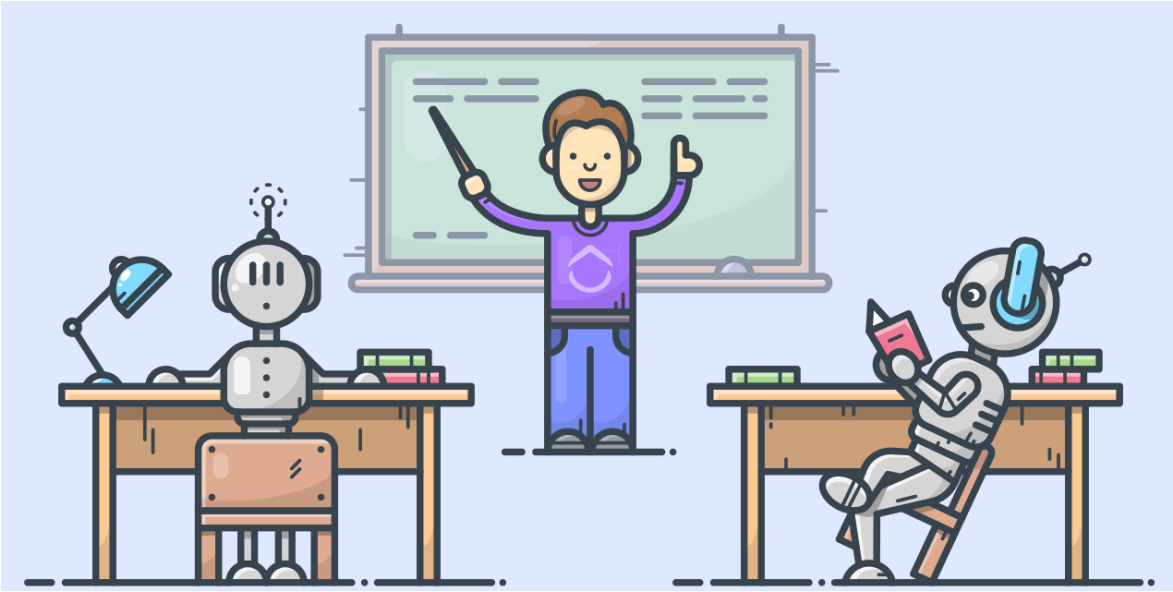
CS5491: Artificial Intelligence

Supervised Learning

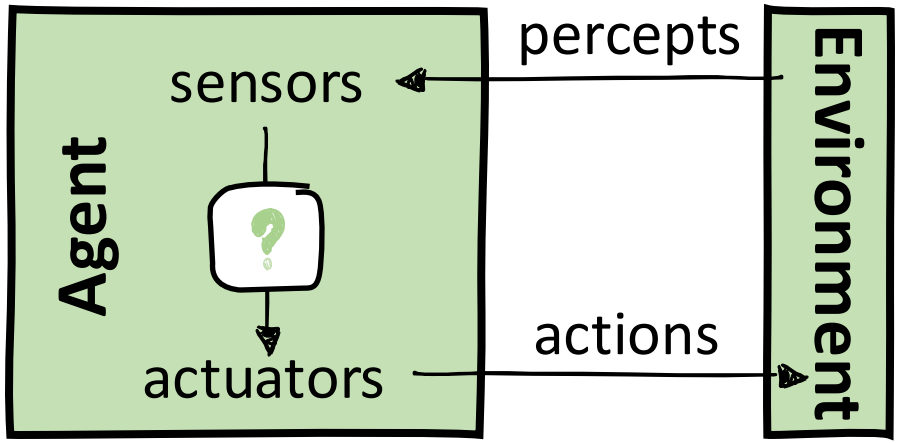


Instructor: Kai Wang

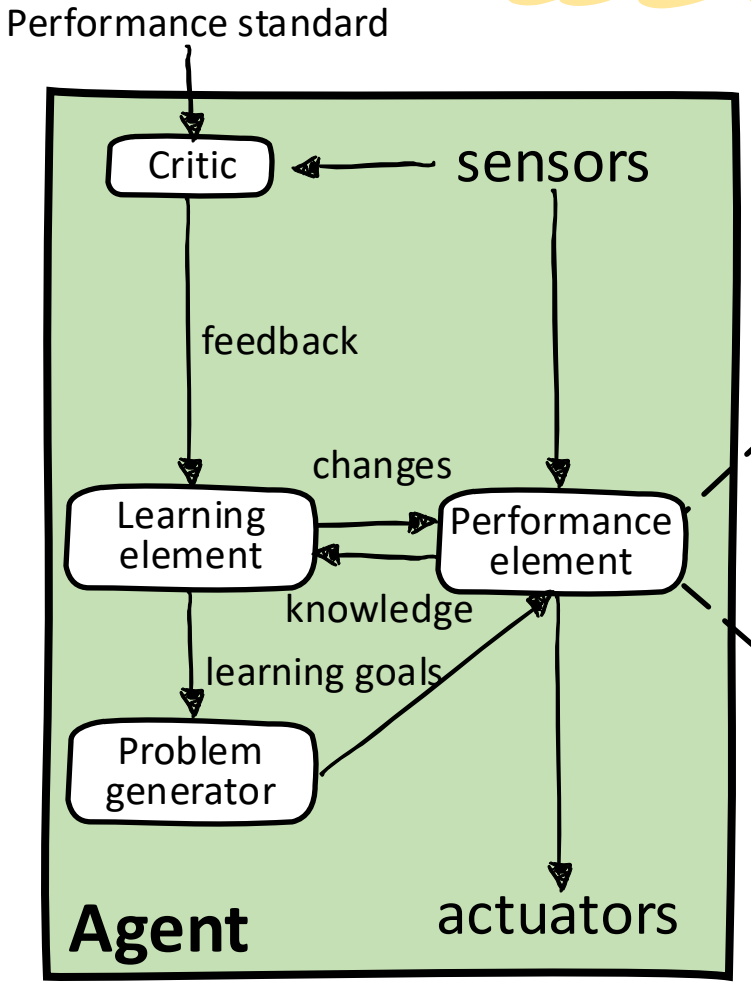
Recap: Agent

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.

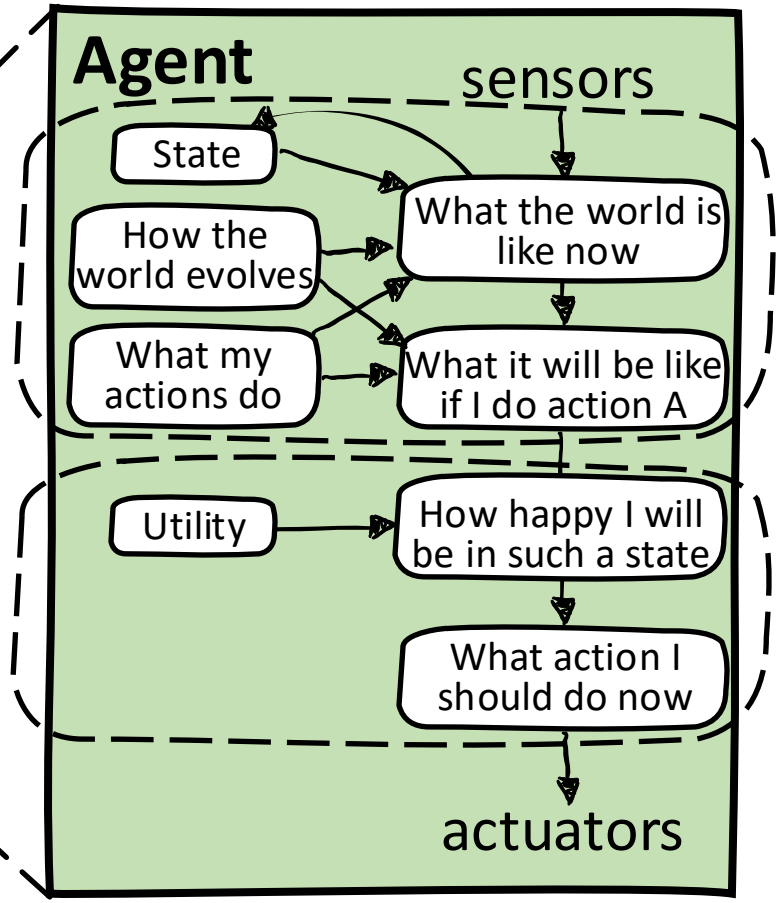
Agent = architecture + program



Recap: The Big Picture



Agents with learning



Knowledge representation and reasoning

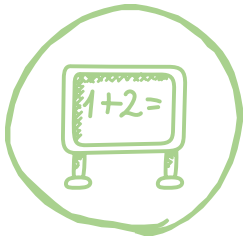
Problem solving

Agents without learning

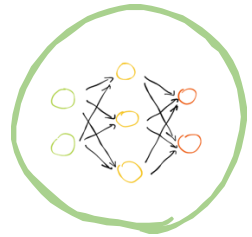
Today

Why Learning?

Categories of Learning



Supervised Learning

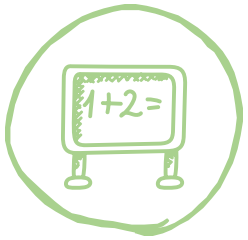


Neural Networks

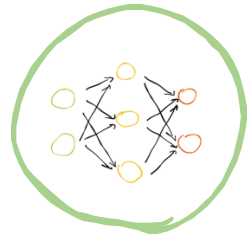
Today

Why Learning?

Categories of Learning



Supervised Learning



Neural Networks

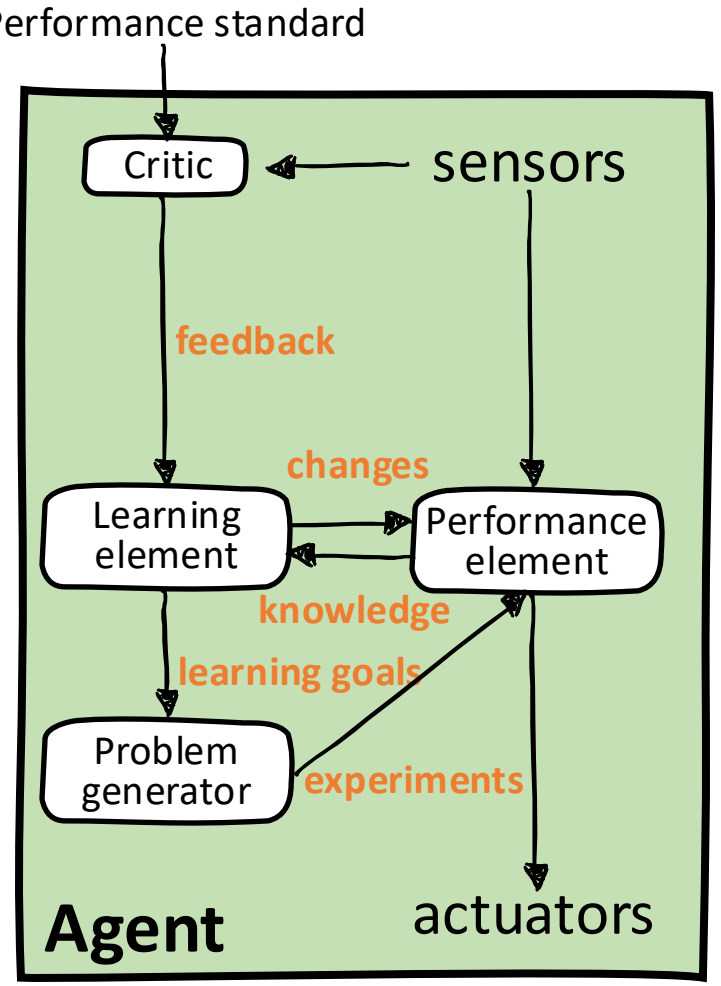
Learning: Why?

- Learning is essential for unknown environments, i.e., when the designer lacks omniscience.
- Learning is useful as a system construction method, i.e., expose the system to reality rather than trying to write it down
- Learning modifies the agent's decision mechanisms to improve performance

Design of Learning Element

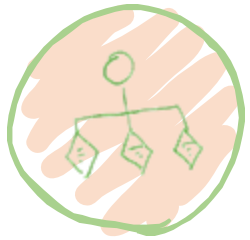
Performance element	Component	Representation	Feedback
Alpha-beta search	Evaluation function	Linear polynomial	Win/loss
Logical planning agent	Transition model (observable envt)	Successor-state axioms	Action outcomes
Utility-based patient monitor	Physiology / sensor model	Dynamic Bayesian network	Observation sequences
Bus image pixel classifier	Classifier (policy)	Neural network	Correct labels

Agents with learning

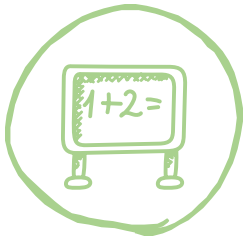


Today

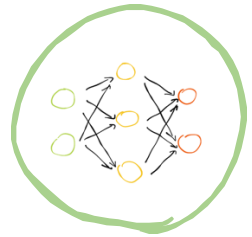
Why Learning?



Categories of Learning



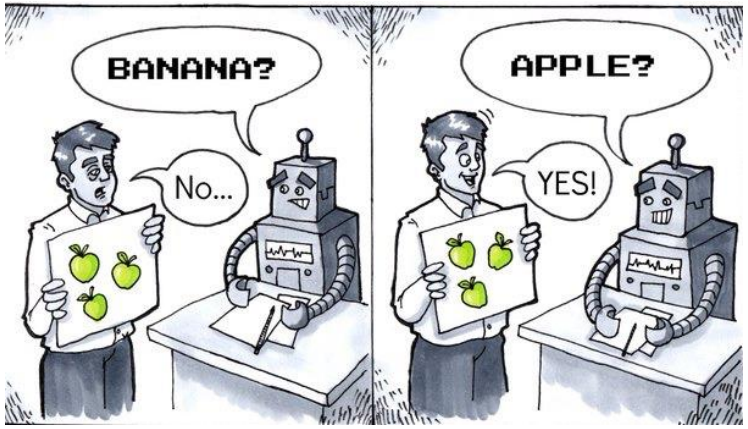
Supervised Learning



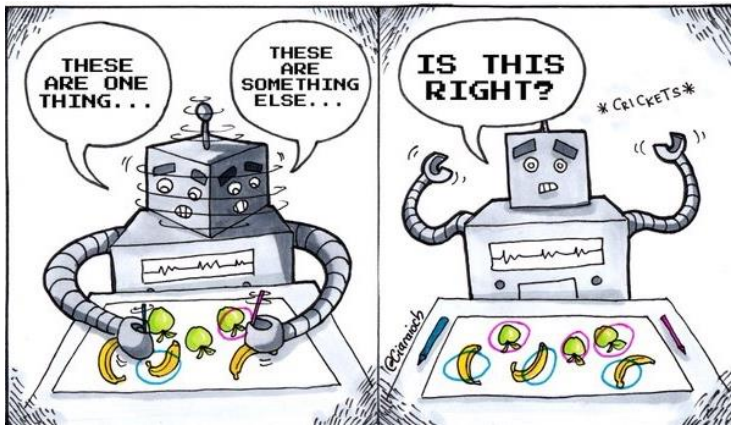
Neural Networks

Categories of Learning

Supervised learning



Unsupervised learning



Reinforcement learning

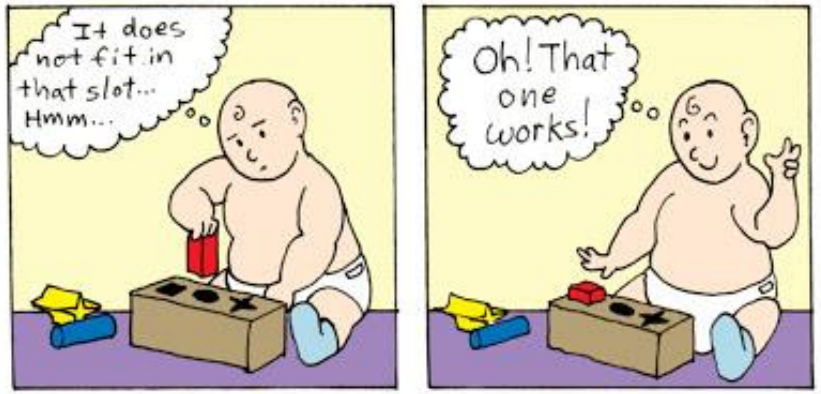
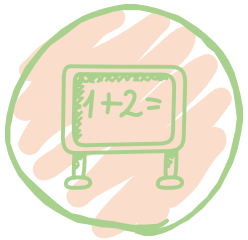


Photo credit to: https://twitter.com/athena_schools/status/1063013435779223553/photo/1
https://sites.google.com/site/acrader4th2014/home/part-3-unity-and-diversity-of-life/trialanderrorlearning

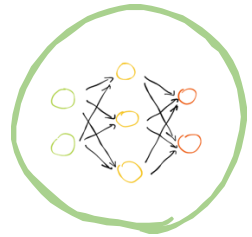
Today

Why Learning?

