DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

December 2 2019

Logistics

- Final exam
  - Wed, Dec 4, 2:50-5pm, in class
  - Office hours: after class today, Tuesday 1-2pm
  - No office hours on Wed
- Final project
  - Presentations: Mon, Dec 9, 1-5pm, ISEC 655
  - 8 minutes per team
  - Final report due on Gradescope on Tue, Dec. 10
  - No late days for final report! Please submit on time

Final Exam Review

What we covered

Ensembles
- Bagging
- Random forests
- Boosting
- AdaBoost

Deep learning
- Feed-forward Neural Nets
- Convolutional Neural Nets
- Architectures
- Forward/back propagation

Linear classification
- Perceptron
- Logistic regression
- LDA
- Linear SVM

Non-linear classification
- kNN
- Decision trees
- Kernel SVM
- Naïve Bayes

- Metrics
- Cross-validation
- Regularization
- Feature selection
- Gradient Descent
- Density Estimation

Linear Regression

Linear algebra

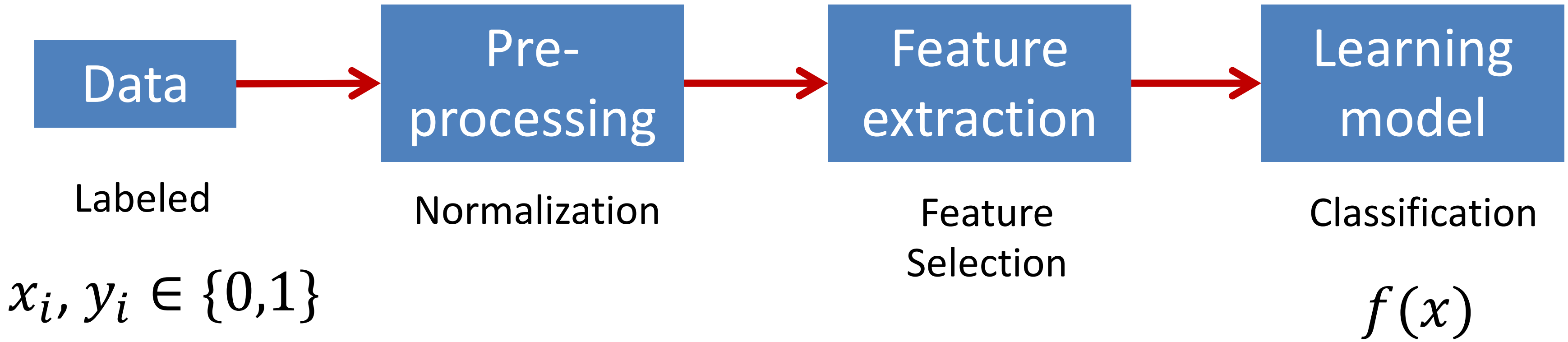
Probability and statistics

Terminology

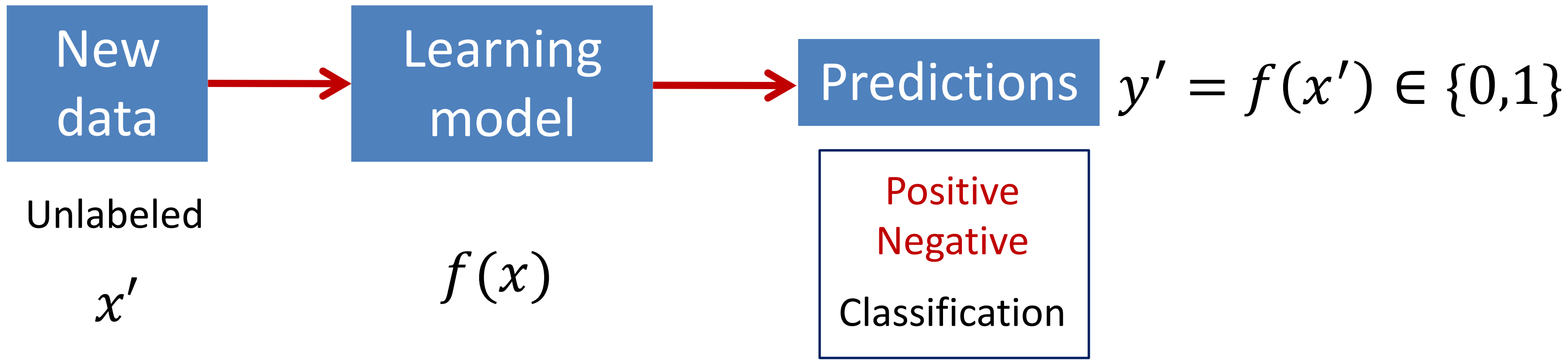
- Hypothesis space $H = \{f: X \rightarrow Y\}$
- Training data $\mathrm{D} = (x_{\mathrm{i}}, y_{\mathrm{i}}) \in X \times Y$
- Features: $x_i \in X$
- Labels / response variables $y_{\mathrm{i}} \in Y$
  - Classification: discrete $y_{\mathrm{i}} \in \{0,1\}$
  - Regression: $y_{\mathrm{i}} \in \mathrm{R}$
- Loss function: $L(f, D)$
  - Measures how well f fits training data
- Training algorithm: Find hypothesis $\hat{f}: X \rightarrow Y$
  - $\hat{f} = \underset{f \in H}{\operatorname{argmin}} \; L(f, D)$

