Outline of the Course

1. The Learning Problem (*April 3*)
2. Is Learning Feasible? (*April 5*)
3. The Linear Model I (*April 10*)
4. Error and Noise (*April 12*)
5. Training versus Testing (*April 17*)
6. Theory of Generalization (*April 19*)
7. The VC Dimension (*April 24*)
8. Bias-Variance Tradeoff (*April 26*)
9. The Linear Model II (*May 1*)
10. Neural Networks (*May 3*)
11. Overfitting (*May 8*)
12. Regularization (*May 10*)
13. Validation (*May 15*)
14. Support Vector Machines (*May 17*)
15. Kernel Methods (*May 22*)
16. Radial Basis Functions (*May 24*)
17. Three Learning Principles (*May 29*)
18. Epilogue (*May 31*)

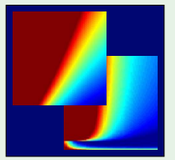
- theory; mathematical
- technique; practical
- analysis; conceptual

Learning From Data

Yaser S. Abu-Mostafa
California Institute of Technology

Lecture 1: The Learning Problem

Sponsored by Caltech's Provost Office, E&AS Division, and IST • Tuesday, April 3, 2012



The learning problem - Outline

- Example of machine learning
- Components of Learning
- A simple model
- Types of learning
- Puzzle

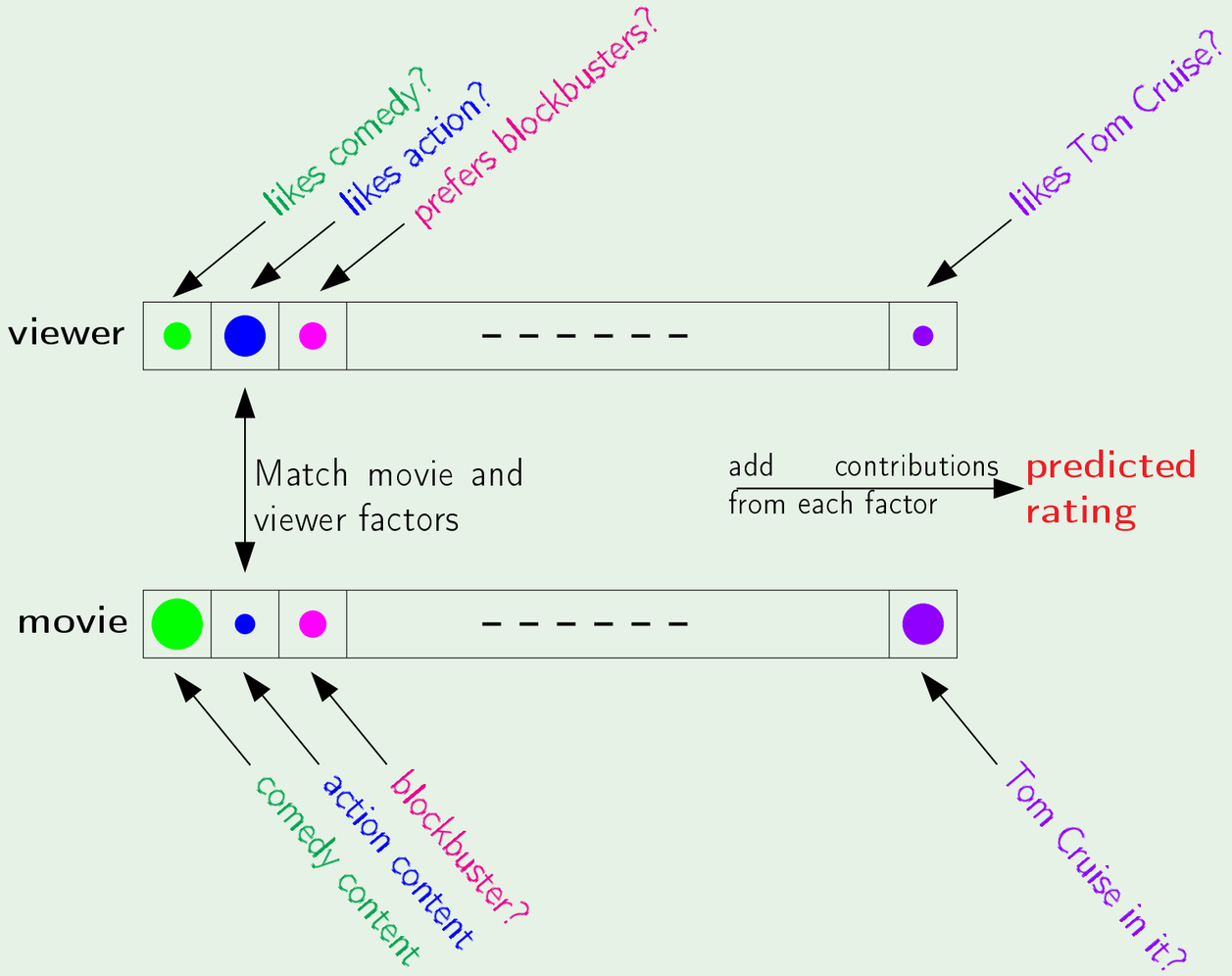
Example: Predicting how a viewer will rate a movie