Categories of Learning



Supervised Learning



Neural Networks

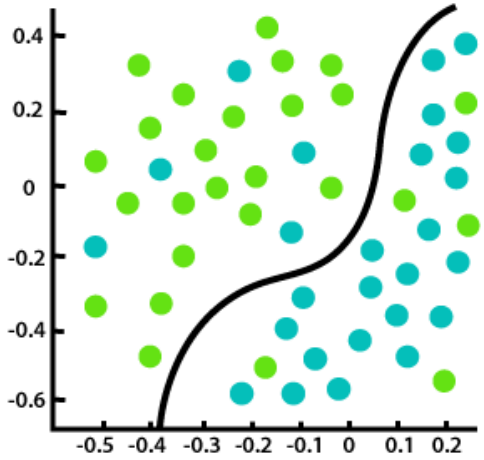
Supervised Learning

- Goal: learn an unknown target function f.
- Input: a training set consisting of labeled examples $(\mathbf{x}_i, y_i)$ where $y_i = f(\mathbf{x}_i)$. Attribute-value pairs characterize $\mathbf{x}_i$ are called features.
  - E.g., $\mathbf{x}_i$ is an image with pixels as features, $f(\mathbf{x}_i)$ is the label "apple".
  - E.g., $\mathbf{x}_i$ is the user profiling features, $f(\mathbf{x}_i)$ is the label "have insurance"
- Output: a hypothesis h that is "close" to f, i.e., h predicts well on unseen examples ("test set").
- Many possible hypothesis families for h
  - Linear models, logistic regression, neural networks, decision trees, examples (nearest-neighbor), grammars, kernelized separators, etc

What to Learn

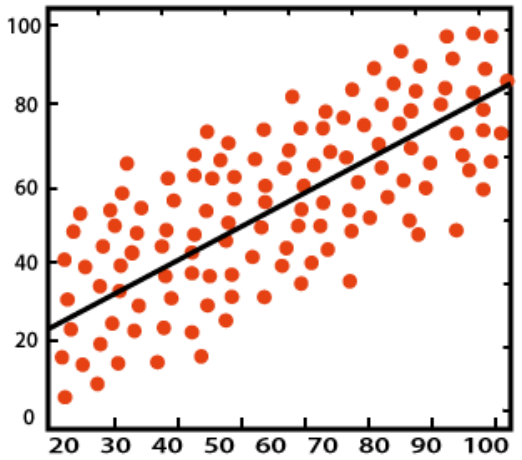
Classification

- The target function is discrete.
- E.g., will it be cold or hot tomorrow?



Regression

- The target function is continuous.
- E.g., what is the temperature going to be tomorrow?



Example for Classification

- According to the attributes of a customer, determine whether he/she will buy an insurance policy.

training set

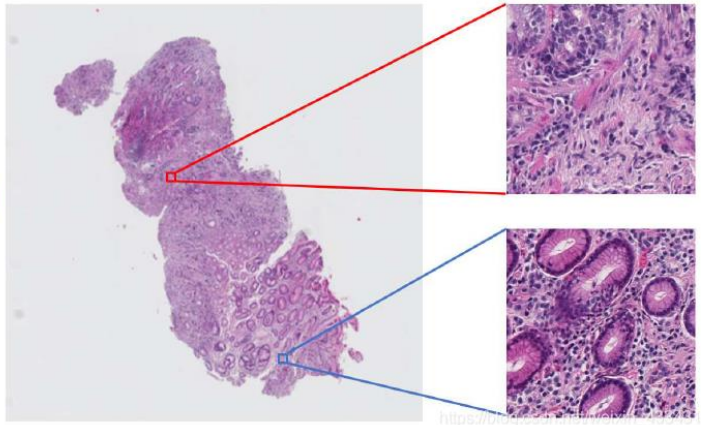
Area	Income	Child	Insurance
rural	high	no	yes
urban	high	yes	yes
urban	low	yes	yes
urban	low	yes	yes
rural	low	no	no
rural	low	no	no
rural	low	no	no
urban	low	no	no

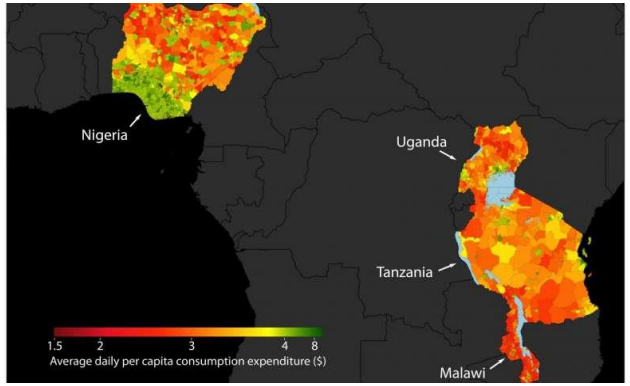
$\mathbf{x}$ $f(\mathbf{x})$

test set

Area	Income	Child	Insurance
urban	high	no	?

More Classification Examples





Example for Regression

- According to the attributes of a house, determine its price.

training set

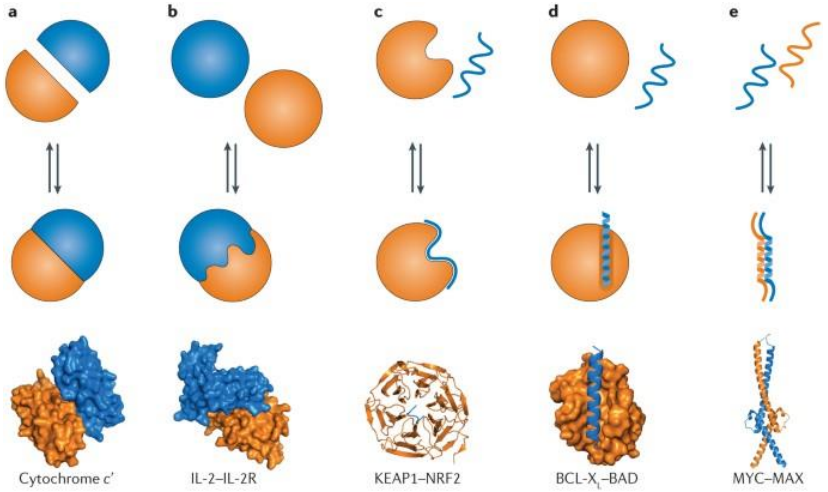
Bedrooms	Floors	Sqft_lot	Price
3	1	5650	221.900k
3	2	7242	538k
2	1	10000	180k
4	1	5000	604k
3	1	8080	510k
2	2	9000	490k

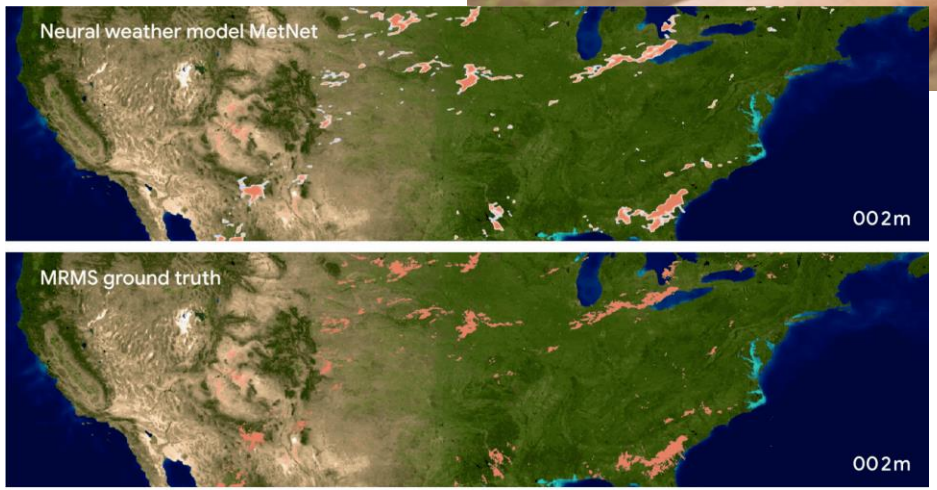
$\mathbf{x}$ $f(\mathbf{x})$

test set

Bedrooms	Floors	Sqft_lot	Price
4	2	9000	?

More Regression Examples





Supervised Learning Considerations

- Which hypothesis space H to choose?
- How to measure degree of fit? (How close $h \in H$ is to f?)
- How to trade off degree of fit vs. complexity?
- How do we find a good h among H?
- How do we know if a good h will predict well?

Supervised Learning Pipeline

1 Preparing data

Area	Income	Child	Insurance
rural	high	no	yes
urban	high	yes	yes
urban	low	yes	yes
urban	low	yes	yes
rural	low	no	no
rural	low	no	no
rural	low	no	no
urban	low	no	no

training set

Area	Income	Child	Insurance
urban	high	no	?

test set

2 Determine the hypothesis space H

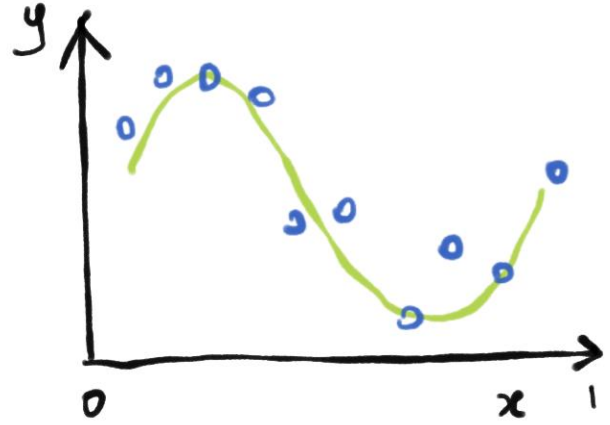
3 Determine degree of fit measure

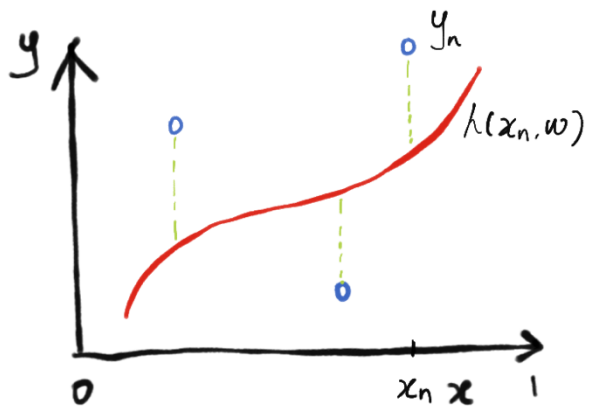
4 Learn a good h

5 Evaluating h

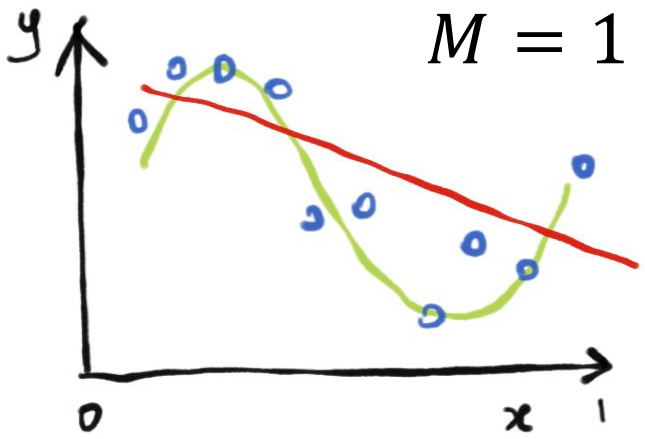
Regression Model: Polynomial Fitting

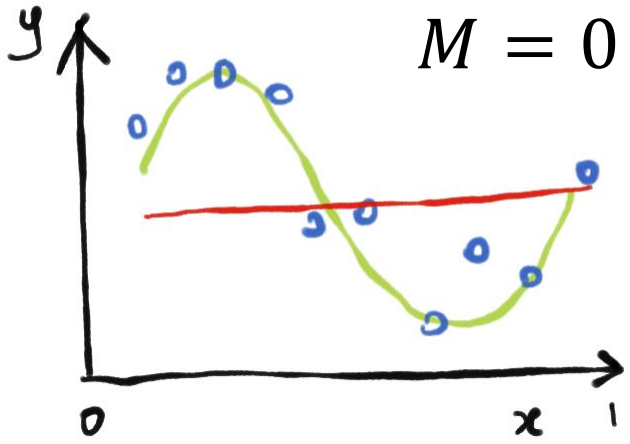
- Hypothesis space: polynomials of degree M
  - $H = \{\mathbf{w}|h(x,\mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots w_M x^M\}$
- Degree of fit/Evaluation: sum of squared loss
  - $E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(h(x_n,\mathbf{w}) - y_n)^2$
- Find a good h by minimizing the error.