Supervised Learning: Classification

Training

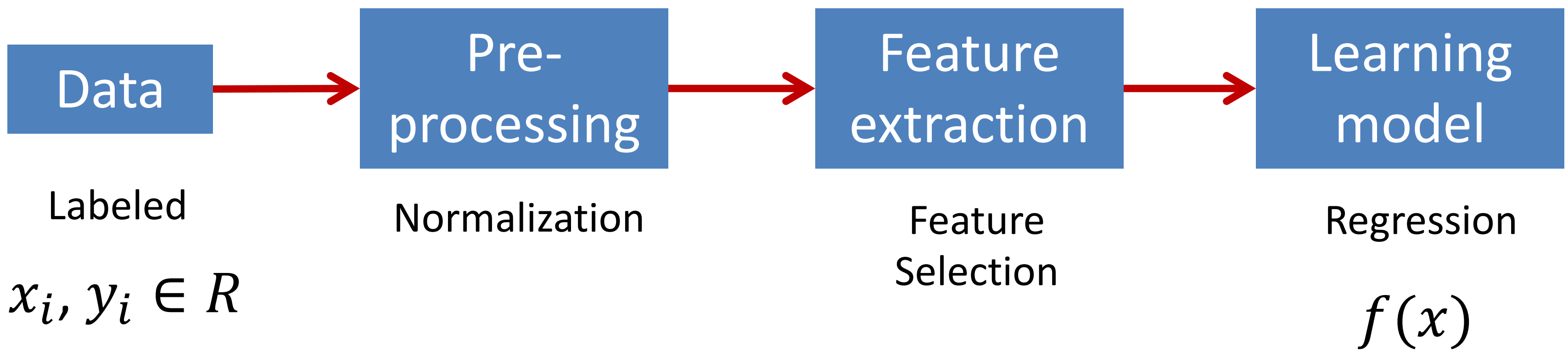


Testing

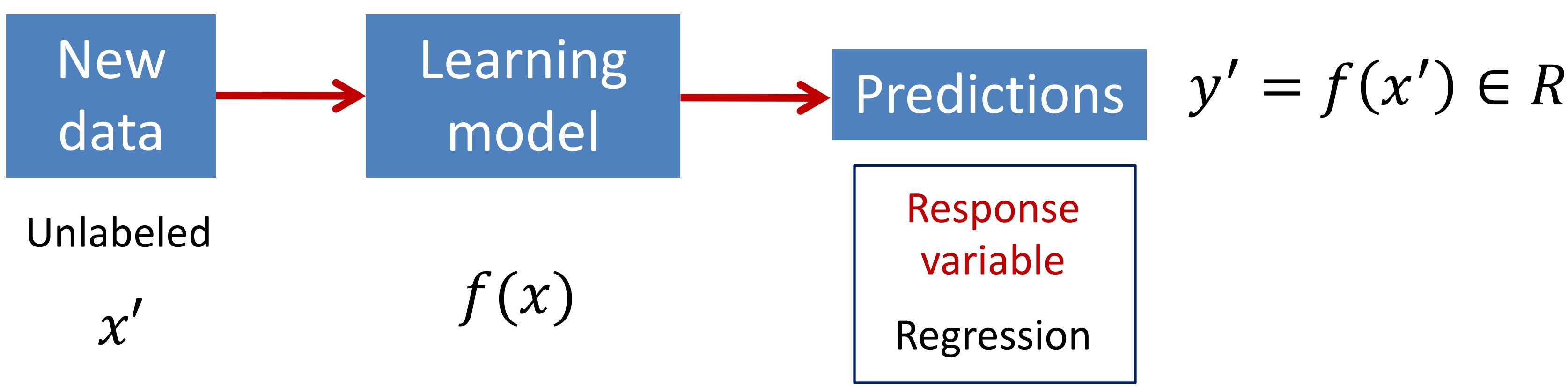


Supervised Learning: Regression

Training



Testing



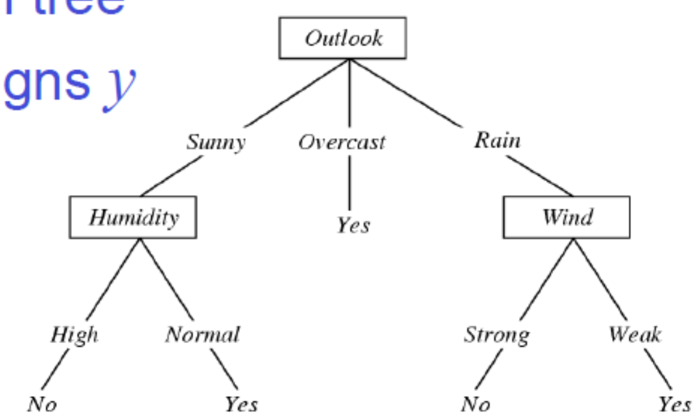
Methods for Feature Selection

- Wrappers
  - Select subset of features that gives best prediction accuracy (using cross-validation)
  - Model-specific
- Filters
  - Compute some statistical metrics (correlation coefficient, mutual information)
  - Select features with statistics higher than threshold
- Embedded methods
  - Feature selection done as part of training
  - Example: Regularization (Lasso, L1 regularization)

Decision Tree Learning

Problem Setting:

- Set of possible instances X
  - each instance x in X is a feature vector
  - e.g., <*Humidity=low, Wind=weak, Outlook=rain, Temp=hot*>
- Unknown target function $f : X \rightarrow Y$
  - Y is discrete valued
- Set of function hypotheses $H=\{ h \mid h : X \rightarrow Y \}$
  - each hypothesis h is a decision tree
  - trees sorts x to leaf, which assigns y



Slide by Tom Mitchell

Learning Decision Trees

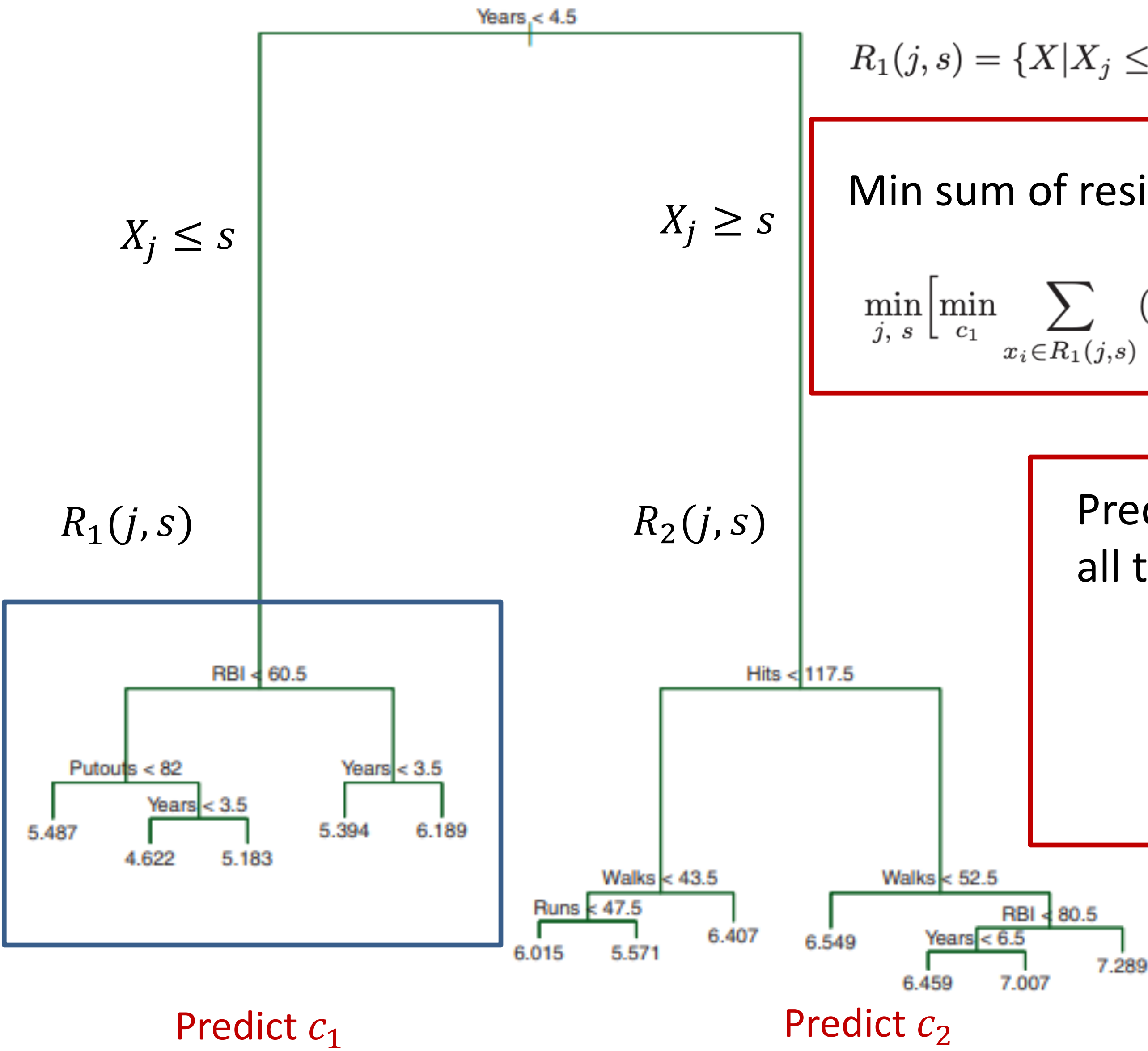
- Start from empty decision tree
- Split on **next best attribute (feature)**
  - Use, for example, information gain to select attribute:

$$\arg\max_i IG(X_i) = \arg\max_i H(Y) - H(Y \mid X_i)$$

- Recurse

ID3 algorithm uses Information Gain
Information Gain reduces uncertainty on Y

Regression Trees



$$R_1(j, s) = \{X|X_j \leq s\} \text{ and } R_2(j, s) = \{X|X_j > s\}.$$

Min sum of residual on both branches

$$\min_{j,\ s}\left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2\right].$$

