



Machine Learning

ZG565

Dr. Sugata Ghosal

Sugata.ghosal@pilani.bits-pilani.ac.in

BITS Pilani
Pilani Campus





Session 1

These slides are prepared by the instructor, with grateful acknowledgement of many others who made their course materials freely available online.

Session Content

- Objective of course
- Evaluation Plan
- What is Machine Learning?
- Application areas of Machine Learning
- Why Machine Learning is important?
- Design a Learning System
- Issues in Machine Learning

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

What We'll Cover in this Course

- **Supervised learning algorithms**
 - Regression
 - Naïve Bayes
 - Logistic regression
 - Decision Tree and Random Forest
 - Support vector machines
 - **Ensemble Techniques**
 - **Unsupervised learning**
 - Clustering
 - **Applications**
-

Books

Text books and Reference book(s)

T1 Tom M. Mitchell: **Machine Learning**, The McGraw-Hill Companies

R1 Christopher M. Bishop: **Pattern Recognition & Machine Learning**,
Springer

R2 P. Tan, et al. **Introduction to Data Mining**, Pearson

R3 C.J.C. BURGES: **A Tutorial on Support Vector Machines for
Pattern Recognition**, Kluwer Academic Publishers, Boston.

Evaluation Plan

Name	Type	Weight
3 Quiz, best 2 scores will be taken	Online	10%
Assignment-I	Take Home	10%
Assignment-II	Take Home	10%
Mid-Semester Test	Closed Book	30%
Comprehensive Exam	Open Book	40%

Please note there will be no change in submission dates for quiz and assignment

Lab Plan

Lab No.	Lab Objective
1	End to End Machine Learning
2	Linear Regression and Gradient Descent Algorithm
3	Logistic Regression Classifier
4	Decision Tree
5	Naïve Bayes Classifier
6	Random Forest

- **Labs not graded**
- **Most of the Lab recordings available at CSIS virtual labs**
- **Webinars will be conducted for lab sessions**
- **Labs will be conducted in Python**

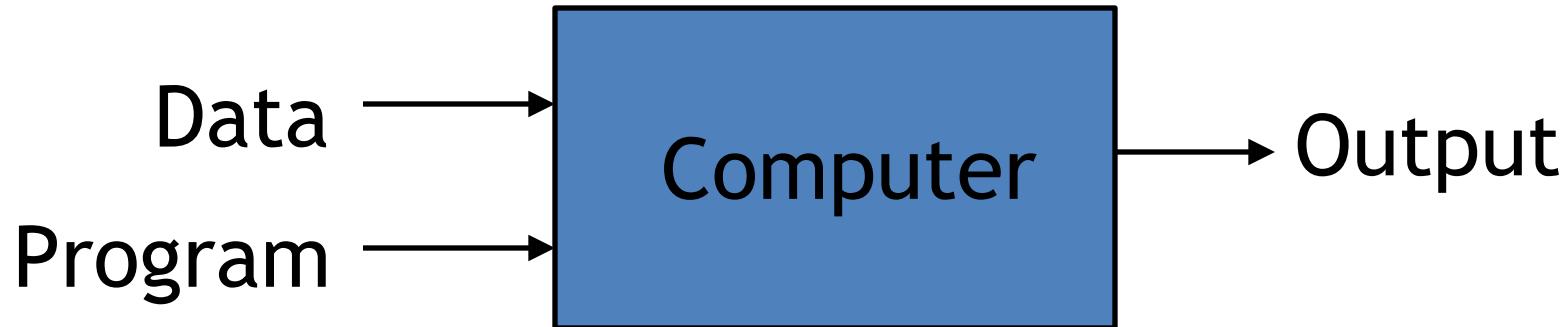
Machine Learning

- **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.

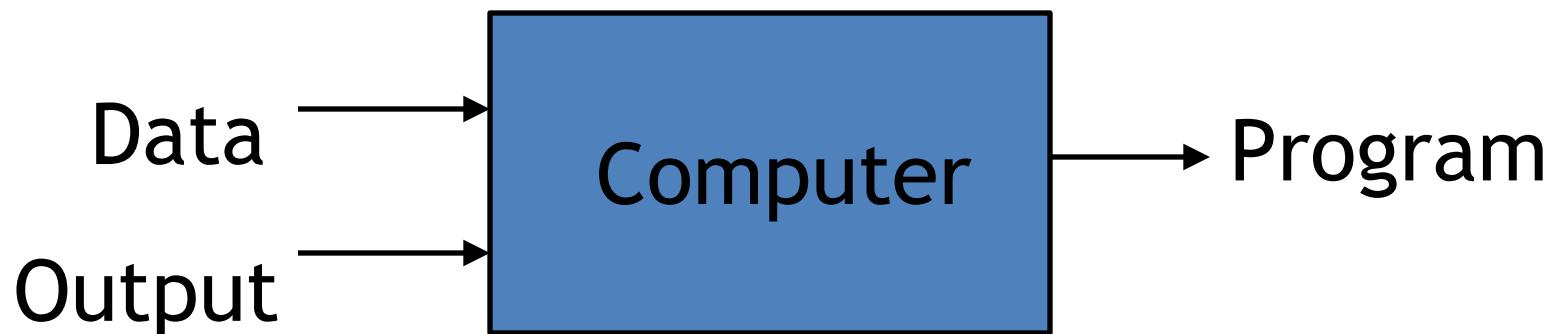
A Few Quotes

- “A breakthrough in machine learning would be worth ten Microsofts” (Bill Gates, Chairman, Microsoft)
 - “Machine learning is the next Internet”
(Tony Tether, Director, DARPA)
 - “Web rankings today are mostly a matter of machine learning” (Prabhakar Raghavan, Dir. Research, Yahoo)
 - “Machine learning is going to result in a real revolution” (Greg Papadopoulos, CTO, Sun)
 - “Machine learning is today’s discontinuity”
(Jerry Yang, CEO, Yahoo)
-

Traditional Programming



Machine Learning



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.

What is Machine Learning?

- To have a learning problem, we must identify
 - The class of tasks
 - The measure of performance to be improved
 - Source of experience

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

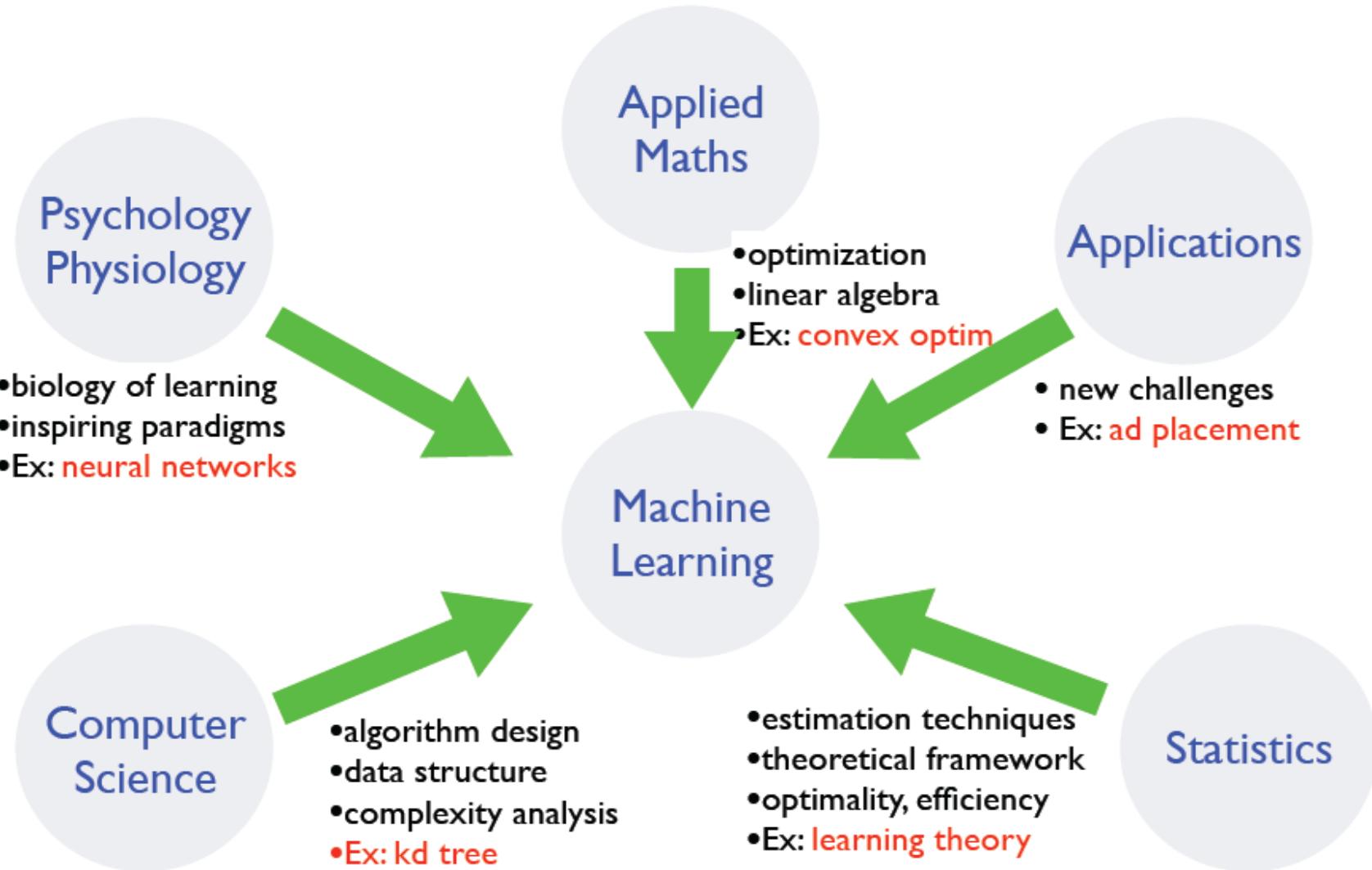
A handwriting recognition learning problem

- **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

Where does ML fit in?



Why is Machine Learning Important?

- Some tasks cannot be defined well, except by examples.
- Relationships and correlations can be hidden within large amounts of data. Machine Learning may be able to find these relationships.
- Human designers often produce machines that do not work as well as desired in the environments in which they are used.

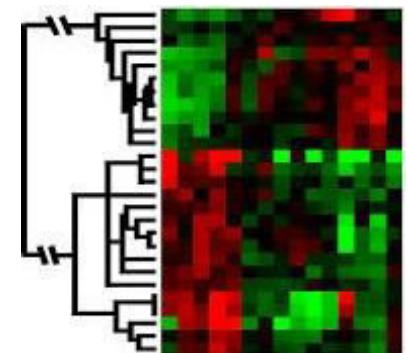
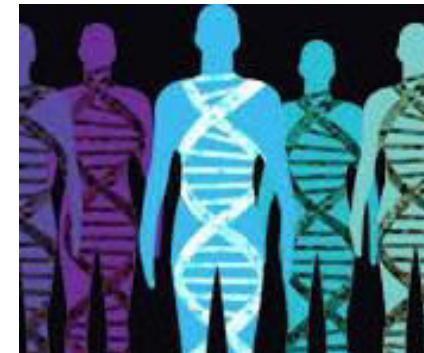
Why is Machine Learning Important ?

- The amount of knowledge available about certain tasks might be too large for explicit encoding by humans (e.g., medical diagnostic).
- New knowledge about tasks is constantly being discovered by humans. It may be difficult to continuously re-design systems “by hand”.

When Do We Use Machine Learning?

ML is used when:

- Human expertise does not exist (navigating on Mars)
- Humans can't explain their expertise (speech recognition)
- Models must be customized (personalized medicine)
- Models are based on huge amounts of data (genomics)



Learning isn't always useful:

- There is no need to “learn” to calculate payroll

Application Domains

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

Many more emerging...

State of the Art Applications of Machine Learning

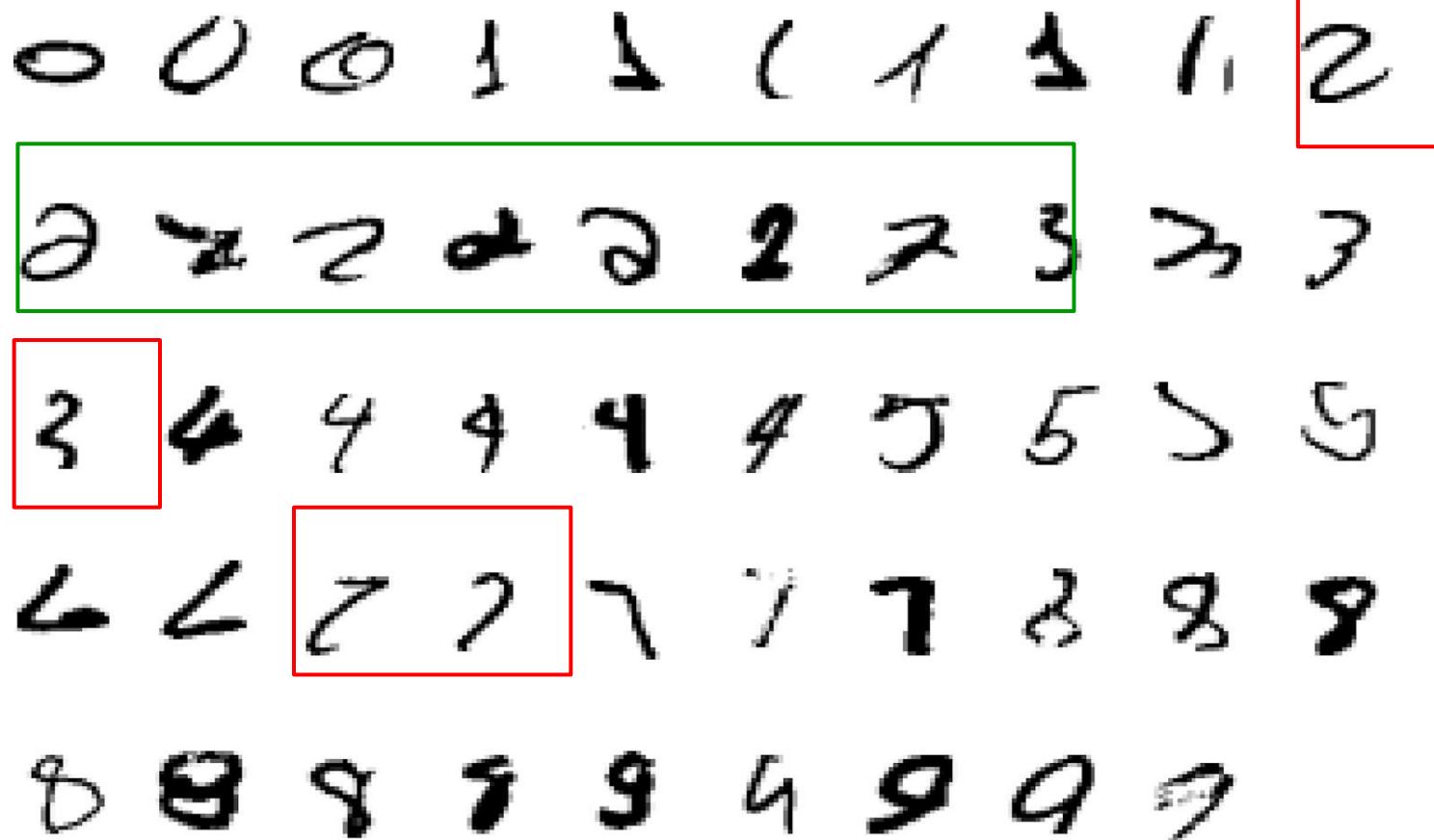
.