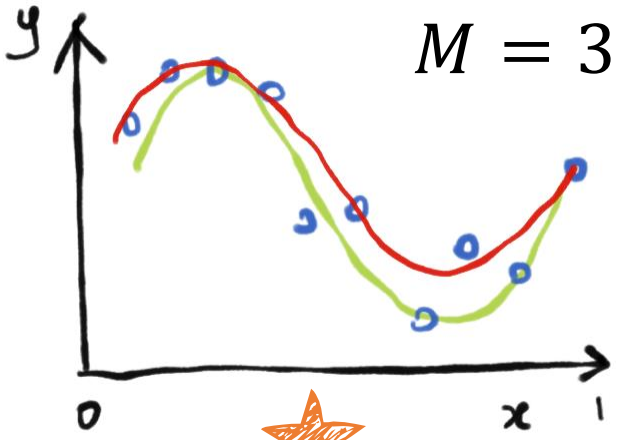


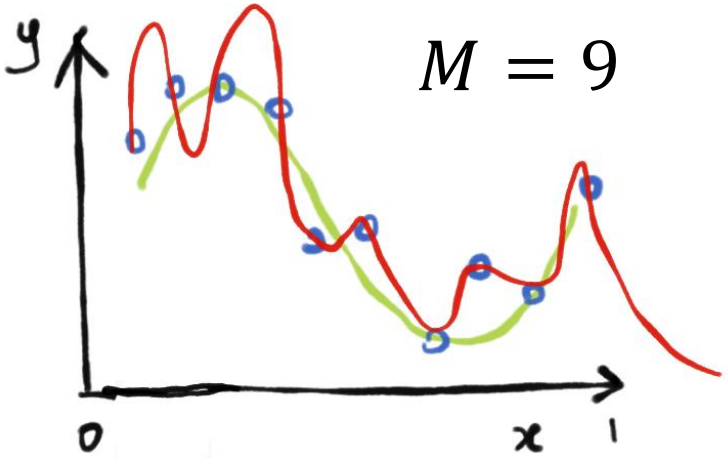
Under-fitting and Overfitting





Under-fitting
(high error, low complexity)





Over-fitting
(low error, high complexity)

Preventing Overfitting: Regularization

- Ridge Regression: Adding a l_2 regularization term
  - $E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(h(x_n, \mathbf{w}) - y_n)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$

$$\|\mathbf{w}\|^2 = w_0^2 + w_1^2 + \cdots + w_M^2$$

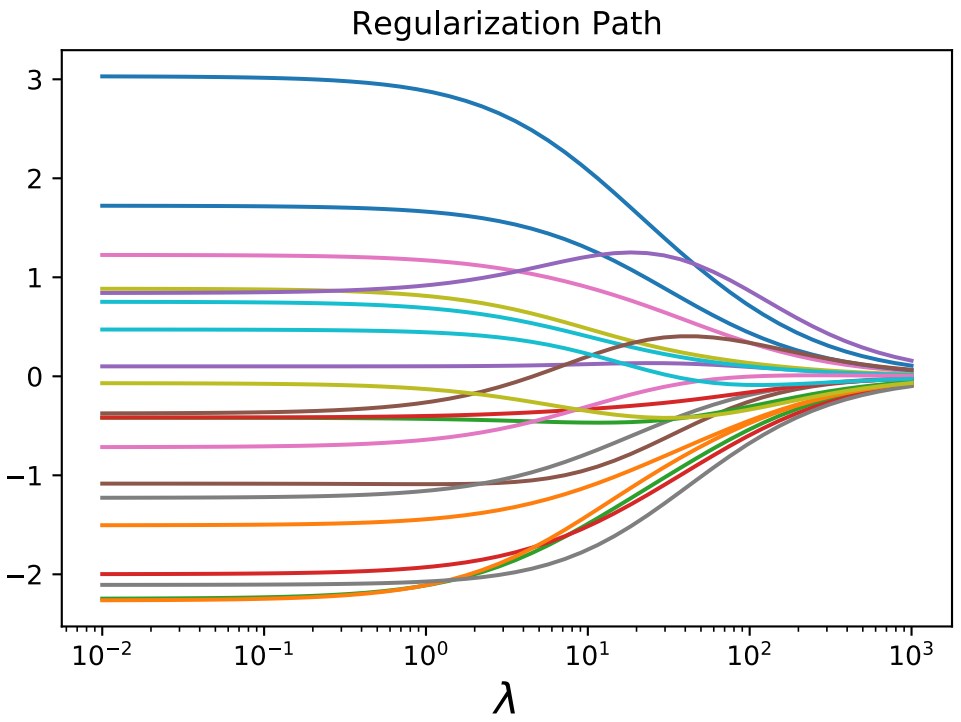
	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

- Lasso Regression: Adding a l_1 regularization term
  - $E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(h(x_n, \mathbf{w}) - y_n)^2 + \frac{\lambda}{2}|\mathbf{w}|^1$

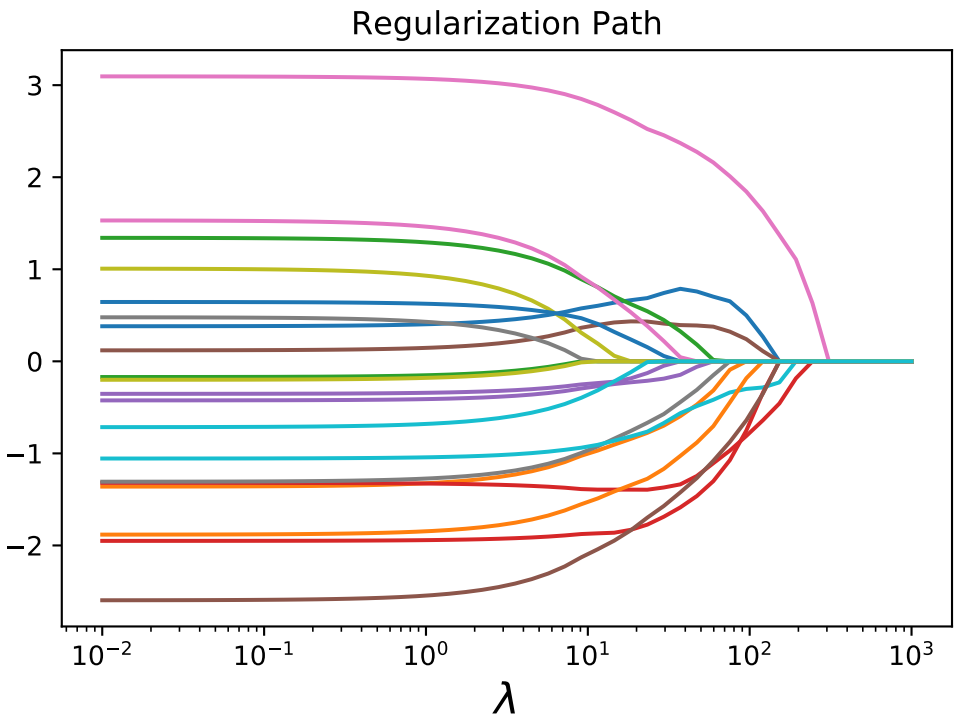
$$|\mathbf{w}|^1 = |w_0| + |w_1| + |w_M|$$

Effect of Regularization

Ridge

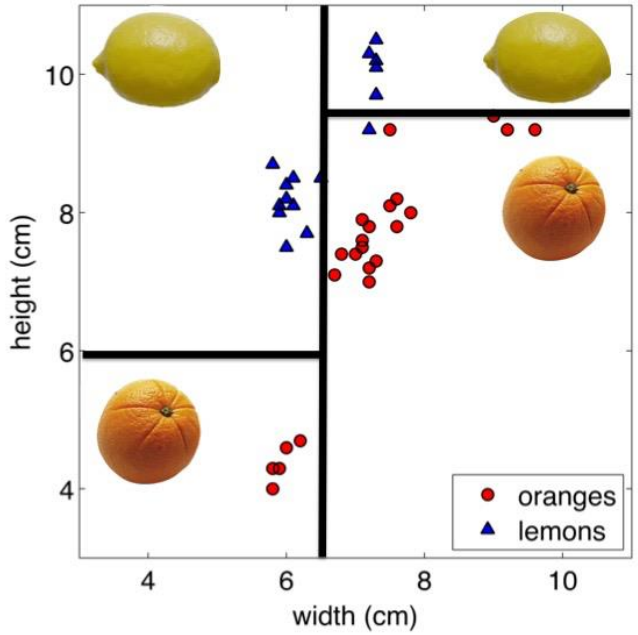


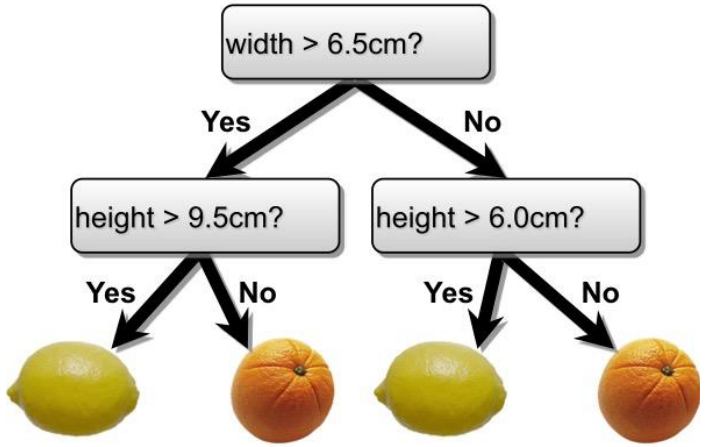
Lasso



Classification Model: Decision Tree

- Hypothesis space: a decision tree
  - $H = \{DT | h(\mathbf{x}, DT) = DT(\mathbf{x})\}$





Credit to: https://www.cs.toronto.edu/~urtasun/courses/CSC411_Fall16/06_trees_handout.pdf

Decision Tree Algorithm

node = root of decision tree

Main loop:

1. A ← the "best" decision attribute for the next *node*.
2. Assign A as decision attribute for *node*.
3. For each value of A, create a new descendant of *node*.
4. Sort training examples to leaf nodes.
5. If training examples are perfectly classified, stop. Else, recurse over new leaf nodes.

Entropy

Entropy is used to measure how informative is a node. If we are given a probability distribution $p = (p_1, p_2, \cdots, p_n)$, then the information conveyed by this distribution, also called the entropy of p is $I(p) = -(p_1 \log p_1 + p_2 \log p_2 + \cdots + p_n \log p_n)$. The smaller the entropy, the more informative we have.

- $p = (0.5,0.5)$ then $I(p) = 1$.
- $p = (0.67,0.33)$ then $I(p) = 0.92$.
- $p = (1,0)$ then $I(p) = 0$.

Decision Tree Algorithm: Example

Area	Income	Child	Insurance
rural	high	no	yes
urban	high	yes	yes
urban	low	yes	yes
urban	low	yes	yes
rural	low	no	no
rural	low	no	no
rural	low	no	no
urban	low	no	no

- $I(\text{Yes}) = -\frac{4}{8}\log\frac{4}{8} - \frac{4}{8}\log\frac{4}{8} = 1$
- $I(\text{Yes}_{\text{rural}}) = -\frac{1}{4}\log\frac{1}{4} - \frac{3}{4}\log\frac{3}{4} = 0.8113$
- $I(\text{Yes}_{\text{urban}}) = -\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4} = 0.8113$
- $I(\text{Area,Yes}) = \frac{1}{2}I(\text{Yes}_{\text{rural}}) + \frac{1}{2}I(\text{Yes}_{\text{urban}}) = 0.8113$
- $\text{Gain(Area,Yes)} = I(\text{Yes}) - I(\text{Area,Yes}) = 0.1887$

Decision Tree Algorithm: Example

Area	Income	Child	Insurance
rural	high	no	yes
urban	high	yes	yes
urban	low	yes	yes
urban	low	yes	yes
rural	low	no	no
rural	low	no	no
rural	low	no	no
urban	low	no	no

- $I(\text{Yes}) = -\frac{4}{8}\log\frac{4}{8} - \frac{4}{8}\log\frac{4}{8} = 1$
- $I(\text{Yes}_{\text{high}}) = -\frac{2}{2}\log\frac{2}{2} - \frac{0}{0}\log\frac{0}{0} = 0$
- $I(\text{Yes}_{\text{low}}) = -\frac{2}{6}\log\frac{2}{6} - \frac{4}{6}\log\frac{4}{6} = 0.9183$
- $I(\text{Income,Yes}) = \frac{2}{8}I(\text{Yes}_{\text{high}}) + \frac{6}{8}I(\text{Yes}_{\text{low}}) = 0.6887$
- $\text{Gain(Income,Yes)} = I(\text{Yes}) - I(\text{Income,Yes}) = 0.3113$

Decision Tree Algorithm: Example

Area	Income	Child	Insurance
rural	high	no	yes
urban	high	yes	yes
urban	low	yes	yes
urban	low	yes	yes
rural	low	no	no
rural	low	no	no
rural	low	no	no
urban	low	no	no

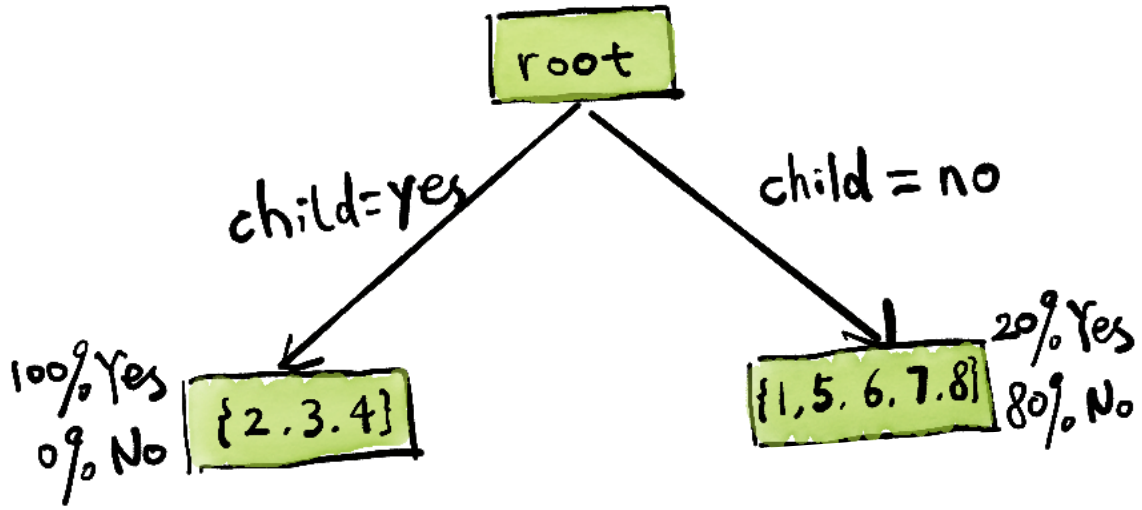
- $I(\text{Yes}) = -\frac{4}{8}\log\frac{4}{8} - \frac{4}{8}\log\frac{4}{8} = 1$
- $I(\text{Yes}_{\text{yes}}) = -\frac{3}{3}\log\frac{3}{3} - \frac{0}{0}\log\frac{0}{0} = 0$
- $I(\text{Yes}_{\text{no}}) = -\frac{1}{5}\log\frac{1}{5} - \frac{4}{5}\log\frac{4}{5} = 0.7219$
- $I(\text{Child,Yes}) = \frac{3}{8}I(\text{Yes}_{\text{yes}}) + \frac{5}{8}I(\text{Yes}_{\text{no}}) = 0.4512$
- $\text{Gain(Child,Yes)} = I(\text{Yes}) - I(\text{Child,Yes}) = 0.5488$

Decision Tree Algorithm: Example

Gain(Area,Yes) = 0.1887

Gain(Income,Yes) = 0.3113

Gain(Child,Yes) = 0.5488



	Area	Income	Child	Insurance
1	rural	high	no	yes
2	urban	high	yes	yes
3	urban	low	yes	yes
4	urban	low	yes	yes
5	rural	low	no	no
6	rural	low	no	no
7	rural	low	no	no
8	urban	low	no	no

Decision Tree Algorithm: Example

	Area	Income	Child	Insurance
1	rural	high	no	yes
2	urban	high	yes	yes
3	urban	low	yes	yes
4	urban	low	yes	yes
5	rural	low	no	no
6	rural	low	no	no
7	rural	low	no	no
8	urban	low	no	no

- $I(\text{Yes}) = -\frac{1}{5}\log\frac{1}{5} - \frac{4}{5}\log\frac{4}{5} = 0.7219$
- $I(\text{Yes}_{\text{rural}}) = -\frac{1}{4}\log\frac{1}{4} - \frac{3}{4}\log\frac{3}{4} = 0.8113$
- $I(\text{Yes}_{\text{urban}}) = -\frac{0}{0}\log\frac{0}{0} - \frac{1}{1}\log\frac{1}{1} = 0$
- $I(\text{Area,Yes}) = \frac{4}{5}I(\text{Yes}_{\text{rural}}) + \frac{1}{5}I(\text{Yes}_{\text{urban}}) = 0.6490$
- $\text{Gain(Area,Yes)} = I(\text{Yes}) - I(\text{Area,Yes}) = 0.0729$