Predict average response of all training data at each leaf

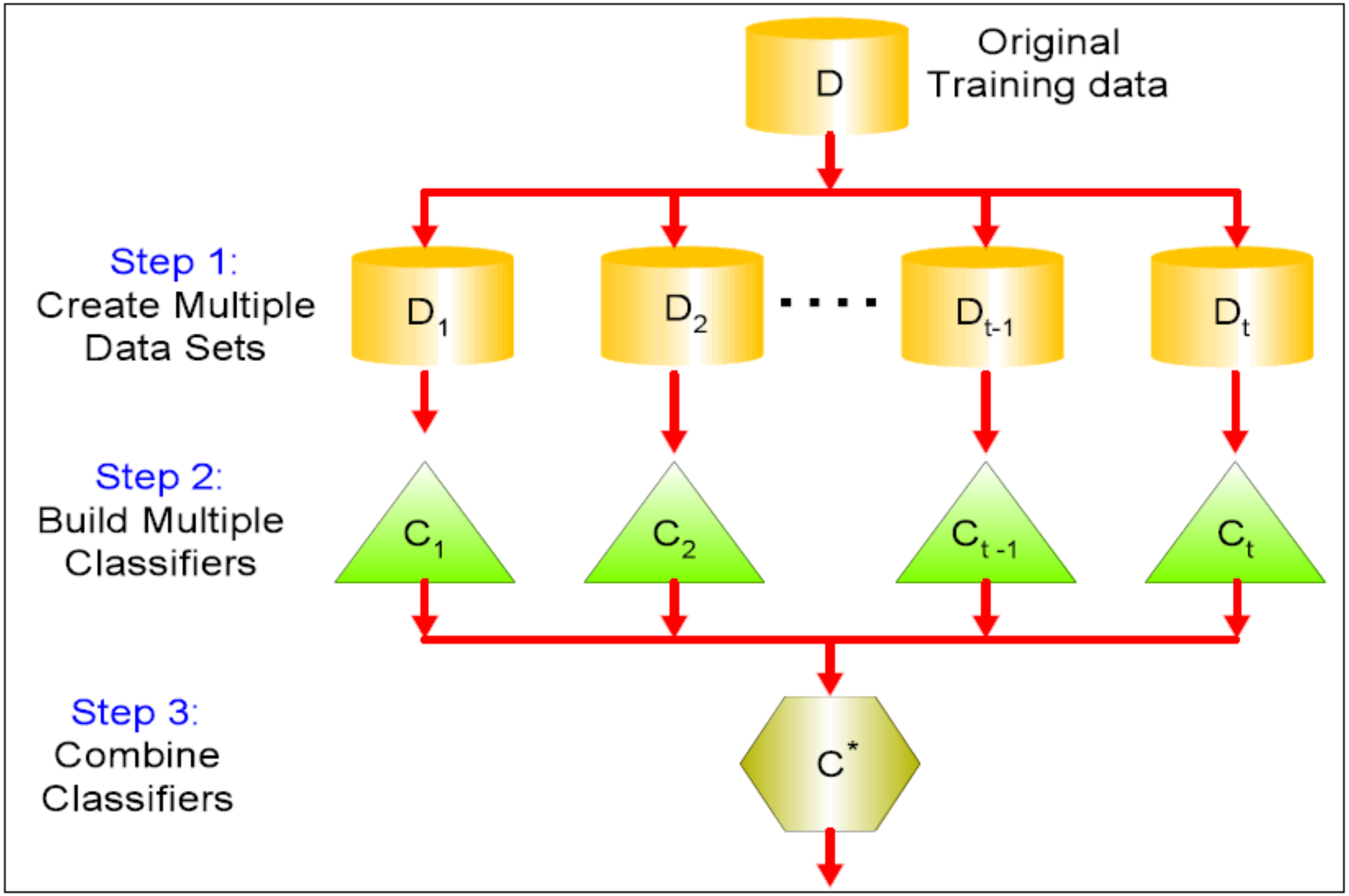
$$\hat{c}_1 = \text{ave}(y_i | x_i \in R_1(j, s))$$

$$\hat{c}_2 = \text{ave}(y_i | x_i \in R_2(j, s)).$$

Decision trees topics

- Entropy, Conditional entropy
- Information gain
- How to train a decision tree
  - Recursive algorithm
  - Impurity metrics (Gini, information gain)
- How to evaluate a decision tree
- Strategies to prevent overfitting
  - Pruning

Ensembles



Majority Votes

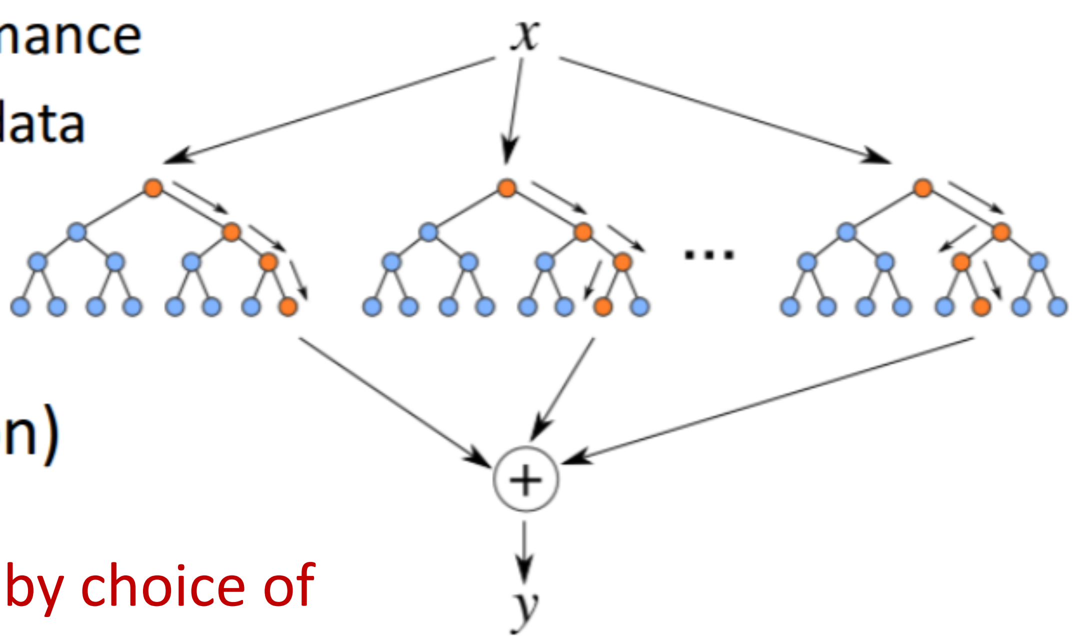
Bagging

- Leo Breiman (1994)
- Take repeated bootstrap samples from training set D
- *Bootstrap sampling*: Given set D containing N training examples, create D' by drawing N examples at random with replacement from D.

- Bagging:
  - Create k bootstrap samples $D_1 \ldots D_k$.
  - Train distinct classifier on each D_i.
  - Classify new instance by majority vote / average.