10% improvement = 1 million dollar prize

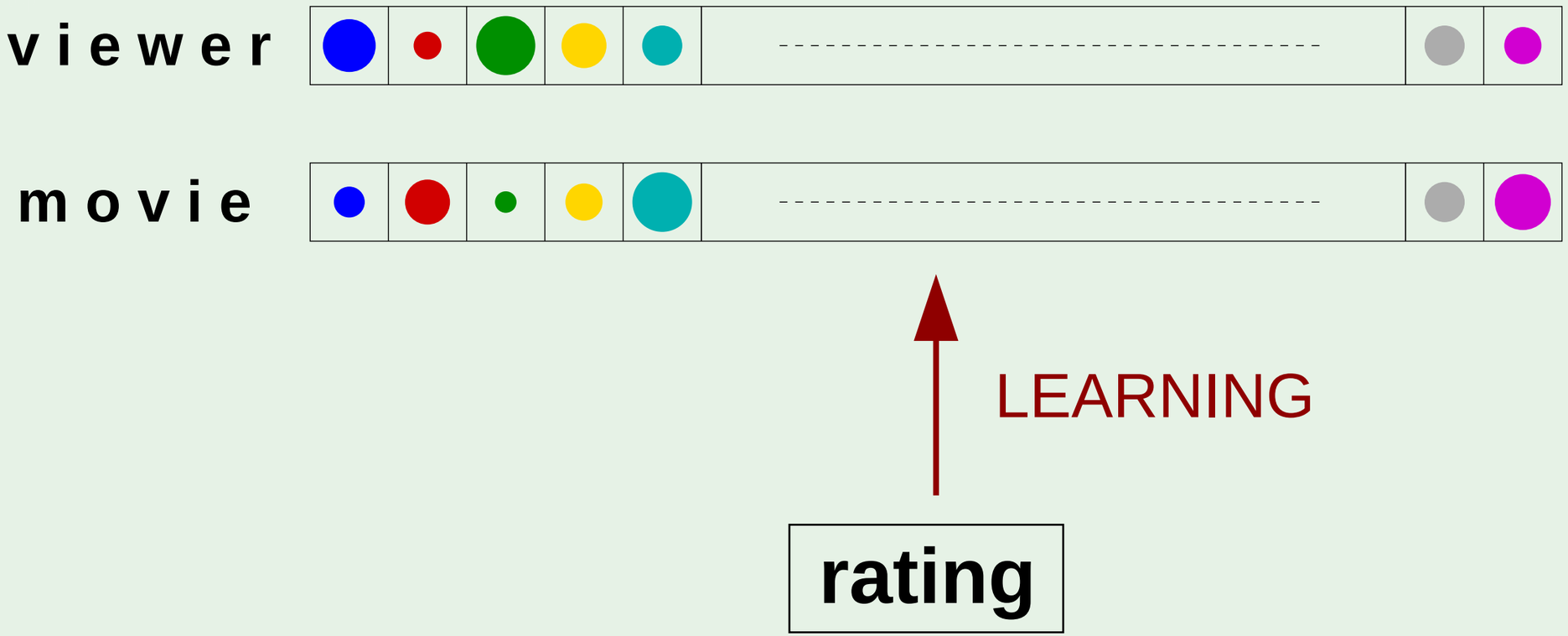
The essence of machine learning:

- A pattern exists.
- We cannot pin it down mathematically.
- We have data on it.