Decision Tree Algorithm: Example

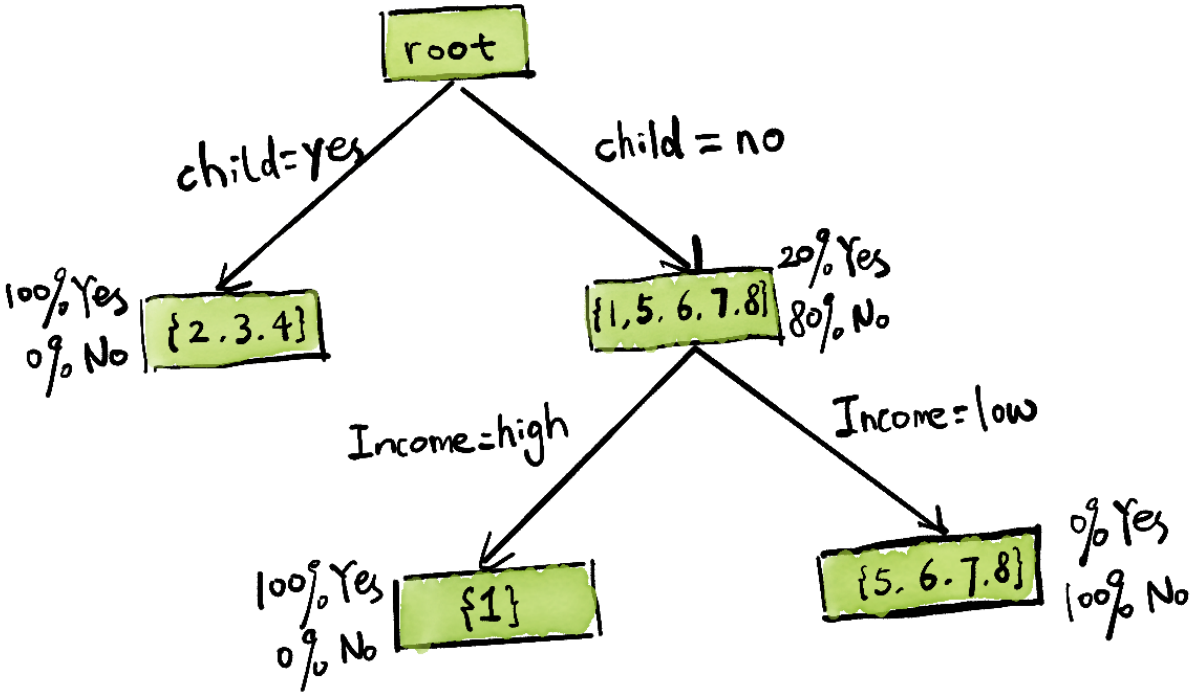
- $I(\text{Yes}) = -\frac{1}{5}\log\frac{1}{5} - \frac{4}{5}\log\frac{4}{5} = 0.7219$
- $I(\text{Yes}_{\text{high}}) = -\frac{1}{1}\log\frac{1}{1} - \frac{0}{0}\log\frac{0}{0} = 0$
- $I(\text{Yes}_{\text{low}}) = -\frac{0}{0}\log\frac{0}{0} - \frac{4}{4}\log\frac{4}{4} = 0$
- $I(\text{Income,Yes}) = \frac{1}{5}I(\text{Yes}_{\text{high}}) + \frac{4}{5}I(\text{Yes}_{\text{low}}) = 0$
- $\text{Gain(Income,Yes)} = I(\text{Yes}) - I(\text{Income,Yes}) = 0.7219$

	Area	Income	Child	Insurance
1	rural	high	no	yes
2	urban	high	yes	yes
3	urban	low	yes	yes
4	urban	low	yes	yes
5	rural	low	no	no
6	rural	low	no	no
7	rural	low	no	no
8	urban	low	no	no

Decision Tree Algorithm: Example

Gain(Area,Yes) = 0.0729

Gain(Income,Yes) = 0.7219



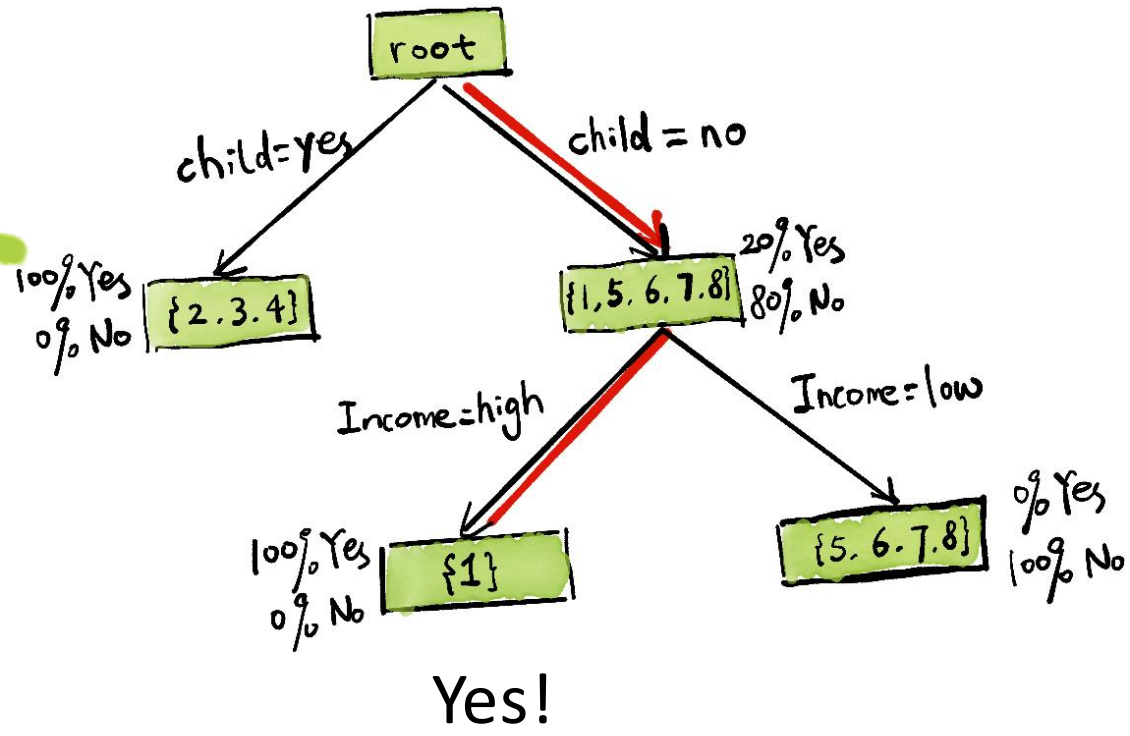
	Area	Income	Child	Insurance
1	rural	high	no	yes
2	urban	high	yes	yes
3	urban	low	yes	yes
4	urban	low	yes	yes
5	rural	low	no	no
6	rural	low	no	no
7	rural	low	no	no
8	urban	low	no	no

Evaluation

- Precision = $\frac{TP}{TP+FP}$
- Recall = $\frac{TP}{TP+FN}$
- Accuracy = $\frac{TP+TN}{TP+FP+FN+TN}$

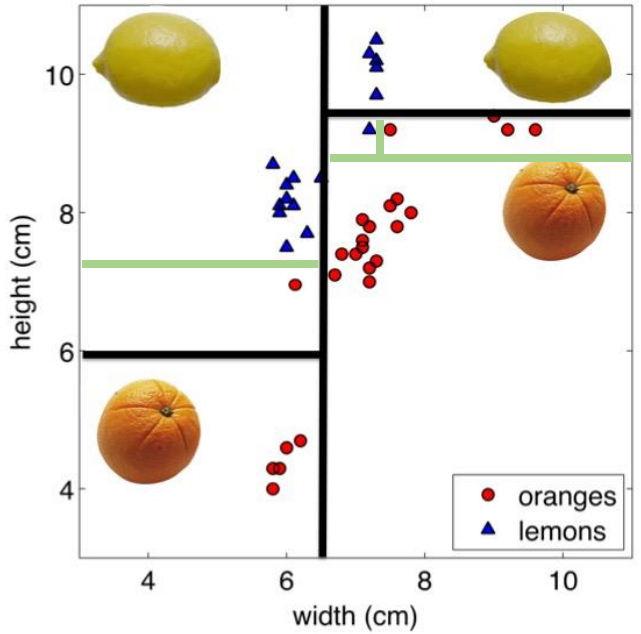
	True Class: Positive	True Class: Negative
Predicted Class: Positive	TP	FP
Predicted Class: Negative	FN	TN

Area	Income	Child	Insurance
urban	high	no	?



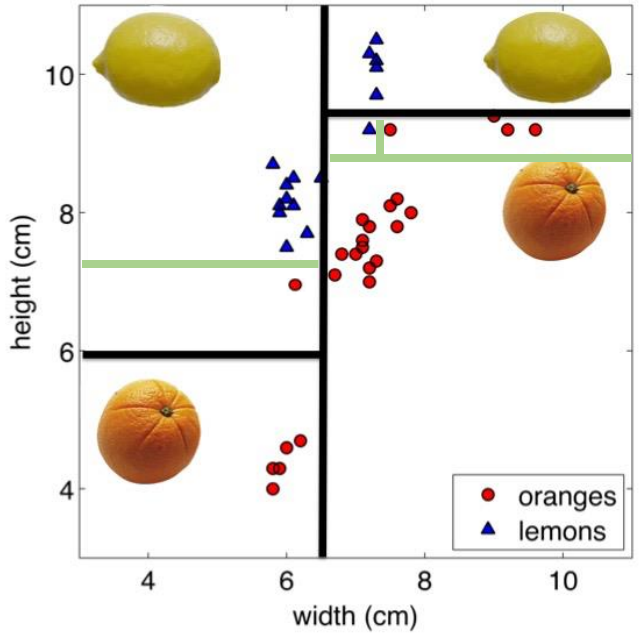
Overfitting

- Overfitting occurs because many possible kinds of “noise”
  - Two examples have same attribute/value pairs, but different classifications
  - Some values of attributes are incorrect because of errors in the data acquisition or preprocessing
  - The instance was labeled incorrectly



Avoid Overfitting

- Avoid overfitting
  - Limit the height of the tree.
  - Avoid growing when data split is not statistically significant
  - Acquire more training data
  - Select the smallest possible tree by measuring the performance on a hold-out validation set.



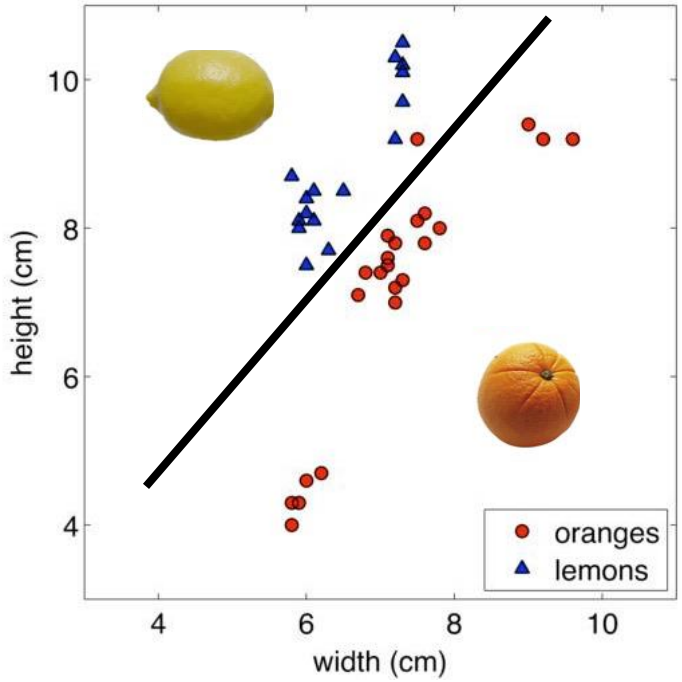
Summary of Decision Tree

- Fast and simple to implement
- Can convert to rules
- Handles noisy data.

- Univariate splits / partitioning using only one attribute at a time – limits types of possible trees
- Large decision trees may be hard to understand
- Requires all the data to be loaded

Classification Model: Logistic Regression

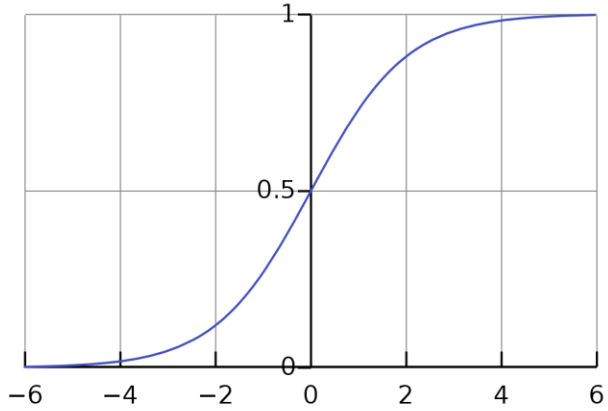
- Hypothesis space: a line for two features or a hyperplane for more features
  - $H = \{\mathbf{w} | h(\mathbf{w}, \mathbf{x}) = w_0 + w_1 x(1) + w_2 x(2) + \cdots + w_m x(m) = \mathbf{w}^T \mathbf{x}\}$



Cross Entropy

- Measure degree of fit
  - Sigmoid function: map the predicted value to a 0/1 value

$$\sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T x}}$$



$$\sigma(z) = \frac{1}{1+e^{-z}}$$

  - Cross-entropy loss function:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} -y_n \log \sigma(\mathbf{w}^T\mathbf{x}_n) - (1-y_n)\log(1-\sigma(\mathbf{w}^T\mathbf{x}_n))$$

- Preventing overfitting: $l2$ or $l1$ regularization

Gradient Methods

- Gradient methods compute

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3}\right)$$

  to increase (ascent) / reduce (descent) f, e.g., by $\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f(\mathbf{x})$.

- Adjusting the step size α :
  - Line search
  - Newton-Raphson iterates $\mathbf{x} \leftarrow \mathbf{x} - \mathbf{H}_f^{-1}(\mathbf{x}) \nabla f(\mathbf{x})$ where $\mathbf{H}_{ij} = \partial^2 f / \partial x_i \partial x_j$.