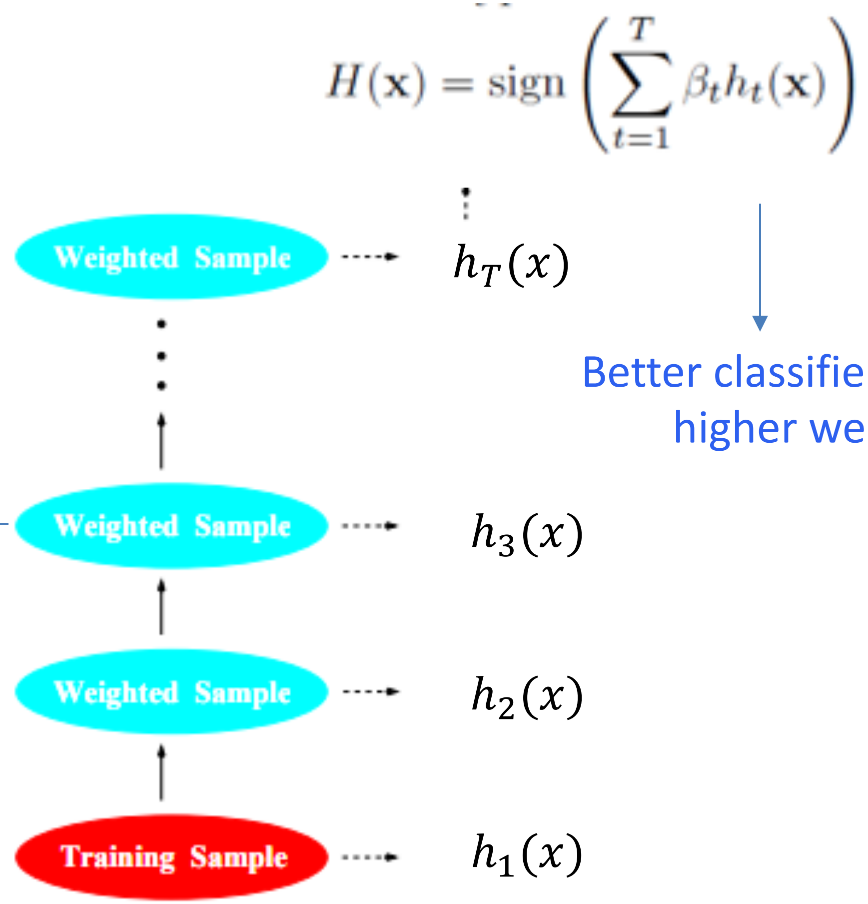
Random Forests

- Construct decision trees on bootstrap replicas
  - Restrict the node decisions to a small subset of features picked randomly for each node
- Do not prune the trees
  - Estimate tree performance on out-of-bootstrap data
- Average the output of all trees (or choose mode decision)



Trees are de-correlated by choice of random subset of features

Overview of AdaBoost



Better classifiers will get higher weights

- Mis-classified examples get higher weights
- Correct examples get lower weights

Uniform weights

FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

AdaBoost

Initialize a vector of n uniform weights $\mathbf{w}_1$

for $t = 1, \ldots, T$

  Train model h_t on X, y with weights $\mathbf{w}_t$

  Compute the weighted training error of h_t

  Choose $\beta_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

  Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right)$$

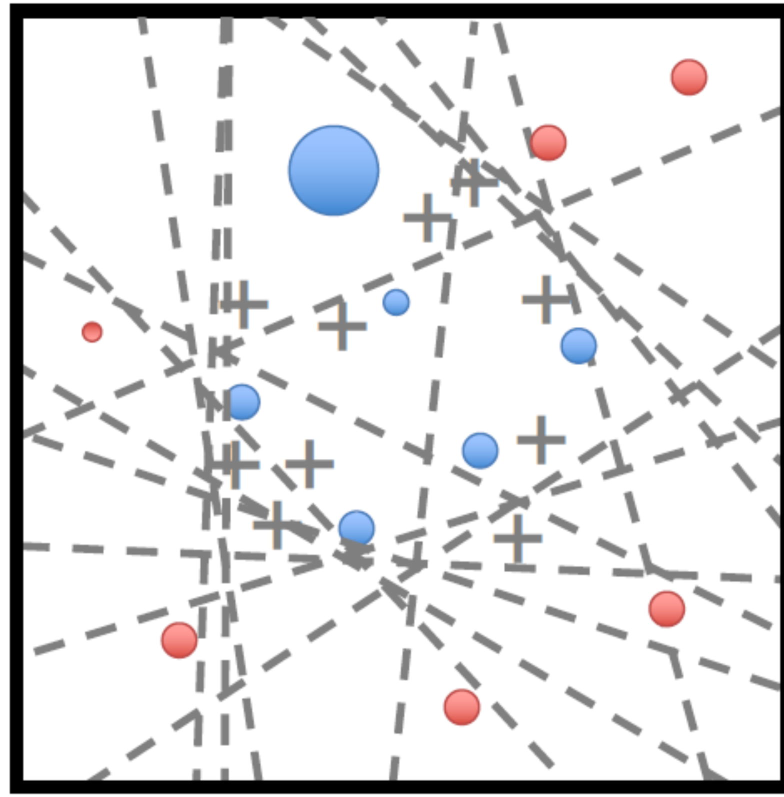
  Normalize $\mathbf{w}_{t+1}$ to be a distribution

end for

Return the hypothesis

$$H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \beta_t h_t(\mathbf{x})\right)$$

$t = \mathrm{T}$



- Final model is a weighted combination of members
  - Each member weighted by its importance

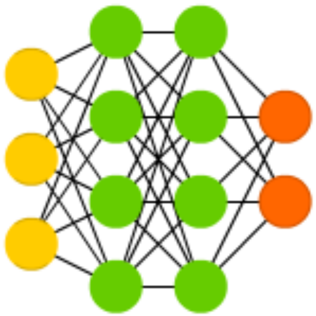
Ensemble Topics

- Bagging
  - Parallel training
  - Bootstrap samples
  - Out-of-bag (OOB) error
  - Random forest
- Boosting
  - Sequential training
  - Training with weights
  - AdaBoost

Neural Network Architectures

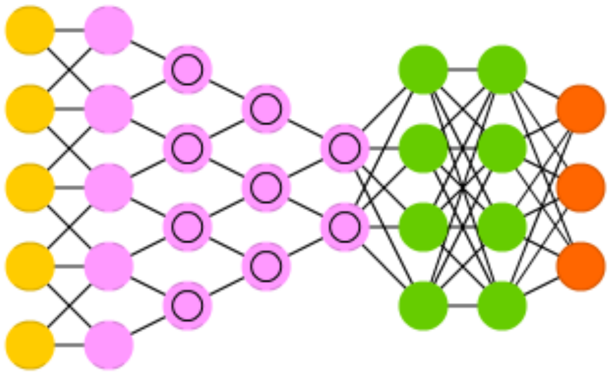
Feed-Forward Networks

- Neurons from each layer connect to neurons from next layer



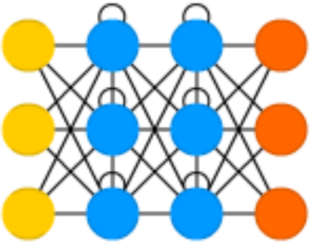
Convolutional Networks

- Includes convolution layer for feature reduction
- Learns hierarchical representations



