M.Tech DSE
Machine Learning

BITS Pilani
Pilani Campus

Dr. Monali Mavani

Source: "Introduction to Machine Learning, 2nd edition, Ethem Alpaydin , "Machine Learning", Tom M. Mitchell, Slides of Prof. Chetana from BITS Pilani and many others who made their course materials freely available online.

Course Introduction

- **Objective of course**
  - Introduction to the basic concepts and techniques of Machine Learning
  - Gain experience of doing independent study and research in the field of Machine Learning
  - Develop skills of using recent machine learning software tools to evaluate learning algorithms and model selection for solving practical problems

- **Evaluation scheme**
  - **Quiz (10% - 3 quiz: best 2 of 3)**
  - **2 Assignments (20%)**
  - **Mid-semester exam (30%)**
  - **Comprehensive exam (40%)**

- **Virtual labs**
  - http://elearn.bits-pilani.ac.in

What We’ll Cover in this Course

Adobe Acrobat
Document

Books

Text books and Reference book(s)

T1	Tom M. Mitchell: Machine Learning, The McGraw-Hill Companies
T2	Christopher M. Bhishop: Pattern Recognition & Machine Learning, Springer

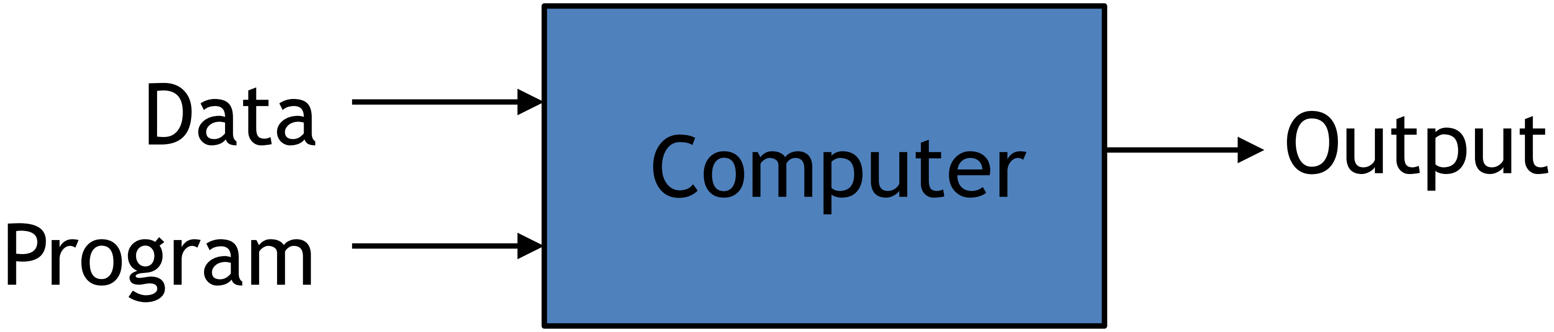
R1	**C.J.C. BURGES: A Tutorial on Support Vector Machines for Pattern Recognition, Kluwer Academic Publishers, Boston.**

Agenda

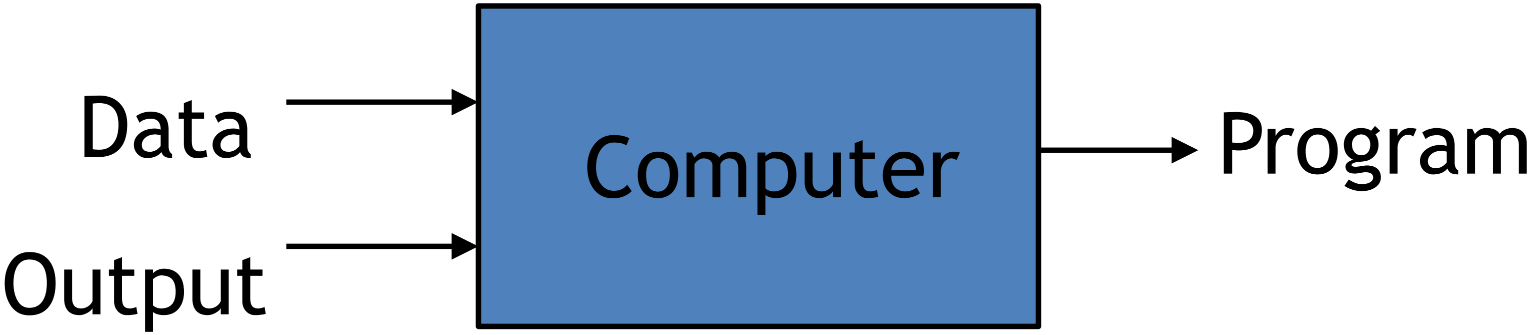
- What is Machine Learning?
- Application areas of Machine Learning
- Why Machine Learning is important?
- Design a Learning System
- Issues in Machine Learning
- Types of Machine Learning
- Supervised Machine Learning
- Model fit-selection-assessment

What is Machine Learning?

Traditional Programming



Machine Learning



Slide credit: Pedro Domingos

Machine Learning

- **Machine learning** is a scientific discipline that explores the construction and study of algorithms that can learn from data.
- Such algorithms operate by building a model based on inputs and using that to make predictions or decisions, rather than following only explicitly programmed instructions.

What is Machine Learning?

Definition by Tom Mitchell (1998):

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.” Example: playing checkers.

E = the experience of playing many games of checkers

T = the task of playing checkers.

P = the probability that the program will win the next game.

Example of Learning Problems

A Checker Learning Problem

- **Task T:** Playing Checkers
- **Performance Measure P:** Percent of games won against opponents
- **Training Experience E:** To be selected ==> Games Played against itself

A handwriting recognition learning problem

- **Task T:** recognizing and classifying handwritten words within images
- **Performance measure P**: percent of words correctly classified
- **Training Experience E:** a database of handwritten words with given classifications

A robot driving learning problem

- **Task T:** driving on public four-lane highways using vision sensors
- **Performance measure P:** average distance travelled before an error (as judged by human)
- **Training experience E:** a sequence of images and steering commands recorded while observing a human driver

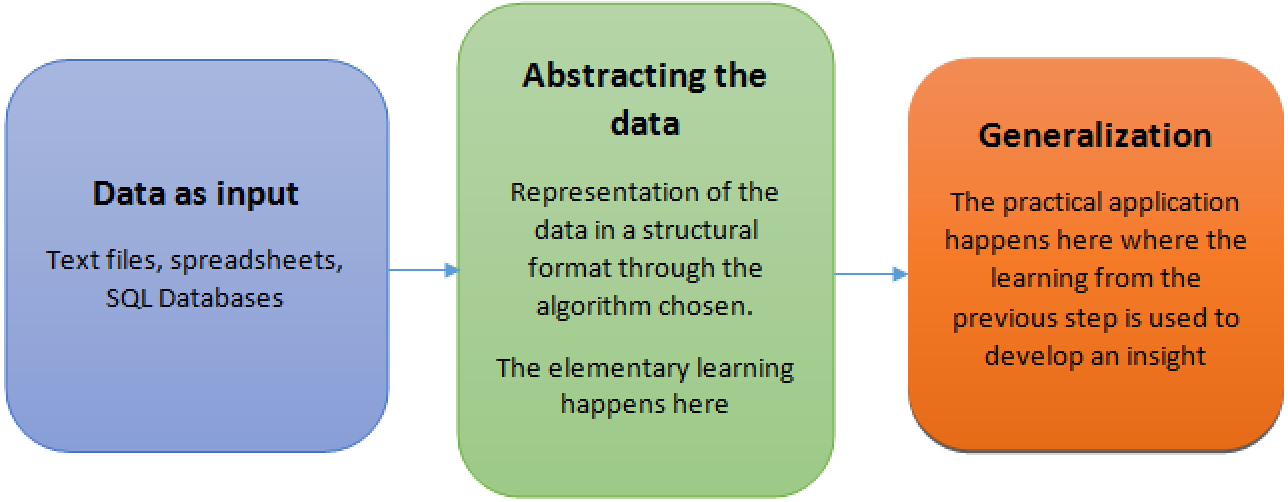
Why “Learn” ?

- Machine learning is programming computers to optimize a performance criterion using example data or past experience.
- There is no need to “learn” to calculate payroll
- Learning is used when:
  - Human expertise does not exist (navigating on Mars),
  - Humans are unable to explain their expertise (speech recognition)
  - Models must be customized (personalized medicine)
  - Models are based on huge amounts of data (genomics)

Problems not to be solved using ML

- Tasks in which humans are very effective
- Tasks in which frequent human intervention is needed
- Simple tasks which can be implemented using traditional programming paradigms
- Situations where training data is not sufficient

Machine Learning steps


Data as input
Text files, spreadsheets, SQL Databases
Abstracting the data
Representation of the data in a structural format through the algorithm chosen.
The elementary learning happens here
Generalization
The practical application happens here where the learning from the previous step is used to develop an insight

1.Collecting data

2.Preparing the data

3.**Training a model** : Representation of data in the form of the model. Build a model that is *a good and useful approximation* to the data

4.Evaluating the model

5.Improving the performance : Learning

Application Domains

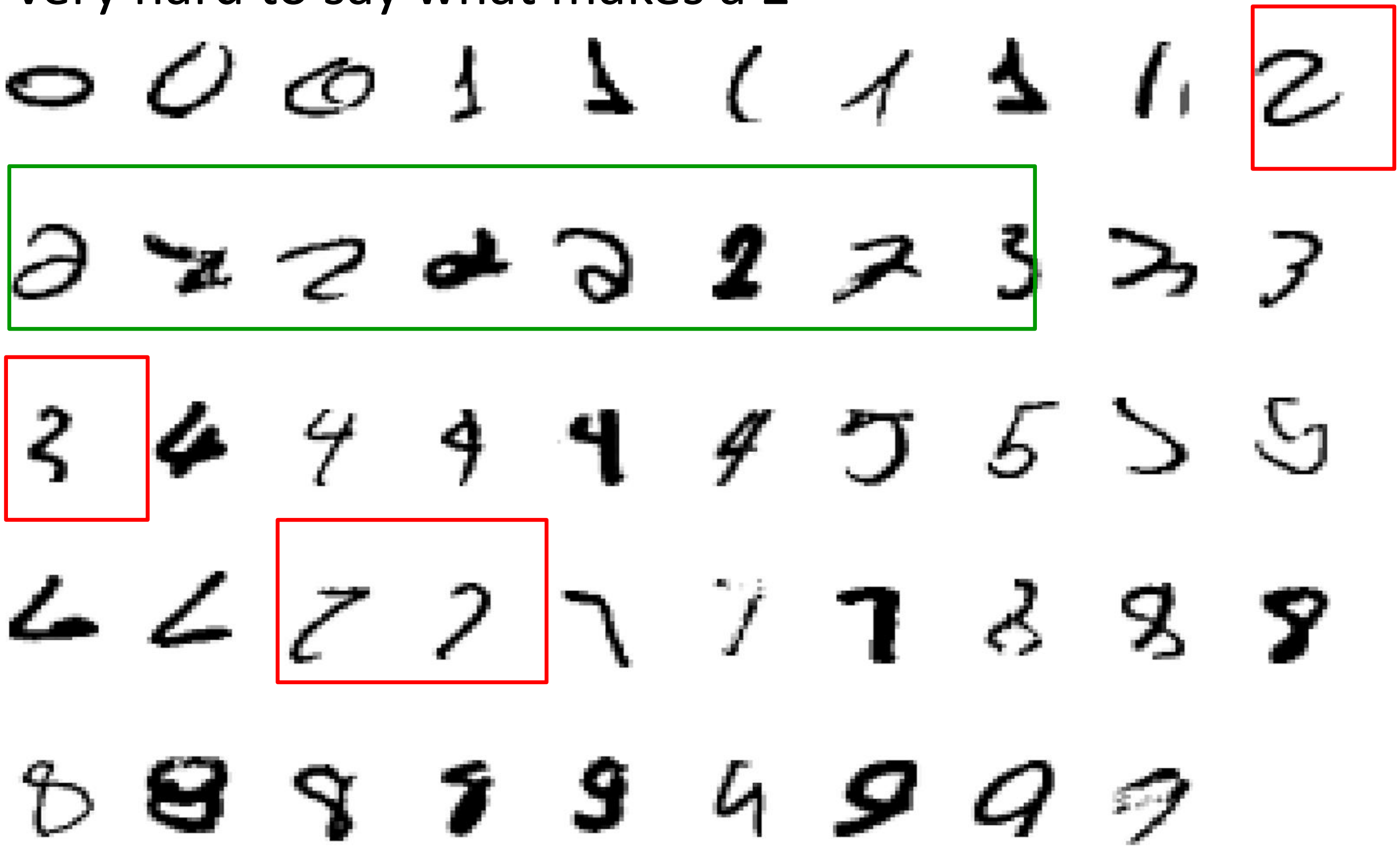
- Web search
- Computational biology
- Finance
- E-commerce
- Space exploration
- Robotics
- Information extraction
- Social networks
- Language Processing