Gradient Descent

- The Gradient for Logistic Regression

$$\frac{\partial E(\mathbf{w})}{\partial w_i} = \frac{\partial}{\partial w_i}\sum_{n=1}^{N} -y_n \log \sigma(\mathbf{w}^T\mathbf{x}_n) - (1-y_n)\log\big(1-\sigma(\mathbf{w}^T\mathbf{x}_n)\big)$$

$$= -\left[\sum_{n=1}^{N} \frac{\partial}{\partial w_i} y_n \log \sigma(\mathbf{w}^T\mathbf{x}_n) + \frac{\partial}{\partial w_i}(1-y_n)\log\big(1-\sigma(\mathbf{w}^T\mathbf{x}_n)\big)\right]$$

$$= -\left[\sum_{n=1}^{N} \frac{y_n}{\sigma(\mathbf{w}^T\mathbf{x}_n)} - \frac{1-y_n}{1-\sigma(\mathbf{w}^T\mathbf{x}_n)}\right]\frac{\partial\sigma(\mathbf{w}^T\mathbf{x}_n)}{\partial w_i}$$

Gradient Descent

- The Gradient for Logistic Regression

$$\frac{\partial E(\mathbf{w})}{\partial w_i} = \frac{\partial}{\partial w_i}\sum_{n=1}^{N} -y_n \log \sigma(\mathbf{w}^T\mathbf{x}_n) - (1-y_n)\log\big(1-\sigma(\mathbf{w}^T\mathbf{x}_n)\big)$$

$$= -\left[\sum_{n=1}^{N}\frac{y_n}{\sigma(\mathbf{w}^T\mathbf{x}_n)} - \frac{1-y_n}{1-\sigma(\mathbf{w}^T\mathbf{x}_n)}\right]\frac{\partial\sigma(\mathbf{w}^T\mathbf{x}_n)}{\partial w_i}$$

$$= \sum_{n=1}^{N}[\sigma(\mathbf{w}^T\mathbf{x}_n) - y_n]x_{ni}$$

Summary of Logistic Regression


Easily extended to multiple classes


Does not require all the data to be loaded in

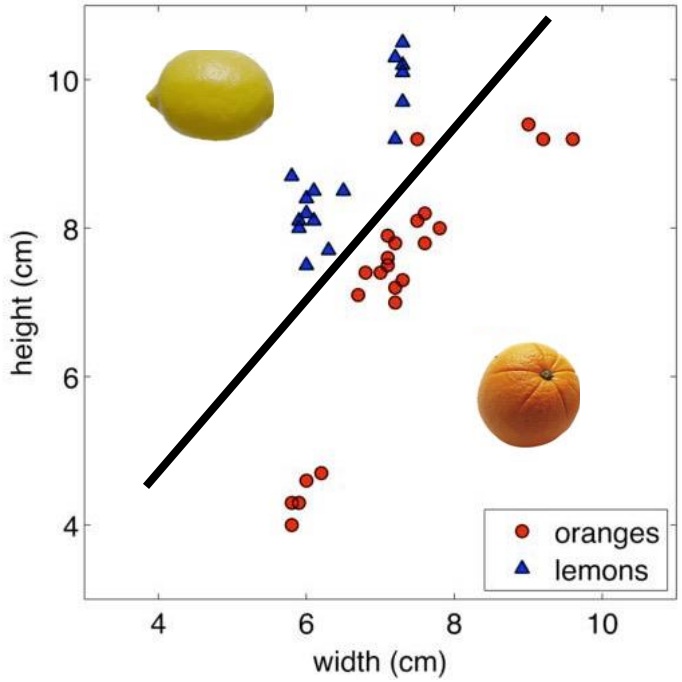

Quick to train


Good accuracy for many simple datasets


Linear decision boundary

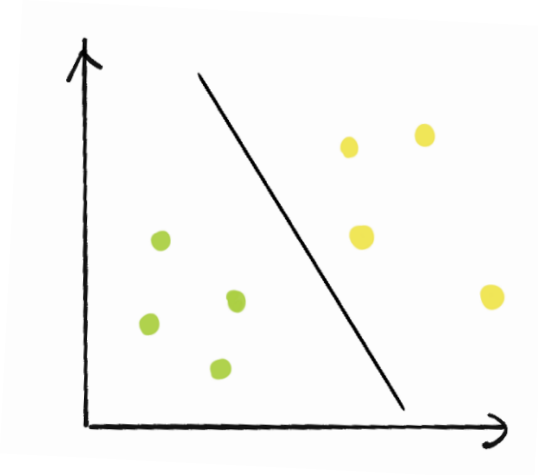
Classification Model: Support Vector Machine (Linear)

- Hypothesis space: a line for two features or a hyperplane for more features
  - $H = \{\mathbf{w} | h(\mathbf{w}, \mathbf{x}) = w_0 + w_1 x(1) + w_2 x(2) + \cdots + w_m x(m) = \mathbf{w}^T \mathbf{x}\}$

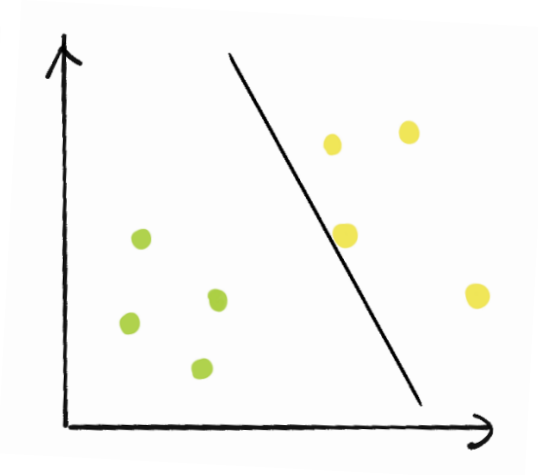


Maximum Margin

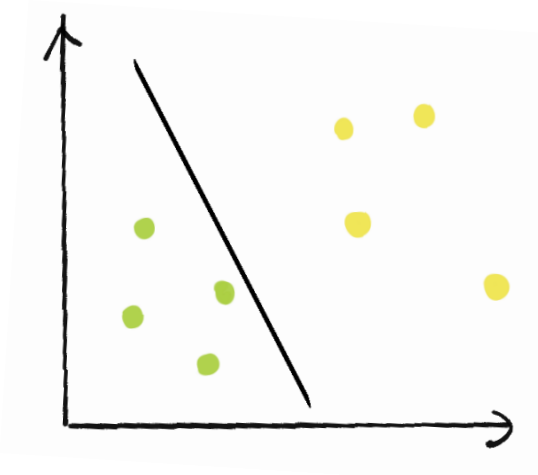
Clicker question: Which decision boundary is more desired?



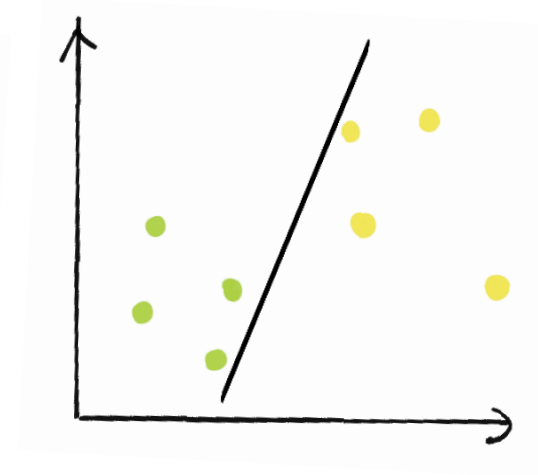
A



B

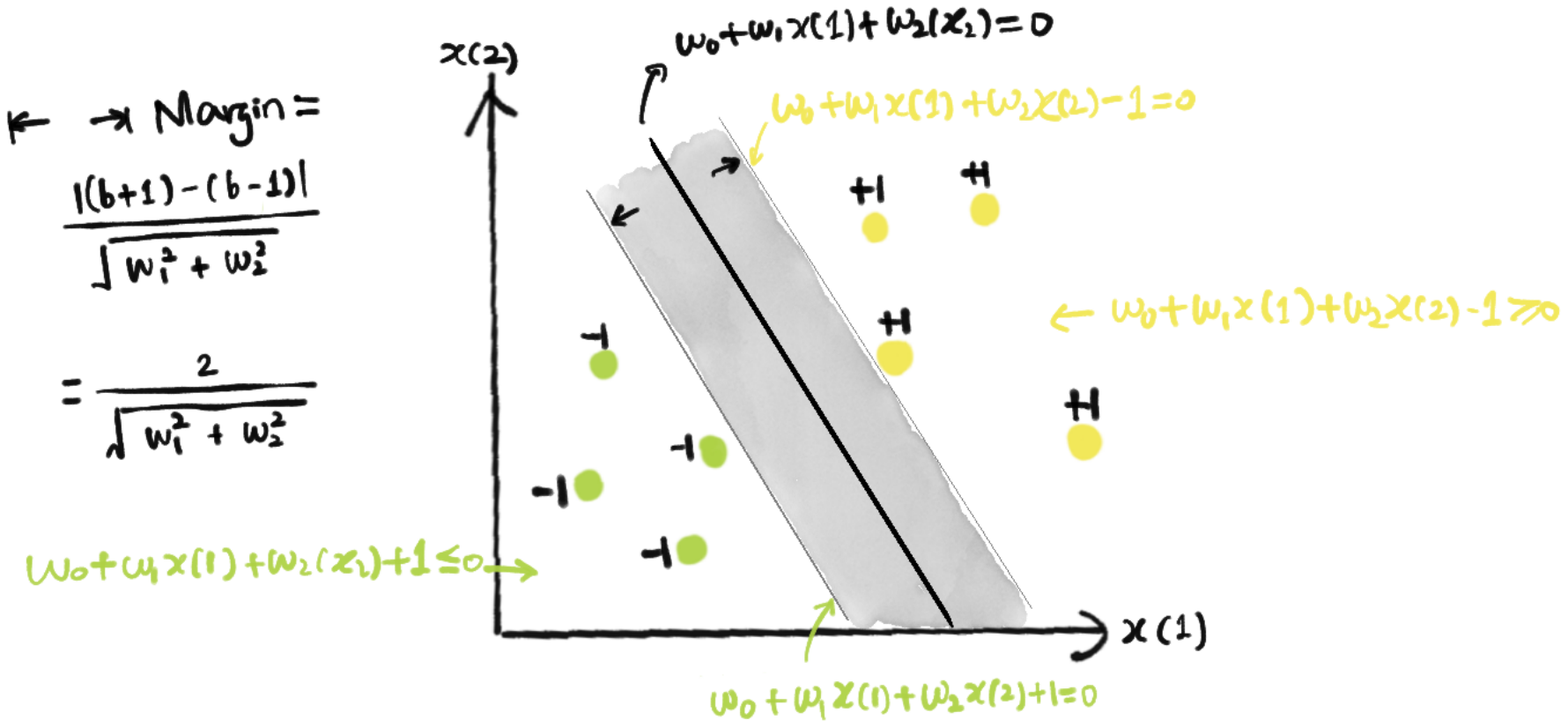


C



D

Maximum Margin



$$y\,[w_0 + w_1 x(1) + w_2 x(2)] \geq 1$$

Maximum Margin

- Measure degree of fit
  - Minimize

$$margin = \frac{\sqrt{w_1^2 + w_2^2}}{2}$$

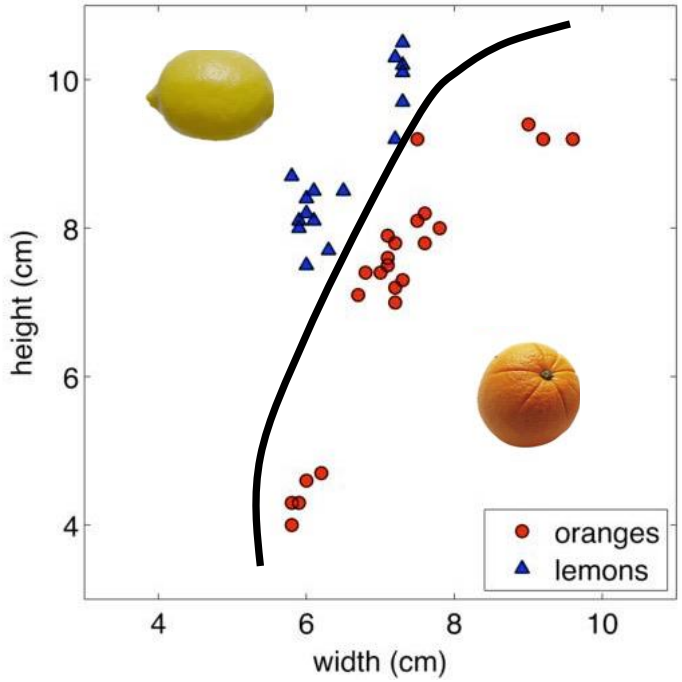
  - Subject to

$$y_n(w_1 x_n(1) + w_2 x_n(2) + w_0) \geq 1, \forall n$$

- Solve the problem with Lagrangian solver

Classification Model: Support Vector Machine (Non-linear)

- Hypothesis space: a line for two features or a hyperplane for more features
  - $H = \{\mathbf{w} | h(\mathbf{w}, \mathbf{x}) = w_0 + w_1\phi(x(1)) + w_2\phi(x(2)) + \cdots + w_m\phi(x(m)) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})\}$



Summary of SVM

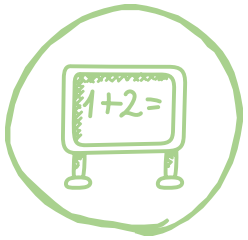
- Provides good generalization ability
- Supports a non-linear transformation
- Robust in high dimensions

- Not suitable for large datasets
- Selection of the non-linear projection function.

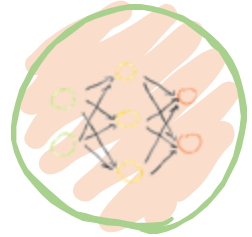
Today

Why Learning?

Categories of Learning



Supervised Learning

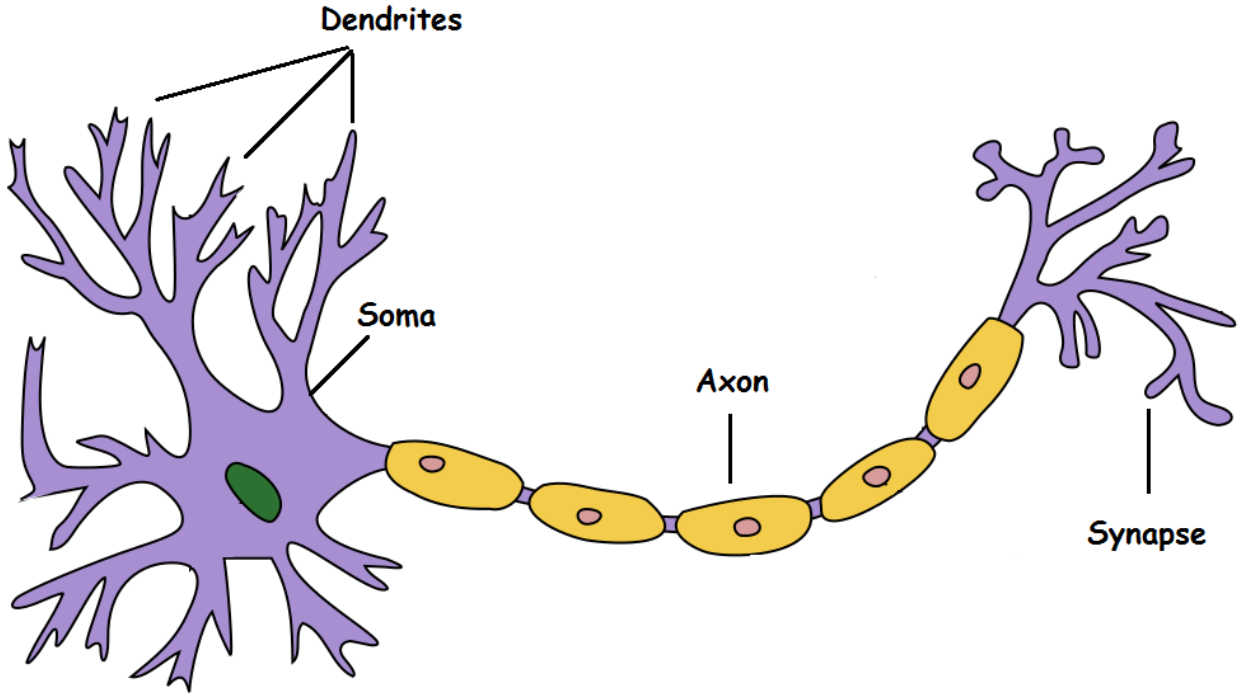


Neural Networks

Neuron in the Brain

- The human brain is made up of 10^{11} neurons > 20 types, 10^{14} synapses, 1ms-10ms cycle time.

- Neurons receive electric signals at the dendrites and send them to the axon.