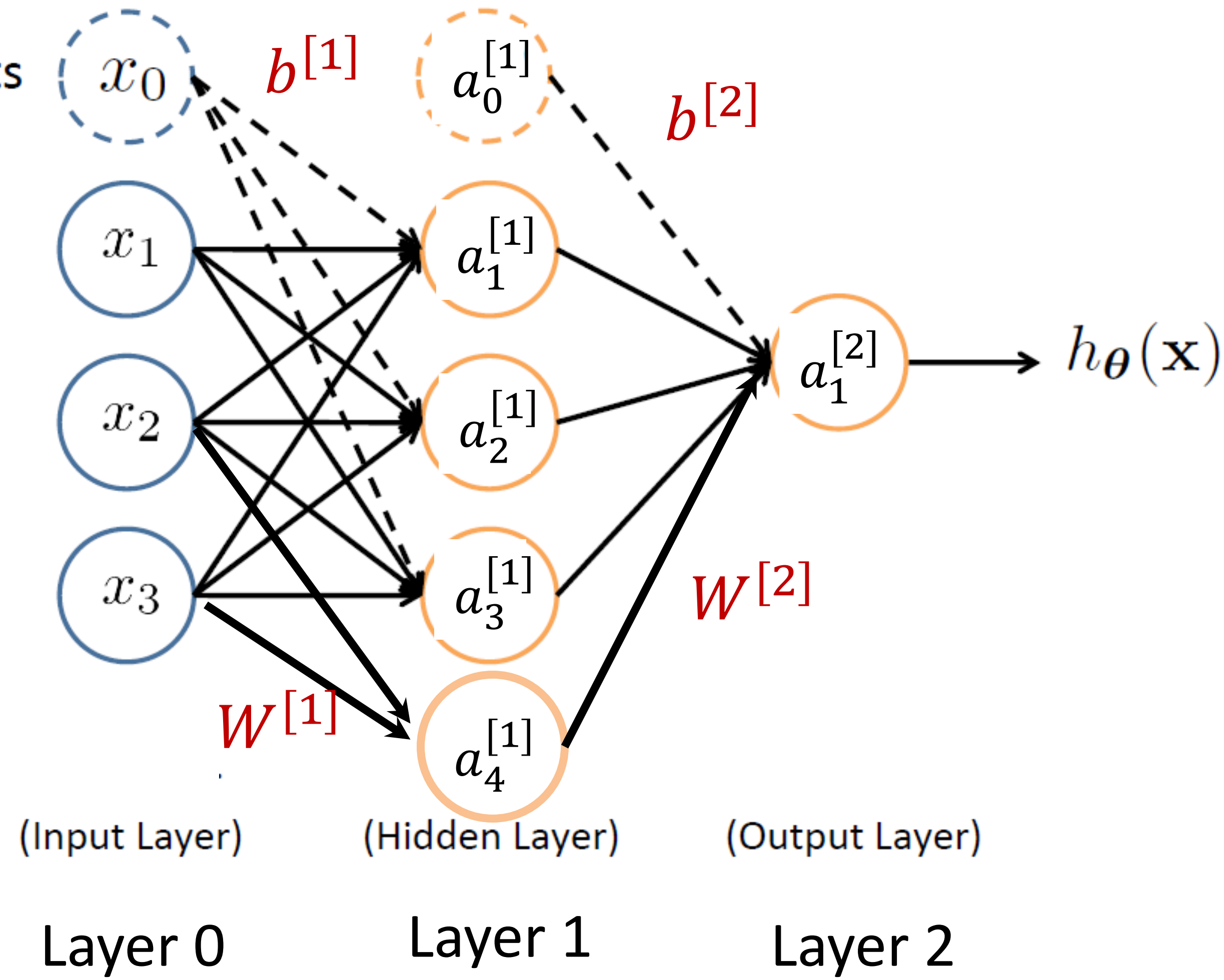
Recurrent Networks

- Keep hidden state
- Have cycles in computational graph



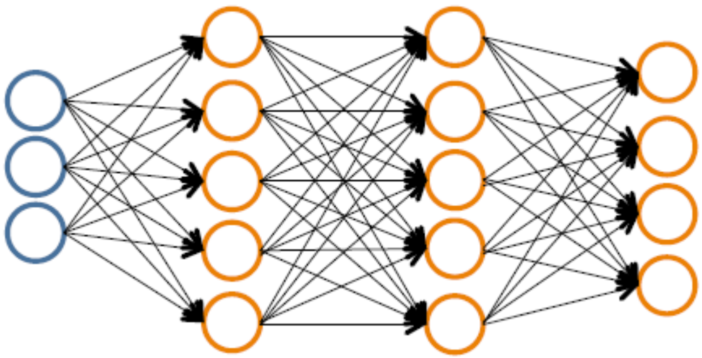
Feed-Forward Neural Network

Training example
$x = (x_1, x_2, x_3)$



No cycles $\theta = (b^{[1]}, W^{[1]}, b^{[2]}, W^{[2]})$

Neural Network Classification



Given:

$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$

$\mathbf{s} \in \mathbb{N}^{+L}$ contains # nodes at each layer

- $s_0 = d$ (# features)

Binary classification

y = 0 or 1

1 output unit $(s_{L-1} = 1)$

Sigmoid

Multi-class classification (K classes)

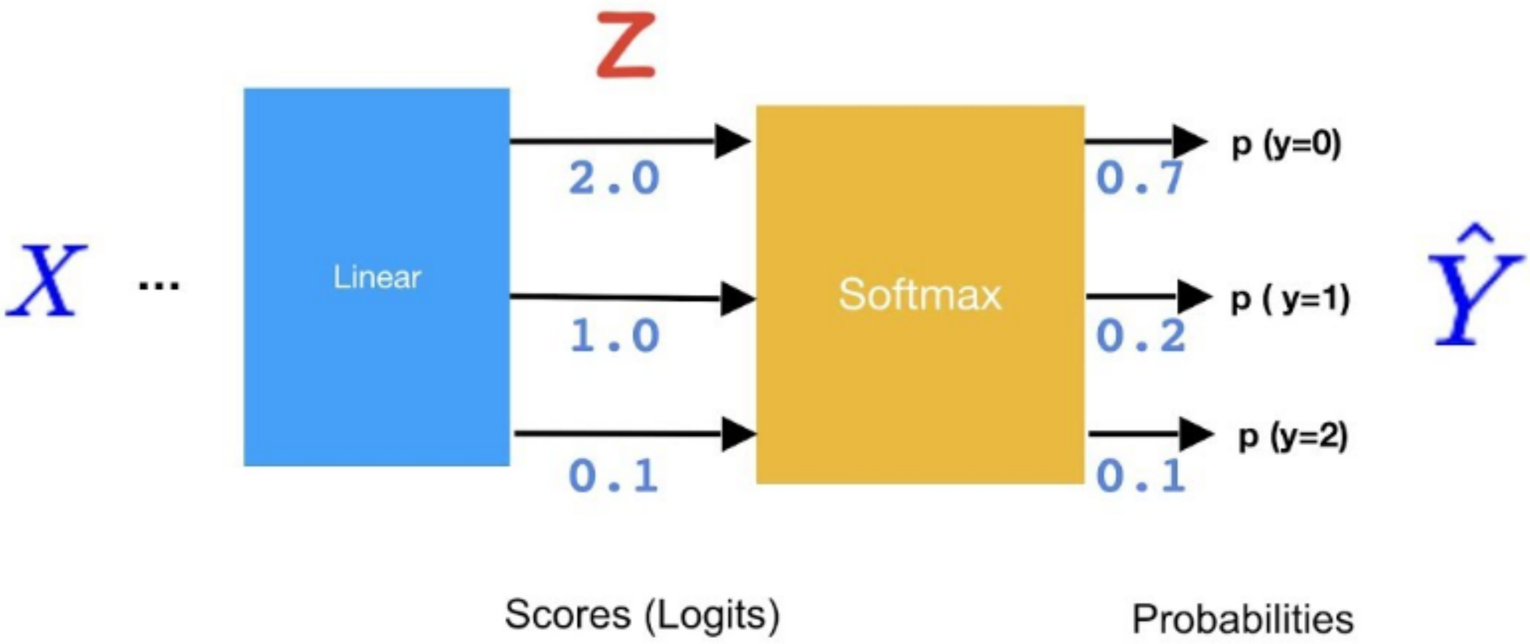
$\mathbf{y} \in \mathbb{R}^K$ e.g. $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

pedestrian car motorcycle truck

K output units $(s_{L-1} = K)$

Softmax

Softmax classifier



$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

- Predict the class with highest probability
- Generalization of sigmoid/logistic regression to multi-class

Feed-Forward Neural Networks Topics

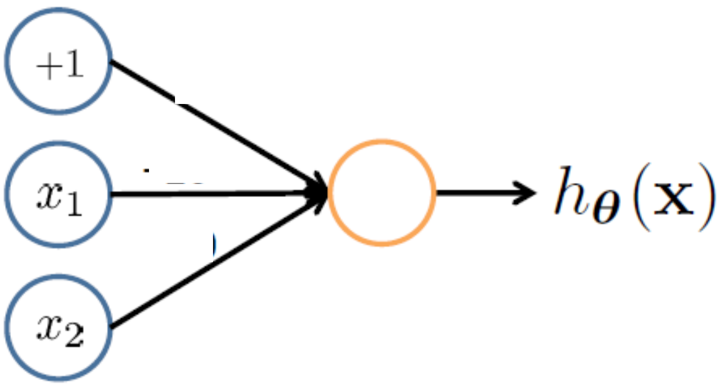
- Forward propagation
  - Linear operations and activations
- Activation functions
  - Examples, non-linearity
- Design networks for simple operations
- Estimate number of parameters
  - Count both weights and biases
- Activations for binary / multi-class classification
- Regularization (dropout and weight decay)