Many more emerging…

Slide credit: Pedro Domingos

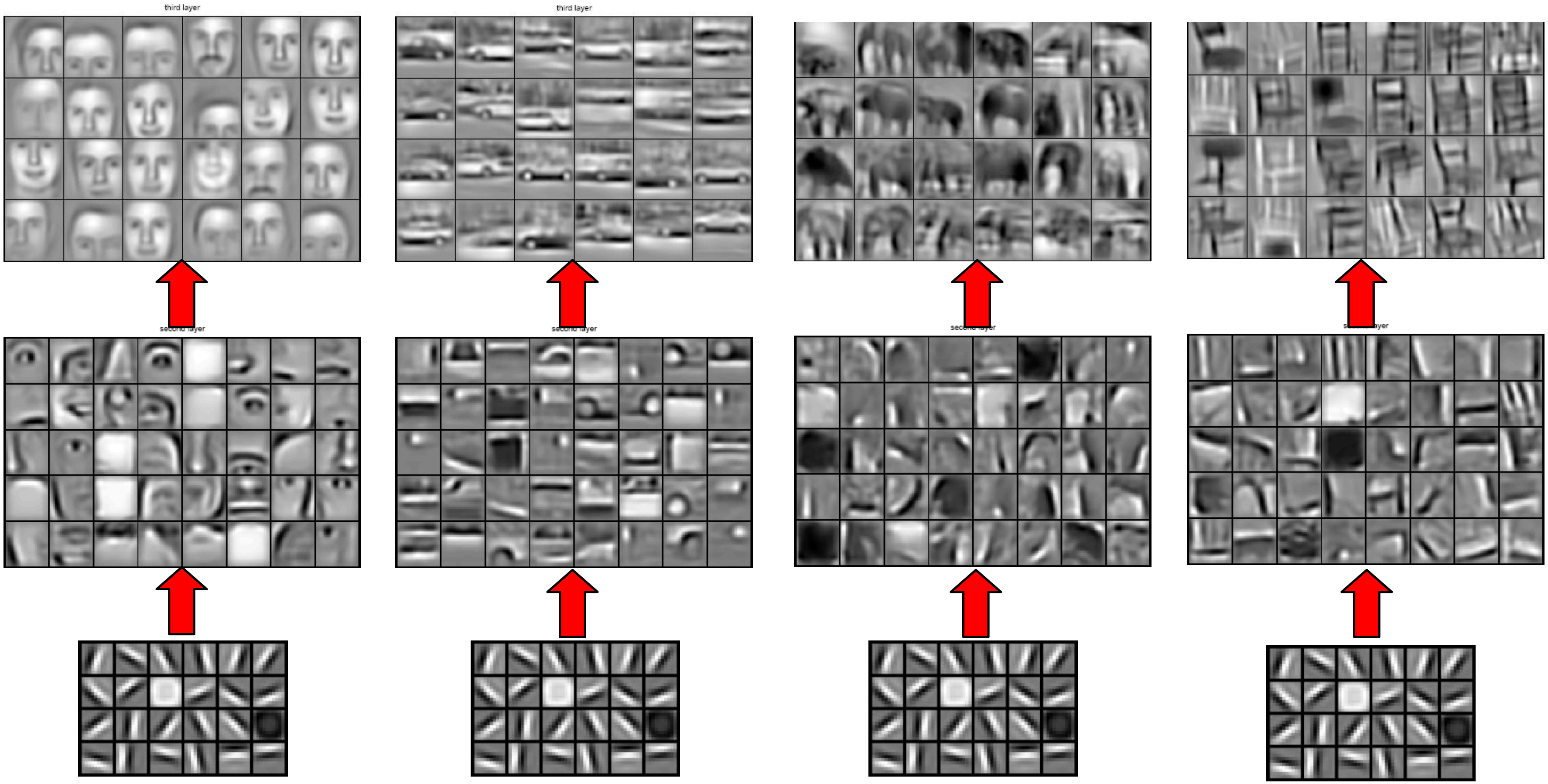
Pattern recognition

It is very hard to say what makes a 2



Slide credit: Geoffrey Hinton

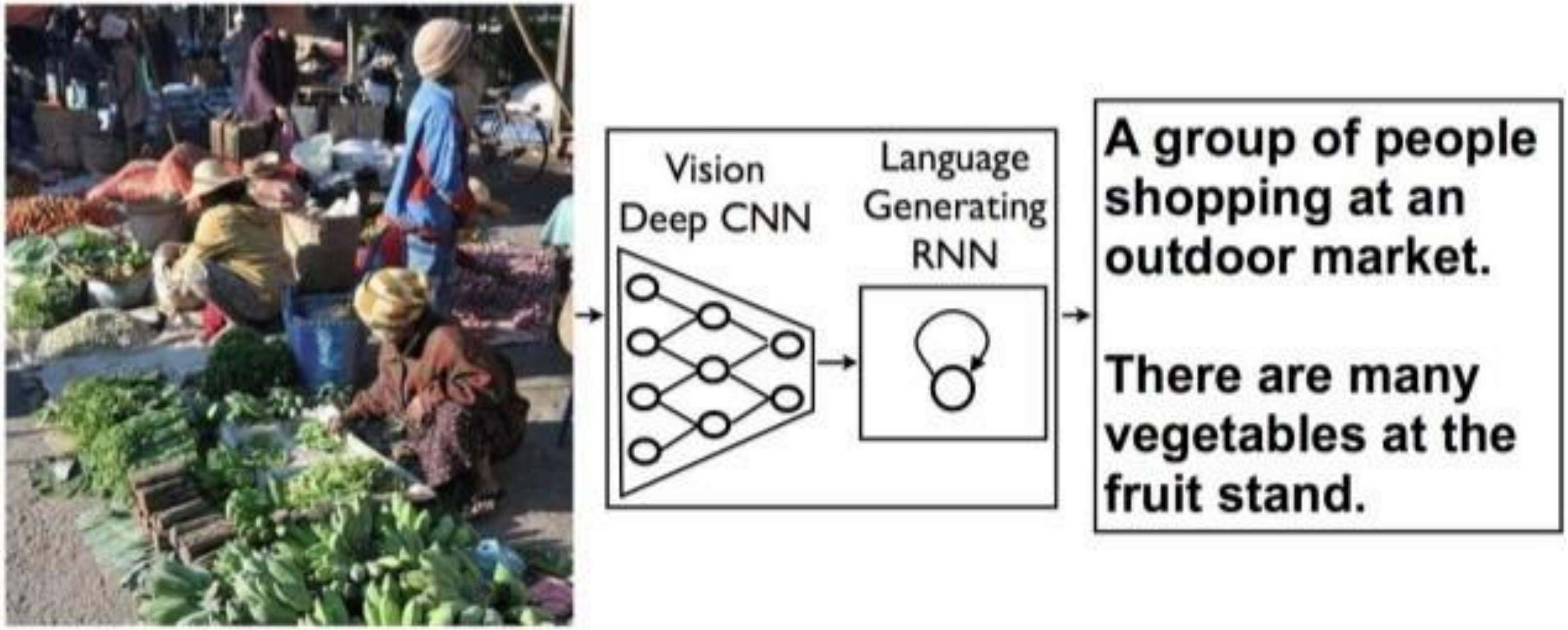
Learning of Object Parts



Slide credit: Andrew Ng

Image Captioning

ML algorithms can learn to generate captions for images



Vision Deep CNN

Language Generating RNN

A group of people shopping at an outdoor market.

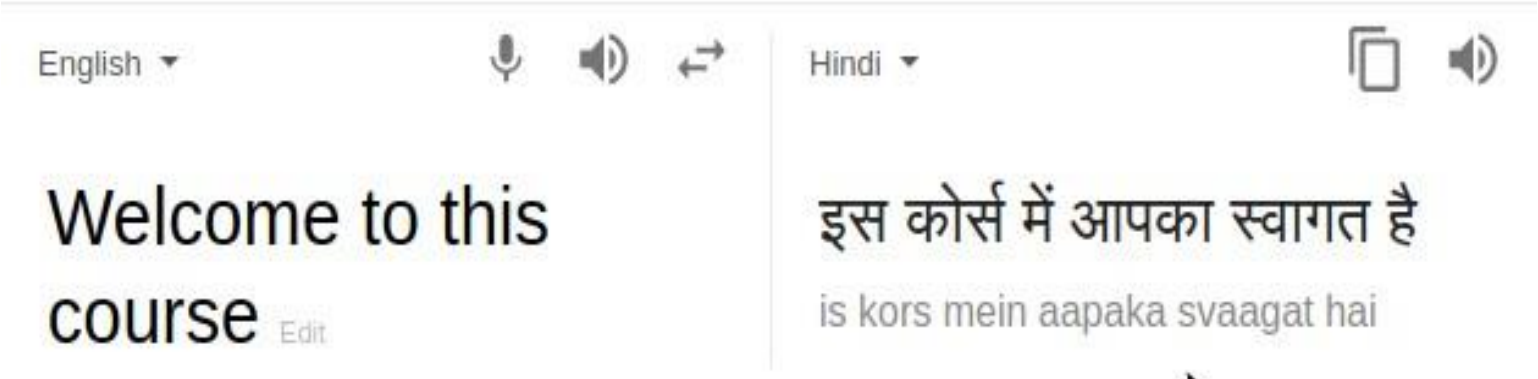
There are many vegetables at the fruit stand.

http://arxiv.org/abs/1411.4555 “Show and Tell: A Neural Image Caption Generator”

Slide Credit: Piyush Rai

Natural Language Processing

ML algorithms can learn to translate text



Welcome to this course

इस कोर्स में आपका स्वागत है

is kors mein aapaka svaagat hai

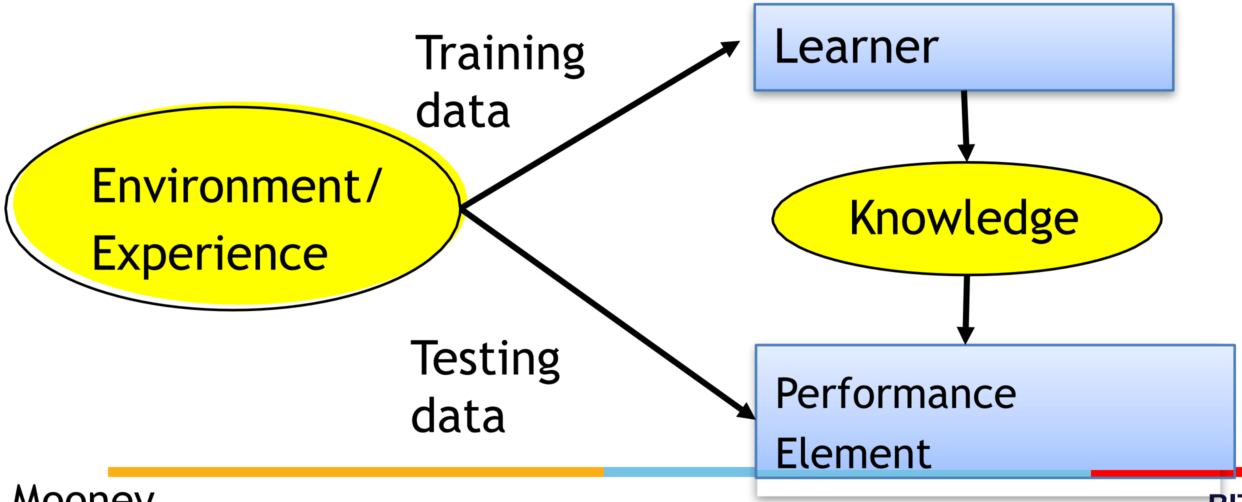
(even “transliterate”)

Slide Credit: Piyush Rai

ML as a Learning System

Designing a Learning System

- Choose the training experience(data)
- Choose exactly what is to be learned
  - i.e. the ***target function***
- Choose how to represent the target function
- Choose a learning algorithm to infer the target function from the experience



