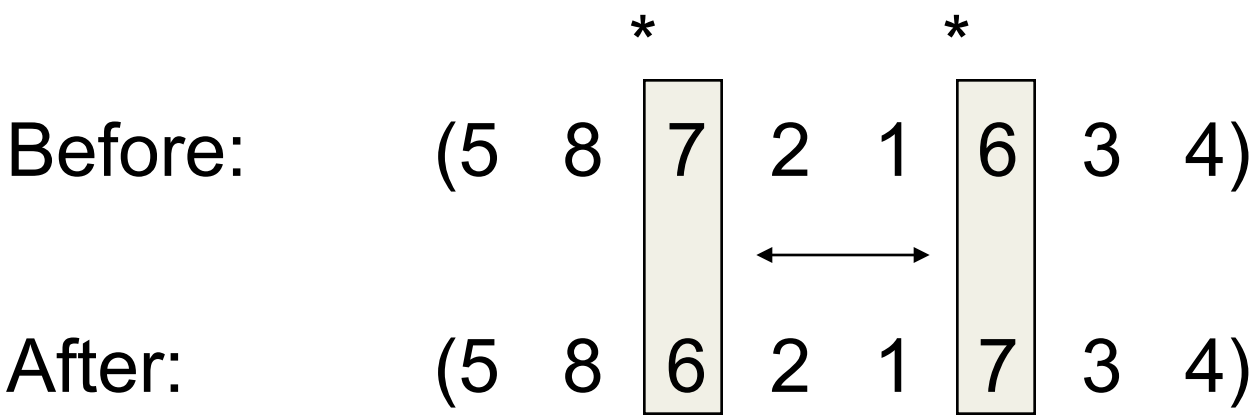
Crossover combines inversion and recombination:

			*			*		
Parent1	(3	5	7	2	1	6	4	8)
Parent2	(2	5	7	6	8	1	3	4)
Child	(2	5	7	2	1	6	3	4)

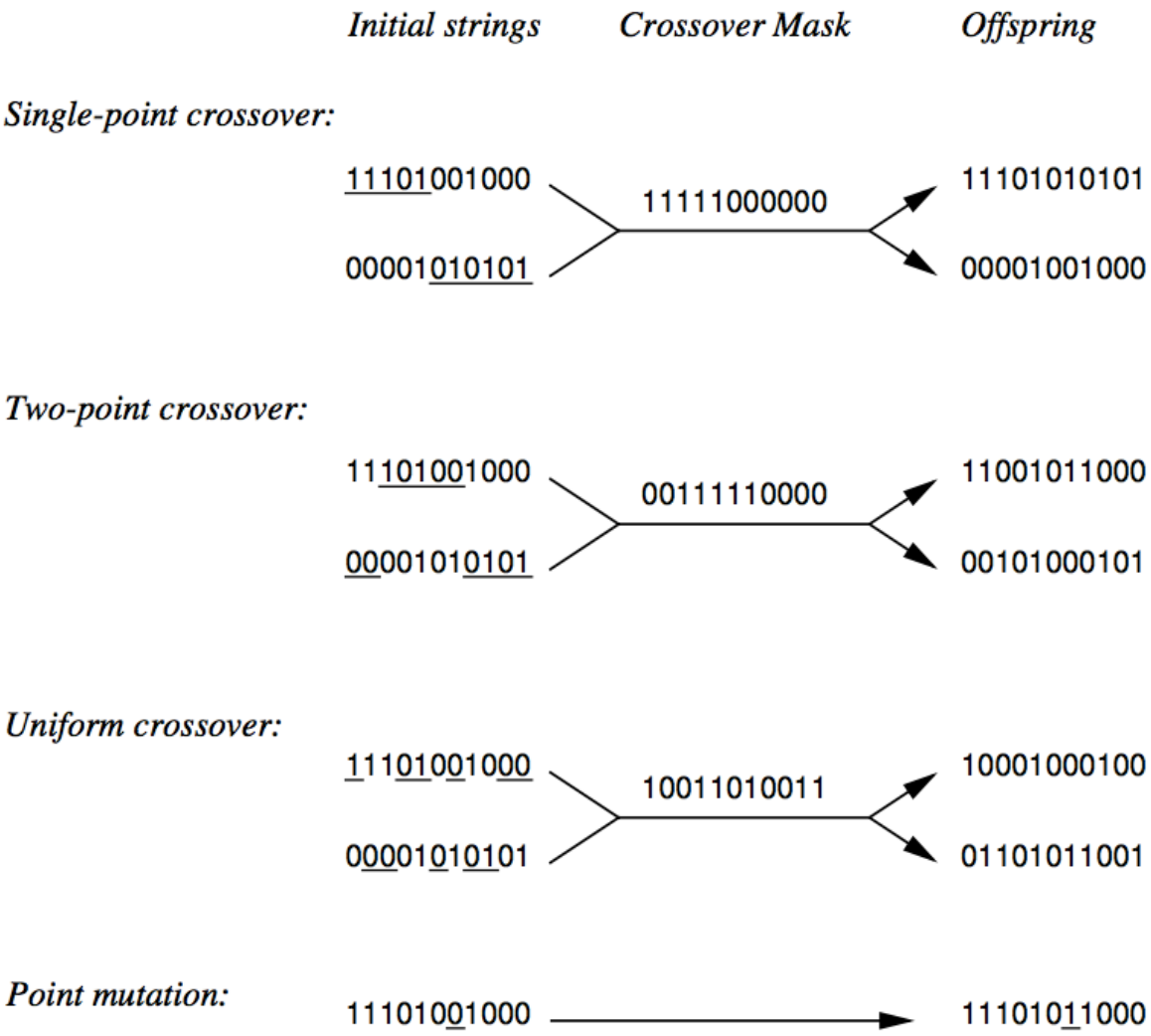
This operator is called the *Order1* crossover.

Mutation

Mutation involves reordering of the list:



Mutation


Initial strings
Crossover Mask
Offspring
Single-point crossover:
11101001000
00001010101
11111000000
11101010101
00001001000
Two-point crossover:
11101001000
00001010101
00111110000
11001011000
00101000101
Uniform crossover:
11101001000
00001010101
10011010011
10001000100
01101011001
Point mutation:
11101001000
11101011000

Genetic Algorithms

$t := 0;$
Compute initial population $\mathcal{B}_0$;

WHILE *stopping condition not fulfilled* **DO**
BEGIN
  select individuals for reproduction;
  create offsprings by crossing individuals;
  eventually mutate some individuals;
  compute new generation
END

* Bodenhofer