The Brain vs. Artificial Neural Networks

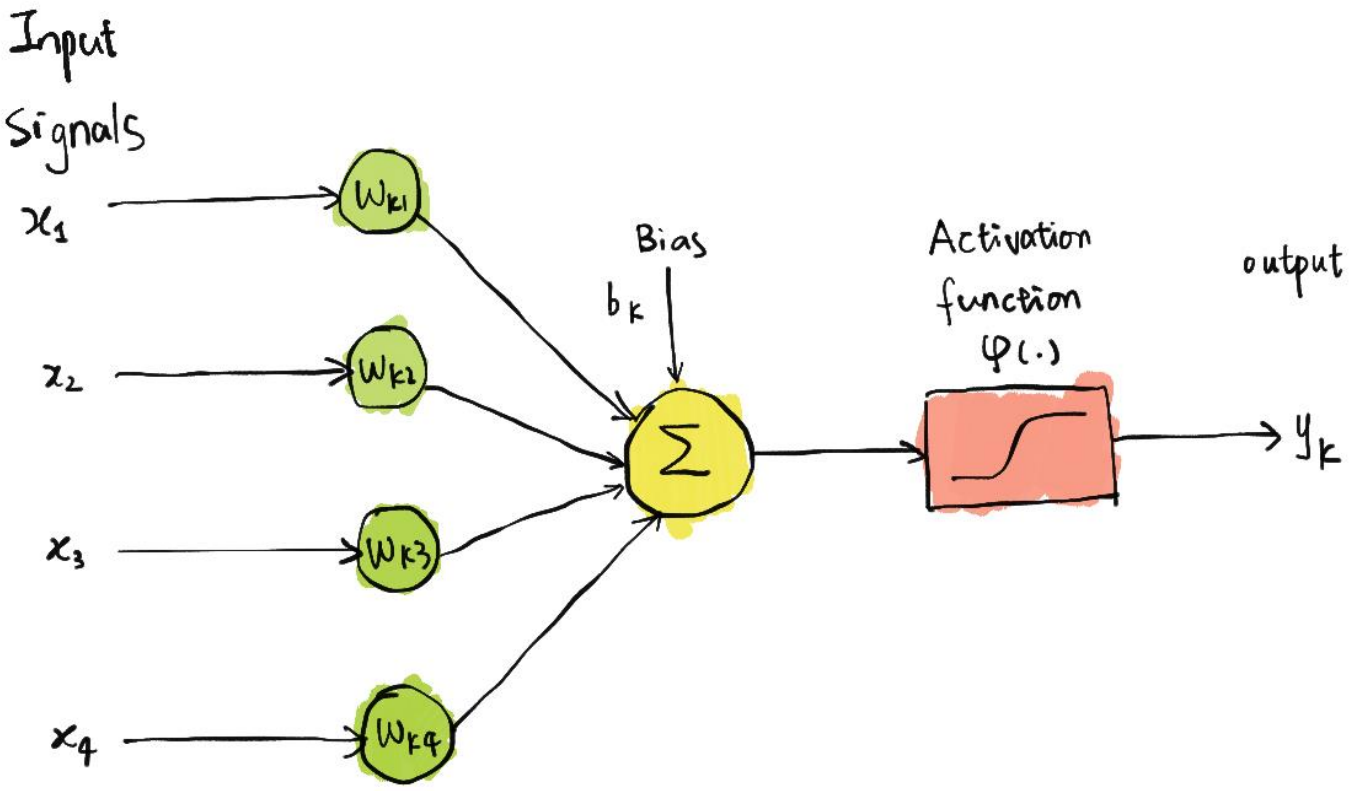
- Similarities
  - Neurons, connections between neurons
  - Learning = change of connections, not change of neurons
  - Massive parallel processing

- But artificial neural networks are much simpler
  - Computation within neuron vastly simplified
  - Discrete time steps
  - Typically some form of supervised learning with massive number of stimuli

McCulloch–Pitts Neuron

- A gross over-simplification of real neurons
- Lay foundation to understanding of what networks of simple units can do.
- Output is a squashed linear function of the inputs.

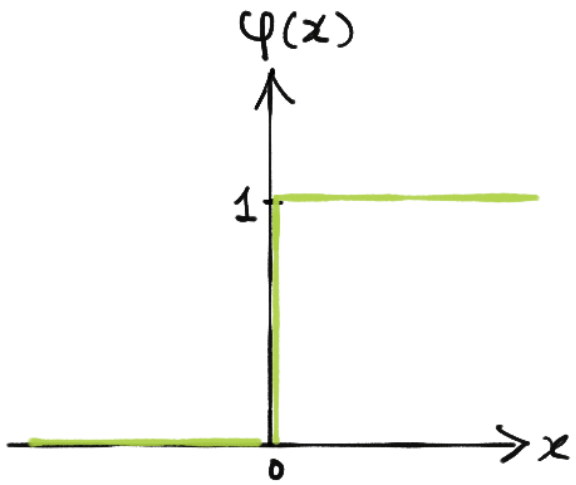
$$y_k = \varphi(\sum_j W_{kj} x_j)$$



Activity Function

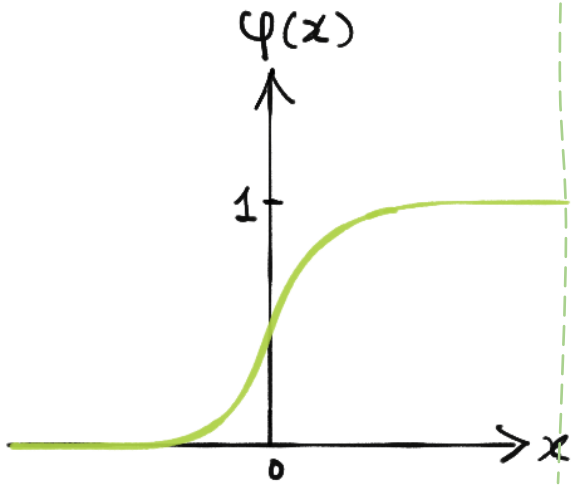
- Introduces "non-linearity" to empower sufficient expressiveness.

Step function



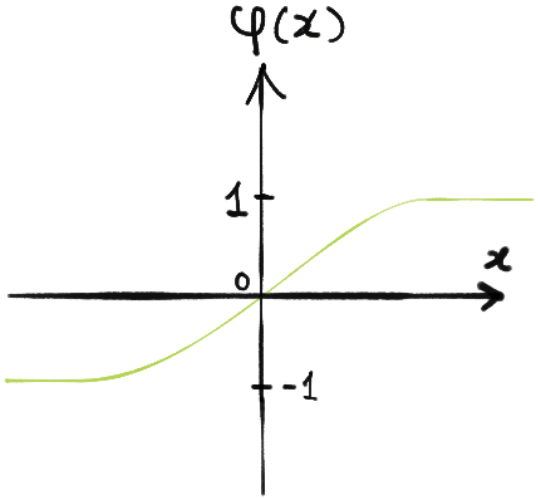
$$\varphi(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

Sigmoid function



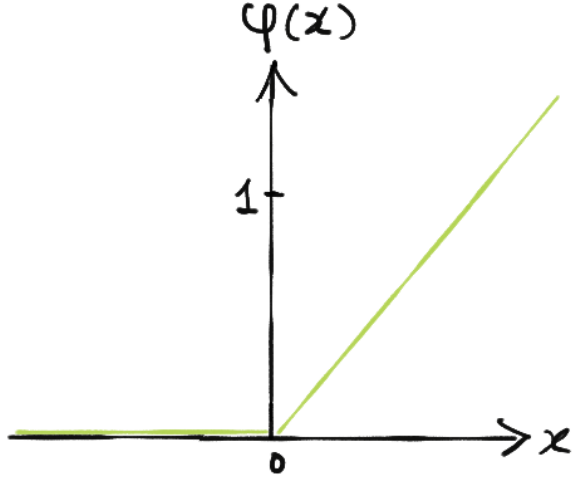
$$\varphi(x) = \frac{1}{1 + e^{-x}}$$

Tanh function



$$\varphi(x) = \tanh x$$

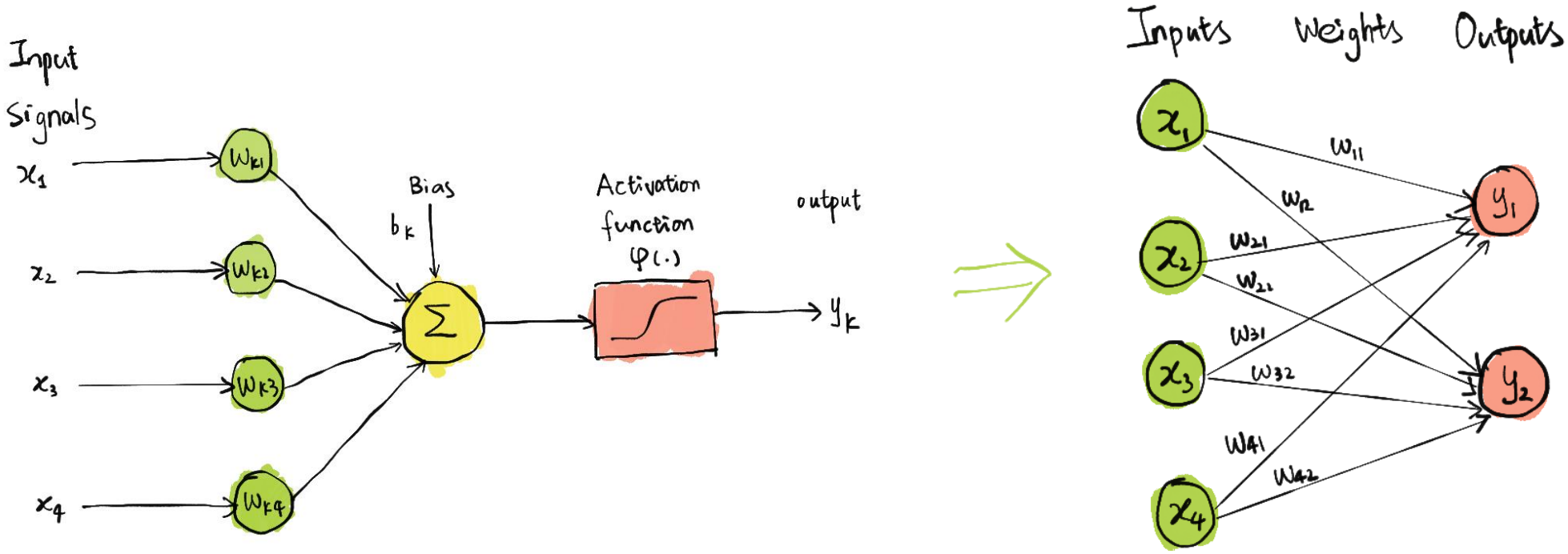
Relu function



$$\varphi(x) = \max(0, x)$$

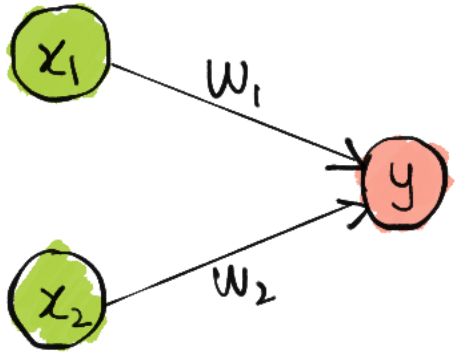
Single-layer Perceptrons

- The simplest neural network.
- Outputs all operate separately with no shared weights

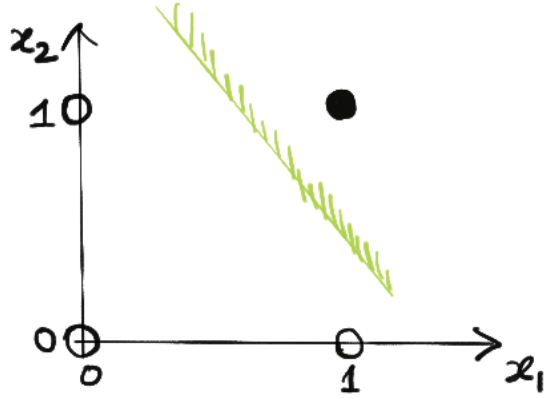


Expressiveness of Single-layer Perceptrons

- Consider the step activation function for a single-layer perceptron.
- Consider using the perceptron to achieve AND, OR, and XOR.
- Can only represent a linear separator in the input space.



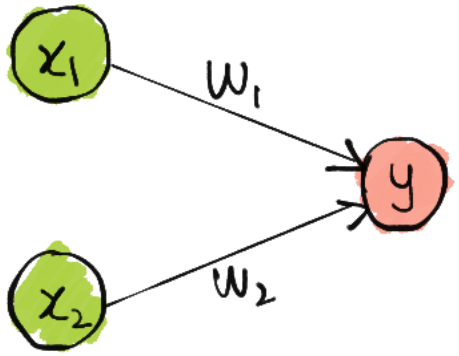
x_1	x_2	y
1	1	1
1	0	0
0	1	0
0	0	0



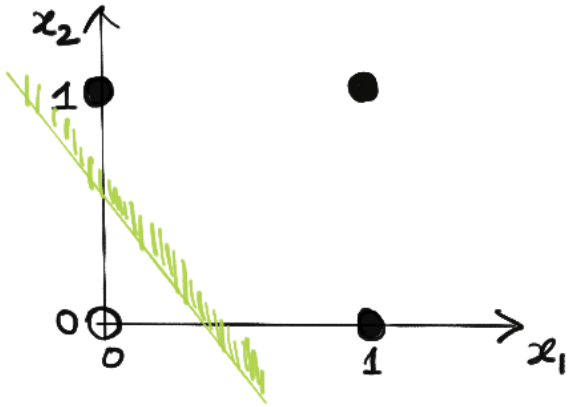
$$y_k = 1, w_1 x_1 + w_2 x_2 + b \geq 0$$

Expressiveness of Single-layer Perceptrons

- Consider the step activation function for a single-layer perceptron.
- Consider using the perceptron to achieve AND, OR, and XOR.
- Can only represent a linear separator in the input space.



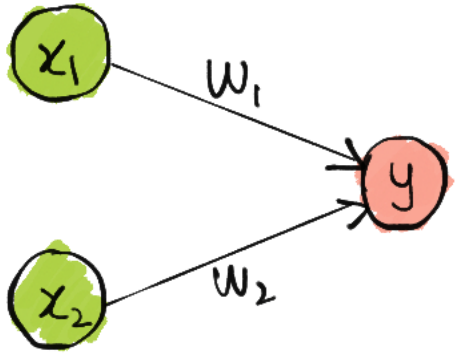
x_1	x_2	y
1	1	1
1	0	1
0	1	1
0	0	0



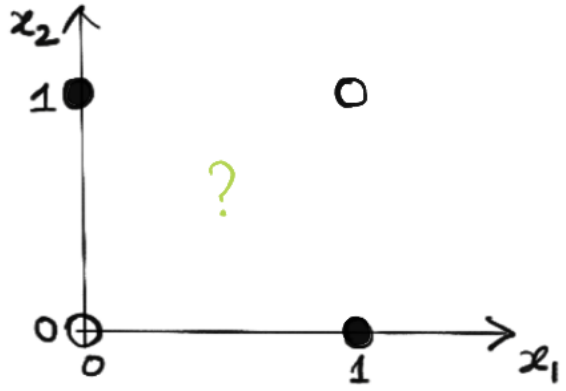
$$y_k = 1, w_1 x_1 + w_2 x_2 + b \geq 0$$

Expressiveness of Single-layer Perceptrons

- Consider the step activation function for a single-layer perceptron.
- Consider using the perceptron to achieve AND, OR, and XOR.
- Can only represent a linear separator in the input space.