Slide Credit: Ray Mooney

Designing a Learning System: An Example

1. Problem Description (Ex: Playing checkers)

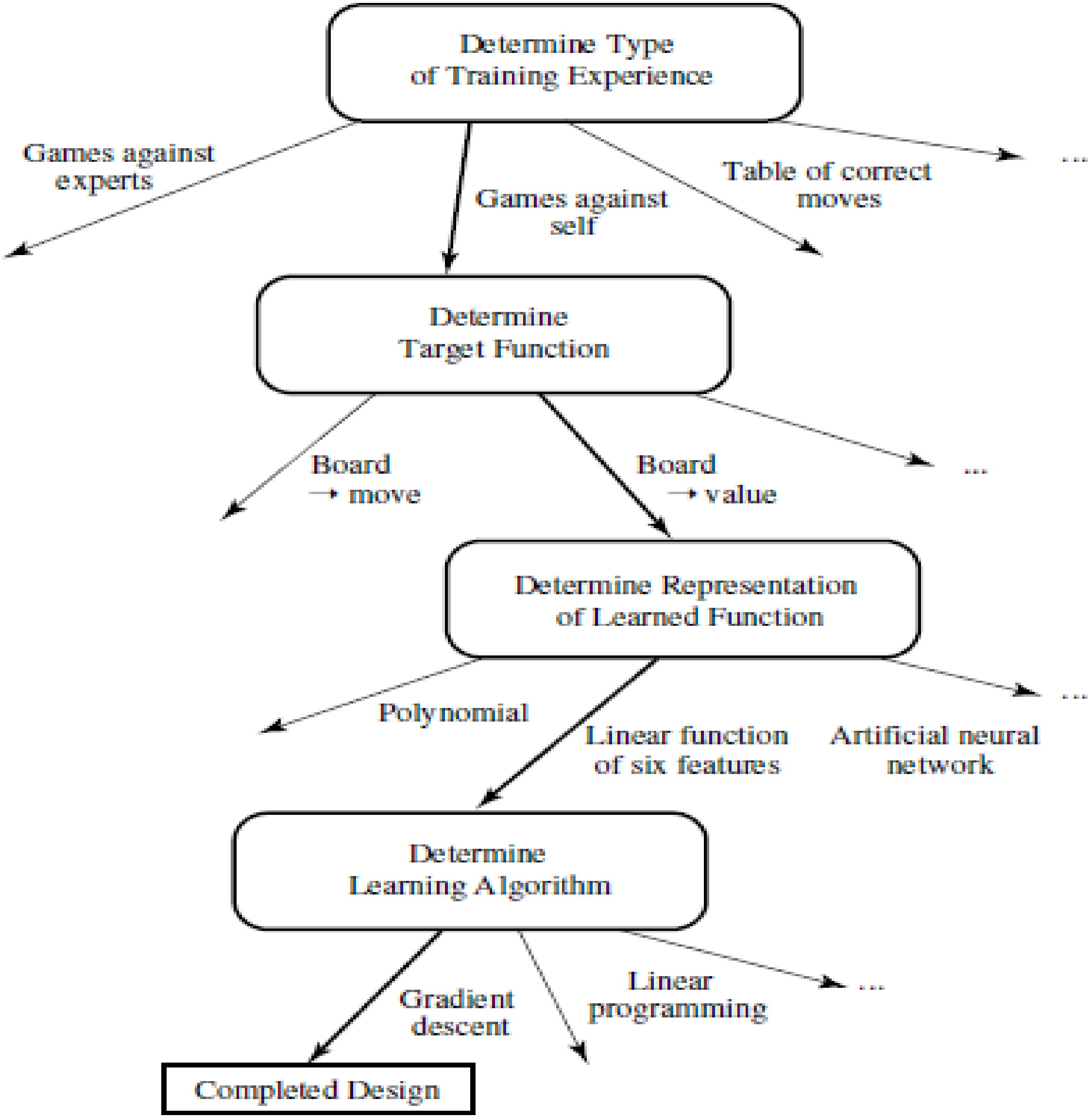
2. Choosing the Training Experience (data expressed as features)

3. Choosing the Target Function to be learnt (Ex: deciding next board position

4. Choosing a Representation for the Target Function (design a function as linear etc)

5. Choosing a Function Approximation Algorithm (parameters learning using loss function)

6. Final Design


Determine Type of Training Experience
Games against experts
Games against self
Table of correct moves
...
Determine Target Function
Board → move
Board → value
...
Determine Representation of Learned Function
Polynomial
Linear function of six features
Artificial neural network
...
Determine Learning Algorithm
Gradient descent
Linear programming
...
Completed Design

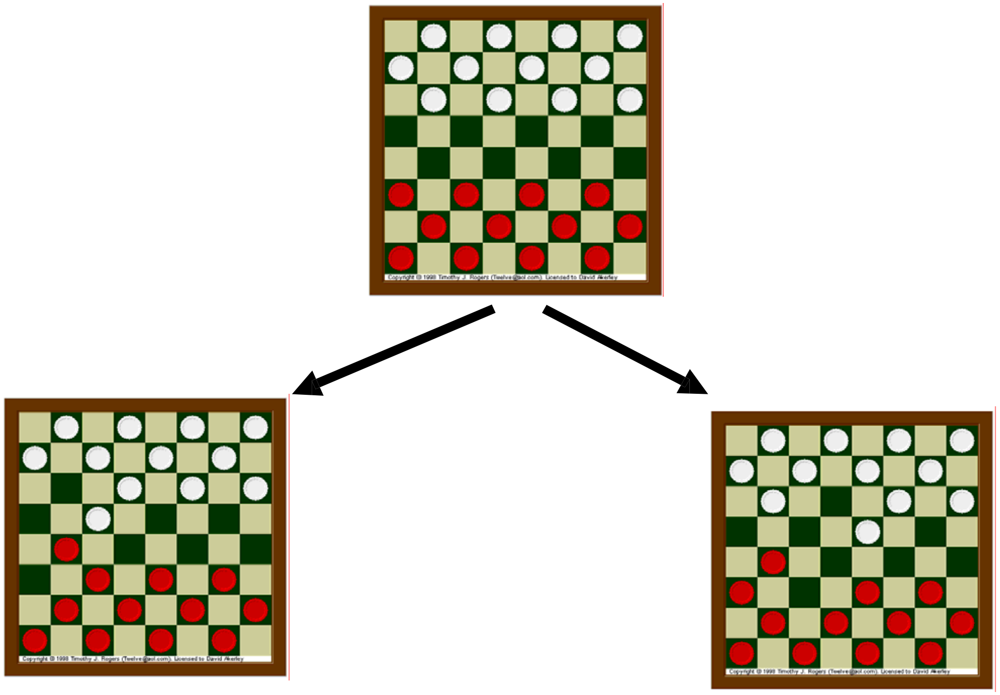
Choosing the Training Experience

The type of training experience **E** available to a system can have significant impact on success or failure of the learning system.

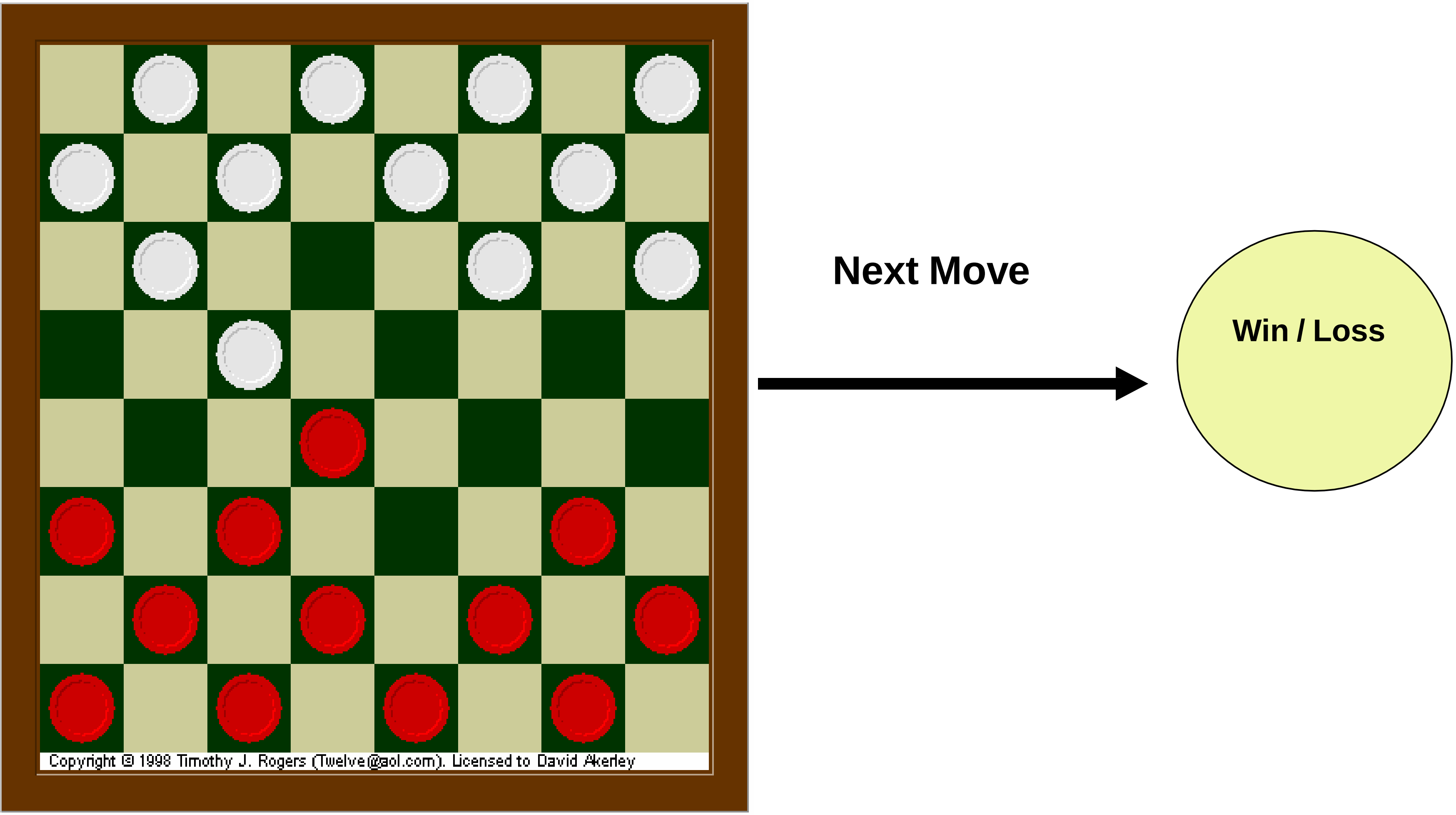
Choosing the Training Experience

One key attribute is whether the training experience provides **direct or indirect feedback regarding the choices made** by the performance system

Direct Learning



Indirect Learning


Copyright © 1998 Timothy J. Rogers (Twelve@aol.com). Licensed to David Akerley

Next Move

Win / Loss

Choosing the training experience

- In ***indirect training***, *information about the correctness of specific moves early in the game must be inferred indirectly from the fact that the game was eventually won or lost*.
- The learner faces an additional problem of **credit assignment**, or determining the degree to which each move in the sequence deserves credit or blame for the final outcome.
- *Credit assignment can be a particularly difficult problem because the game can be lost even when early moves are optimal, if these are followed later by poor moves*. Hence, learning from **direct training feedback is typically easier than learning from indirect feedback.**

Choosing the training experience -

How well the training experience represents the distribution of examples over which the final system performance P must be measured.