Application Types

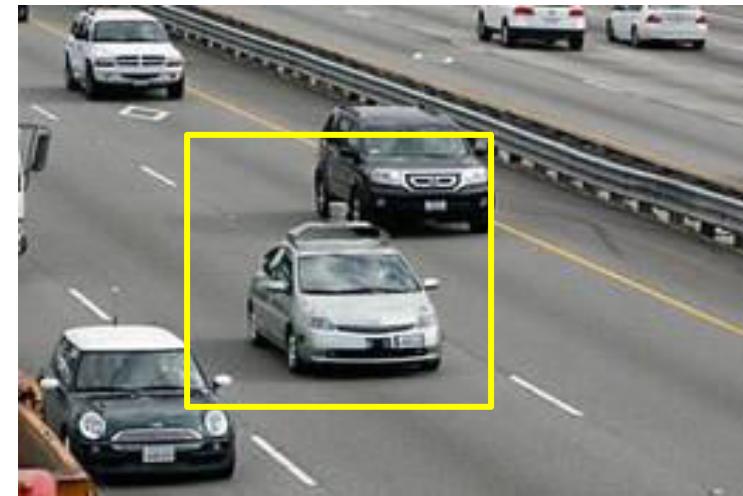
- Medical diagnosis
- Credit card applications or transactions
- Fraud detection in e-commerce
- Worm detection in network packets
- Spam filtering in email
- Recommended articles in a newspaper
- Recommended books, movies, music, or jokes
- Financial investments
- DNA sequences
- Spoken words
- Handwritten letters
- Astronomical images

Pattern recognition

It is very hard to say what makes a 2



Autonomous Cars

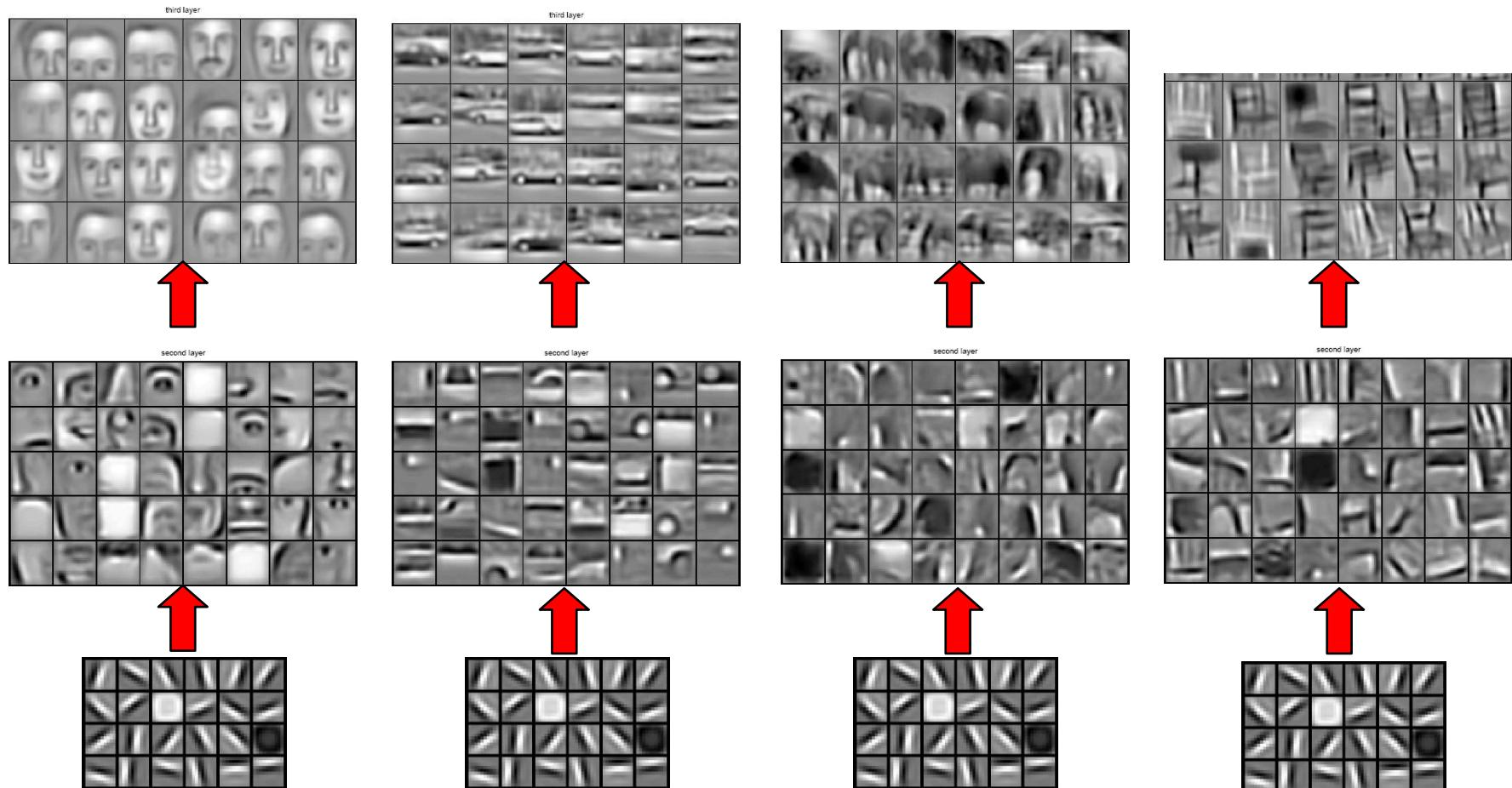


- Nevada made it legal for autonomous cars to drive on roads in June 2011
- As of 2013, four states (Nevada, Florida, California, and Michigan) have legalized autonomous cars

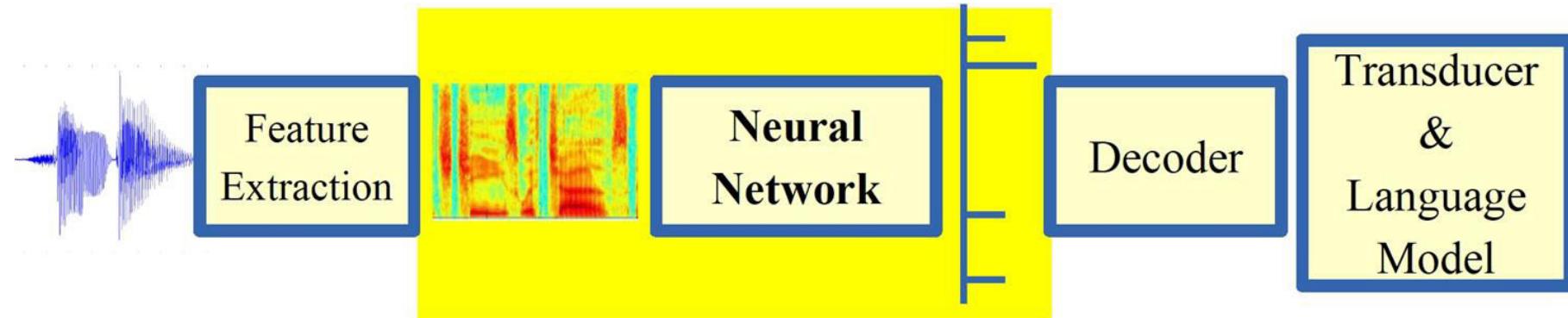
UPenn's Autonomous Car →



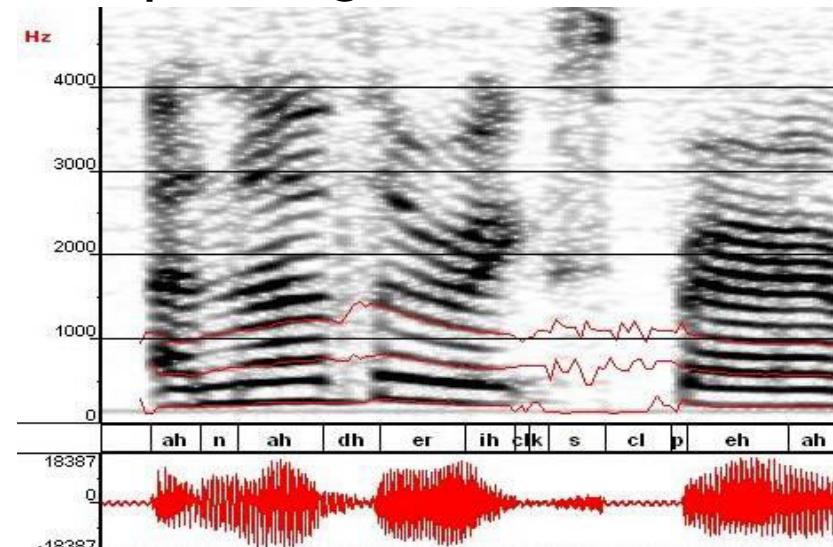
Learning of Object Parts



Automatic Speech Recognition



ML used to predict phoneme states from sound spectrogram Deep Learning Based Results



# Hidden Layers	1	2	4	8	10	12
Word Error Rate %	16.0	12.8	11.4	10.9	11.0	11.1

Baseline Gaussian Mixture Model based word error rate = 15.4%

[Zeiler et al. "On rectified linear units for speech recognition" ICASSP 2013]

Robotics



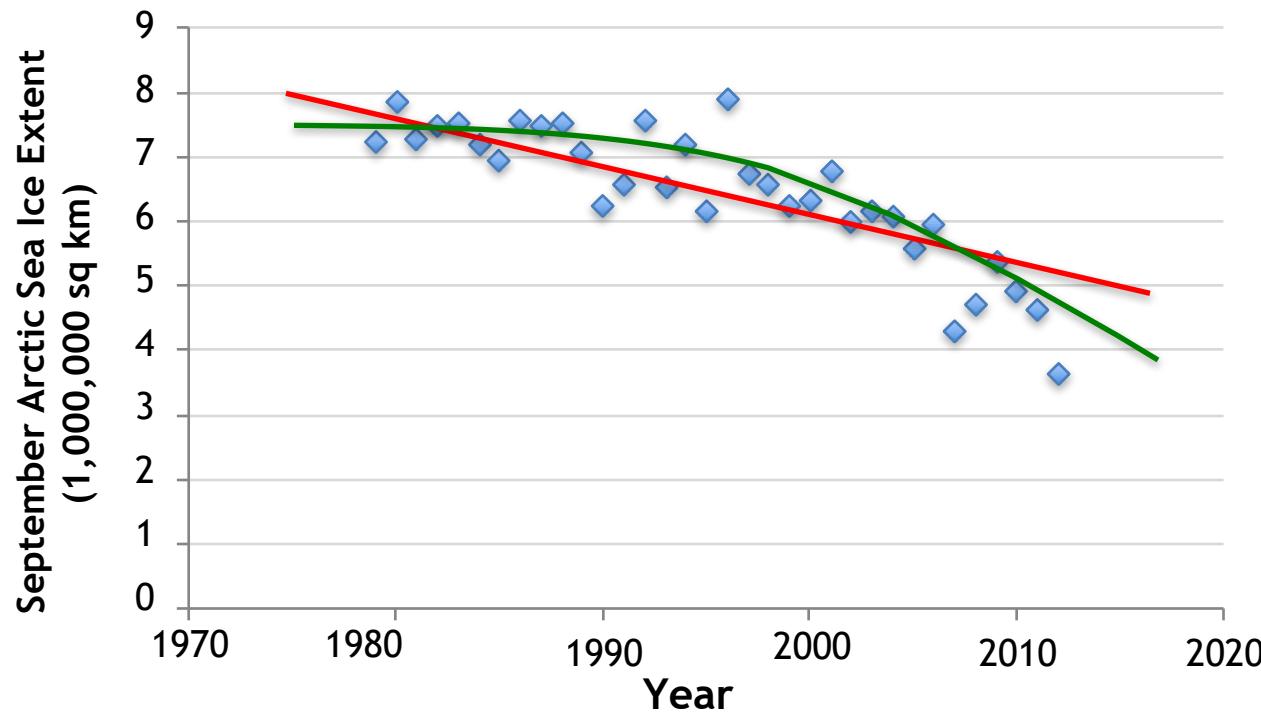
Cleaning Robots

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

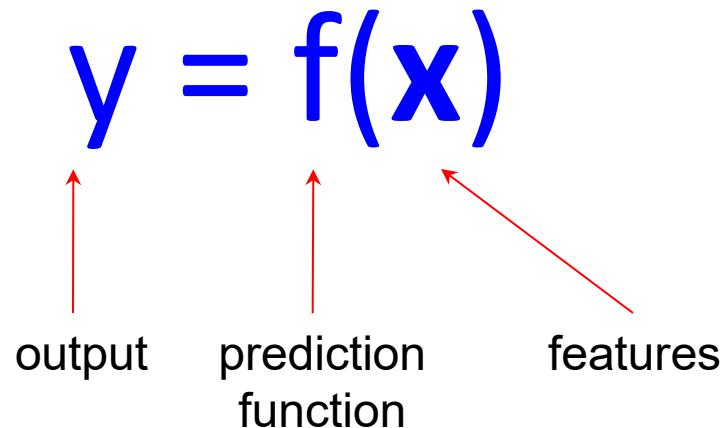
Supervised Learning: Regression

- Given $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
- 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



- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$, 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 $y = f(\mathbf{x})$

Classification Example

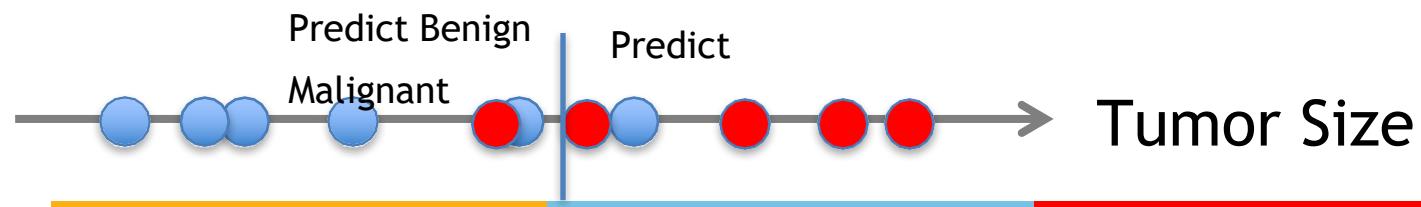
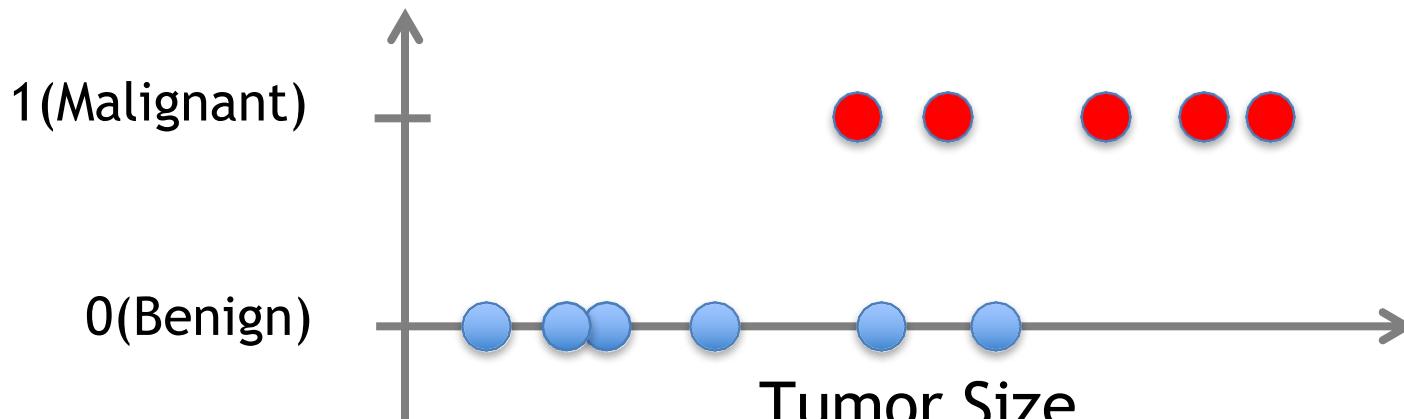
- Apply a prediction function to a feature representation of the image to get the desired output:

$$f(\text{apple}) = \text{"apple"}$$
$$f(\text{tomato}) = \text{"tomato"}$$
$$f(\text{cow}) = \text{"cow"}$$