Movie rating - a solution



The learning approach



Components of learning

Metaphor: Credit approval

Applicant information:

age	23 years
gender	male
annual salary	$30,000
years in residence	1 year
years in job	1 year
current debt	$15,000
...	...

Approve credit?

Components of learning

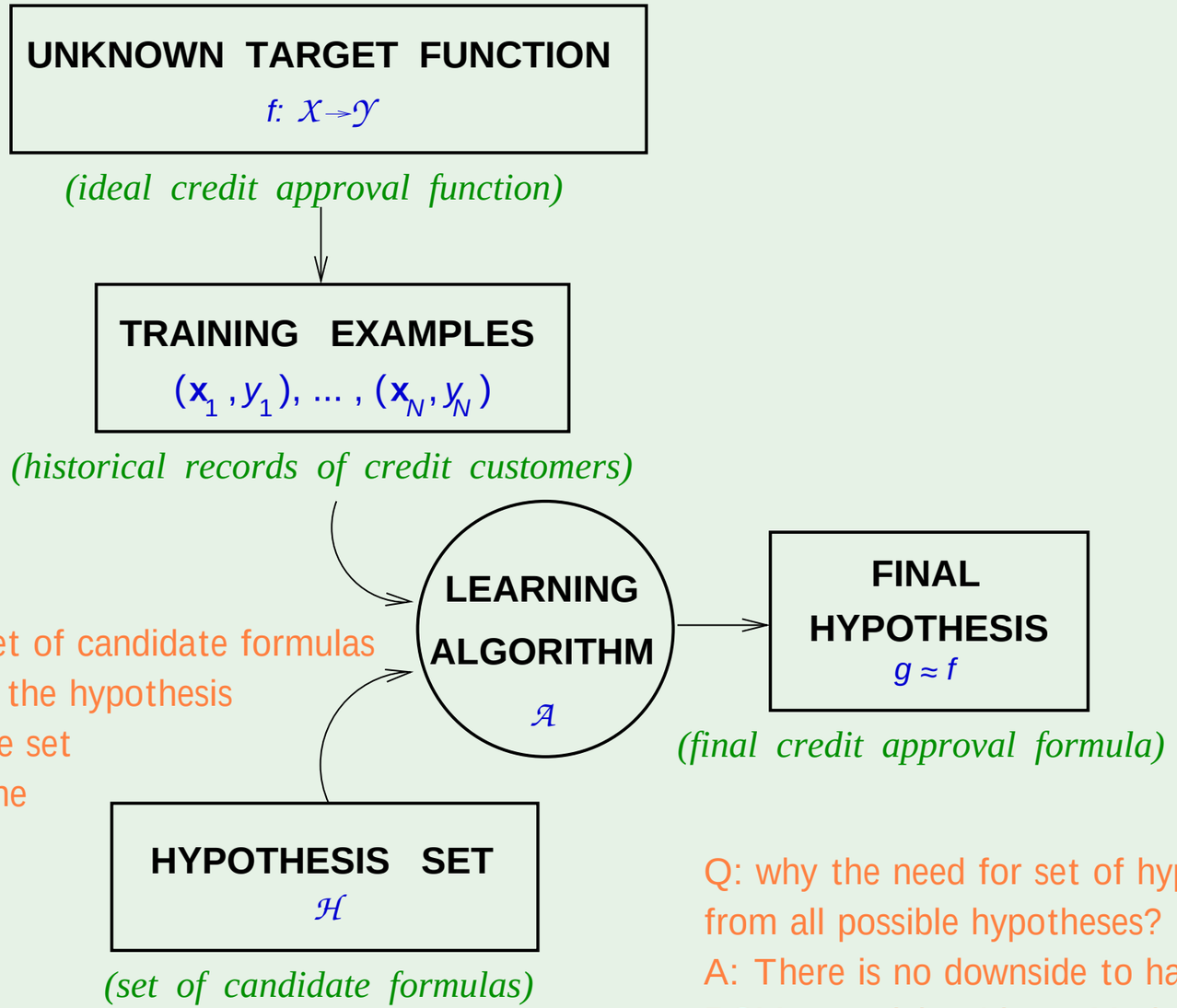
Formalization:

- Input: $\mathbf{x}$ (*customer application*)
- Output: y (*good/bad customer?*)
- Target function: $f : \mathcal{X} \to \mathcal{Y}$ (*ideal credit approval formula*)
- Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$ (*historical records*)

  -- Dataset D of input-output examples where yn = f(xn) for n = 1,...,N. Inputs correspond to previous customers and the correct credit decision for them in hindsight

  $\downarrow \quad \downarrow \quad \downarrow$
- Hypothesis: $g : \mathcal{X} \to \mathcal{Y}$ (*formula to be used*)

We modify the hypothesis g to attempt to approximate f



The algorithm chooses g from a set of candidate formulas under consideration, which we call the hypothesis set H. For instance, H could be the set of all linear formulas from which the algorithm would choose the best linear fit to the data.

Q: why the need for set of hypotheses H, why not let A pick from all possible hypotheses?
A: There is no downside to having H since from a practical POV, you pick a Linear Model/NN/SVM at the beginning. The upside to H is that it is pivotal to if we can learn and how well we learn from the data.

Solution components

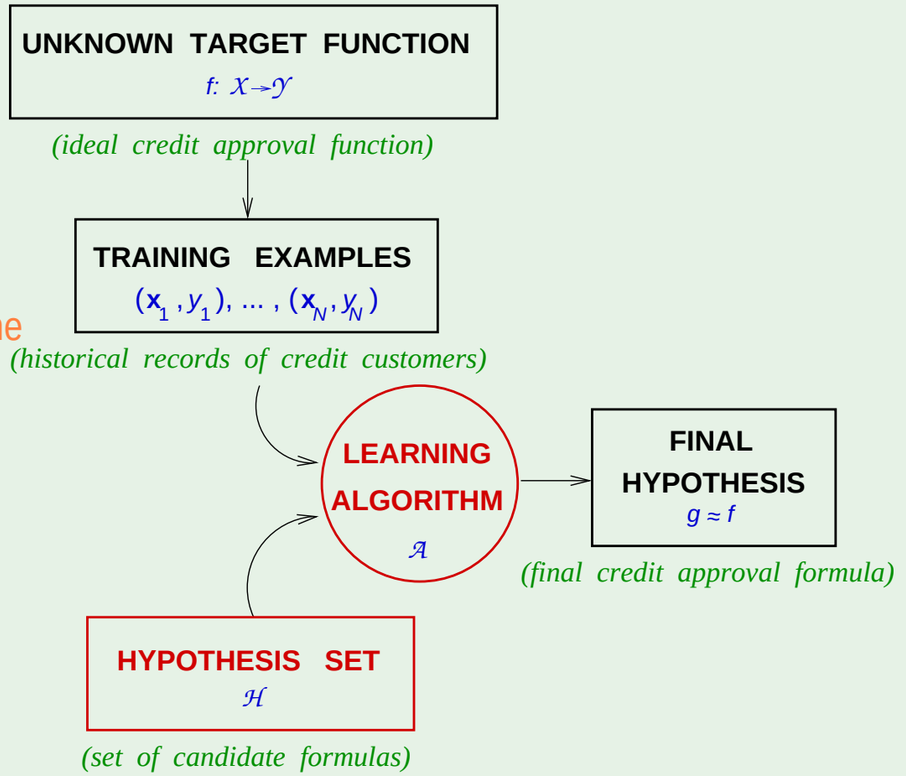
The 2 solution components of the learning problem:

- The Hypothesis Set (e.g. perceptron, NN, SVM - or atleast the set of all possible weights for whatever selected model)

  $\mathcal{H} = \{h\}$ $\quad$ $g \in \mathcal{H}$

- The Learning Algorithm (e.g. perceptron learning model, backprop., quadratic programming)

Together, they are referred to as the *learning model*.