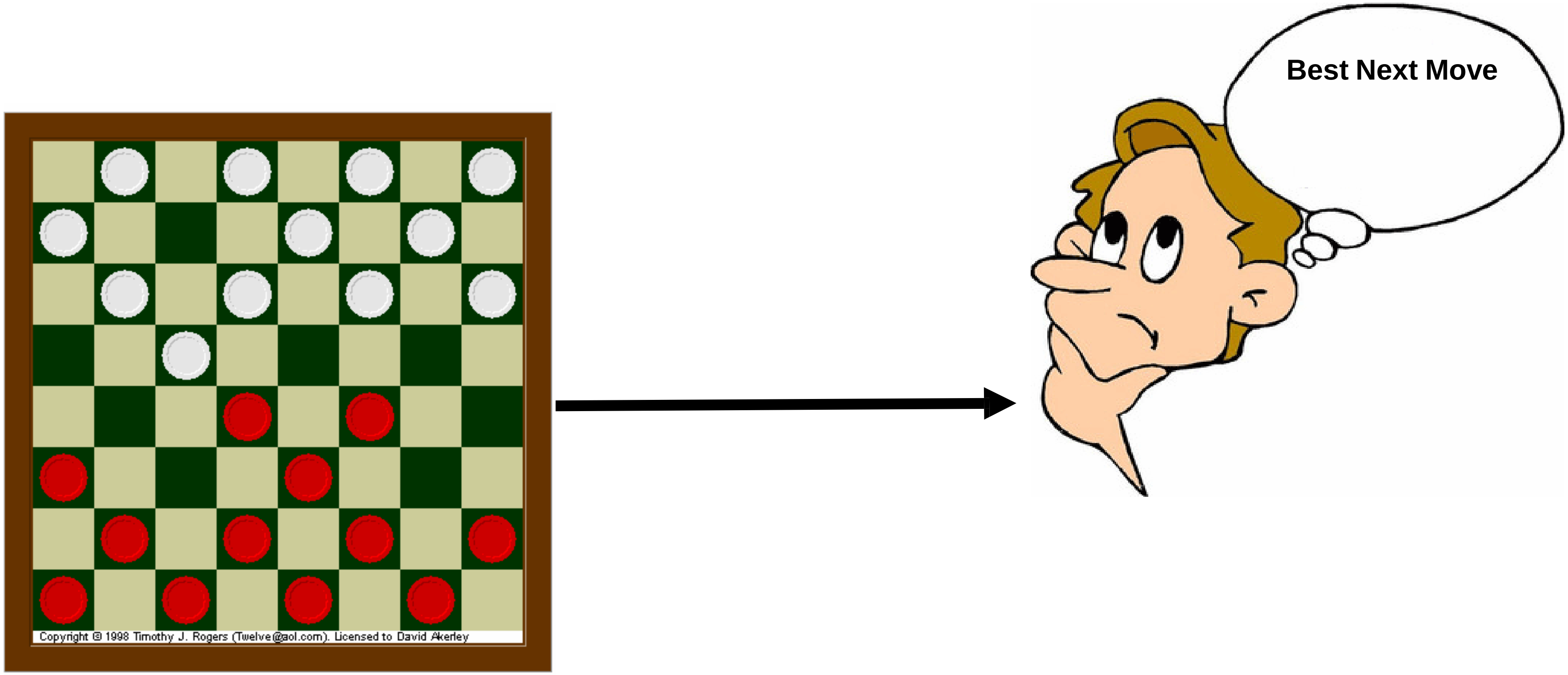
(Learning will be most reliable when the training example follow a distribution similar to that of future test examples)

E.G - the performance metric ***P*** is the percent of games the system wins in the world tournament. If its training experience ***E*** consists only of games played against itself, there is an obvious danger that this training experience might not be fully representative of the distribution of situations over which it will later be tested.

Assumptions

- Let us assume that our system will train by playing games against itself.
- And it is allowed to generate as much training data as time permits.

Main objective of the checker playing


Copyright © 1998 Timothy J. Rogers (Twelve@aol.com). Licensed to David Akerley
Best Next Move

Target Function

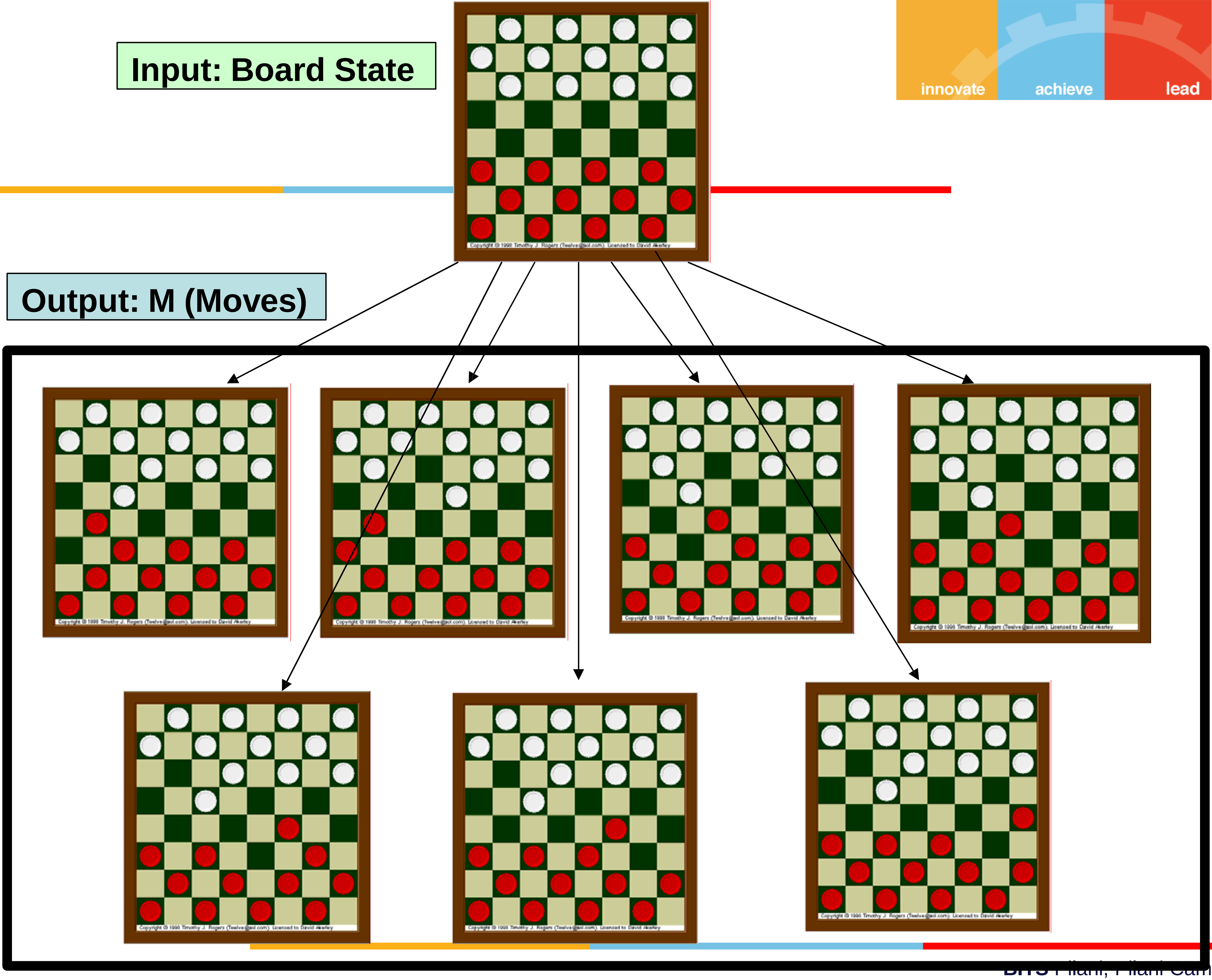
Choose Move: B → M

Choose Move is a function

where input **B** is the set of legal board states and produces **M** which is the set of legal moves

Input: Board State

Output: M (Moves)


Copyright © 1998 Timothy J. Rogers (Twelve@aol.com). Licensed to David Akerley

Target Function

Choose Move: B → M

M = *Choose Move* (B)

***ChooseMove*, is difficult to learn directly.**

Alternative Target Function

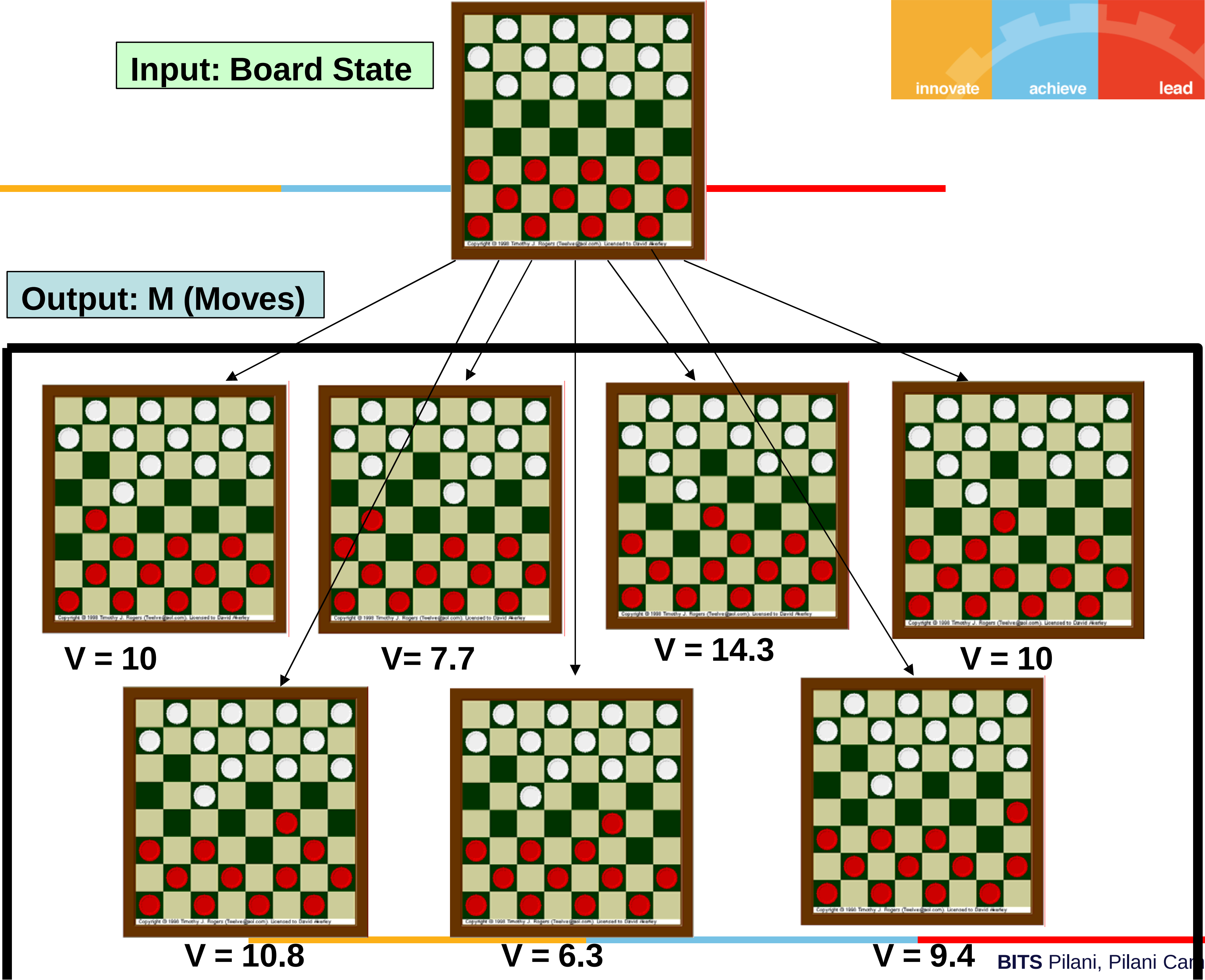
$$V : \mathrm{B} \rightarrow \mathrm{R}$$

V is the target function,
Input B is the board state and
Output R denotes a set of real number

An easier and related target function to learn is function V: B --> R, which assigns a numerical score to each board. The better the board positions B, the higher the score R.

Input: Board State

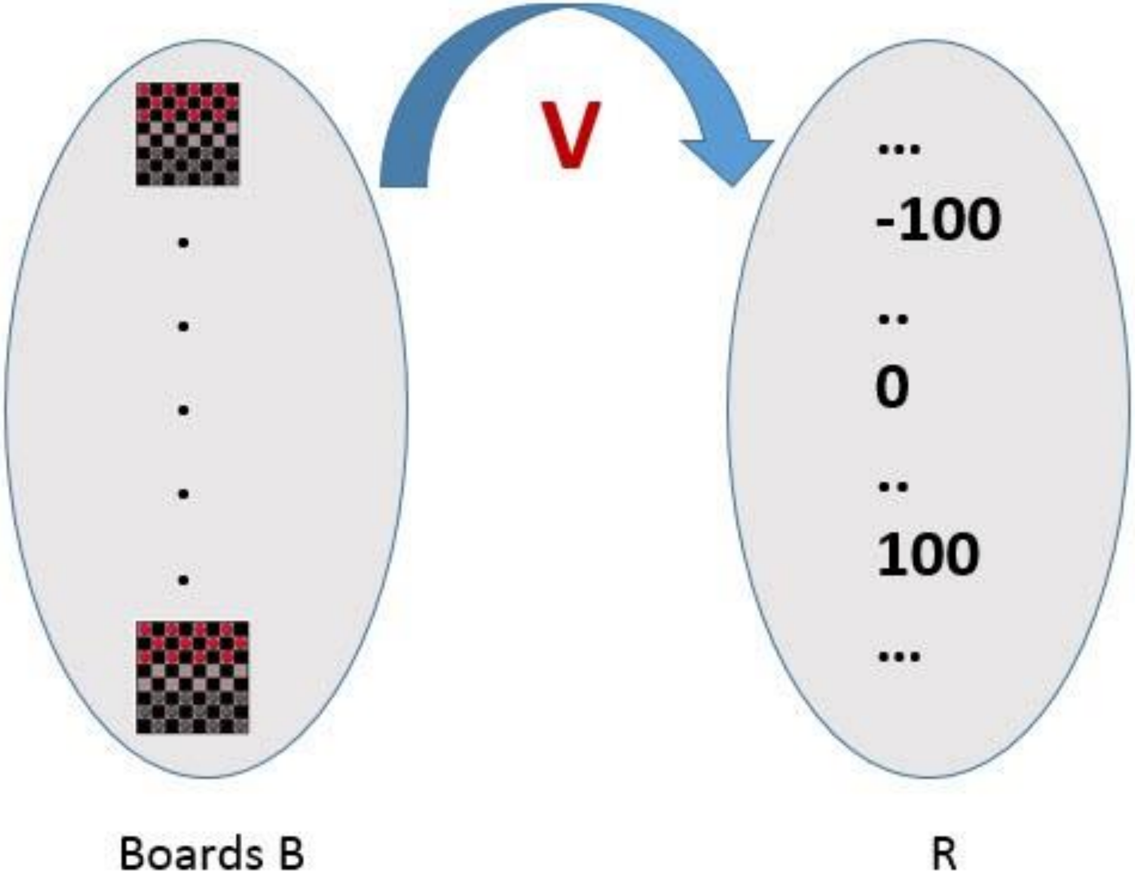
Output: M (Moves)



Choosing the Target Function

- Let us therefore define the target value ***V(b)*** for an arbitrary board state ***b*** in ***B***, as follows:
  1. If ***b*** is a final board state that is won, then ***V(b) = 100***
  2. If ***b*** is a final board state that is lost, then ***V(b) = -100***
  3. If ***b*** is a final board state that is draw, then ***V(b) = 0***
  4. If ***b*** is a not a final state in the game, then ***V(b) = V(b')***, where ***b'*** is the best final board state that can be achieved starting from ***b*** and playing optimally until the end of the game.