Supervised Learning: Classification

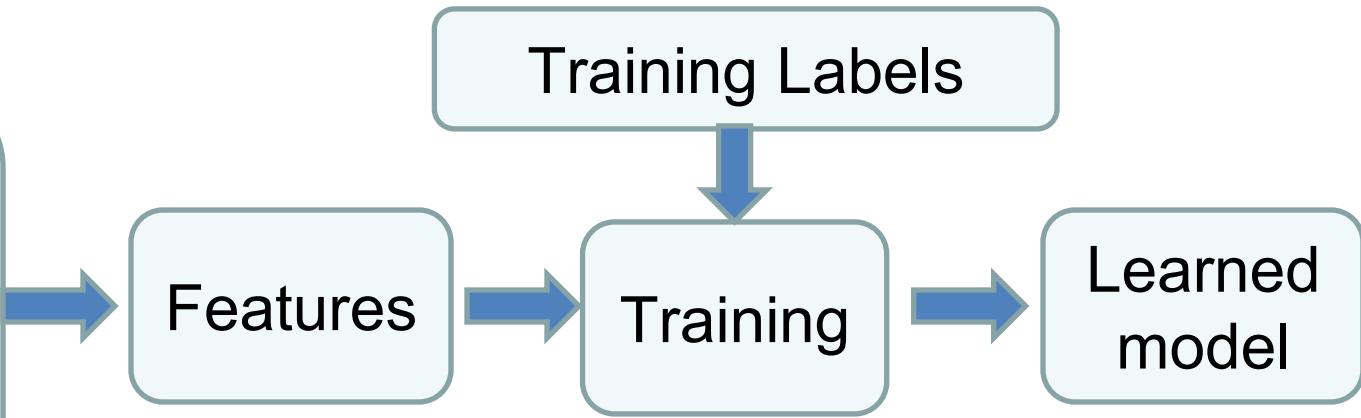
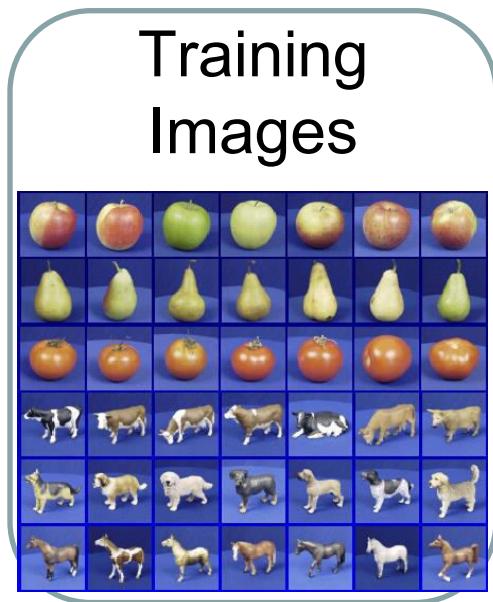
- Given $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
- Learn a function $f(x)$ to predict y given x
 - y is categorical == classification

Breast Cancer (Malignant / Benign)

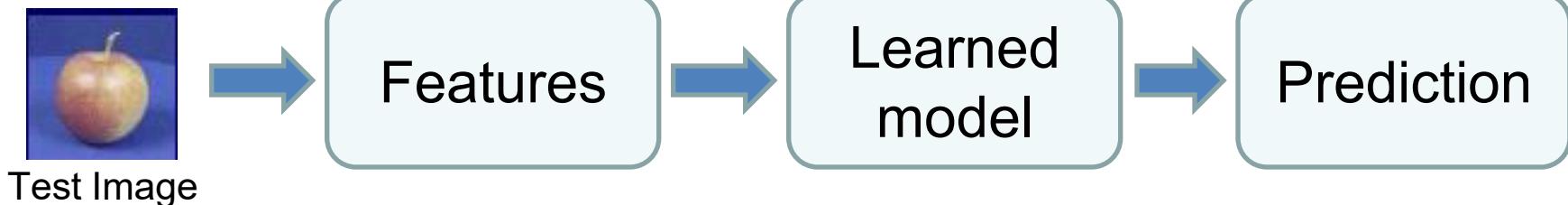


Classification

Training

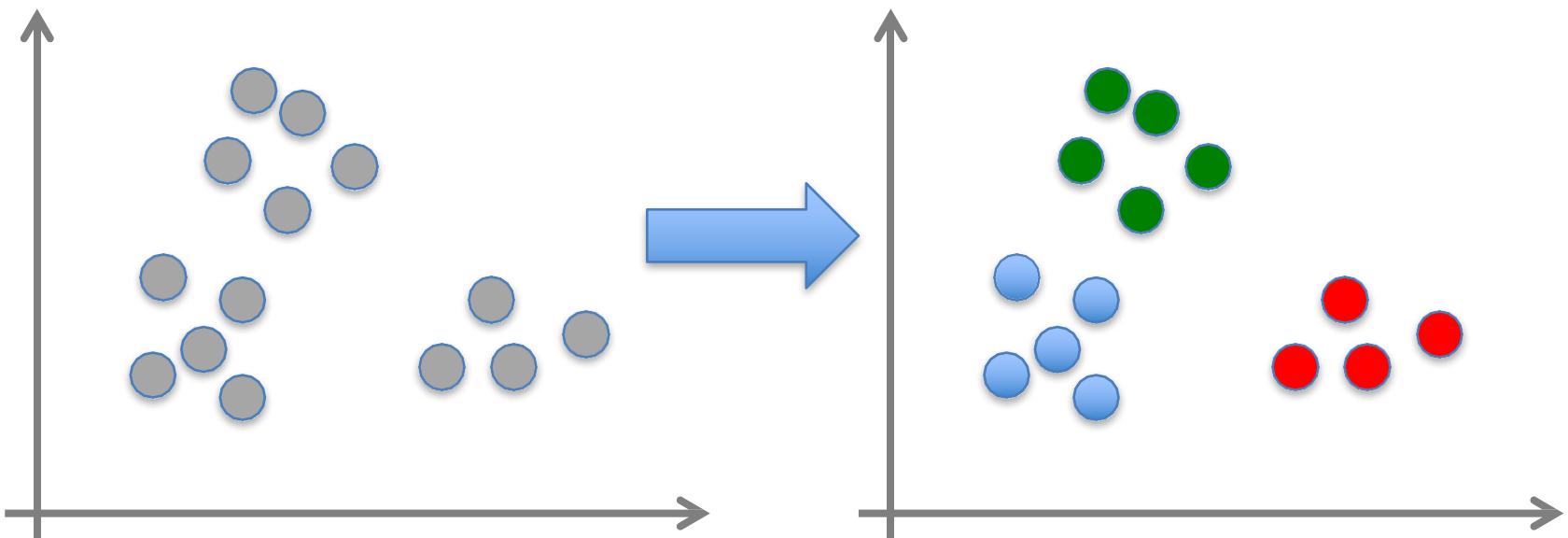


Testing



Unsupervised Learning

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



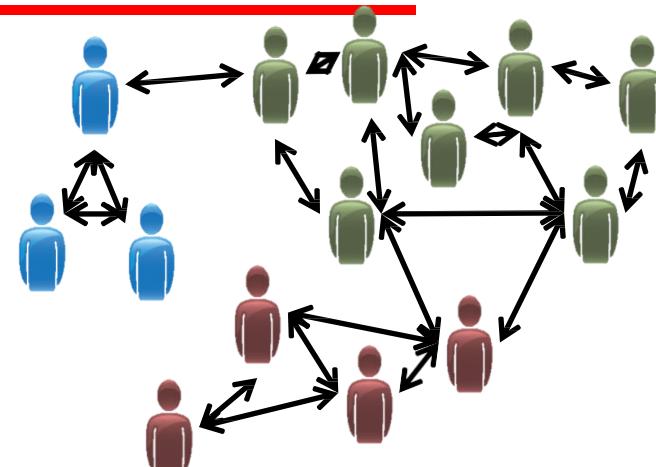
Unsupervised Learning



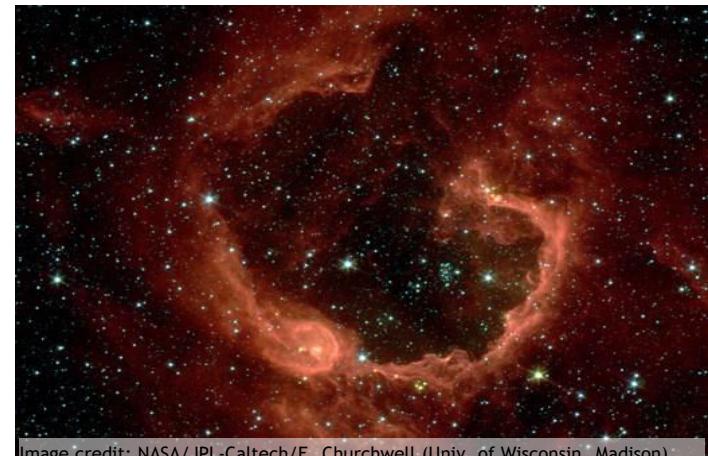
Organize computing clusters



Market segmentation



Social network analysis



Astronomical data analysis

Reinforcement Learning

- No predefined data
- Semi-supervised learning model in machine learning,
- 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

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

Open source ML programming tools

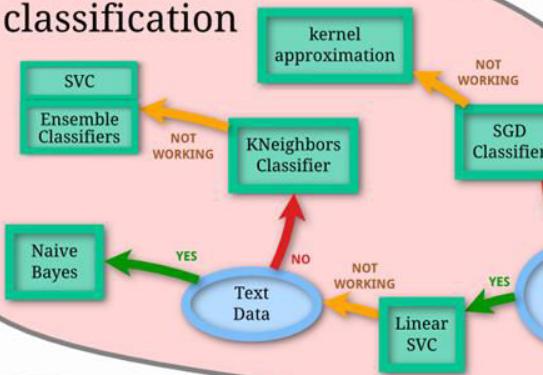
Platform	Algorithms or Features		
Scikit Learn	Linux, Mac OS, Windows	Python, C, C++	Classification, Regression, Clustering Preprocessing, Model Selection Dimensionality reduction.
PyTorch	Linux, Mac OS, Windows	Python, C++	Autograd Module, Optimization Module NN Module
TensorFlow	Linux, Mac OS, Windows	Python, C++	Provides a library for dataflow programming.
Weka	Linux, Mac OS, Windows	Java	Data preparation, Classification Regression, Clustering, Visualization Association rules mining

Open source ML programming tools

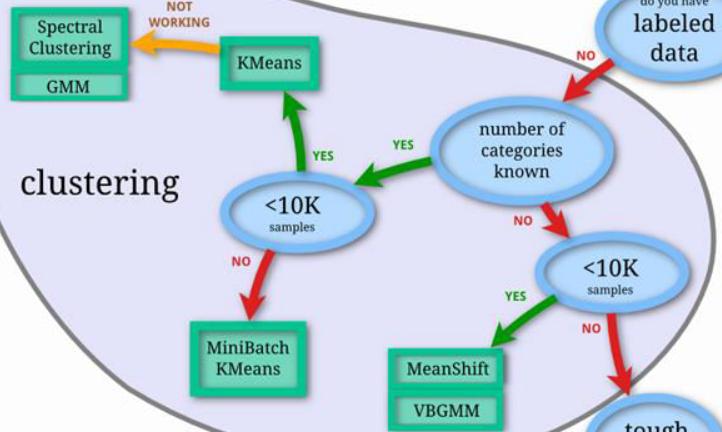
Colab	Cloud Service	-	Supports libraries of PyTorch, Keras, TensorFlow, and OpenCV
Apache Mahout	Cross-platform	Java Scala	Preprocessors, Regression Clustering, Recommenders Distributed Linear Algebra.
Accors.Net	Cross-platform	C#	Classification, Regression, Distribution Clustering, Hypothesis Tests & Kernel Methods, Image, Audio & Signal & Vision
Shogun	Windows Linux, UNIX Mac OS	C++	Regression, Classification, Clustering Support vector machines. Dimensionality reduction, Online learning etc.
Keras.io	Cross-platform	Python	API for neural networks