A simple hypothesis set - the 'perceptron'

For input $\mathbf{x} = (x_1, \cdots, x_d)$ 'attributes of a customer'

$$\text{Approve credit if} \quad \sum_{i=1}^{d} w_i x_i > \text{threshold},$$

$$\text{Deny credit if} \quad \sum_{i=1}^{d} w_i x_i < \text{threshold}.$$

The linear addition of w*x makes this a perceptron

This linear formula $h \in \mathcal{H}$ can be written as

$$h(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - \text{threshold}\right)$$

Red parameters determine the exact hypothesis - these are the aspects we vary using the learning algorithm

$$h(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i\, x_i\right) + \quad w_0 \quad\right)$$

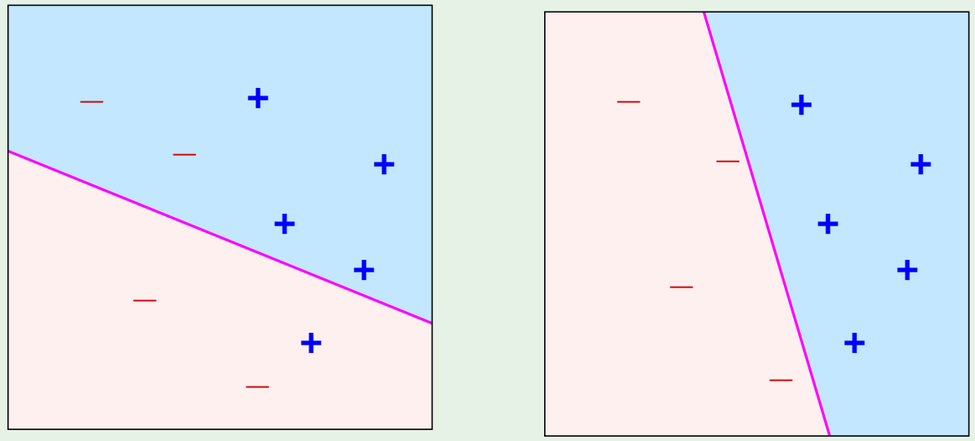
A choice of wi determines the location of the purple separation line below. (Note: change threshold (= -b (the bias term)) to -w0 and let x0 = 1 to simplify the expression)

Introduce an artificial coordinate $x_0 = 1$:

$$h(\mathbf{x}) = \text{sign}\left(\sum_{i=0}^{d} w_i\, x_i\right)$$

In vector form, the perceptron implements

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^{\mathsf{T}}\mathbf{x})$$



'linearly separable' data

A simple learning algorithm - PLA

The perceptron implements

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^\mathsf{T}\mathbf{x})$$

Given the training set:

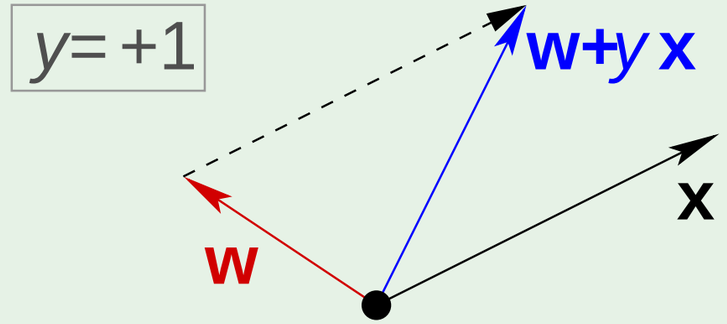
$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$$

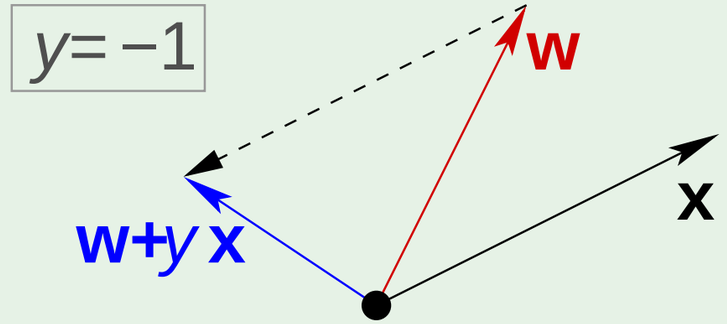
pick a misclassified point:

$$\text{sign}(\mathbf{w}^\mathsf{T}\mathbf{x}_n) \neq y_n$$

and update the weight vector:

$$\mathbf{w} \leftarrow \mathbf{w} + y_n\mathbf{x}_n$$





So: since yn = + or - 1, add or subtract xn to w,
so now the hypothesis will correctly classify this point - see diagrams