Choosing the Target Function



Choosing the Target Function

- Reduced the learning task to the problem of discovering an operational description of the ideal target function ***V*** – it is difficult to learn ***V*** perfectly.
- We often expect learning algorithms to acquire only some ***approximation*** to the target function -- is called ***function approximation.*** [The actual function can often not be learned and must be approximated]

Choosing a Representation for the Target Function

- **<u>Expressiveness versus Training set size</u>**
  - More expressive the representation of the target function, the closer to the "truth" we can get.
  - More expressive the representation, the more training examples are necessary to choose among the large number of "representable" possibilities.

- **<u>Example of a representation</u>:**
  - x1 = # of red pieces on the board, x2 = # of black pieces on the board
  - X3= # of black king on the board
  - x4 = # of red king on the board
  - X5 = # of black pieces threatened by black
  - x6 = # of red pieces threatened by black

$$\boxed{\hat{V}(b) = w0+w1.x1+w2.x2+w3.x3+w4.x4+w5.x5+w6.x6}$$

wi's are adjustable or "learnable" coefficients

Choosing a Representation for the Target Function

- Task T: playing checkers
- Performance measure ***P: %*** of games won in the world tournament
- Training experience E: games played against itself
- Target function: V : Board → R
- Target function representation

$$V'(b) = w0 + w1x1 + w2x2 + w3x3 + w4x4 + w5x5 + w6x6$$

The problem of learning a checkers strategy reduces to the problem of learning values for the coefficients *w0* through *w6* in the target function representation

Choosing a Function Approximation Algorithm

- **Generating Training Examples of the form** **$<b, V_{train}(b)>$** [e.g. <x1=3, x2=0, x3=1, x4=0, x5=0, x6=0, +100 (=blacks won)]
  - x1= # of black pieces on the board
  - x2 = # of red pieces on the board
  - x3 = # of black king on the board
  - x4 = # of red king on the board
  - x5 = # of black pieces threatened by red
  - x6 = # of red pieces threatened by black

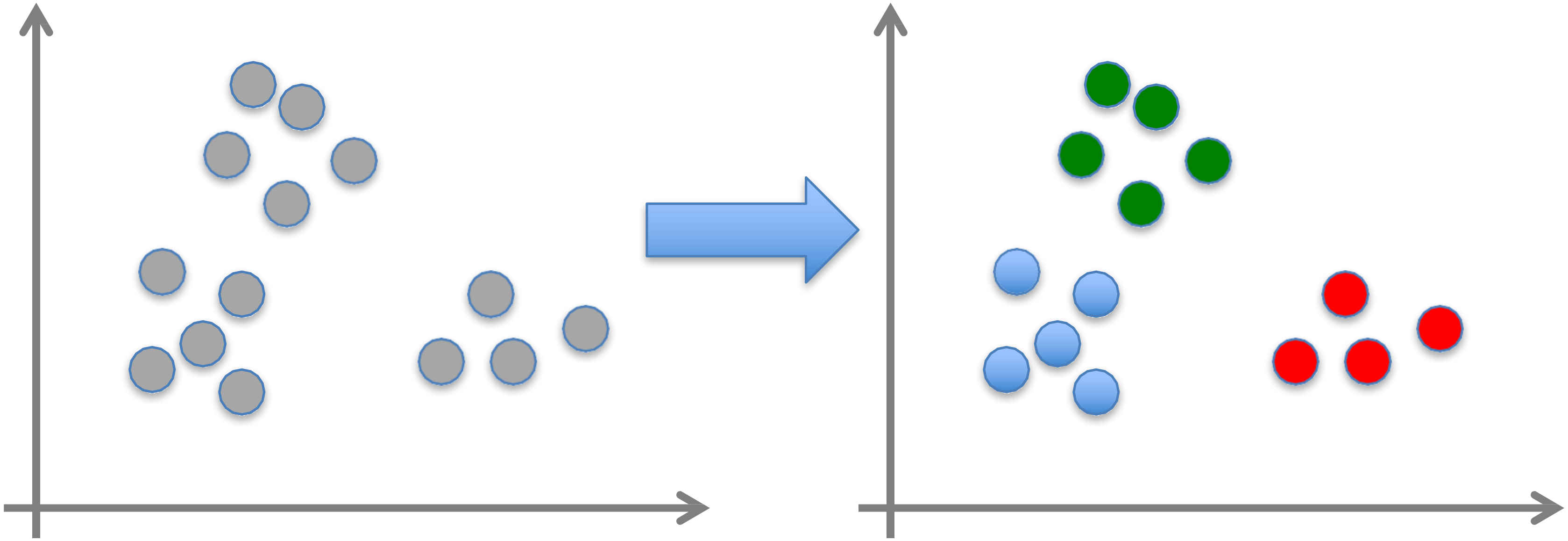
Types of Learning

- Supervised (inductive) learning
  - Given: training data, desired outputs (labels)
- Unsupervised learning
  - Given: training data (without desired outputs)
- Semi-supervised learning
  - Given: training data + a few desired outputs
- Reinforcement learning
  - Given: rewards from sequence of actions

Slide Credit: Eric Eaton

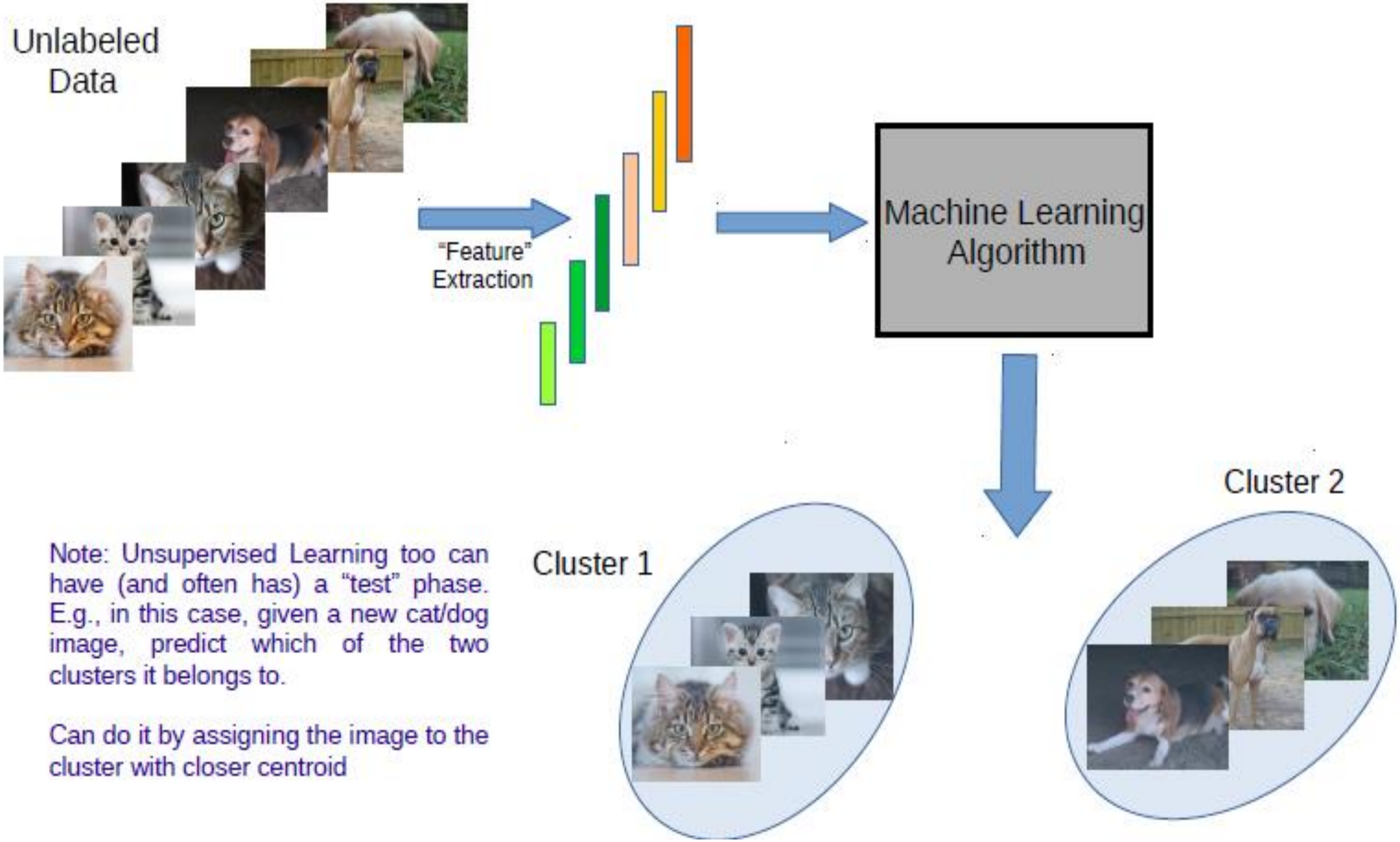
Unsupervised Learning

- Given x_1, x_2, …, x_n (without labels)
- Output hidden structure behind the x's
  - E.g., clustering



Slide Credit: Eric Eaton

A Typical Unupervised Learning Workflow (for Clustering)



Note: Unsupervised Learning too can have (and often has) a "test" phase. E.g., in this case, given a new cat/dog image, predict which of the two clusters it belongs to.