scikit-learn algorithm cheat-sheet

classification



clustering

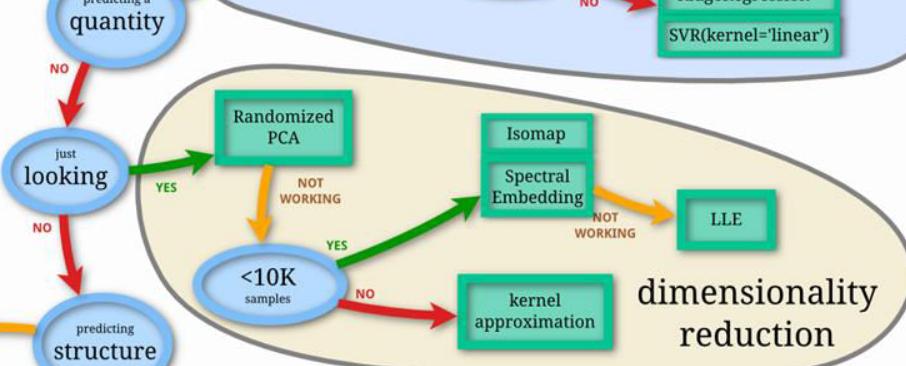


Back

scikit
learn

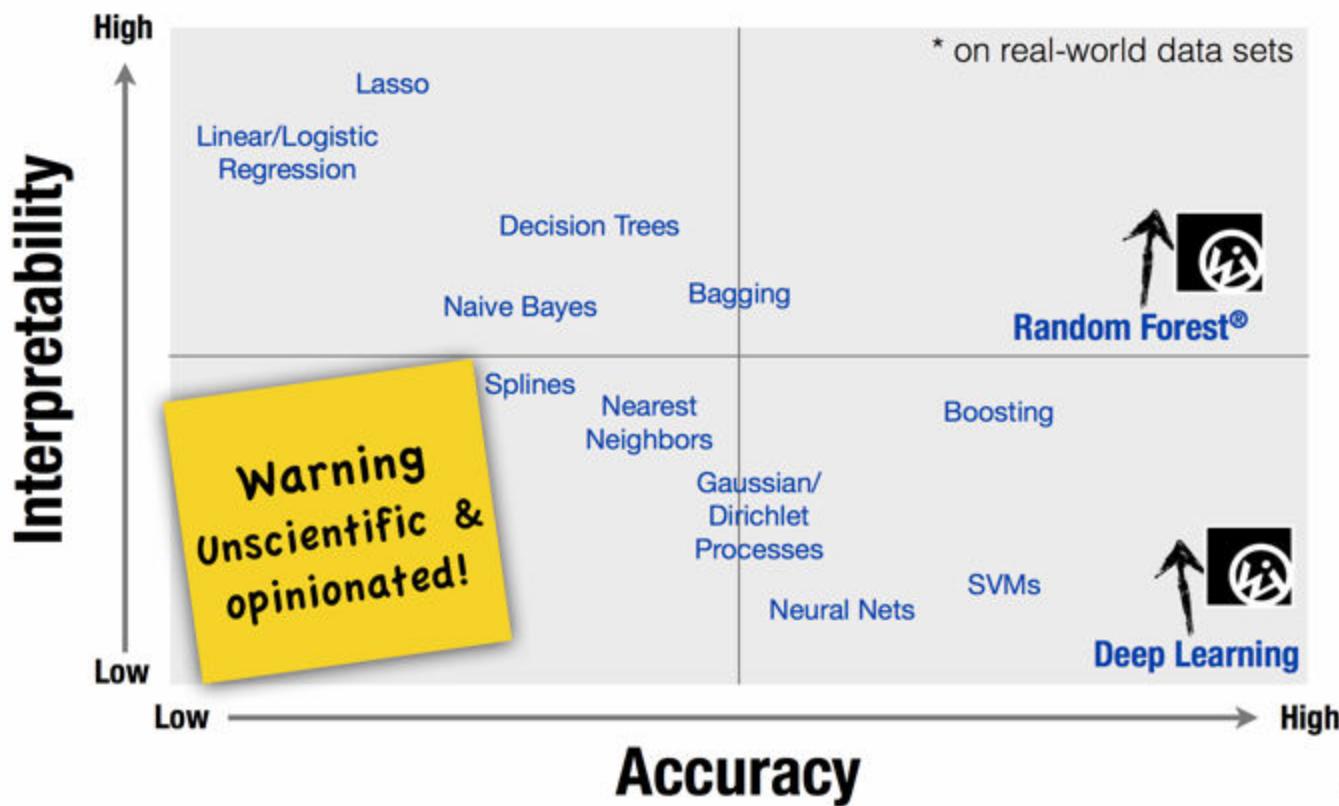


regression



dimensionality reduction

ML Algorithmic Trade-Off



ML in a Nutshell

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

Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- etc.

Evaluating Performance

- If y is discrete:
 - Accuracy: # correctly classified / # all test examples
 - Precision/recall
 - True Positive, False Positive, True Negative, False Negative
 - Precision = $TP / (TP + FP) = \# \text{predicted true pos} / \# \text{predicted pos}$
 - Recall = $TP / (TP + FN) = \# \text{predicted true pos} / \# \text{true pos}$
 - F-measure
$$= 2PR / (P + R)$$
- Want evaluation metric to be in some range, e.g. [0 1]
 - 0 = worst possible classifier, 1 = best possible classifier

Evaluating Performance

- If y is continuous:
 - Sum-of-Squared-Differences (SSD) error between predicted and true y :

$$E = \sum_{i=1}^n (f(x_i) - y_i)^2$$

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?



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

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

Loop

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

Recommended Readings

- Tom M. Mitchell, Machine Learning, The McGraw-Hill Companies, Inc. International Edition 1997. **[Ch. 1]**
- [**\[Web\]**](http://www.cs.princeton.edu/courses/archive/spr08/cos511/)
- [**\[Web\]**](https://www.softwaretestinghelp.com/machine-learning-tools/)

**END
of
Session 1**



Machine Learning

ZG565

Dr. Sugata Ghosal

Sugata.ghosal@pilani.bits-pilani.ac.in

BITS Pilani
Pilani Campus





Session 2
Date – 19th November 2022
Time – 4:15 PM to 6:15 PM

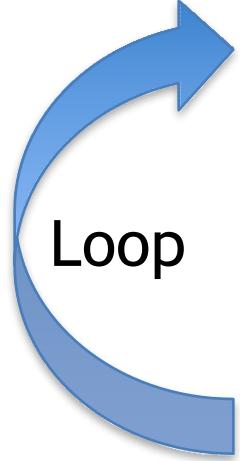
These slides are prepared by the instructor, with grateful acknowledgement of many others who made their course materials freely available online.

Session Content

- Data - Types, Attributes
- Data Quality
- Data Preprocessing
- Performance Metrics
- Challenges of ML
- Model Evaluation, Selection
- Homework
 - Review Lab Capsule 1 from the “Machine Learning” Virtual Labs

ML in a Nutshell

- Tens of thousands of machine learning algorithms
 - Hundreds new every year
- Every ML algorithm has three components
 - **Data Representation**
 - **Parameter Optimization**
 - **Model Evaluation, Selection**



Loop

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

Definition of Data

- Collection of ***data objects*** and their ***attributes***
- An ***attribute*** is a property or characteristic of an object
 - Examples: eye color of a person, temperature, etc.
 - aka variable, field, characteristic, dimension, or feature
- A collection of attributes describe an ***object***
 - Object is also known as record, point, case, sample, entity, or instance

Attributes

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Objects

Attribute Values

- ***Attribute values*** are numbers or symbols assigned to an attribute for a particular object
- Distinction between attributes and attribute values
 - Same attribute can be mapped to different attribute values
 - Example: height can be measured in feet or meters
 - Different attributes can be mapped to the same set of values
 - Example: Attribute values for ID and age are integers
 - But properties of attribute can be different than the properties of the values used to represent the attribute

Types of Attributes

- There are different types of attributes
 - **Nominal**
 - Examples: ID numbers, zip codes
 - **Ordinal**
 - Examples: rankings (e.g., taste of potato chips on a scale from 1-10), grades, height {tall, medium, short}
 - **Interval**
 - Examples: calendar dates, temperatures in Celsius or Fahrenheit.
 - **Ratio**
 - Examples: temperature in Kelvin, length, counts, elapsed time (e.g., time to run a race)

Properties of Attribute Values

- The type of an attribute depends on which of the following properties/operations it possesses:
 - Distinctness: $= \neq$
 - Order: $< >$
 - Differences are $+ -$
meaningful :
 - Ratios are $* /$
meaningful
 - Nominal attribute: distinctness
 - Ordinal attribute: distinctness & order
 - Interval attribute: distinctness, order & meaningful differences
 - Ratio attribute: all 4 properties/operations

Difference Between Ratio and Interval

- Is it physically meaningful to say that a temperature of 10° is twice that of 5° on
 - the Celsius scale?
 - the Fahrenheit scale?
 - the Kelvin scale?
- Consider measuring the height above average
 - If Bill's height is three inches above average and Bob's height is six inches above average, then would we say that Bob is twice as tall as Bill?
 - Is this situation analogous to that of temperature?

Attribute Type	Description	Examples	Operations
Categorical Qualitative	Nominal Nominal attribute values only distinguish. ($=$, \neq)	zip codes, employee ID numbers, eye color, sex: { <i>male</i> , <i>female</i> }	mode, entropy, contingency correlation, χ^2 test
	Ordinal Ordinal attribute values also order objects. ($<$, $>$)	hardness of minerals, { <i>good</i> , <i>better</i> , <i>best</i> }, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Numeric Quantitative	Interval For interval attributes, differences between values are meaningful. (+, -)	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, <i>t</i> and <i>F</i> tests
	Ratio For ratio variables, both differences and ratios are meaningful. (*, /)	temperature in Kelvin, monetary quantities, counts, age, mass, length, current	geometric mean, harmonic mean, percent variation

This categorization of attributes is due to S. S. Stevens

Attribute Type	Transformation	Comments
Categorical Qualitative	Nominal Any permutation of values	If all employee ID numbers were reassigned, would it make any difference?
	Ordinal An order preserving change of values, i.e., $new_value = f(old_value)$ where f is a monotonic function	An attribute encompassing the notion of good, better best can be represented equally well by the values {1, 2, 3} or by { 0.5, 1, 10}.
Numeric Quantitative	$new_value = a * old_value + b$ where a and b are constants	Thus, the Fahrenheit and Celsius temperature scales differ in terms of where their zero value is and the size of a unit (degree).
	$new_value = a * old_value$	Length can be measured in meters or feet.

This categorization of attributes is due to S. S. Stevens

Discrete and Continuous Attributes

- ## Discrete Attribute

- Has only a finite or countably infinite set of values
- Examples: zip codes, counts, or the set of words in a collection of documents
- Often represented as integer variables.
- Note: **binary attributes** are a special case of discrete attributes

- ## Continuous Attribute

- Has real numbers as attribute values
- Examples: temperature, height, or weight.
- Practically, real values can only be measured and represented using a finite number of digits.
- Continuous attributes are typically represented as floating-point variables.

Asymmetric Attributes

- Only presence (a non-zero attribute value) is regarded as important
 - Words present in documents
 - Items present in customer transactions
- If we met a friend in the grocery store would we ever say the following?

“I see our purchases are very similar since we didn’t buy most of the same things.”

Key Messages for Attribute Types

- The types of operations you choose should be “meaningful” for the type of data you have
 - Distinctness, order, meaningful intervals, and meaningful ratios are only four (among many possible) properties of data
 - The data type you see – often numbers or strings – may not capture all the properties or may suggest properties that are not present
 - Analysis may depend on these other properties of the data
 - Many statistical analyses depend only on the distribution
 - In the end, what is meaningful can be specific to domain
-

Important Characteristics of Data

- Dimensionality (number of attributes)
 - High dimensional data brings a number of challenges
 - Sparsity
 - Only presence counts
 - Resolution
 - Patterns depend on the scale
 - Size
 - Type of analysis may depend on size of data
-

Types of data sets

- Record
 - Data Matrix
 - Document Data
 - Transaction Data
 - Graph
 - World Wide Web
 - Molecular Structures
 - Ordered
 - Spatial Data
 - Temporal Data
 - Sequential Data
 - Genetic Sequence Data
-

Record Data

Data that consists of a collection of records, each of which consists of a fixed set of attributes

<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Data Matrix

- If data objects have the same fixed set of numeric attributes, then the data objects can be thought of as points in a multi-dimensional space, where each dimension represents a distinct attribute
- Such a data set can be represented by an m by n matrix, where there are m rows, one for each object, and n columns, one for each attribute

Projection of x Load	Projection of y load	Distance	Load	Thickness
10.23	5.27	15.22	2.7	1.2
12.65	6.25	16.22	2.2	1.1

Document Data

- Each document becomes a ‘term’ vector
 - Each term is a component (attribute) of the vector
 - The value of each component is the number of times the corresponding term occurs in the document.

	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

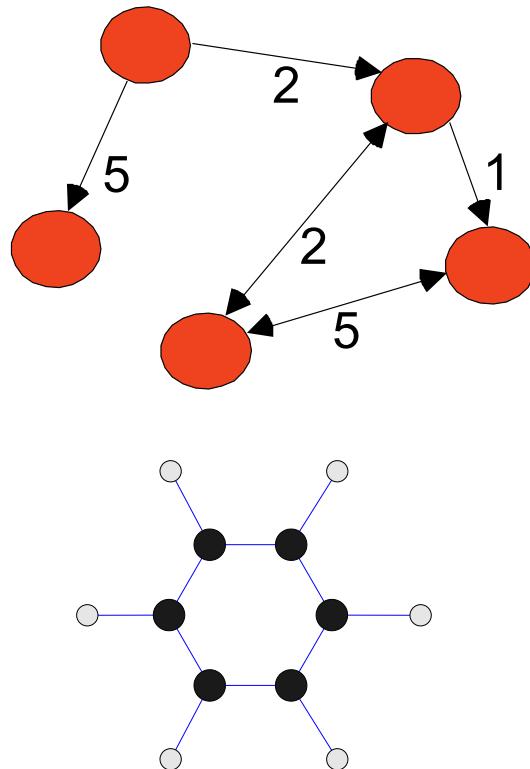
Transaction Data

- A special type of data, where
 - Each transaction involves a set of items.
 - For example, consider a grocery store. The set of products purchased by a customer during one shopping trip constitute a transaction, while the individual products that were purchased are the items.
 - Can represent transaction data as record data

<i>TID</i>	<i>Items</i>
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Graph Data

Examples: Generic graph, a molecule, and webpages



Benzene Molecule: C₆H₆

Useful Links:

- [Bibliography](#)
- Other Useful Web sites
 - [ACM SIGKDD](#)
 - [KDnuggets](#)
 - [The Data Mine](#)

Knowledge Discovery and Data Mining Bibliography

(Gets updated frequently, so visit often!)

- [Books](#)
- [General Data Mining](#)

Book References in Data Mining and Knowledge Discovery

Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Iyer, "Advances in Knowledge Discovery and Data Mining", AAAI Press/the MIT Press, 1996.

J. Ross Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers, 1993. Michael Berry and Gordon Linoff, "Data Mining Techniques (For Marketing, Sales, and Customer Support)", John Wiley & Sons, 1997.

General Data Mining

Usama Fayyad, "Mining Databases: Towards Algorithms for Knowledge Discovery", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 21, no. 1, March 1998.

Christopher Matheus, Philip Chan, and Gregory Piatetsky-Shapiro, "Systems for knowledge Discovery in databases", IEEE Transactions on Knowledge and Data Engineering, 5(6):903-913, December 1993.