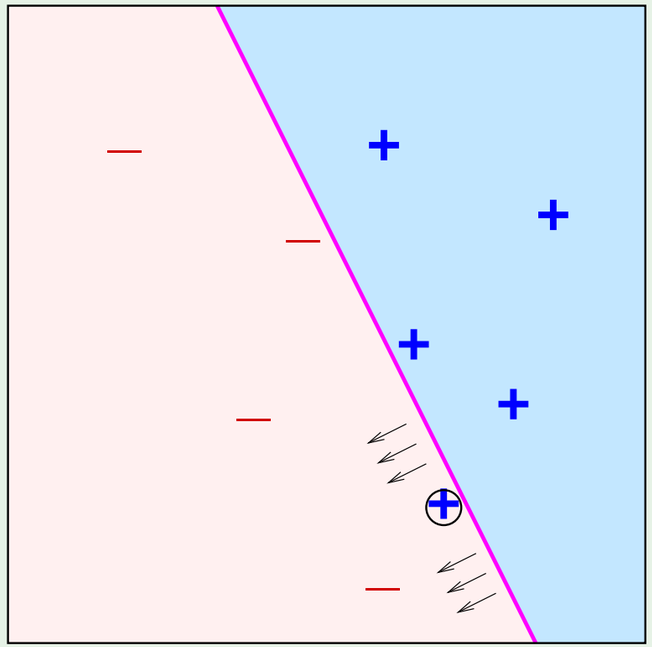
Iterations of PLA

If the data set is linearly separable, there will be a vector w that makes h classify all training examples correctly

- One iteration of the PLA:

$$\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$$

where $(\mathbf{x}, y)$ is a misclassified training point.



- At iteration $t = 1, 2, 3, \cdots$, pick a misclassified point from

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$$

and run a PLA iteration on it.

- That's it!

Within the infinite space of all weight vectors , the perceptron algorithm manages to find a weight vector that works, using a simple iterative process. This illustrates how a learning algorithm can effectively search an infinite hypothesis set using a finite number of simple steps. This feature is characteristic of many techniques that are used in learning (some of which are far more sophisticated than the PLA).

The learning problem - Outline

- Example of machine learning
- Components of learning
- A simple model
- Types of learning
- Puzzle

Basic premise of learning

"using a set of observations to uncover an underlying process"

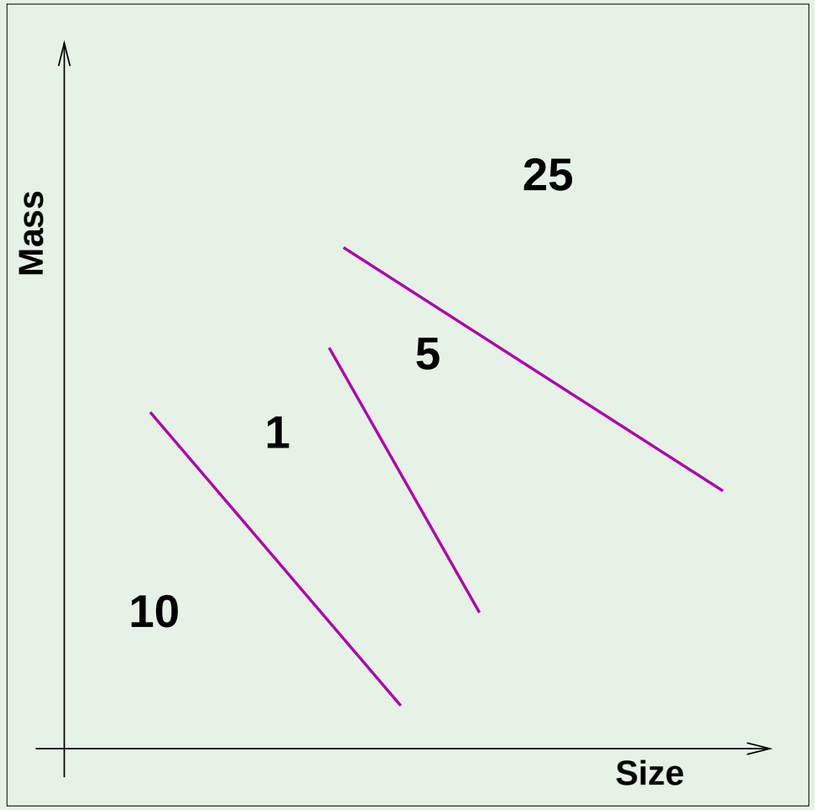
broad premise $\Longrightarrow$ many variations