Representing Boolean Functions

Simple example: AND

$x_1, x_2 \in \{0, 1\}$

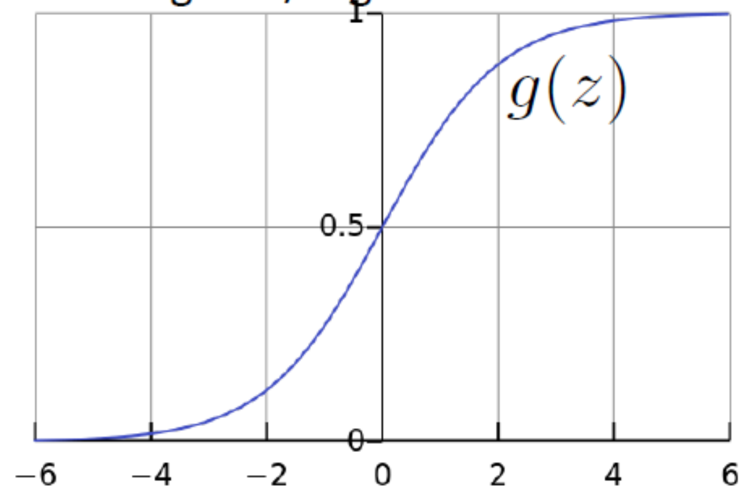
$y = x_1 \text{ AND } x_2$



$h_\Theta(\mathbf{x}) = g(\ ? \ + \ ? \ x_1 + \ ? \ x_2)$

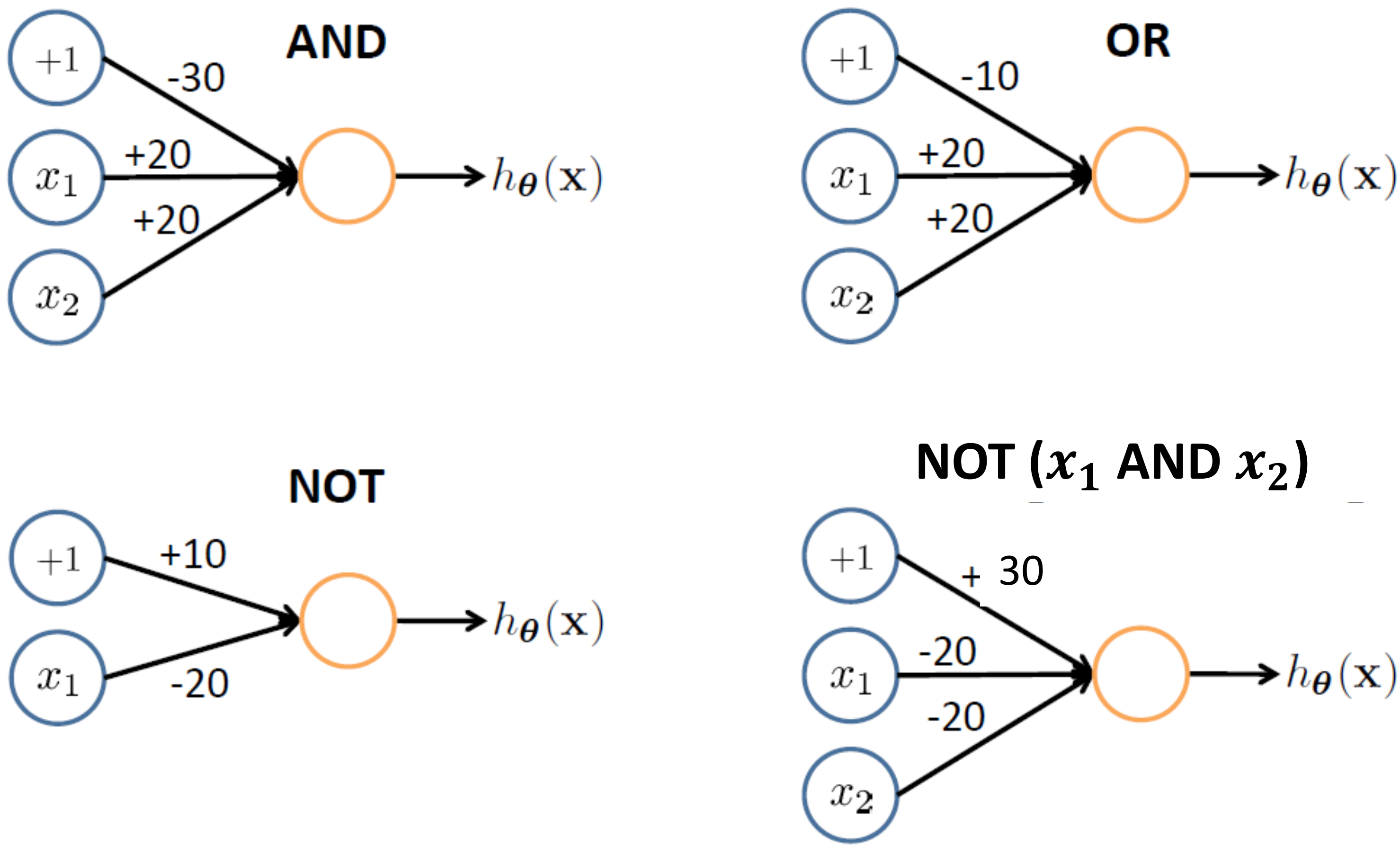
Logistic unit

Logistic / Sigmoid Function



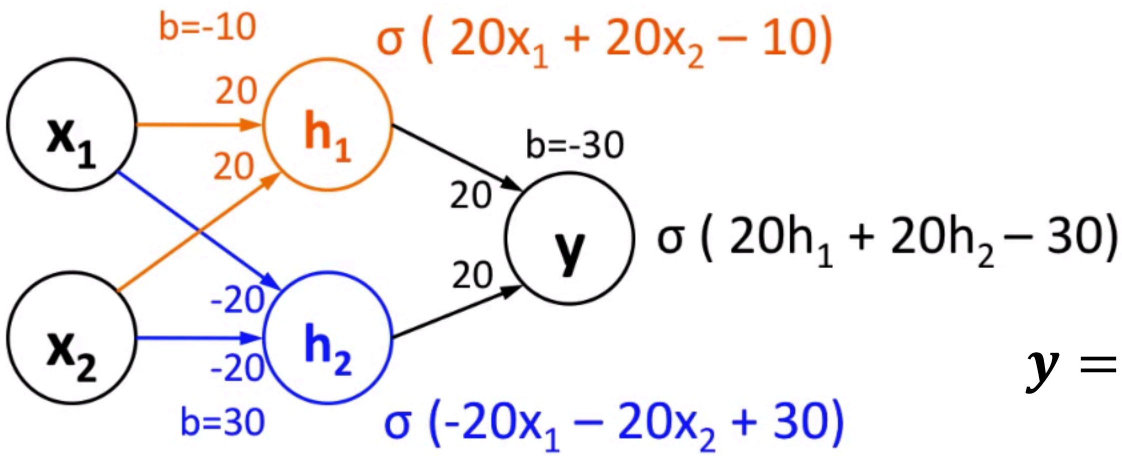
x_1	x_2	$h_\Theta(\mathbf{x})$
0	0	0
0	1	0
1	0	0
1	1	1