Can do it by assigning the image to the cluster with closer centroid

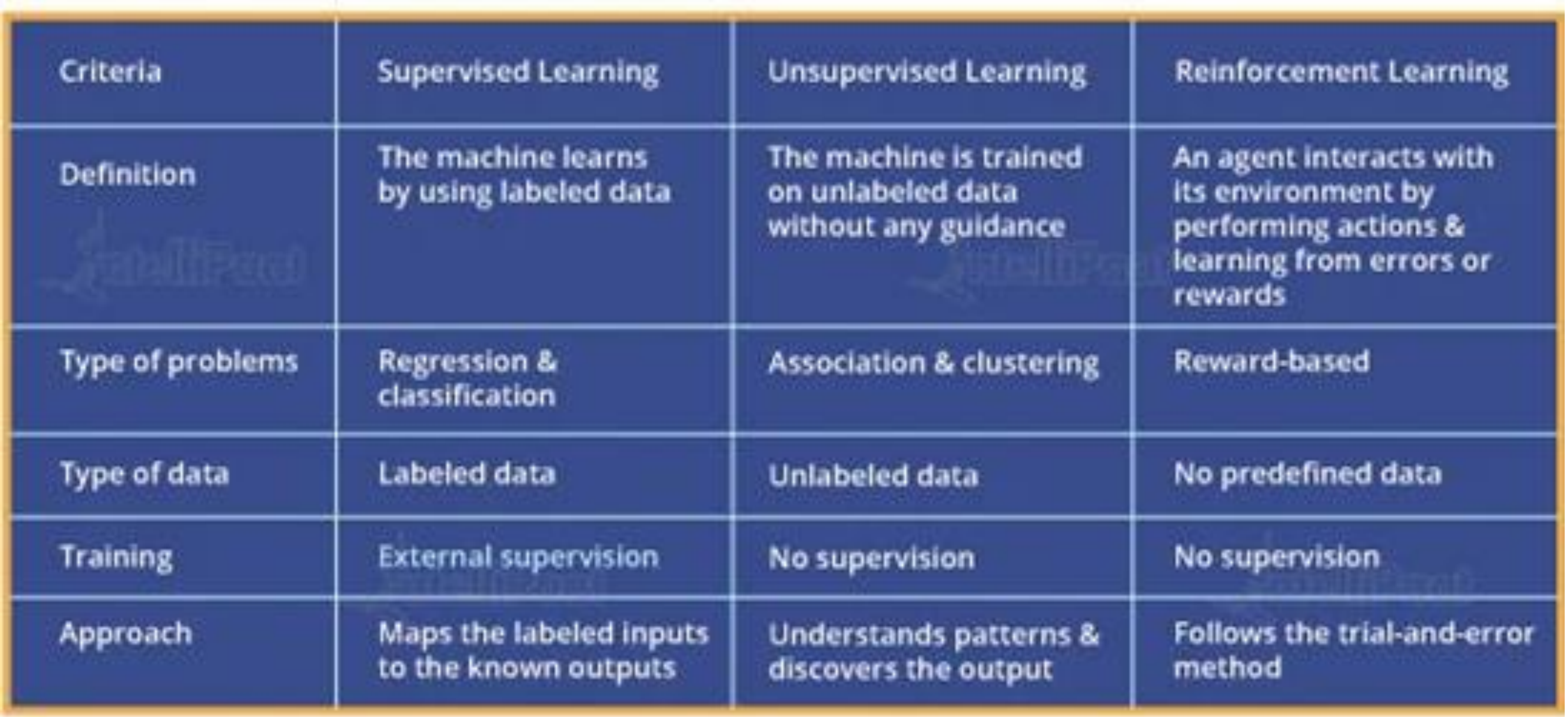
Slide Credit: Piyush Rai

Reinforcement Learning

- No predefined data
- Allow an agent to take actions and interact with an environment so as to maximize the total rewards.
- Examples:
  - Resources management in computer clusters
  - Game playing
  - Robot in a maze

Slide Credit: Eric Eaton

Supervised, Unsupervised and Reinforcement Learning Comparison



Criteria	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Definition	The machine learns by using labeled data	The machine is trained on unlabeled data without any guidance	An agent interacts with its environment by performing actions & learning from errors or rewards
Type of problems	Regression & classification	Association & clustering	Reward-based
Type of data	Labeled data	Unlabeled data	No predefined data
Training	External supervision	No supervision	No supervision
Approach	Maps the labeled inputs to the known outputs	Understands patterns & discovers the output	Follows the trial-and-error method

Supervised (inductive) learning

Terminology

- **Training example.** An example of the form $\langle \mathbf{x}, f(\mathbf{x}) \rangle$.
- **Target function (target concept).** The true function f.
- **Hypothesis**. A proposed function h believed to be similar to f.
- **Concept**. A boolean function. Examples for which $f(\mathbf{x}) = 1$ are called **positive examples** or **positive instances** of the concept. Examples for which $f(\mathbf{x}) = 0$ are called **negative examples** or **negative instances.**
- **Classifier**. A discrete-valued function. The possible values $f(\mathbf{x}) \in \{1, \ldots, K\}$ are called the **classes** or **class labels**.
- **Hypothesis Space**. The space of all hypotheses that can, in principle, be output by a learning algorithm.
- **Version Space**. The space of all hypotheses in the hypothesis space that have not yet been ruled out by a training example.

Inductive Learning Hypothesis

Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples

Inductive Learning

- **Inductive learning** or "**Prediction**":
  - **Given** examples of a function *(X, F(X))*
  - **Predict** function *F(X)* for new examples *X*
- Classification

  F(X) = Discrete
- Regression

  F(X) = Continuous
- Probability estimation

  F(X) = Probability*(X):*

Learning a Class from Examples

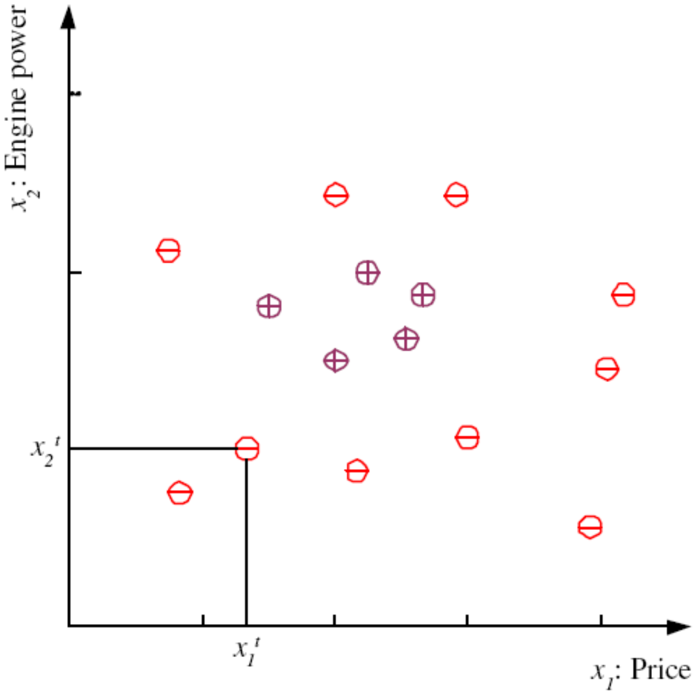
- Class C of a “family car”
  - Prediction: Is car *x* a family car?
  - Knowledge extraction: What do people expect from a family car?
- Output:

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

  x_1: price, x_2 : engine power

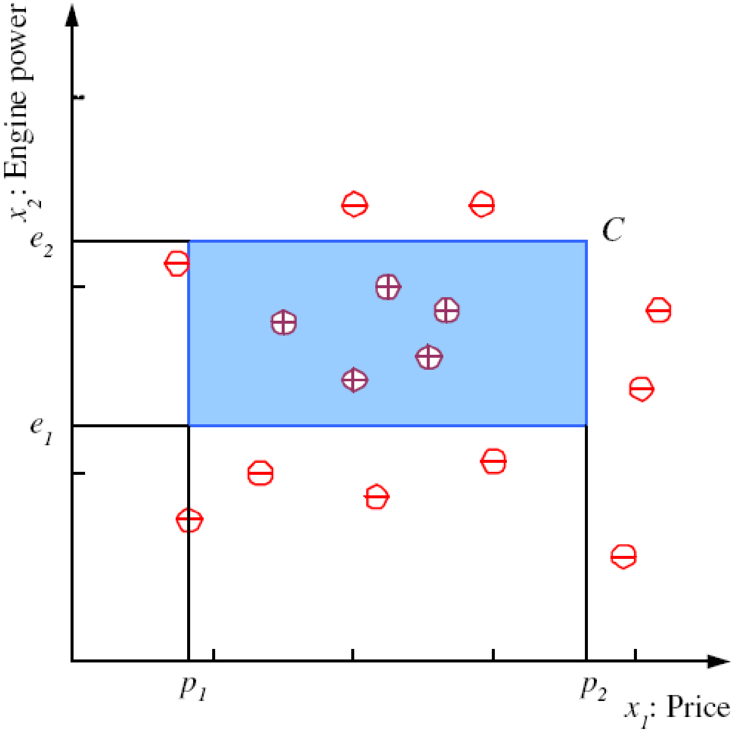
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Training set $\mathcal{X}$

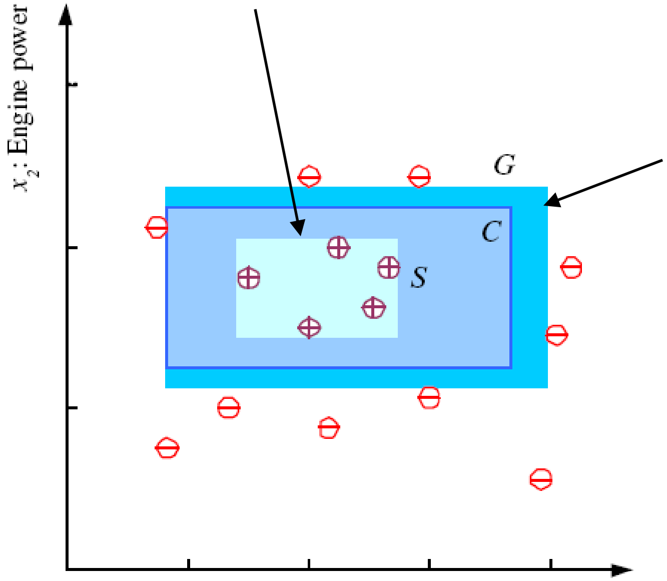


Class C

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



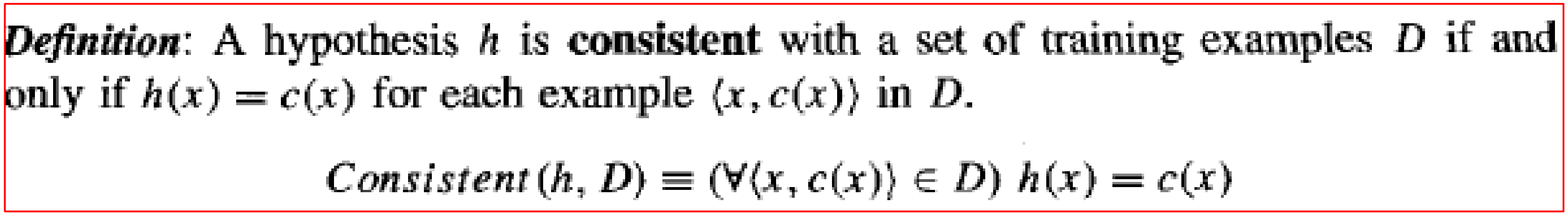
S, G, and the Version Space



most specific hypothesis, *S*

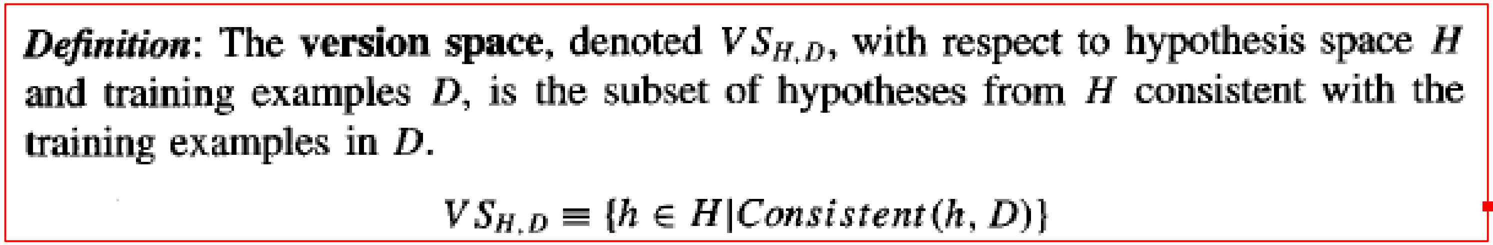
most general hypothesis, *G*

- the hypothesis class H is the set of all possible rectangles.
- *h* ∈ H, between *S* and *G* is consistent and make up the version space



***Definition*:** A hypothesis h is **consistent** with a set of training examples D if and only if $h(x) = c(x)$ for each example $\langle x, c(x)\rangle$ in D.