Ordered Data

Sequences of transactions

Items/Events



(A B) (D) (C E)
(B D) (C) (E)
(C D) (B) (A E)



**An element of
the sequence**

Ordered Data

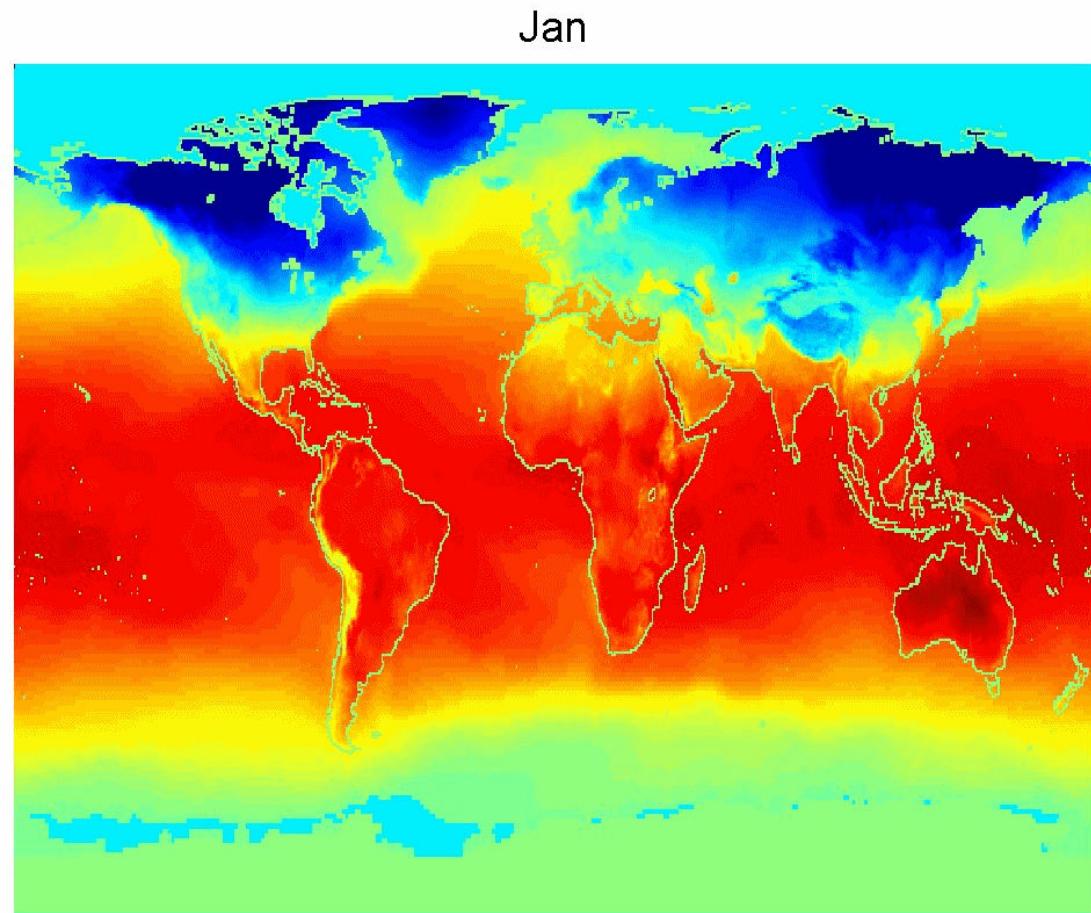
Genomic sequence data

**GGTTCCGCCTTCAGCCCCGCGCC
CGCAGGGCCCGCCCCGCGCCGTG
GAGAAGGGCCCGCCTGGCGGGCG
GGGGGAGGCAGGGGCCGCCCAGC
CCAACCGAGTCCGACCAGGTGCC
CCCTCTGCTCGGCCTAGACCTGA
GCTCATTAGGCAGCAGCGGACAG
GCCAAGTAGAACACGCGAAGCGC
TGGGCTGCCTGCTGCGACCAGGG**

Ordered Data

Spatiotemporal Data

Average Monthly
Temperature of land
and ocean



Data Quality

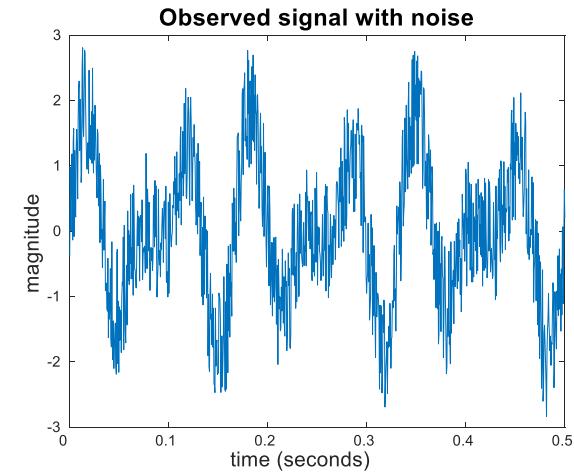
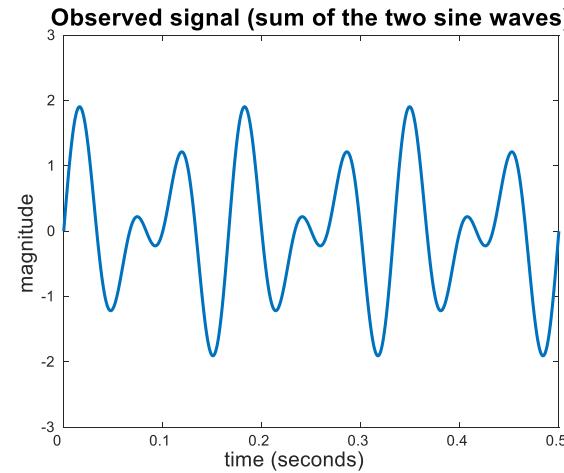
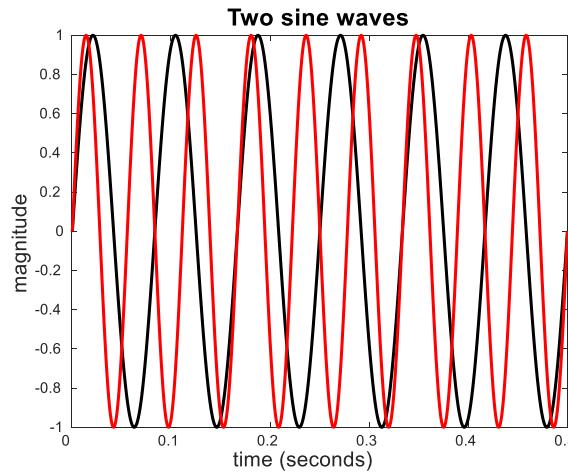
- Poor data quality negatively affects many data processing efforts
- ML example: a classification model for detecting people who are loan risks is built using poor data
 - Some credit-worthy candidates are denied loans
 - More loans are given to individuals that default

Data Quality ...

- What kinds of data quality problems?
 - How can we detect problems with the data?
 - What can we do about these problems?
-
- Examples of data quality problems:
 - Noise and outliers
 - Wrong data
 - Fake data
 - Missing values
 - Duplicate data

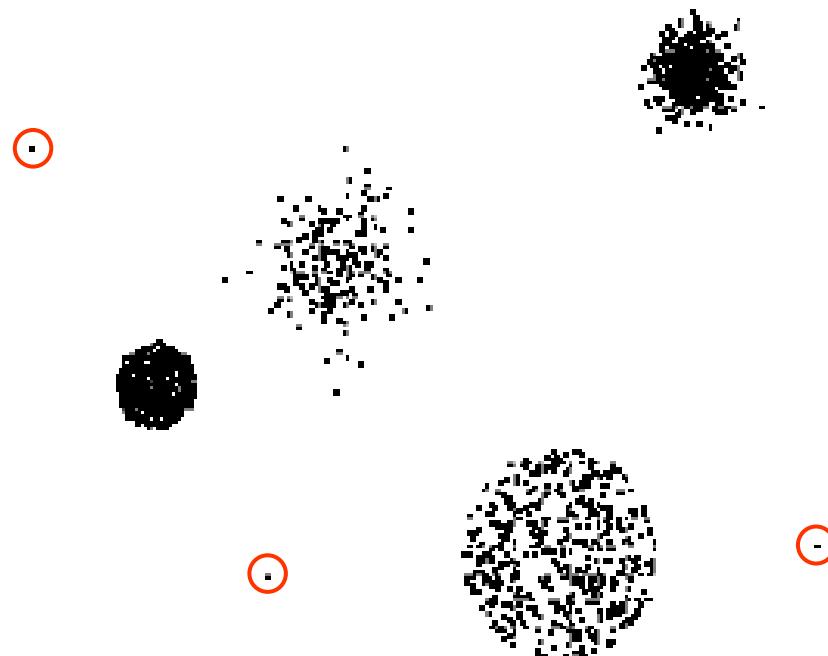
Noise

- For objects, noise is an extraneous object
- For attributes, noise refers to modification of original values
 - Examples: distortion of a person's voice when talking on a poor phone and "snow" on television screen
 - The figures below show two sine waves of the same magnitude and different frequencies, the waves combined, and the two sine waves with random noise
 - The magnitude and shape of the original signal is distorted



Outliers

- **Outliers** are data objects with characteristics that are considerably different than most of the other data objects in the data set
 - **Case 1:** Outliers are noise that interferes with data analysis
 - **Case 2:** Outliers are the goal of our analysis
 - Credit card fraud
 - Intrusion detection
- Causes?



Missing Values

- Reasons for missing values
 - Information is not collected
(e.g., people decline to give their age and weight)
 - Attributes may not be applicable to all cases
(e.g., annual income is not applicable to children)
- Handling missing values
 - Eliminate data objects or variables
 - Estimate missing values
 - Example: time series of temperature
 - Example: census results
 - Ignore the missing value during analysis

Duplicate Data

- Data set may include data objects that are duplicates, or almost duplicates of one another
 - Major issue when merging data from heterogeneous sources
 - Examples:
 - Same person with multiple email addresses
 - Data cleaning
 - Process of dealing with duplicate data issues
-

Data Preprocessing

- Aggregation
- Sampling
- Discretization and Binarization
- Attribute Transformation
- Dimensionality Reduction
- Feature subset selection
- Feature creation

Aggregation

- Combining two or more attributes (or objects) into a single attribute (or object)
- Purpose
 - Data reduction - reduce the number of attributes or objects
 - Change of scale
 - Cities aggregated into regions, states, countries, etc.
 - Days aggregated into weeks, months, or years
 - More “stable” data - aggregated data tends to have less variability

Table 2.4. Data set containing information about customer purchases.

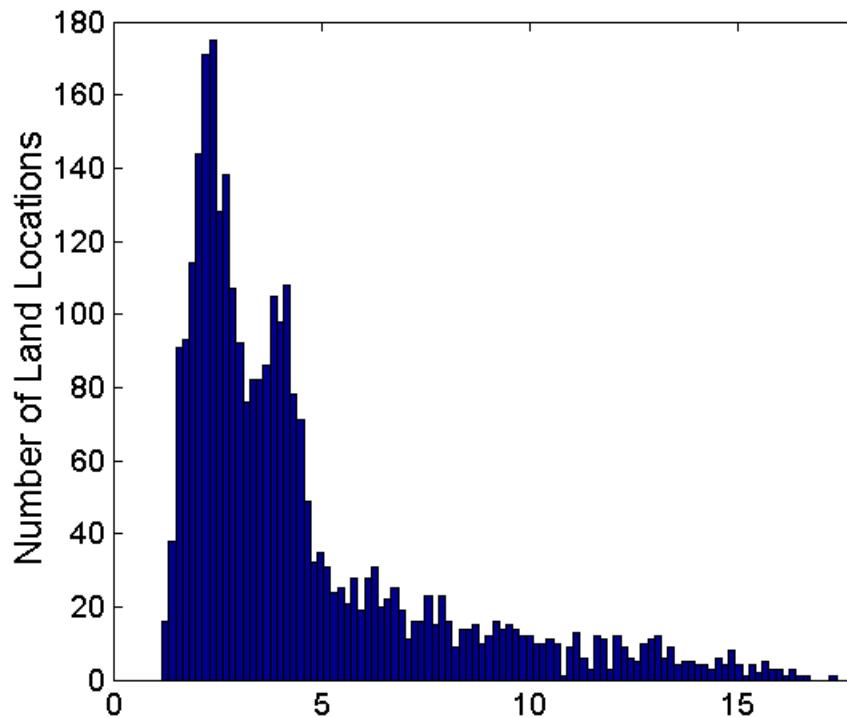
Transaction ID	Item	Store Location	Date	Price	...
:	:	:	:	:	
101123	Watch	Chicago	09/06/04	\$25.99	...
101123	Battery	Chicago	09/06/04	\$5.99	...
101124	Shoes	Minneapolis	09/06/04	\$75.00	...
:	:	:	:	:	

Example: Precipitation in Australia

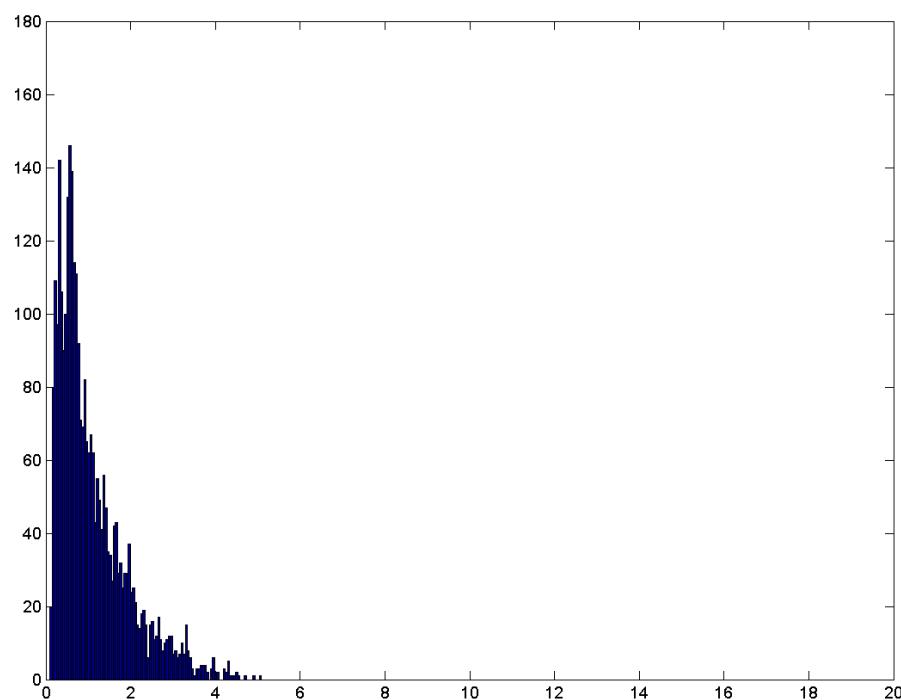
- This example is based on precipitation in Australia from the period 1982 to 1993.
The next slide shows
 - A histogram for the standard deviation of average monthly precipitation for 3,030 0.5° by 0.5° grid cells in Australia, and
 - A histogram for the standard deviation of the average yearly precipitation for the same locations.
- The average yearly precipitation has less variability than the average monthly precipitation.
- All precipitation measurements (and their standard deviations) are in centimeters.

Example: Precipitation in Australia ...