Representing Boolean Functions



XOR with 1 Hidden layer

$h_1 = x_1$ **OR** x_2

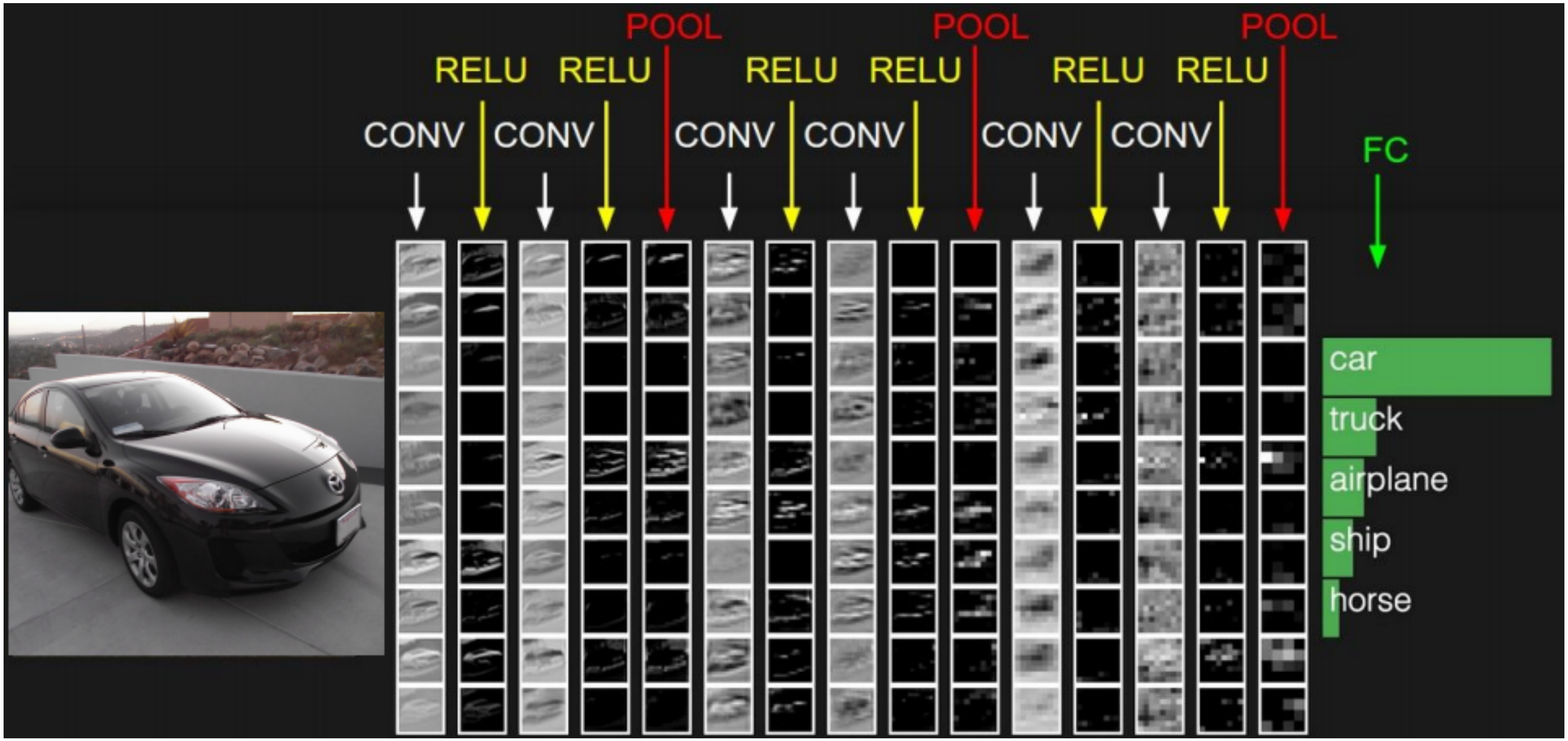


$y = h_1$ **AND** h_2

$h_2 =$ **NOT** (x_1 **AND** x_2)

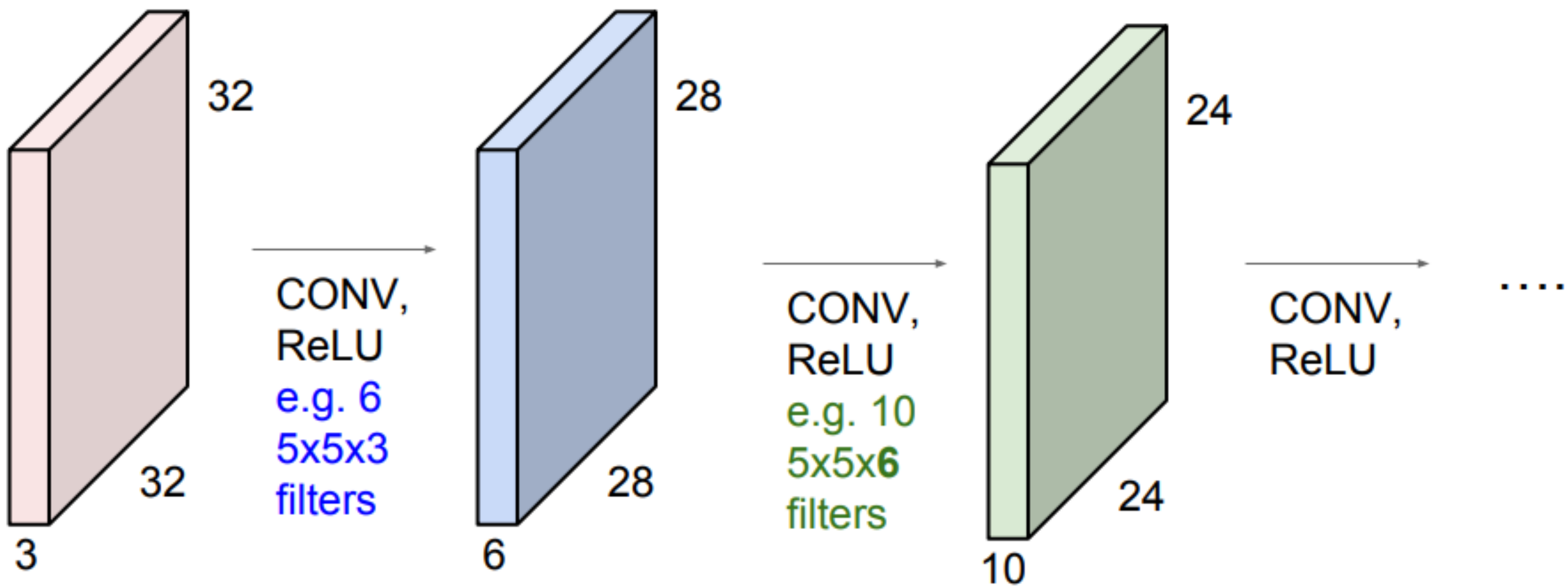
x_1 **XOR** x_2 **=** (x_1 **OR** x_2 **) AND (NOT** (x_1 **AND** x_2**))**

Convolutional Nets



Convolutional Nets

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



Topics

- Input and output size
- Number of parameters
- Convolution operation (stride, pad)
- Max pooling

Training NN with Backpropagation

Given training set $(x_1, y_1), \dots, (x_N, y_N)$
Initialize all parameters $W^{[\ell]}, b^{[\ell]}$ randomly, for all layers ℓ
Loop

Set $\Delta_{ij}^{(l)} = 0 \quad \forall l, i, j$ (Used to accumulate gradient)
For each training instance $(\mathbf{x}_i, y_i)$:
Set $\mathbf{a}^{(1)} = \mathbf{x}_i$
Compute $\{\mathbf{a}^{(2)}, \dots, \mathbf{a}^{(L)}\}$ via forward propagation
Compute $\boldsymbol{\delta}^{(L)} = \mathbf{a}^{(L)} - y_i$
Compute errors $\{\boldsymbol{\delta}^{(L-1)}, \dots, \boldsymbol{\delta}^{(2)}\}$
Compute gradients $\Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

EPOCH

Update weights via gradient step

- $W_{ij}^{[\ell]} = W_{ij}^{[\ell]} - \alpha \frac{\Delta_{ij}^{[\ell]}}{N}$
- Similar for $b_{ij}^{[\ell]}$