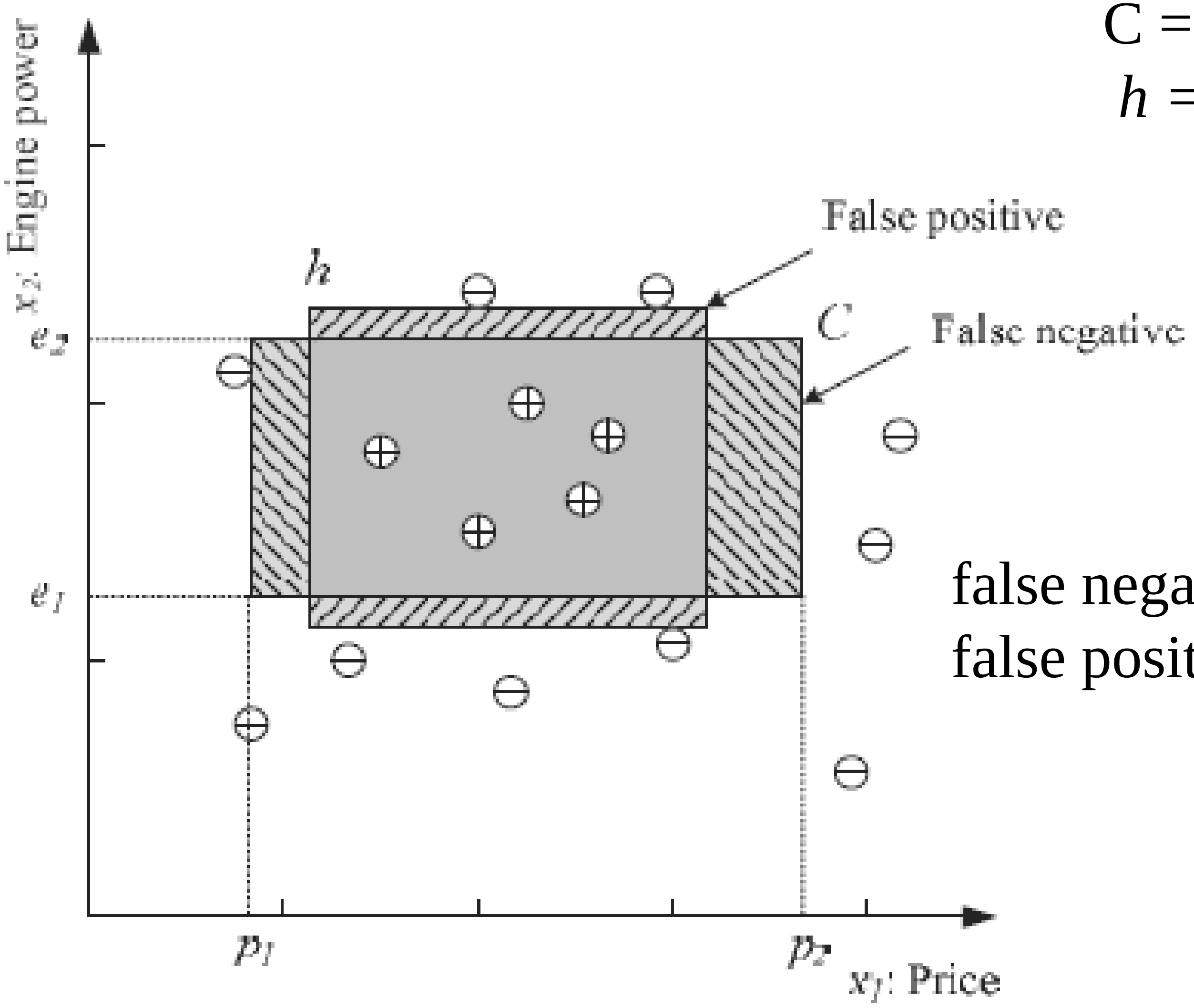
$$Consistent(h, D) \equiv (\forall \langle x, c(x)\rangle \in D)\ h(x) = c(x)$$



***Definition*:** The **version space**, denoted $VS_{H,D}$, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with the training examples in D.

$$VS_{H,D} \equiv \{h \in H | Consistent(h, D)\}$$

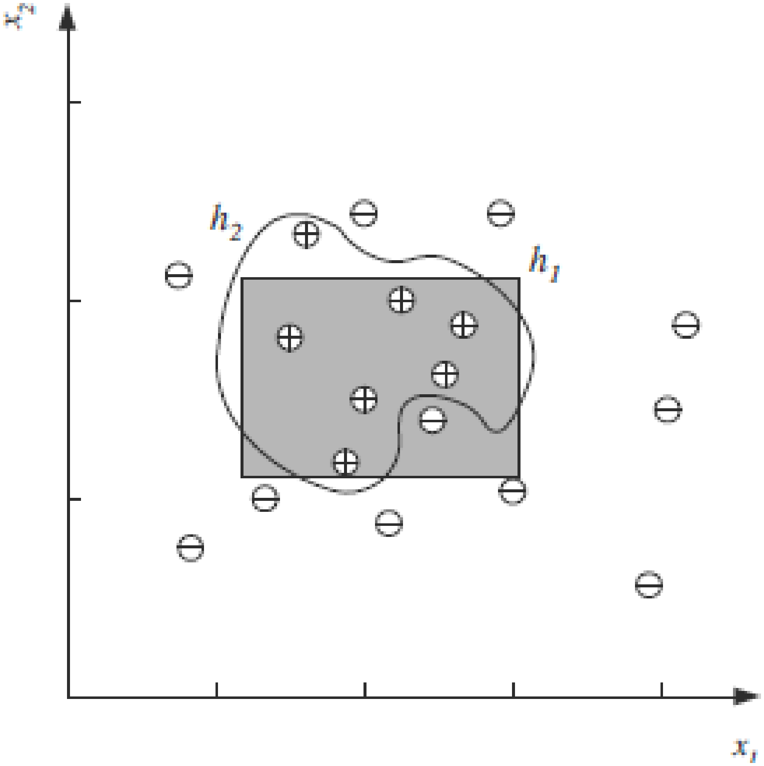
Empirical Error



C = actual class
h = induced hypothesis.

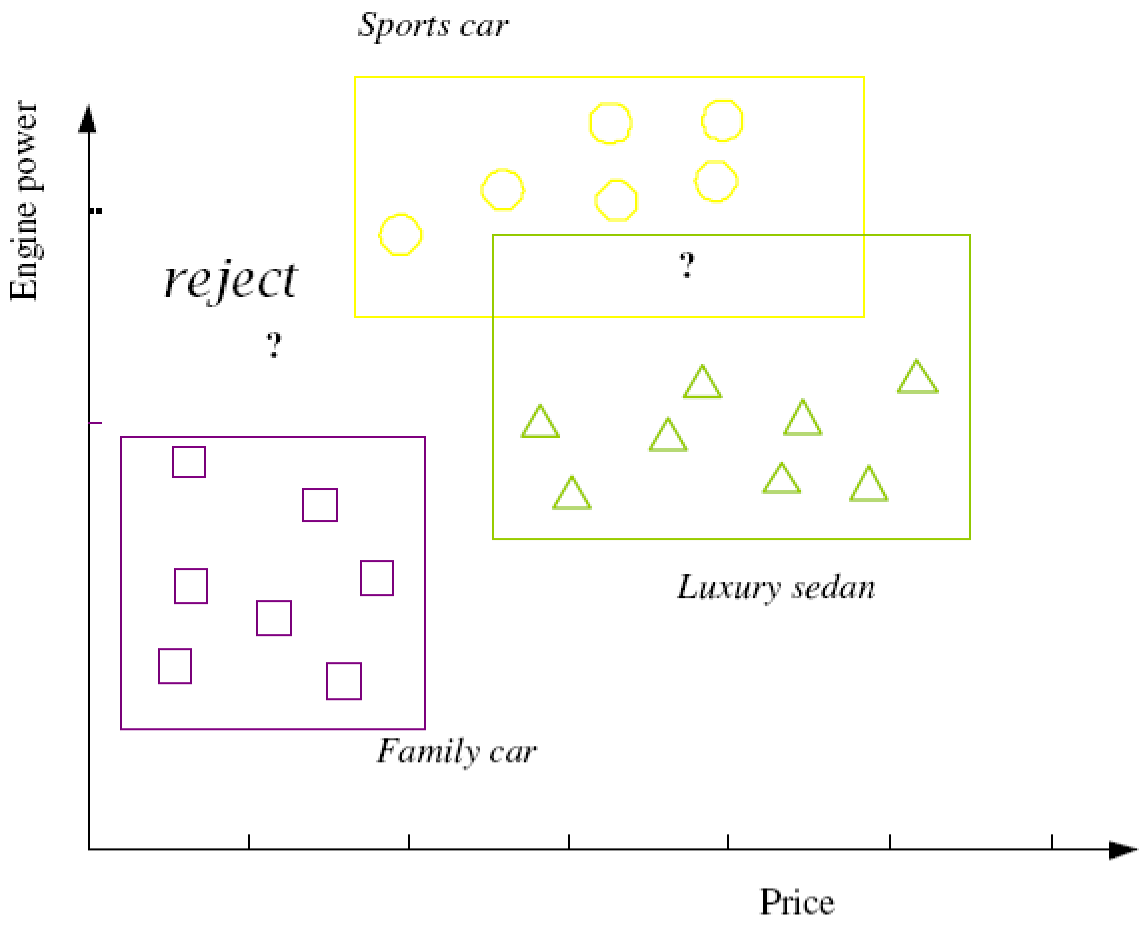
false negative =>C is 1 but *h* is 0
false positive => C is 0 but *h* is 1

Noise



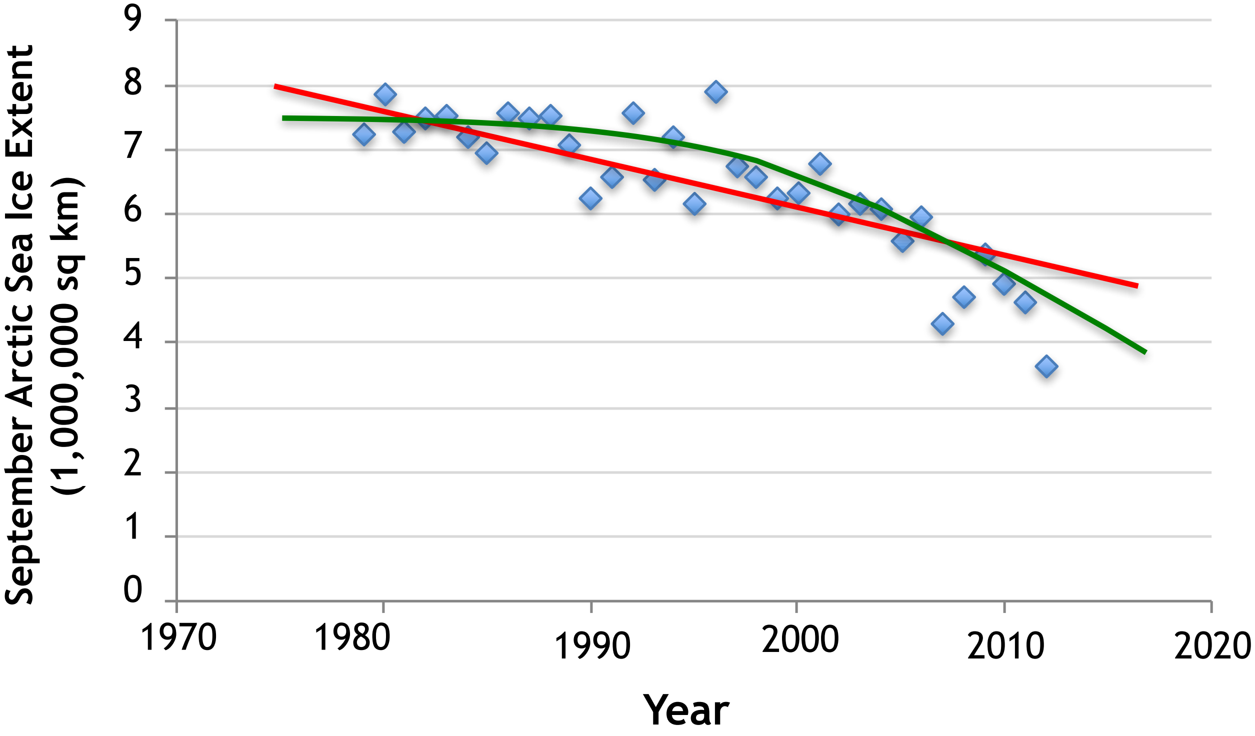
- Imprecision in recording the input attributes
- Errors in labeling the data points
- Additional hidden/latent attributes that affect the label of an instance

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



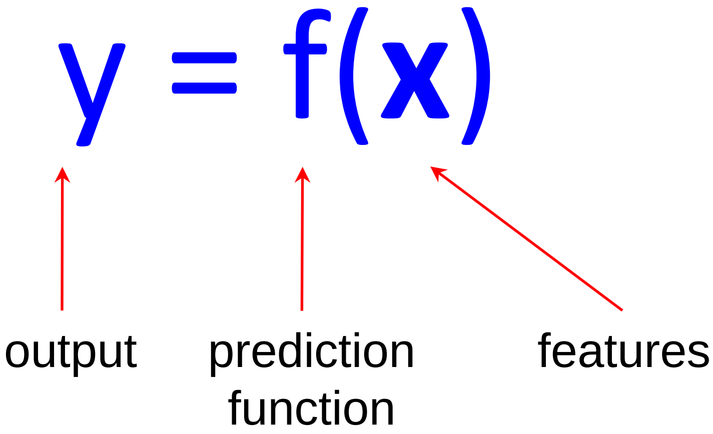
Supervised Learning: Regression

- Given $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Learn a function $f(x)$ to predict y given x



Data from G. Witt. Journal of Statistics Education, Volume 21, Number 1 (2013) Slide Credit: Eric Eaton