Variation of Precipitation in Australia



Standard Deviation of Average Monthly Precipitation



Standard Deviation of Average Yearly Precipitation

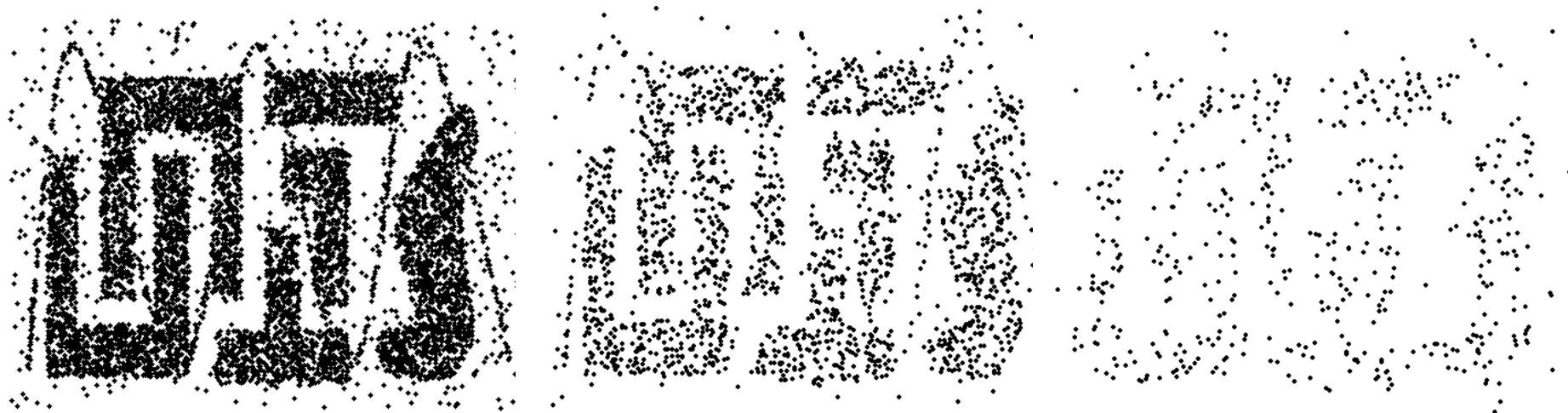
Sampling

- Sampling is the main technique employed for data reduction.
 - It is often used for both the preliminary investigation of the data and the final data analysis.
- Statisticians often sample because **obtaining** the entire set of data of interest is too expensive or time consuming.
- Sampling is typically used in data mining because **processing** the entire set of data of interest is too expensive or time consuming.

Sampling ...

- The key principle for effective sampling is the following:
 - Using a sample will work almost as well as using the entire data set, if the sample is **representative**
 - A sample is **representative** if it has approximately the same properties (of interest) as the original set of data

Sample Size



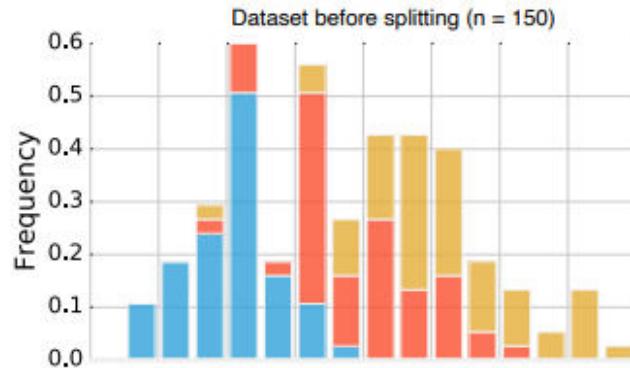
8000 points

2000 Points

500 Points



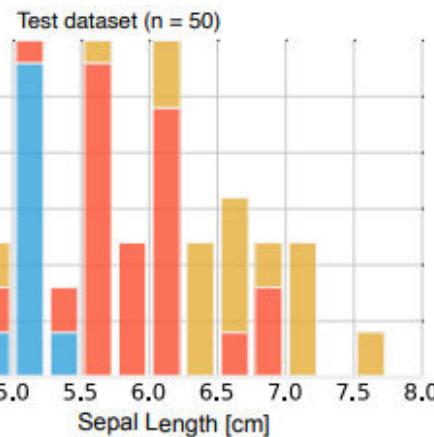
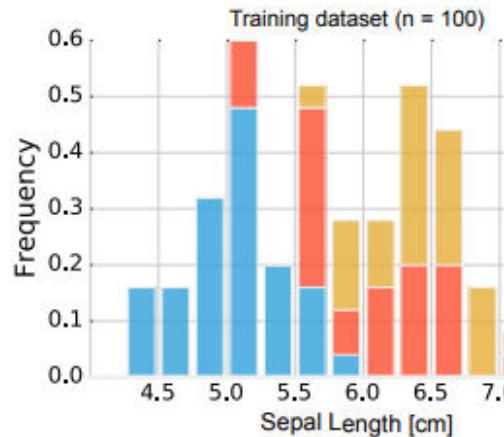
Issues with Subsampling (Independence Violation)



IRIS Dataset of Flowers

50 Setosa, 50 Versicolor, and 50 Virginica

- Setosa
- Versicolor
- Virginica



- Random subsampling can assign 2/3 (100) to training set and 1/3 (50) to the test set
- Training set → 38 x Setosa, 28 x Versicolor, 34 x Virginica
- Test set → 12 x Setosa, 22 x Versicolor, 16 x Virginica

Types of Sampling

- **Simple Random Sampling**
 - There is an equal probability of selecting any particular item
 - Sampling without replacement
 - As each item is selected, it is removed from the population
 - Sampling with replacement
 - Objects are not removed from the population as they are selected for the sample.
 - In sampling with replacement, the same object can be picked up more than once
 - **Stratified sampling**
 - Split the data into several partitions; then draw random samples from each partition
-

Building Classifiers with Imbalanced Training Set

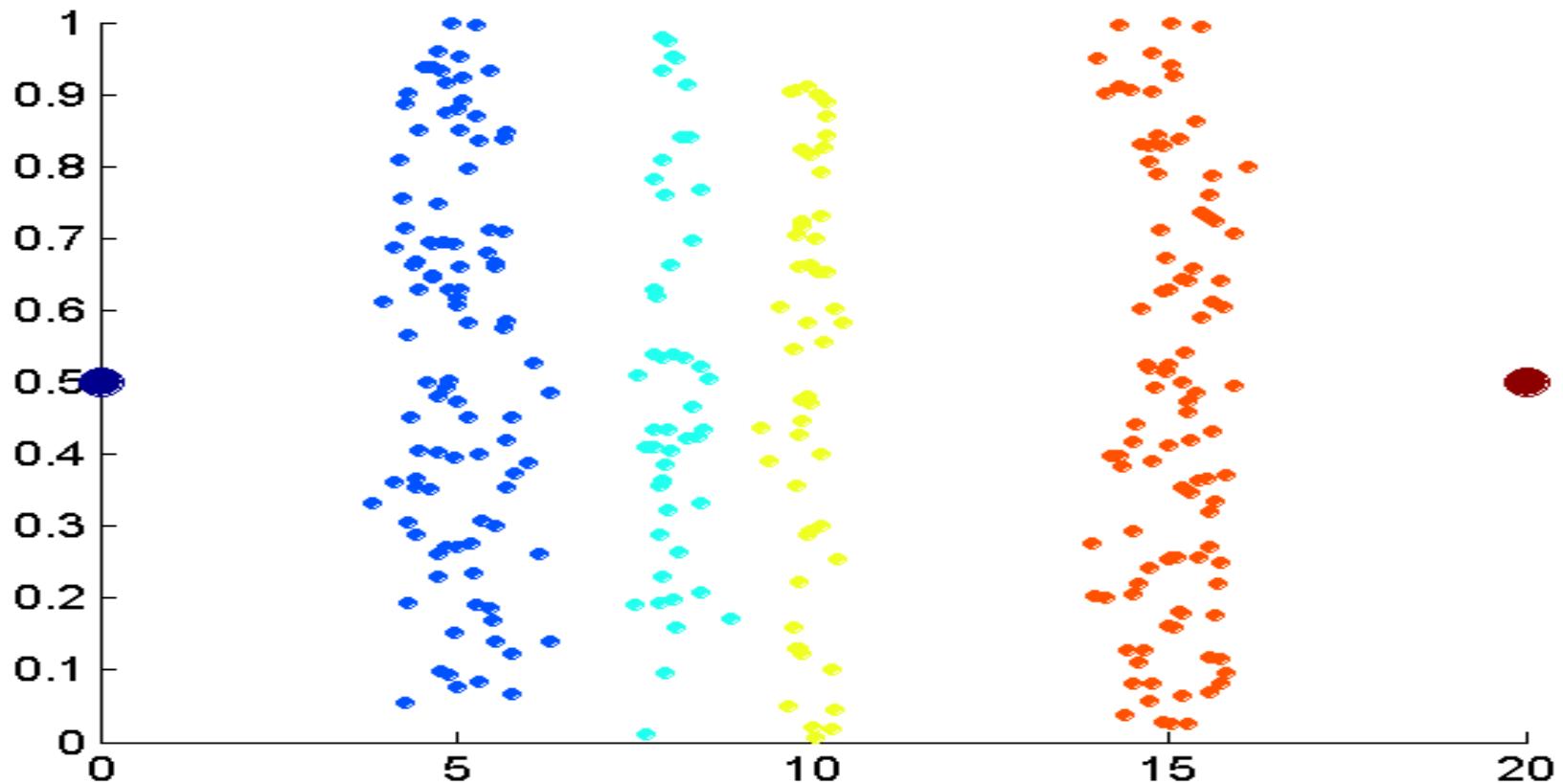


- Modify the distribution of training data so that rare class is well-represented in training set
 - Undersample the majority class
 - Oversample the rare class
-

Discretization

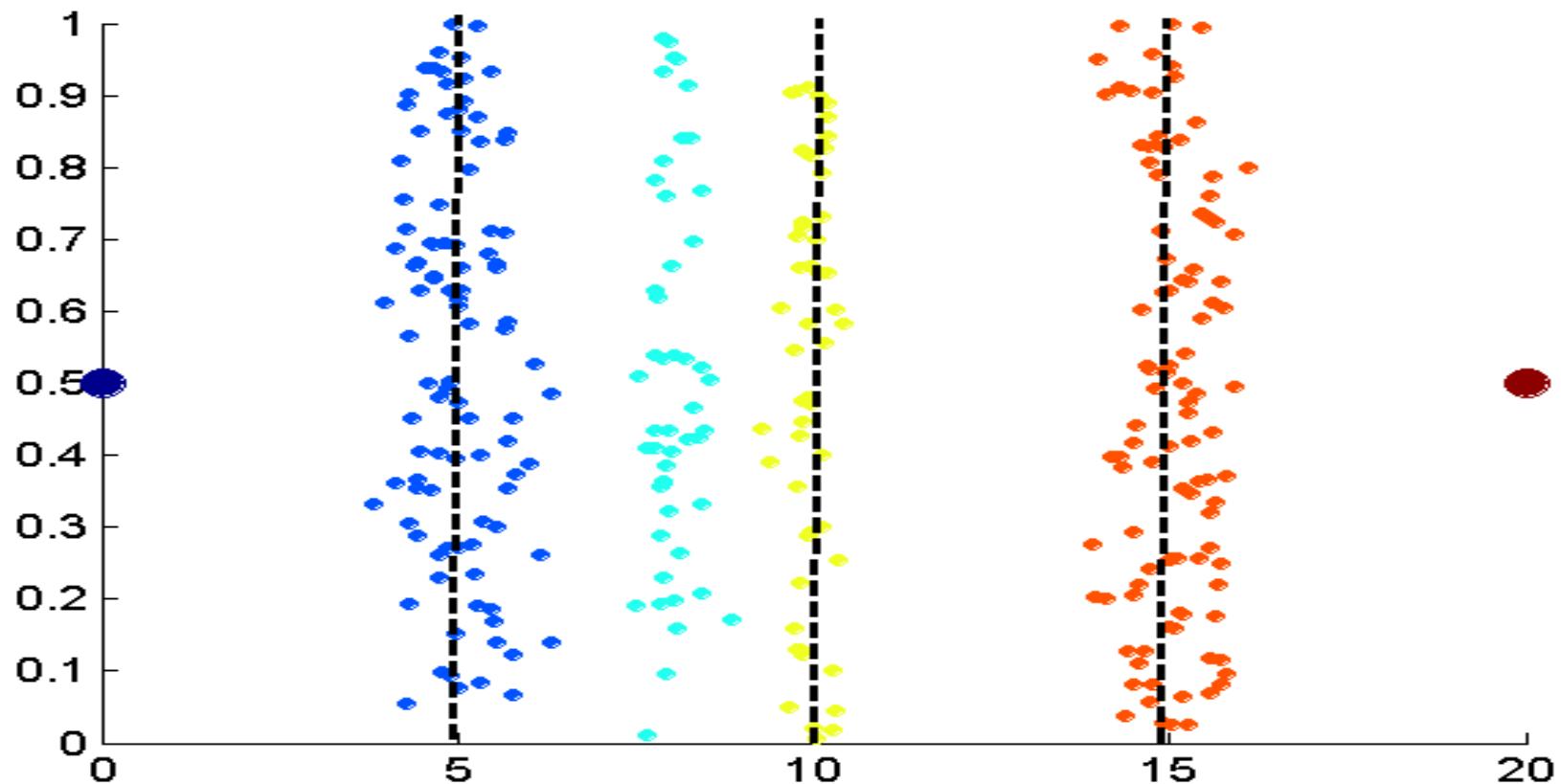
- **Discretization** is the process of converting a continuous attribute into an ordinal attribute
 - A potentially infinite number of values are mapped into a small number of categories
 - Discretization is used in both unsupervised and supervised settings
-

Unsupervised Discretization



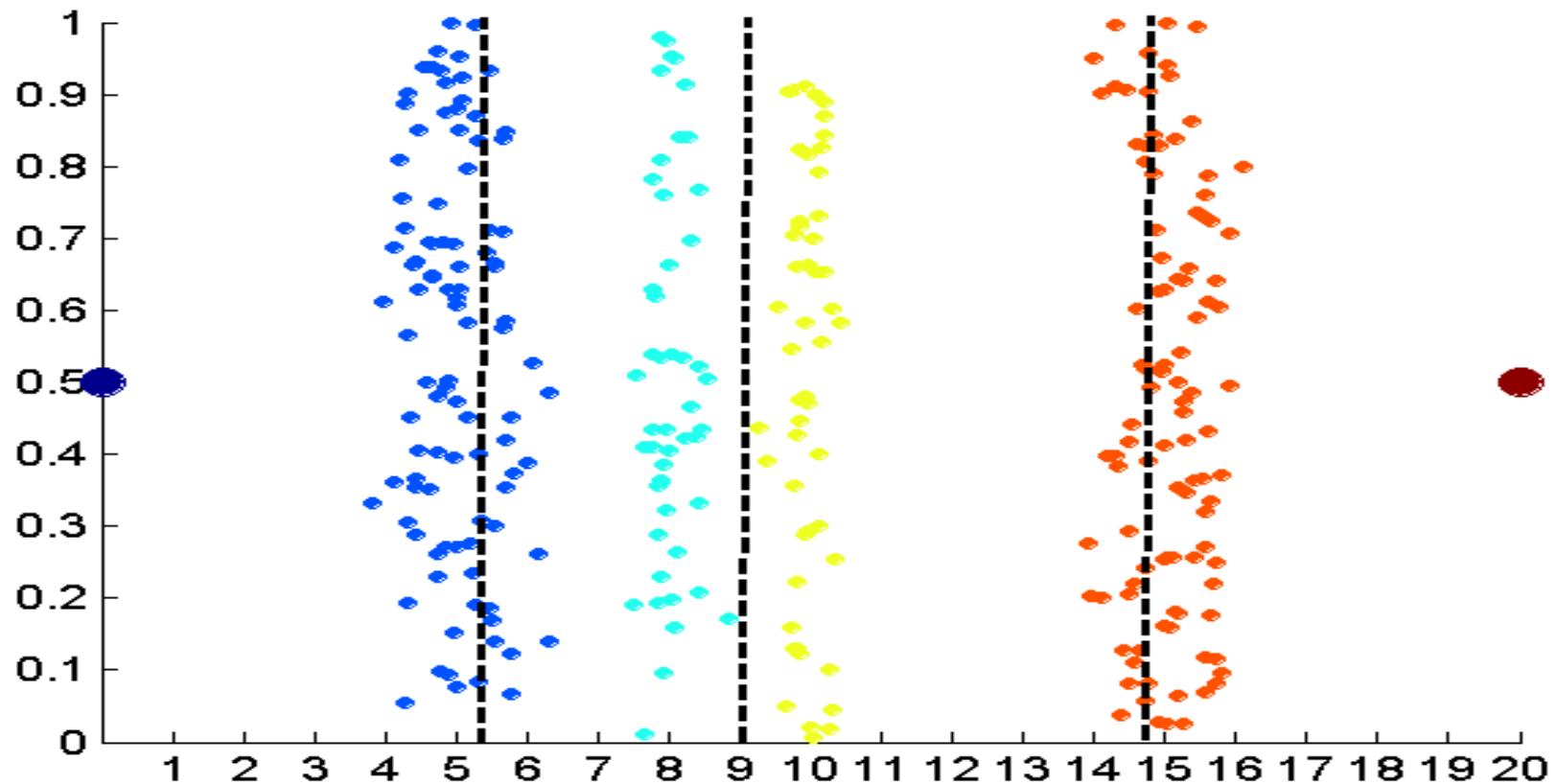
Data consists of four groups of points and two outliers. Data is one-dimensional, but a random y component is added to reduce overlap.