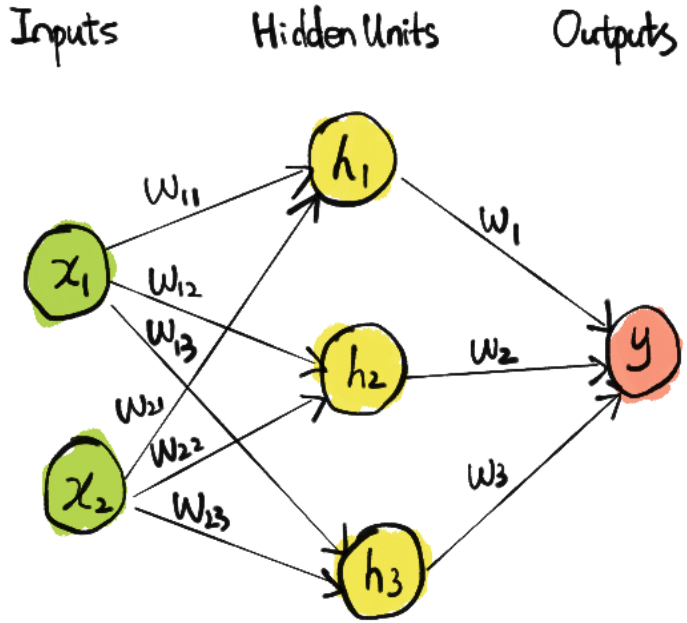
x_1	x_2	y
1	1	0
1	0	1
0	1	1
0	0	0



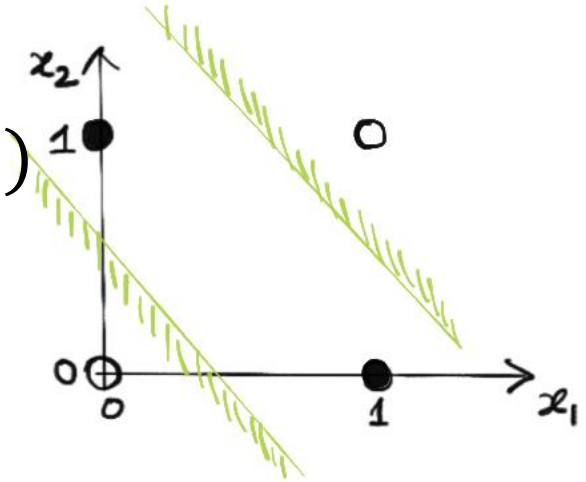
$$y_k = 1, w_1 x_1 + w_2 x_2 + b \geq 0$$

Multi-layer Perceptrons (MLP)

- Layers are usually fully connected.
- Introduce more nonlinearity.
- The numbers of hidden units are typically chosen by hand.



$$y = \varphi[w_1\varphi(w_{11}x_1 + w_{21}x_2) + w_2\varphi(w_{12}x_1 + w_{22}x_2) + w_3\varphi(w_{13}x_1 + w_{23}x_2)]$$



Simple MLP for XOR

- Define the loss function to be the sum of squared loss

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\hat{y}^n(x_1^n, x_2^n, \mathbf{w}) - y^n)^2$$

x_1	x_2	y
1	1	0
1	0	1
0	1	1
0	0	0

- Initialize all the weights to be zero.

$$w_{11} = w_{12} = w_{13} = w_{21} = w_{22} = w_{23} = w_1 = w_2 = w_3 = 1$$

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}$$

Derivative by Chain Rule

$$\frac{\partial E(w_1)}{\partial w_1} = \frac{\partial E(w_1)}{\partial \hat{y}^n} \cdot \frac{\partial \hat{y}^n}{\partial o^n} \cdot \frac{\partial o^n}{\partial w_1}$$

$$\frac{\partial E(w_1)}{\partial \hat{y}^n} = \frac{\partial}{\partial \hat{y}^n}\left[\frac{1}{2}\sum_{n=1}^{N}(\hat{y}^n - y^n)^2\right] = \sum_{n=1}^{N}(\hat{y}^n - y^n)$$

$$\frac{\partial \hat{y}^n}{\partial o^n} = \frac{\partial \text{sigmoid}(o^n)}{\partial o^n} = \text{sigmoid}(o^n)(1-\text{sigmoid}(o^n))$$
$$= \hat{y}^n(1-\hat{y}^n)$$

$$\frac{\partial o^n}{\partial w_1} = \text{sigmoid}(w_{11}x_1^n + w_{21}x_2^n) = \text{sigmoid}(o_1^n)$$

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\hat{y}^n(x_1^n, x_2^n, \mathbf{w}) - y^n)^2$$

$$\hat{y}^n = \varphi[w_1\varphi(w_{11}x_1^n + w_{21}x_2^n) + w_2\varphi(w_{12}x_1^n + w_{22}x_2^n) + w_3\varphi(w_{13}x_1^n + w_{23}x_2^n)]$$

$$\hat{y}^n = \varphi(o^n)$$

$$o^n = w_1\varphi(o_1^n) + w_2\varphi(o_2^n) + w_3\varphi(o_3^n)$$

$$o_1^n = w_{11}x_1^n + w_{21}x_2^n$$

Derivative by Chain Rule

$$\frac{\partial E(w_{11})}{\partial w_{11}} = \frac{\partial E(w_{11})}{\partial \hat{y}^n} \cdot \frac{\partial \hat{y}^n}{\partial o^n} \cdot \frac{\partial o^n}{\partial o_1^n} \cdot \frac{\partial o_1^n}{\partial w_{11}}$$

$$\frac{\partial E(w_1)}{\partial \hat{y}^n} = \frac{\partial}{\partial \hat{y}^n}\left[\frac{1}{2}\sum_{n=1}^{N}(\hat{y}^n - y^n)^2\right] = \sum_{n=1}^{N}(\hat{y}^n - y^n)$$

$$\frac{\partial \hat{y}^n}{\partial o^n} = \frac{\partial \text{sigmoid}(o^n)}{\partial o^n} = \text{sigmoid}(o^n)(1-\text{sigmoid}(o^n))$$
$$= \hat{y}^n(1-\hat{y}^n)$$

$$\frac{\partial o^n}{\partial o_1^n} = w_1 \cdot \frac{\partial \text{sigmoid}(o_1^n)}{\partial o_1^n}$$
$$= w_1 \,\text{sigmoid}(o_1^n)(1-\text{sigmoid}(o_1^n))$$

$$\frac{\partial o_1^n}{\partial w_{11}} = \frac{\partial (w_{11}x_1^n + w_{21}x_2^n)}{\partial w_{11}} = x_1^n$$

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\hat{y}^n(x_1^n, x_2^n, \mathbf{w}) - y^n)^2$$

$$\hat{y}^n = \varphi[w_1\varphi(w_{11}x_1^n + w_{21}x_2^n) + w_2\varphi(w_{12}x_1^n + w_{22}x_2^n) + w_3\varphi(w_{13}x_1^n + w_{23}x_2^n)]$$

$$\hat{y}^n = \varphi(o^n)$$

$$o^n = w_1\varphi(o_1^n) + w_2\varphi(o_2^n) + w_3\varphi(o_3^n)$$

$$o_1^n = w_{11}x_1^n + w_{21}x_2^n$$

Forward Pass

$x_1^1 = 1,\ x_2^1 = 1,\ y^1 = 0$

$o_1^1 = 1 \times 1 + 1 \times 1 = 2$

$o^1 = 1 \times \text{sigmoid}(2) + 1 \times \text{sigmoid}(2) + 1 \times \text{Sigmoid}(2) = 2.6424$

$\hat{y}^1 = \text{sigmoid}(2.6424) = 0.9335$

$x_1^2 = 1,\ x_2^2 = 0,\ y^2 = 1$

$o_1^2 = 1 \times 1 + 1 \times 0 = 1$

$o^2 = 1 \times \text{sigmoid}(1) + 1 \times \text{sigmoid}(1) + 1 \times \text{sigmoid}(1) = 2.1932$

$\hat{y}^2 = \text{sigmoid}(2.1932) = 0.8996$

$x_1^3 = 0,\ x_2^3 = 1,\ y^3 = 1$

$o_1^3 = 1 \times 0 + 1 \times 1 = 1$

$o^3 = 1 \times \text{sigmoid}(1) + 1 \times \text{sigmoid}(1) + 1 \times \text{sigmoid}(1) = 2.1932$

$\hat{y}^3 = \text{sigmoid}(2.1932) = 0.8996$

$x_1^4 = 0,\ x_2^4 = 0,\ y^4 = 0$

$o_1^4 = 1 \times 0 + 1 \times 0 = 0$

$o^4 = 1 \times \text{sigmoid}(0) + 1 \times \text{sigmoid}(0) + 1 \times \text{sigmoid}(0) = 1.5$

$\hat{y}^4 = \text{sigmoid}(1.5) = 0.8176$

Back-propagation

$$\frac{\partial E(w_1)}{\partial w_1} = \sum_{n=1}^{N} (\hat{y}^n - y^n)\,\hat{y}^n(1-\hat{y}^n)\,\text{sigmoid}(o_1^n)$$

$$= (0.9335 - 0)\,0.9335(1-0.9335)\,\text{sigmoid}(2)$$
$$+ (0.8996 - 1)\,0.8996(1-0.8996)\,\text{sigmoid}(1)$$
$$+ (0.8996 - 1)\,0.8996(1-0.8996)\,\text{sigmoid}(1)$$
$$+ (0.8176 - 0)\,0.8176(1-0.8176)\,\text{sigmoid}(0)$$

$$= 0.0988$$

$$w_1 = w_1 - \alpha \frac{\partial E(w_1)}{\partial w_1}$$

$$= 1 - 0.5 \times 0.0988$$

$$= 0.9506$$

Back-propagation

$$\frac{\partial E(w_{11})}{\partial w_{11}} = \sum_{n=1}^{N} (\hat{y}^n - y^n)\, \hat{y}^n (1-\hat{y}^n)\, w_1\, \text{sigmoid}(o_1^n)(1-\text{sigmoid}(o_1^n))\, x_1^n$$

$$= (0.9335 - 0)\, 0.9335 (1-0.9335)\, 1\, \text{sigmoid}(2)(1-\text{sigmoid}(2))\, 1$$
$$+ (0.8996 - 1)\, 0.8996 (1-0.8996)\, 1\, \text{sigmoid}(1)(1-\text{sigmoid}(1))\, 1$$
$$+ (0.8996 - 1)\, 0.8996 (1-0.8996)\, 1\, \text{sigmoid}(1)(1-\text{sigmoid}(1))\, 0$$
$$+ (0.8176 - 0)\, 0.8176 (1-0.8176)\, 1\, \text{sigmoid}(0)(1-\text{sigmoid}(0))\, 0$$
$$= 0.0043$$

$$w_{11} = w_{11} - \alpha \frac{\partial E(w_{11})}{\partial w_{11}}$$
$$= 1 - 0.5 \times 0.0043$$
$$= 0.9979$$

Beyond Multi-layer Perceptrons

- More layers = deep learning

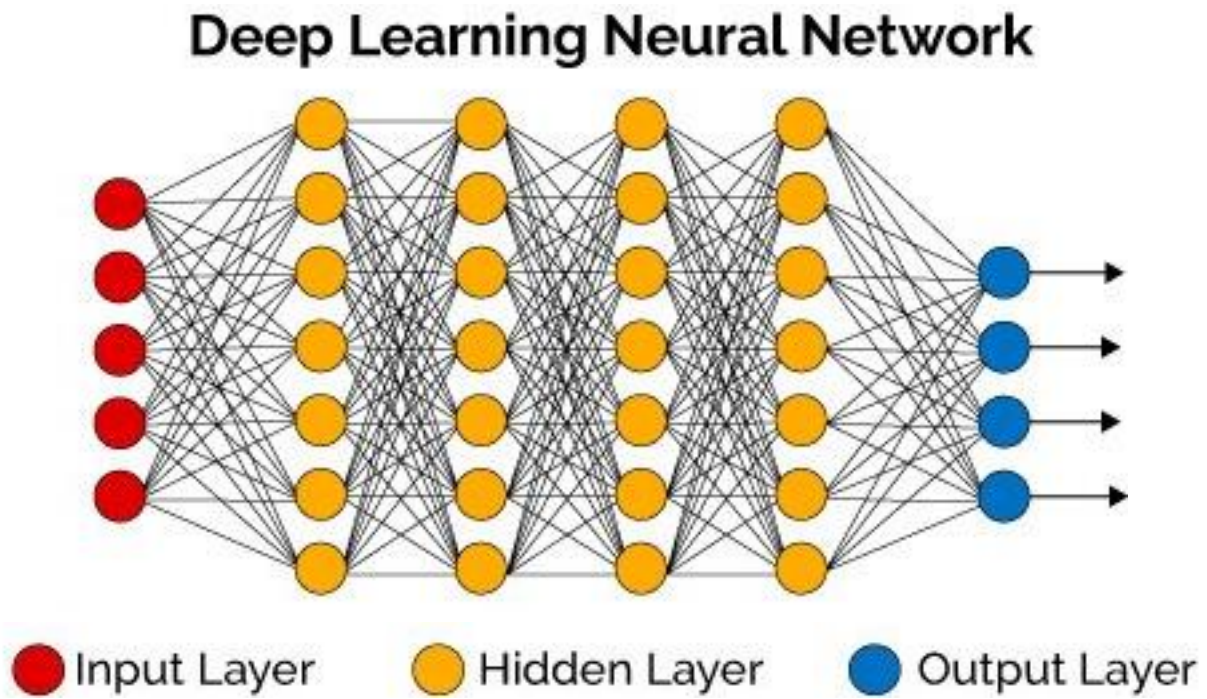
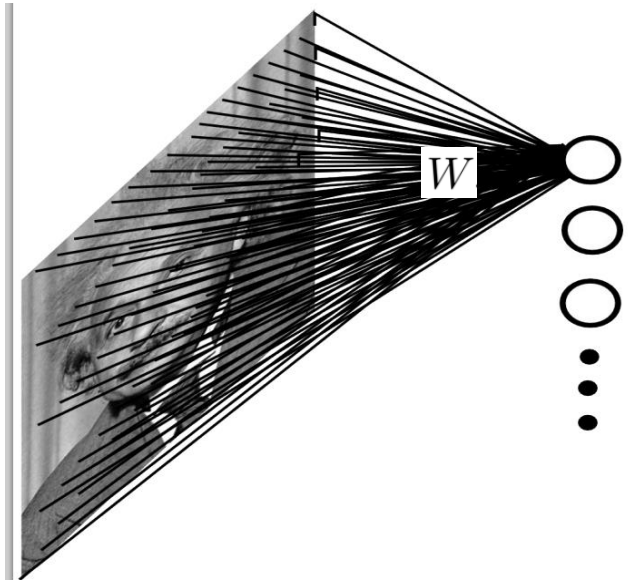
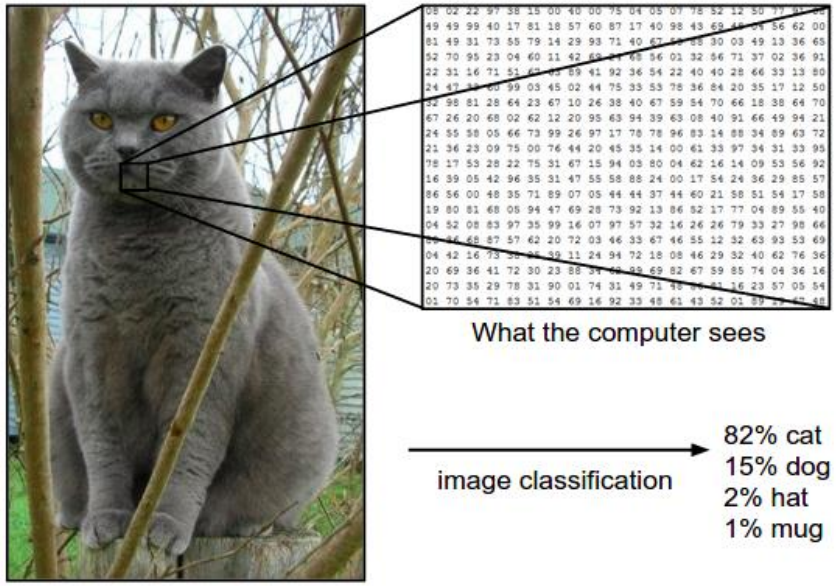


Image as Inputs

- When the input data is an image, the number of parameters is huge to train and learn.
- 1000 x 1000 image connected to 1M hidden units will produce 1B parameters!