Until weights converge or maximum number of epochs is reached

Stochastic Gradient Descent

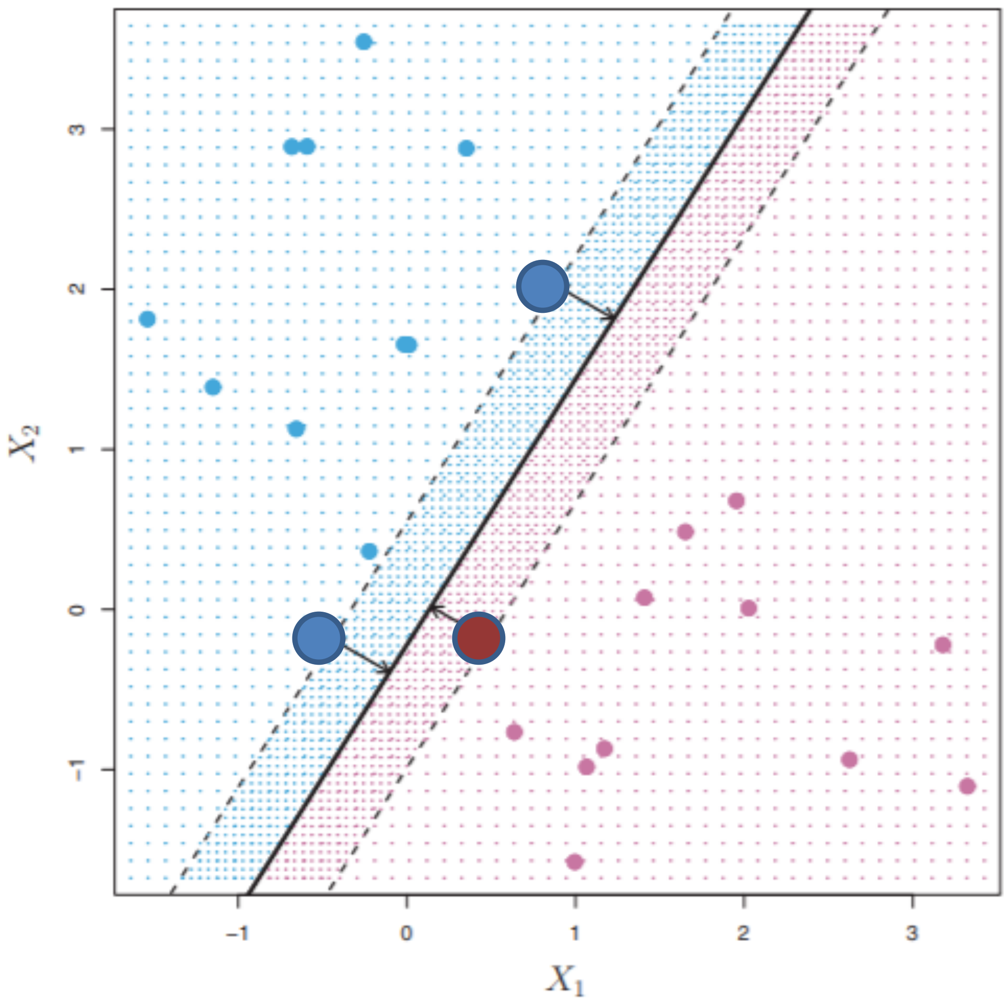
- Initialization
  - For all layers ℓ
    - Set $W^{[\ell]}, b^{[\ell]}$ at random
- Backpropagation
  - Fix learning rate α
  - For all layers ℓ (starting backwards)
    - For all training examples x_i, y_i
      - $W^{[\ell]} = W^{[\ell]} - \alpha \frac{\partial L(\hat{y}_i, y_i)}{\partial W^{[\ell]}}$
      - $b^{[\ell]} = b^{[\ell]} - \alpha \frac{\partial L(\hat{y}_i y_i)}{\partial b^{[\ell]}}$

Incremental version of GD

Mini-batch Gradient Descent

- Initialization
  - For all layers ℓ
    - Set $W^{[\ell]}, b^{[\ell]}$ at random
- Backpropagation
  - Fix learning rate α
  - For all layers ℓ (starting backwards)
    - For all batches b of size B with training examples x_{ib}, y_{ib}
      - $W^{[\ell]} = W^{[\ell]} - \alpha \sum_{i=1}^{B} \frac{\partial L(\hat{y}_{ib}, y_{ib})}{\partial W^{[\ell]}}$
      - $b^{[\ell]} = b^{[\ell]} - \alpha \sum_{i=1}^{B} \frac{\partial L(\hat{y}_{ib}, y_{ib})}{\partial b^{[\ell]}}$

Linear SVM - Max Margin



- Support vectors are "closest" to hyperplane
- If support vectors change, classifier changes

SVM with Kernels

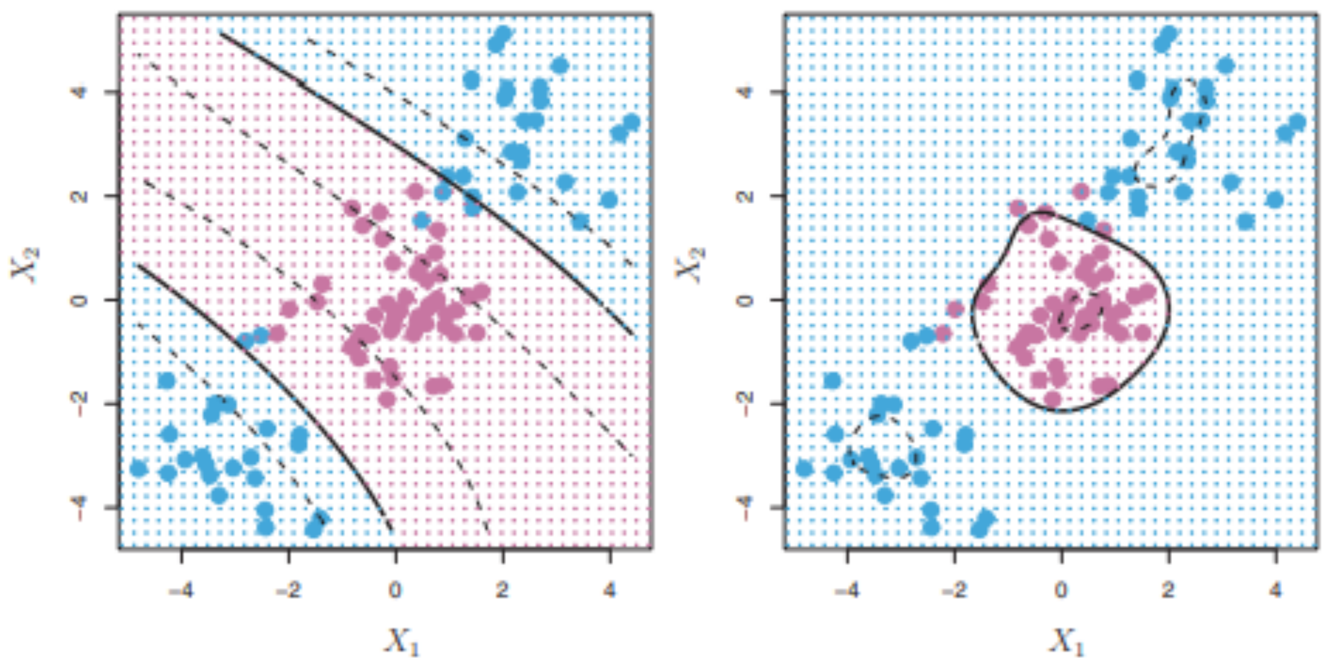


FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

SVM Topics

- Linear SVM
  - Maximum margin
  - Error budget
  - Solution depends only on support vectors
- Kernel SVM
  - Examples of kernels

Comparing classifiers

Algorithm	Interpretable	Model size	Predictive accuracy	Training time	Testing time
Logistic regression	High	Small	Lower	Low	Low
kNN	Medium	Large	Lower	No training	High
LDA	Medium	Small	Lower	Low	Low
Decision trees	High	Medium	Lower	Medium	Low
Ensembles	Low	Large	High	High	High
Naïve Bayes	Medium	Small	Lower	Medium	Low
SVM	Medium	Small	High	High	Low
Neural Networks	Low	Large	High	High	Low


THANK
YOU