Regression



$$y = f(\mathbf{x})$$

output

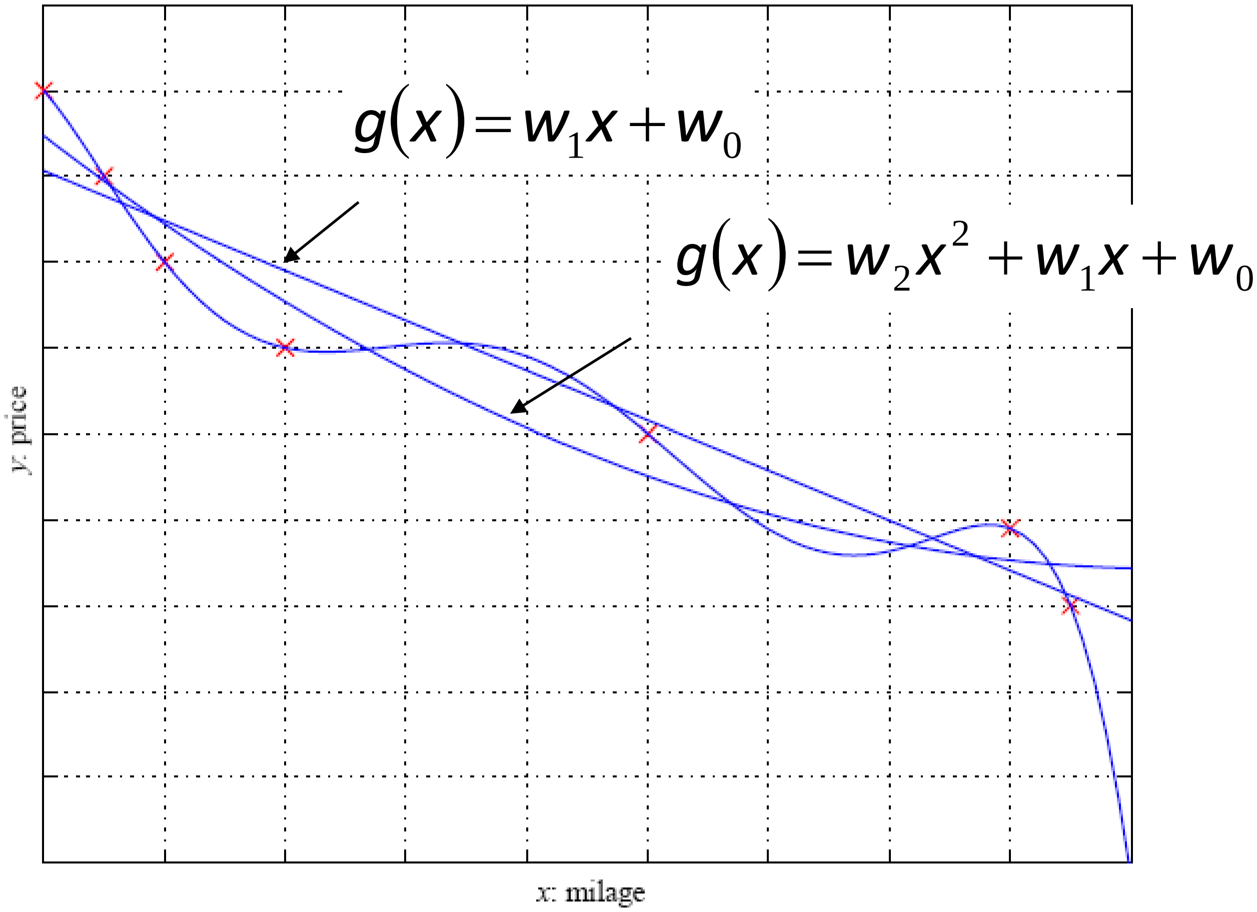
prediction function

features

- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

Slide credit: L. Lazebnik

Regression



$$g(x) = w_1 x + w_0$$

$$g(x) = w_2 x^2 + w_1 x + w_0$$

Inductive Bias

- Learning is an ill-posed problem; data is not sufficient to find a unique solution
- Need to make assumptions
  - – Experience alone doesn't allow us to make conclusions about unseen data instances
- Inductive bias
  - The set of assumptions we make to have learning possible is called the inductive bias of the learning algorithm
  - The need for inductive bias, assumptions about $\mathcal{H}$
- Generalization—that is, how well our hypothesis will correctly classify future examples that are not part of the training set.

Training vs Testing

- What do we want?
  - High accuracy on training data?
  - No, high accuracy on *unseen/new/test data!*
  - Why is this tricky?
- Training data
  - Features (x) and labels (y) used to learn mapping f
- Test data
  - Features used to make a prediction
  - Labels only used to see how well we've learned f!!!
- Validation data
  - Held-out set of the *training data*
  - Can use both features and labels to tune *parameters* of the model we're learning

Train-test error

- Target variable Y , a vector of inputs X, and prediction model f(x)-hat that has been estimated from a training set T. The loss function for measuring errors between Y and f(X)-hat is L(Y, f(X)-hat). Typical choices are

$$L(Y, \hat{f}(X)) = \begin{cases} (Y - \hat{f}(X))^2 & \text{squared error} \\ |Y - \hat{f}(X)| & \text{absolute error.} \end{cases}$$

- Training error is the average loss over the training sample

$$\overline{\text{err}} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$

- Test error, also referred to as generalization error, is the prediction error over an independent test sample. the training set T is fixed, and test error refers to the error for this specific training set

$$\text{Err}_{\mathcal{T}} = \text{E}[L(Y, \hat{f}(X)) | \mathcal{T}]$$

Model selection and assessment

- Model selection: estimating the performance of different models in order to choose the best one.
- Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data.
- To estimate generalization error (measure the quality of inductive bias), we need data unseen during training. We split the data as
  - Training set : used to fit the models;
  - Validation set : used to estimate prediction error for model selection, tune hyperparameters
  - Test (publication) set : is used for assessment of the generalization error of the final chosen model

Cross-Validation

- Resampling when there is few data
- *cross-validation* : choosing the hypothesis that is the most accurate on the validation set is the best one (the one that has the best inductive bias)

$\hat{f}^{-k}(x)$ = the fitted function, computed with the kth part of the data removed

$$\mathrm{CV}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i))$$

K = N , leave-one-out cross-validation

Cross-Validation

Example:
For the k=3 part, we fit the model to the other K −1 parts of the data, and calculate the prediction error of the fitted model when predicting the kth part of the data. We do this for k = 1, 2, . . . ,K and combine the K estimates of prediction error.

1	2	3	4	5
Train	Train	Validation	Train	Train

$\hat{f}^{-k}(x, \alpha)$ = set of models f(x,) indexed by a tuning parameter, with the kth part of the data removed.

$$\mathrm{CV}(\hat{f}, \alpha) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i, \alpha))$$

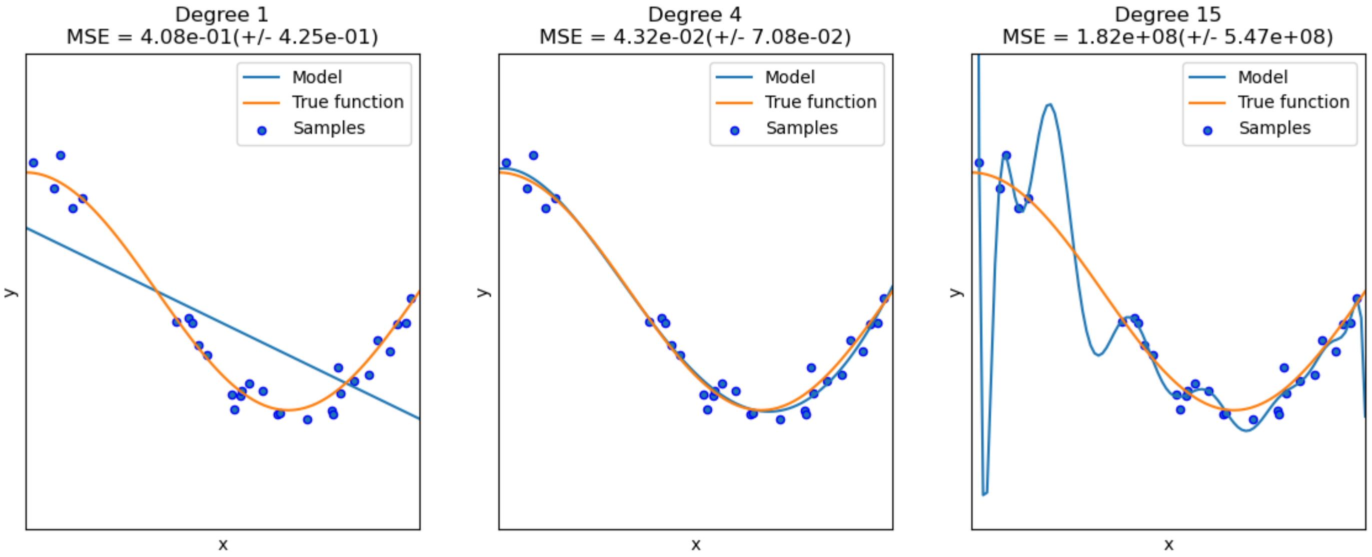
final chosen model is f(x, α-hat), which is used to fit to all the data

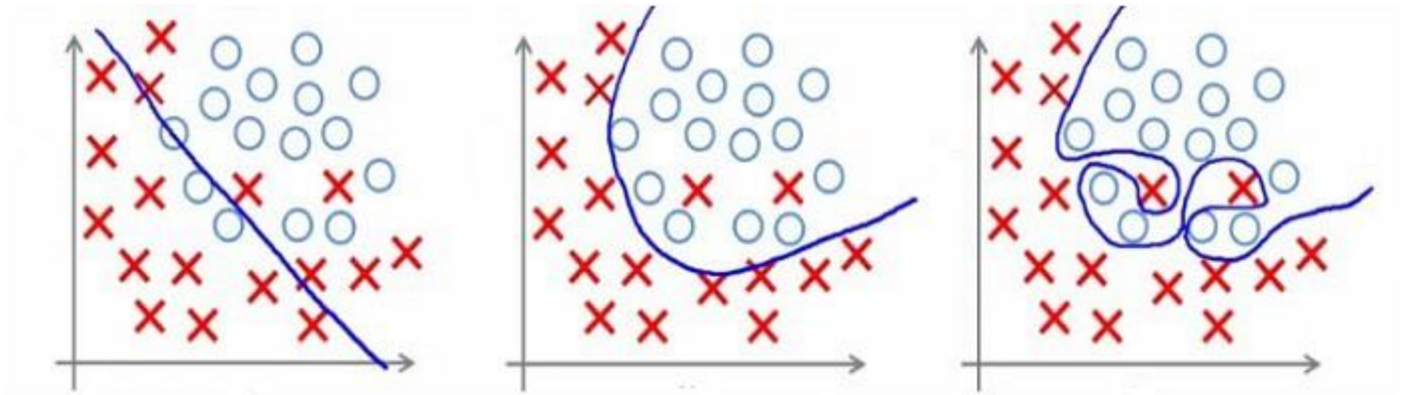
Training vs. Test Distribution

- We generally assume that training and test examples are independently drawn from the same overall distribution of data
  - We call this “i.i.d” which stands for “independent and identically distributed”

Slide credit: Ray Mooney

Overfitting/Underfitting





Under-fitting

Appropriate-fitting

Over-fitting

(too simple to explain the variance)

(forcefitting -- too good to be true)

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* increases, $E\downarrow$

□ As $c\ (\mathcal{H})$ increases, first $E\downarrow$ and then *E* increases

Issues in Machine Learning

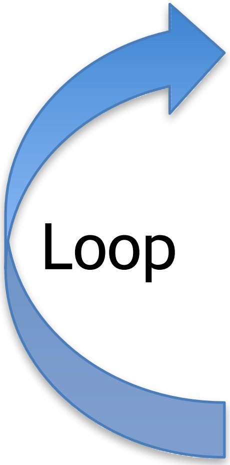
- What algorithms are available for learning a concept? How well do they perform?
- How much training data is sufficient to learn a concept with high confidence?
- When is it useful to use prior knowledge?
- Are some training examples more useful than others?
- What are the best tasks for a system to learn?
- What is the best way for a system to represent its knowledge?

ML in a Nutshell

- Tens of thousands of machine learning algorithms
  - Hundreds new every year
- Every ML algorithm has three components
  - Representation
  - Optimization
  - Evaluation

Slide credit: Pedro Domingos

ML in Practice



Loop

- Understand domain, prior knowledge, and goals
- Data integration, selection, cleaning, pre-processing, etc.
- Learn models
- Interpret results
- Consolidate and deploy discovered knowledge

Based on a slide by Pedro Domingos

Thank you !