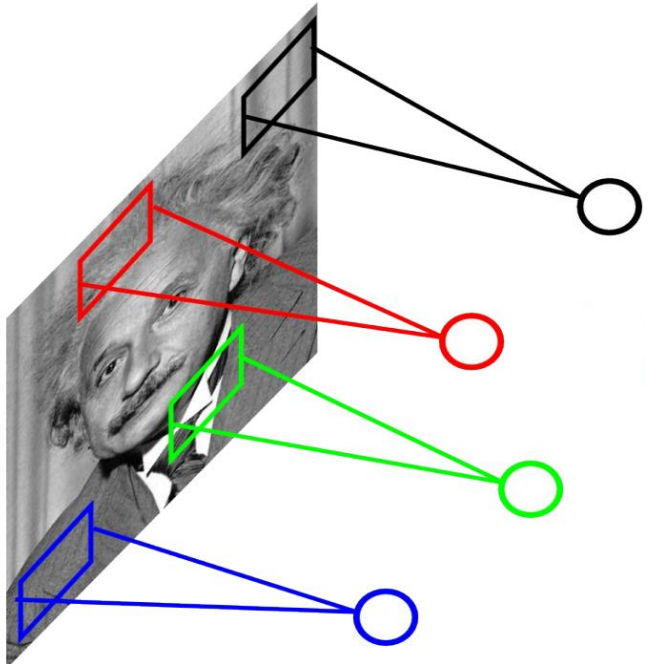




Source: http://cs231n.github.io/classification/

Convolutional Neural Networks

- Reduce connection to local regions, as spatial correlation is local.
- Share the same parameters across different locations.
- 1000 x 1000 image connected to 1M hidden units will reduce to 10,000 parameters with 100 filters in the filter size of 10 x 10!



Convolution

- Convolution is the main building block of a CNN.
- The filter or kernel (green) is the parameters to learn.
- The filter is sliding over out input (blue) and the sum of the convolution goes into the feature map (red).

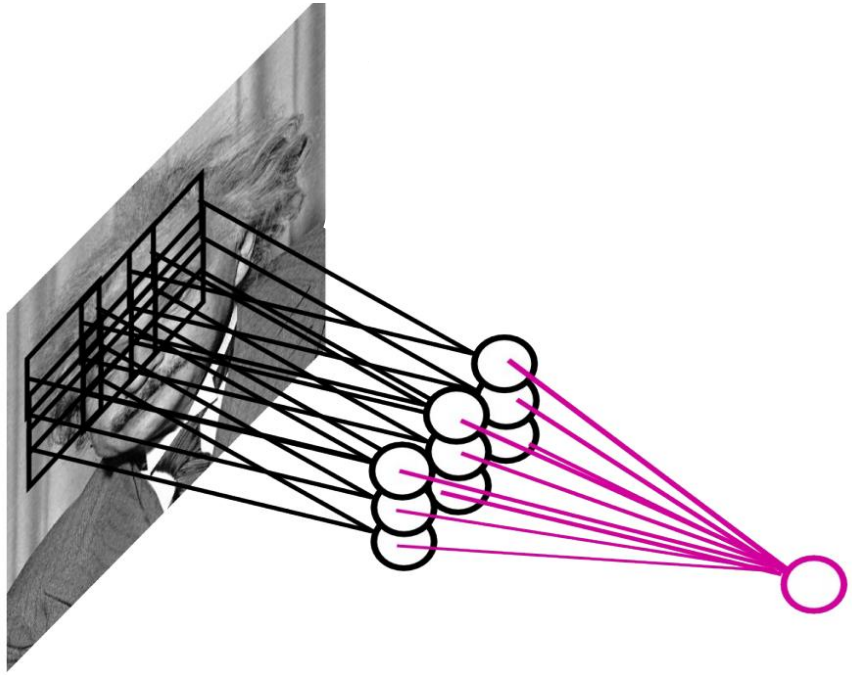
1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

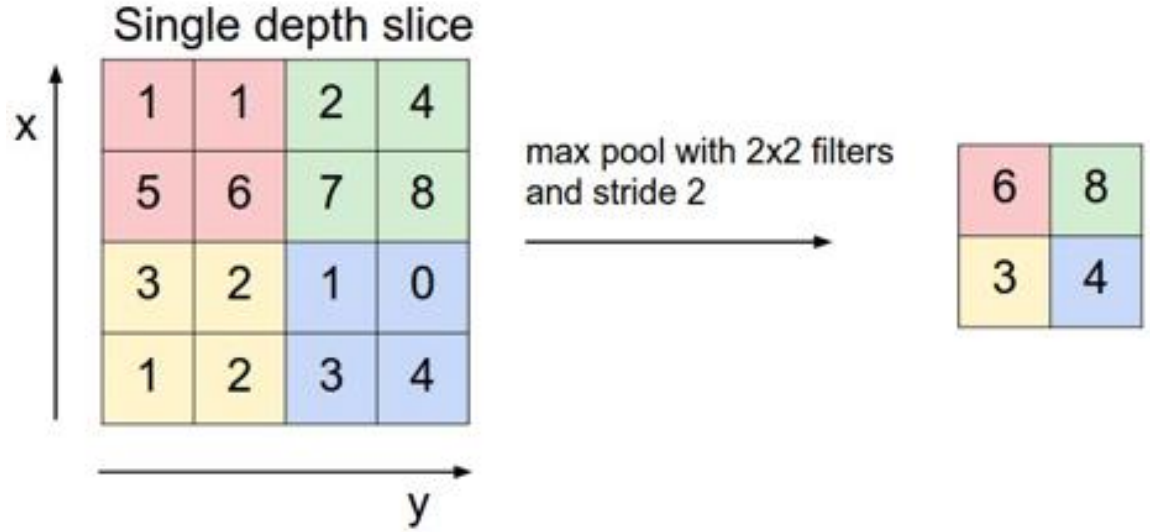
4		

Source: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Max-pooling

- How can we make the learned features robust to the exact location of a pattern?
- Further reduces the dimension of the features

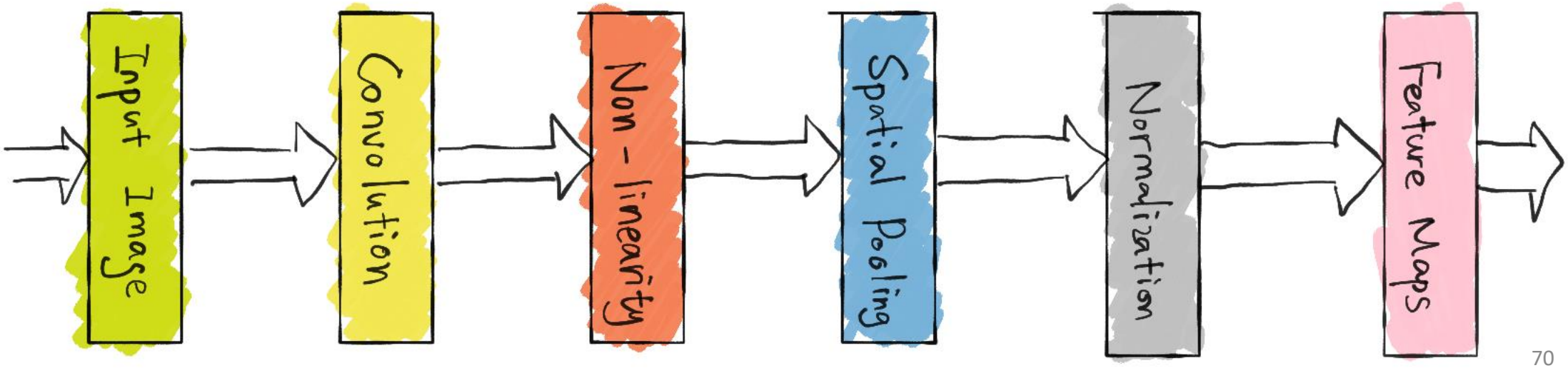




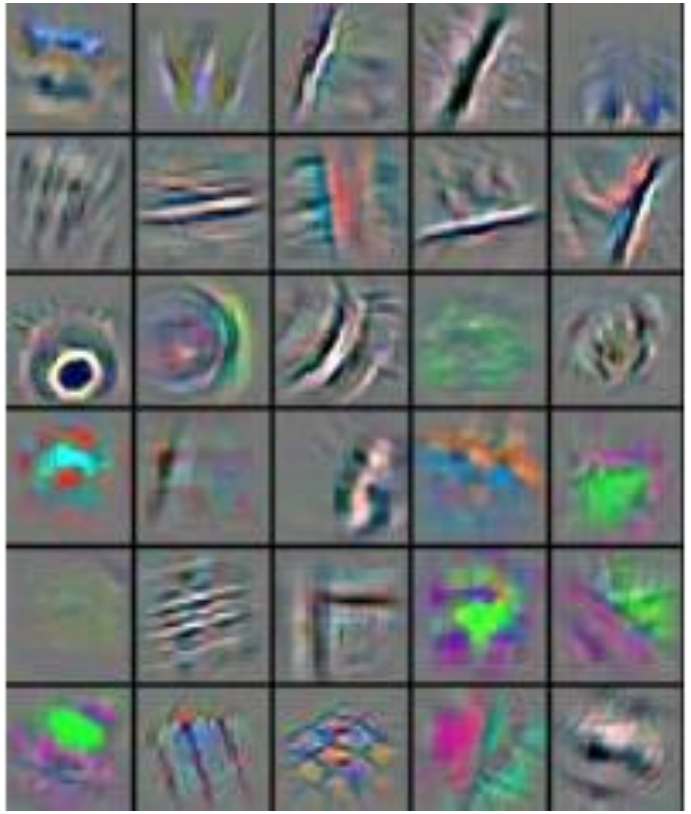
Source: http://cs231n.github.io/convolutional-networks/

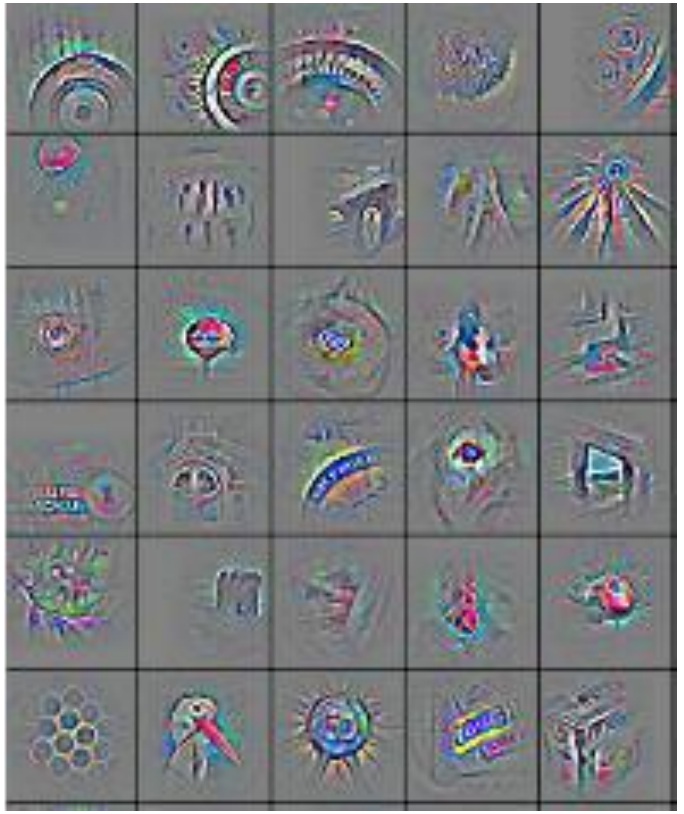
Summary of Typical Convolutional Layers

- Stack all the layers up just like multi-layer perceptrons.
- Pooling and normalization is optional.
- Final layer is usually fully connected neural network with the output size equal to the number of classes.



Visualization of Features at Different Layers



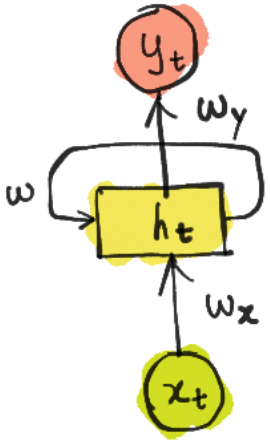


Recurrent Neural Networks

- The sequence of data matters, e.g., for text generation and stock price prediction.
- Conventional neural networks have fixed input/output size.

Convention neural networks

Recurrent Neural Networks

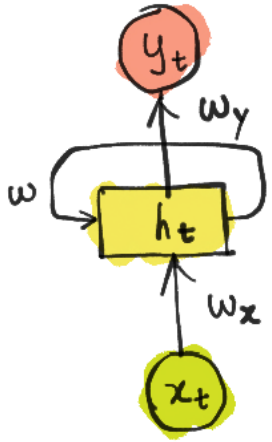


Recurrent Neural Networks

- x_t is the input at time t.
- h_t is the hidden state (memory) at time t.
- y_t is the output at time t.
- w, w_x, w_y are distinct weights and remain the same at all time steps.

$$h_t = w\varphi(h_{t-1}) + w_x x_t$$

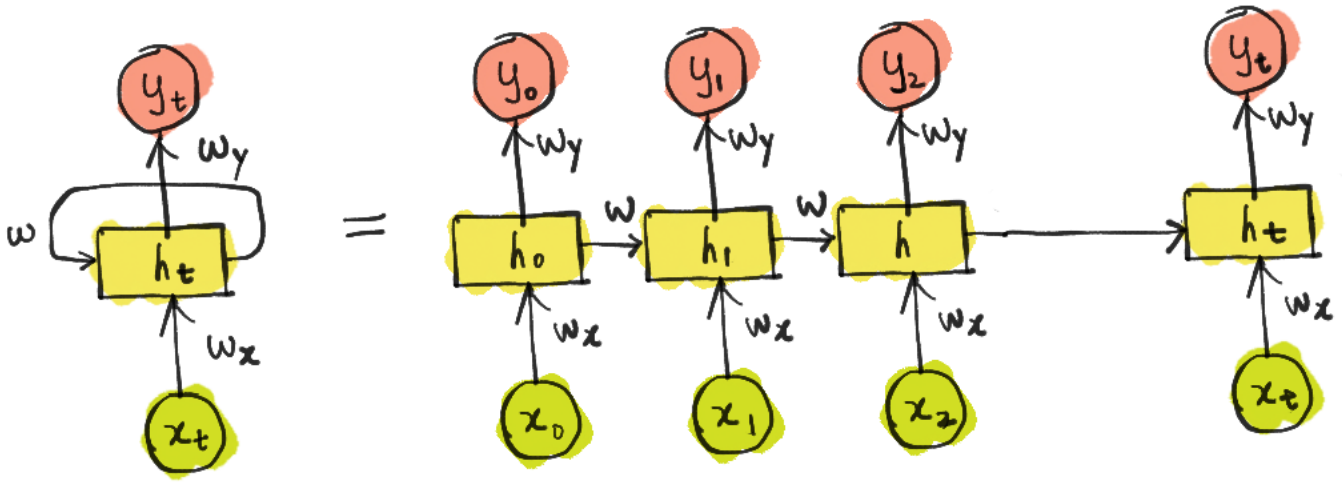
$$y_t = w_y \varphi(h_t)$$



Unrolled RNNs

- RNNs can be thought of as multiple copies of the same network, each passing a message to a successor.

- The same function and the same set of parameters are used at every time step.



Goals

- Understand why learning is needed for intelligent agents.
- Understand what the difference between three categories of learning is.
- Understand the basic supervised learning considerations and pipeline.
- Learn basic procedures and principles of different learning algorithms.
- Learn how to use Python packages to implement the supervised learning.

Important This Week

- Read Chapter 18.7 and 20.

- I strongly encourage you to go through this blog https://machinelearningmastery.com/start-here/ , If you are interested in machine learning and want to do more. They are definitely beneficial.