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Two variations on the way a dataset can be presented in the learning process.
One is active learning (only applicable to supervised learning, where the data set is acquired through queries that
we make. Thus, we get to choose a point x in the input space, and the
supervisor reports to us the target value for x. As you can see, this opens
the possibility for strategic choice of the point

Supervised learning

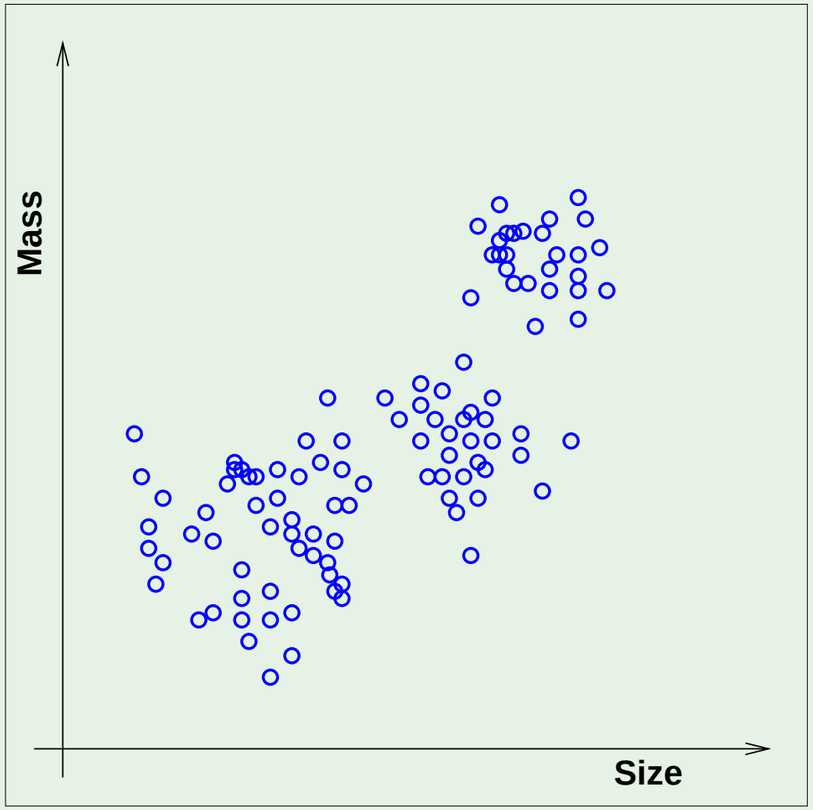
Example from vending machines – **coin recognition**



Unsupervised learning

Instead of (input,correct output), we get (input, ?)