Unsupervised Discretization



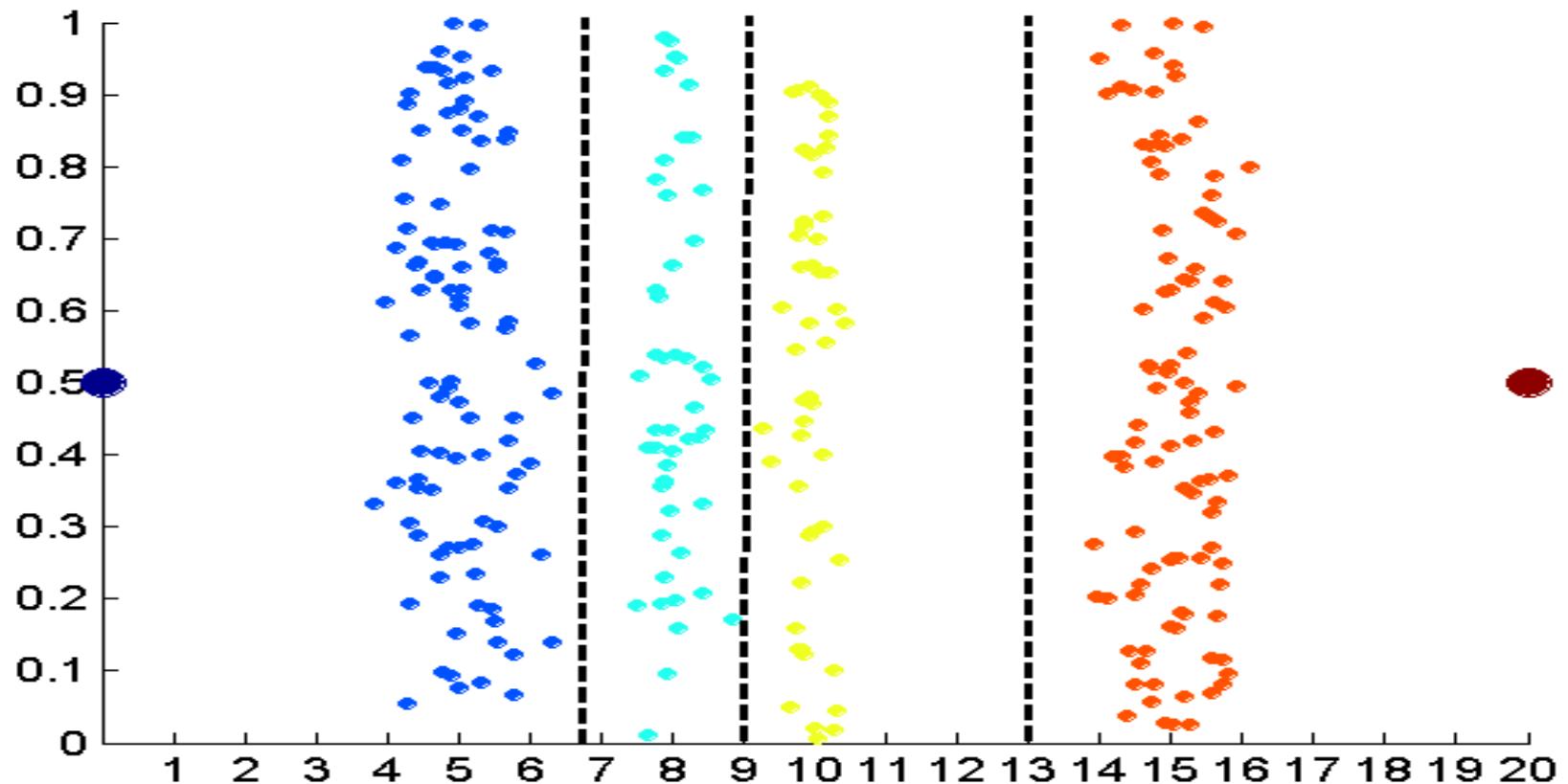
Equal interval width approach used to obtain 4 values.

Unsupervised Discretization



Equal frequency approach used to obtain 4 values.

Unsupervised Discretization



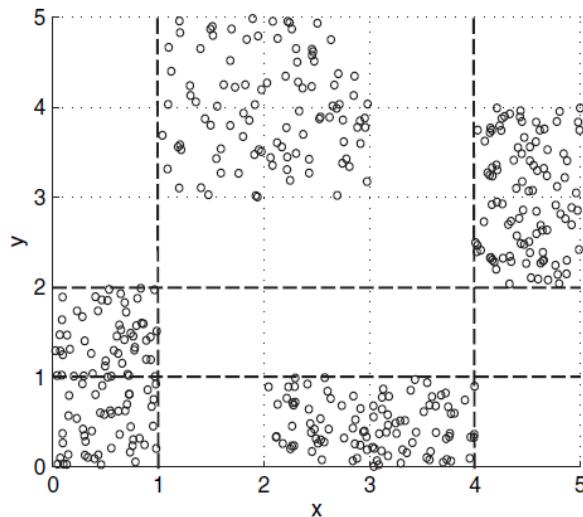
K-means approach to obtain 4 values.



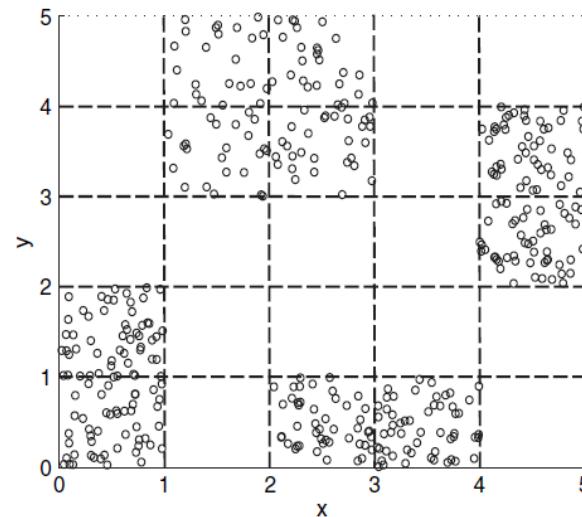
Discretization in Supervised Settings



- Many classification algorithms work best if both the independent and dependent variables have only a few values
- We give an illustration of the usefulness of discretization using the following example.



(a) Three intervals



(b) Five intervals

Figure 2.14. Discretizing x and y attributes for four groups (classes) of points.

Binarization

- Binarization maps a continuous or categorical attribute into one or more binary variables

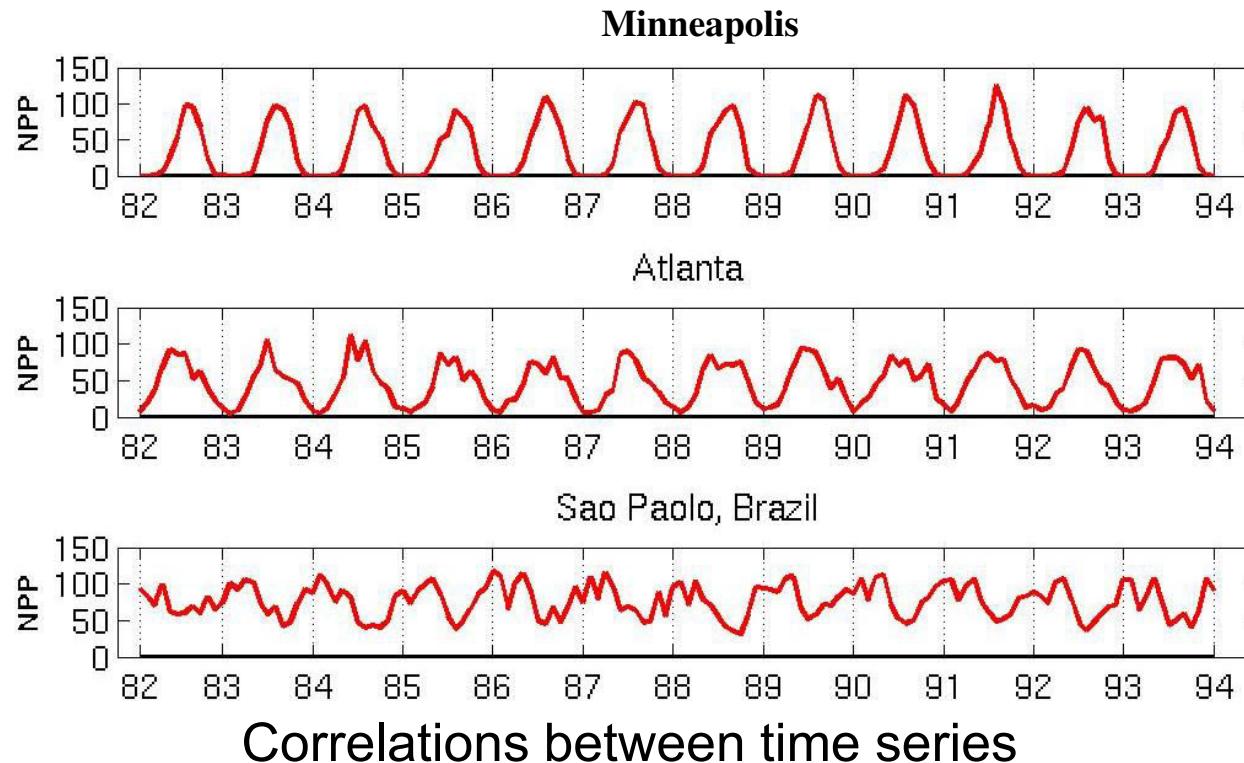
Table 2.6. Conversion of a categorical attribute to five asymmetric binary attributes.

Categorical Value	Integer Value	x_1	x_2	x_3	x_4	x_5
<i>awful</i>	0	1	0	0	0	0
<i>poor</i>	1	0	1	0	0	0
<i>OK</i>	2	0	0	1	0	0
<i>good</i>	3	0	0	0	1	0
<i>great</i>	4	0	0	0	0	1

Attribute Transformation

- An **attribute transform** is a function that maps the entire set of values of a given attribute to a new set of replacement values such that each old value can be identified with one of the new values
 - Simple functions: x^k , $\log(x)$, e^x , $|x|$
 - **Normalization**
 - Refers to various techniques to adjust to differences among attributes in terms of frequency of occurrence, mean, variance, range
 - Take out unwanted, common signal, e.g., seasonality
 - In statistics, **standardization** refers to subtracting off the means and dividing by the standard deviation

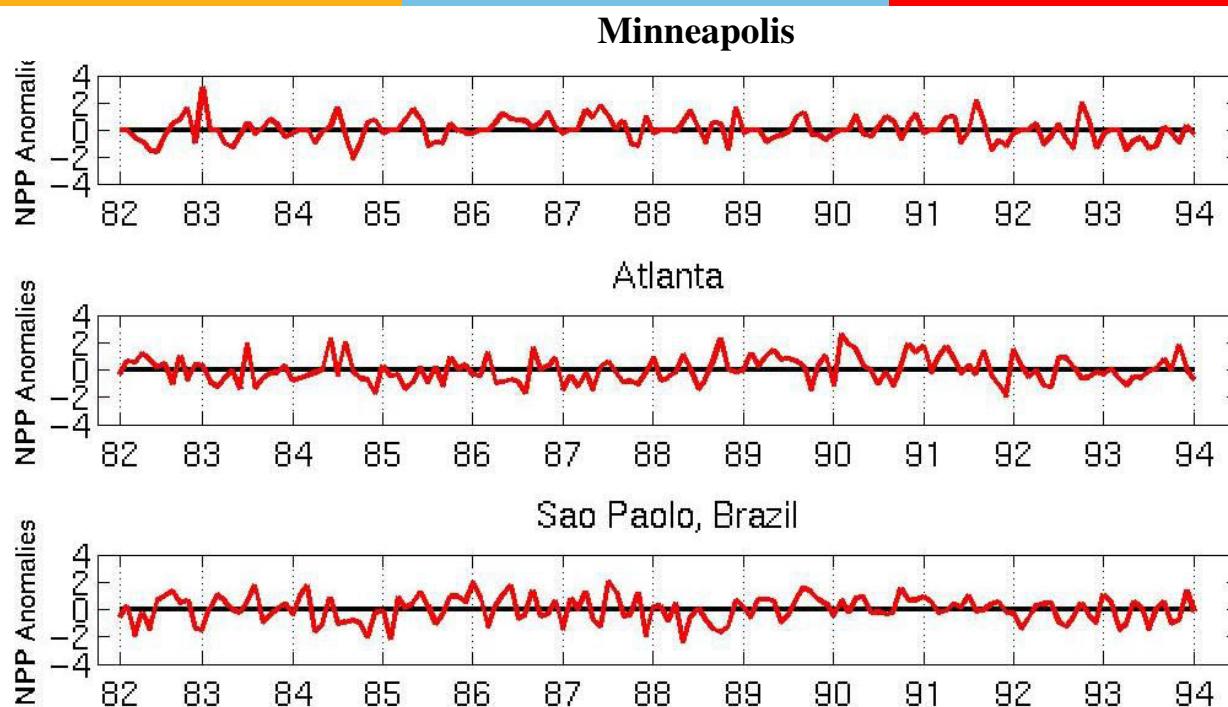
Example: Sample Time Series of Plant Growth



Net Primary Production (NPP) is a measure of plant growth used by ecosystem scientists.