Another variation on dataset presentation is called online learning, where the data set is given to the algorithm one example at a time. This happens when we have streaming data that the algorithm has to process 'on the run ' . For instance, when
the movie recommendation system is deployed, online learning can process new ratings from current users and movies. Online learning is also useful when we have limitations on computing and storage that preclude us from processing the whole data as a batch. We should note
that online learning can be used in different paradigms of learning, not just in supervised

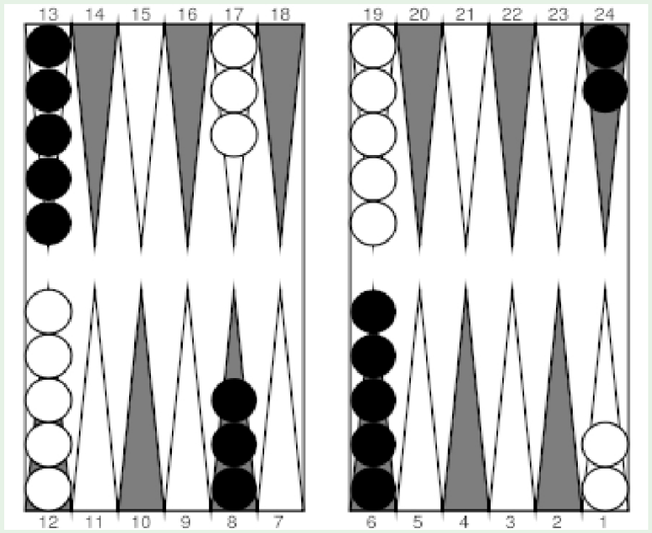


Unsupervised learning can be viewed as the task of spontaneously finding patterns and structure in input data. For instance, if
our task is to categorize a set of books into topics , and we only use general properties of the various books, we can identify books that have similar properties and put them together in one category, without naming that category - unsupervised learning can be a precursor to supervised learning. In other cases, it is a stand-alone technique..

Reinforcement learning

Instead of (input,correct output),

we get (input,*some* output,grade for this output)



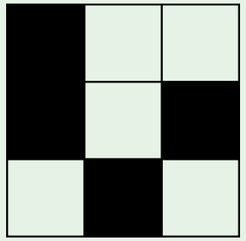
Reinforcement learning: where the training example does not contain the target output , but instead contains some possible output together with a measure of how good that output is.

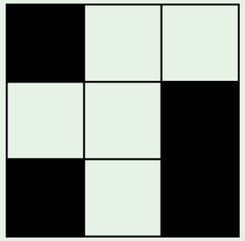
The world champion was a neural network!

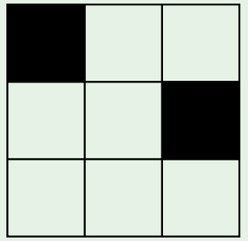
If you use reinforcement learning all you need to do is to take some action and report how well things went and you have a training example. The reinforcement learning algorithm is left with the task of sorting out the information coming from different examples to find the best line of play.

A Learning puzzle

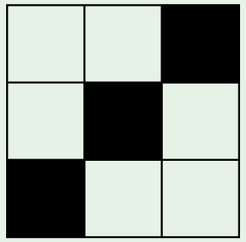
A simple learning task with 6 training examples of a ±1 target function is shown. Try to learn what the function is then apply it to the test input given. Do you get - 1 or + 1? Now, show the problem to your friends and see if they get the same answer .
The chances are the answers were not unanimous, and for good reason. There is simply more than one function that fits the 6 training examples , and
some of these functions have a value of - 1 on the test point and others have a value of + 1 . For instance, if the true f is + 1 when the pattern is symmetric, the value for the test point would be + 1 . If the true f is + 1 when the top left
square of the pattern is white, the value for the test point would be - 1 . Both functions agree with all the examples in the data set, so there isn't enough
information to tell us which would be the correct answer. This does not bode well for the feasibility of learning. To make matters worse, we will now see that the difficulty we experienced in this simple problem is the rule, not the exception.

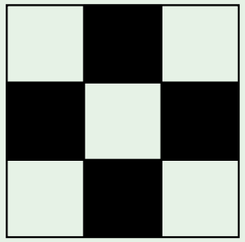


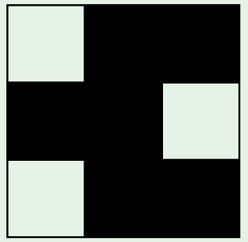




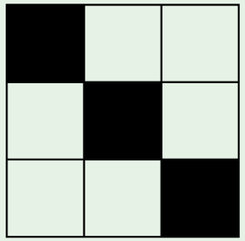
$f = -1$







$f = +1$



$f = ?$