	Minneapolis	Atlanta	Sao Paolo
Minneapolis	1.0000	0.7591	-0.7581
Atlanta	0.7591	1.0000	-0.5739
Sao Paolo	-0.7581	-0.5739	1.0000

Seasonality Accounts for Much Correlation



Normalized using monthly Z Score:

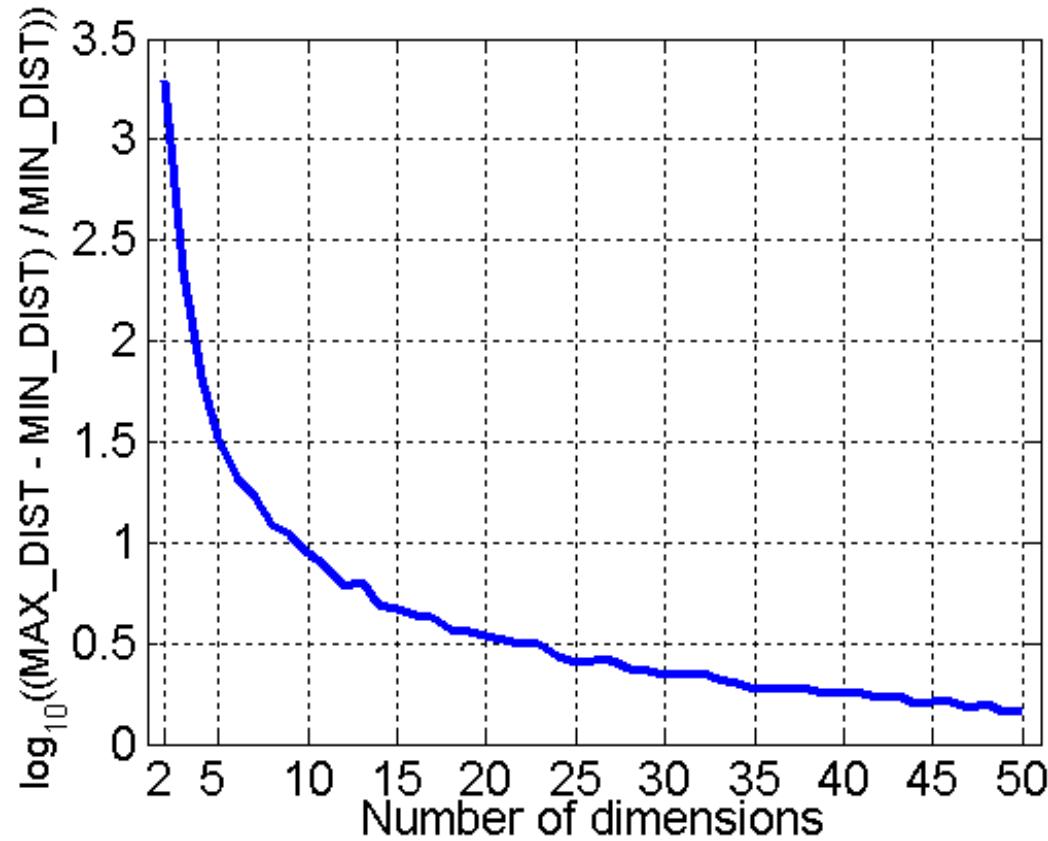
Subtract off monthly mean and divide by monthly standard deviation

Correlations between time series

	Minneapolis	Atlanta	Sao Paolo
Minneapolis	1.0000	0.0492	0.0906
Atlanta	0.0492	1.0000	-0.0154
Sao Paolo	0.0906	-0.0154	1.0000

Curse of Dimensionality

- When dimensionality increases, data becomes increasingly sparse in the space that it occupies
- Definitions of density and distance between points, which are critical for clustering and outlier detection, become less meaningful



- Randomly generate 500 points
- Compute difference between max and min distance between any pair of points

Dimensionality Reduction

- Purpose:
 - Avoid curse of dimensionality
 - Reduce amount of time and memory required by data mining algorithms
 - Allow data to be more easily visualized
 - May help to eliminate irrelevant features or reduce noise
- Techniques
 - Principal Components Analysis (PCA)
 - Singular Value Decomposition
 - Others: supervised and non-linear techniques

Feature Subset Selection

- Another way to reduce dimensionality of data
 - Redundant features
 - Duplicate much or all of the information contained in one or more other attributes
 - Example: purchase price of a product and the amount of sales tax paid
 - Irrelevant features
 - Contain no information that is useful for the data mining task at hand
 - Example: students' ID is often irrelevant to the task of predicting students' GPA
 - Many techniques developed, especially for classification
-

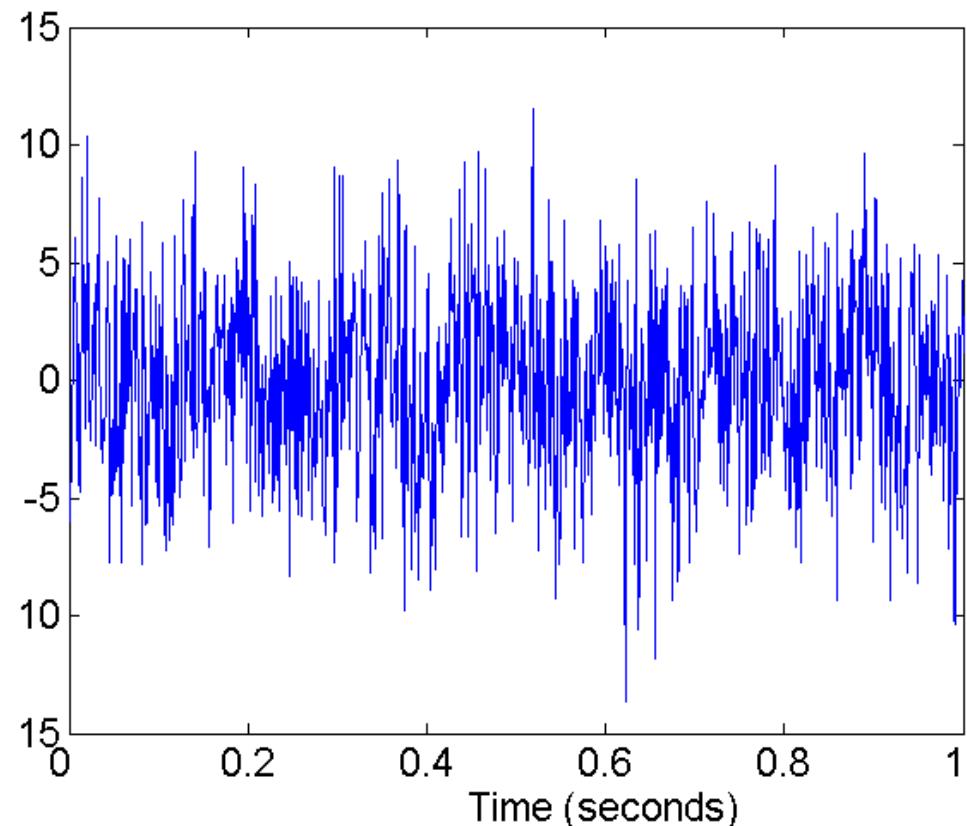
Feature Creation

- Create new attributes that can capture the important information in a data set much more efficiently than the original attributes
- Three general methodologies:
 - Feature extraction
 - Example: extracting edges from images
 - Feature construction
 - Example: dividing mass by volume to get density
 - Mapping data to new space
 - Example: Fourier and wavelet analysis

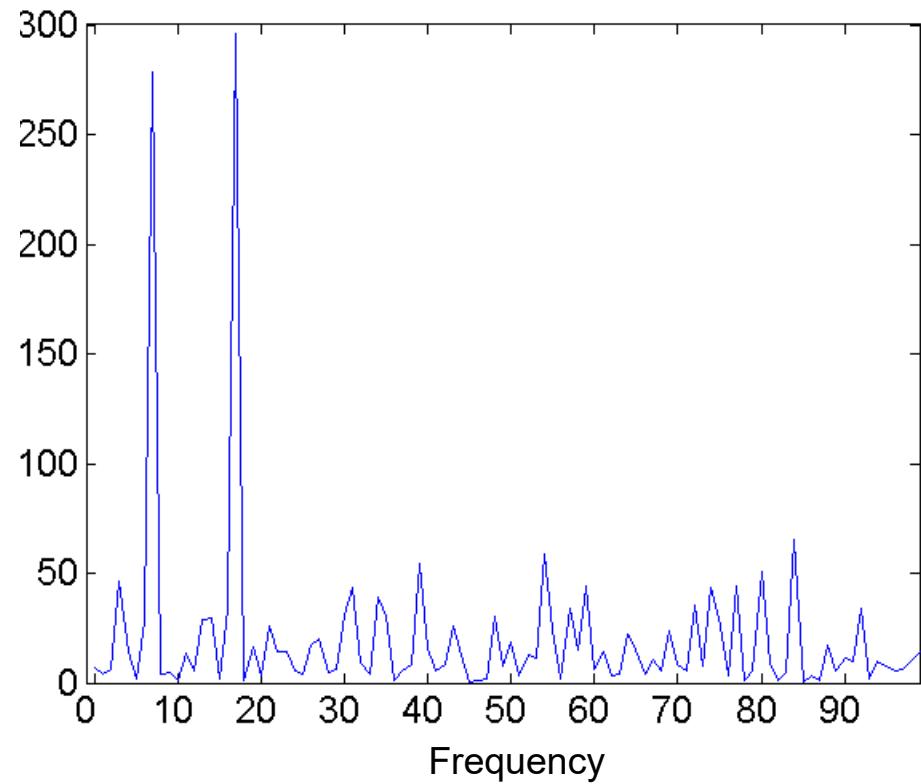
Mapping Data to a New Space



Fourier and wavelet transform



Two Sine Waves + Noise



Frequency

Evaluation Metrics: Confusion Matrix

		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	a	b
	Class>No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Evaluation Metrics: Accuracy

		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class>No	c (FP)	d (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}$$

Class Imbalance Problem

- Lots of classification problems where the classes are skewed (more records from one class than another)
 - Credit card fraud
 - Intrusion detection
 - Defective products in manufacturing assembly line
 - COVID-19 test results on a random sample
- **Key Challenge:**
 - Evaluation measures such as accuracy are not well-suited for imbalanced class

Problem with Accuracy

- Consider a 2-class problem
 - Number of Class NO examples = 990
 - Number of Class YES examples = 10
- If a model predicts everything to be class NO, accuracy is $990/1000 = 99\%$
 - This is misleading because this trivial model does not detect any class YES example
 - Detecting the rare class is usually more interesting (e.g., frauds, intrusions, defects, etc)

		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	0	10
	Class>No	0	990

Which model is better?

A

		PREDICTED	
ACTUAL		Class=Yes	Class>No
	Class=Yes	0	10
	Class>No	0	990

Accuracy: 99%

B

		PREDICTED	
ACTUAL		Class=Yes	Class>No
	Class=Yes	10	0
	Class>No	500	490

Accuracy: 50%

Which model is better?

A

		PREDICTED	
ACTUAL		Class=Yes	Class>No
	Class=Yes	5	5
	Class>No	0	990

B

		PREDICTED	
ACTUAL		Class=Yes	Class>No
	Class=Yes	10	0
	Class>No	500	490

Alternative Measures

		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	a	b
	Class>No	c	d

$$\text{Precision (p)} = \frac{a}{a + c}$$

$$\text{Recall (r)} = \frac{a}{a + b}$$

$$\text{F - measure (F)} = \frac{2rp}{r + p} = \frac{2a}{2a + b + c}$$

Alternative Measures

		PREDICTED CLASS	
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	10	0
	Class>No	10	980

$$\text{Precision (p)} = \frac{10}{10+10} = 0.5$$

$$\text{Recall (r)} = \frac{10}{10+0} = 1$$

$$\text{F - measure (F)} = \frac{2 * 1 * 0.5}{1 + 0.5} = 0.62$$

$$\text{Accuracy} = \frac{990}{1000} = 0.99$$

Alternative Measures

ACTUAL CLASS	PREDICTED CLASS		
		Class=Yes	Class>No
	Class=Yes	10	0
	Class>No	10	980

$$\text{Precision (p)} = \frac{10}{10+10} = 0.5$$

$$\text{Recall (r)} = \frac{10}{10+0} = 1$$

$$\text{F - measure (F)} = \frac{2 * 1 * 0.5}{1 + 0.5} = 0.62$$

$$\text{Accuracy} = \frac{990}{1000} = 0.99$$

ACTUAL CLASS	PREDICTED CLASS		
		Class=Yes	Class>No
	Class=Yes	1	9
	Class>No	0	990

$$\text{Precision (p)} = \frac{1}{1+0} = 1$$

$$\text{Recall (r)} = \frac{1}{1+9} = 0.1$$

$$\text{F - measure (F)} = \frac{2 * 0.1 * 1}{1 + 0.1} = 0.18$$

$$\text{Accuracy} = \frac{991}{1000} = 0.991$$

Which of these classifiers is better?



A

		PREDICTED CLASS	
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	40	10
	Class>No	10	40

Precision (p) = 0.8

Recall (r) = 0.8

F - measure (F) = 0.8

Accuracy = 0.8

B

		PREDICTED CLASS	
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	40	10
	Class>No	1000	4000

Precision (p) = ~ 0.04

Recall (r) = 0.8

F - measure (F) = ~ 0.08

Accuracy = ~ 0.8

Measures of Classification Performance

	PREDICTED CLASS	
ACTUAL CLASS	Yes	No
	Yes	TP
	No	FP

α is the probability that we reject the null hypothesis when it is true. This is a Type I error or a false positive (FP).

β is the probability that we accept the null hypothesis when it is false. This is a Type II error or a false negative (FN).

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}$$

$$\text{ErrorRate} = 1 - \text{accuracy}$$

$$\text{Precision} = \text{Positive Predictive Value} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \text{Sensitivity} = \text{TP Rate} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \text{TN Rate} = \frac{TN}{TN + FP}$$

$$\text{FP Rate} = \alpha = \frac{FP}{TN + FP} = 1 - \text{specificity}$$

$$\text{FN Rate} = \beta = \frac{FN}{FN + TP} = 1 - \text{sensitivity}$$

$$\text{Power} = \text{sensitivity} = 1 - \beta$$

Alternative Measures

A	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	40	10
	Class>No	10	40

Precision (p) = 0.8
 TPR = Recall (r) = 0.8
 FPR = 0.2
 F-measure (F) = 0.8
 Accuracy = 0.8

$$\frac{\text{TPR}}{\text{FPR}} = 4$$

B	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	40	10
	Class>No	1000	4000

Precision (p) = 0.038
 TPR = Recall (r) = 0.8
 FPR = 0.2
 F-measure (F) = 0.07
 Accuracy = 0.8

$$\frac{\text{TPR}}{\text{FPR}} = 4$$

Which of these classifiers is better?

ACTUAL CLASS	PREDICTED CLASS	
	Class=Yes	Class>No
Class=Yes	10	40
Class>No	10	40

Precision (p) = 0.5

TPR = Recall (r) = 0.2

FPR = 0.2

F – measure = 0.28

ACTUAL CLASS	PREDICTED CLASS	
	Class=Yes	Class>No
Class=Yes	25	25
Class>No	25	25

Precision (p) = 0.5

TPR = Recall (r) = 0.5

FPR = 0.5

F – measure = 0.5

ACTUAL CLASS	PREDICTED CLASS	
	Class=Yes	Class>No
Class=Yes	40	10
Class>No	40	10

Precision (p) = 0.5

TPR = Recall (r) = 0.8

FPR = 0.8

F – measure = 0.61

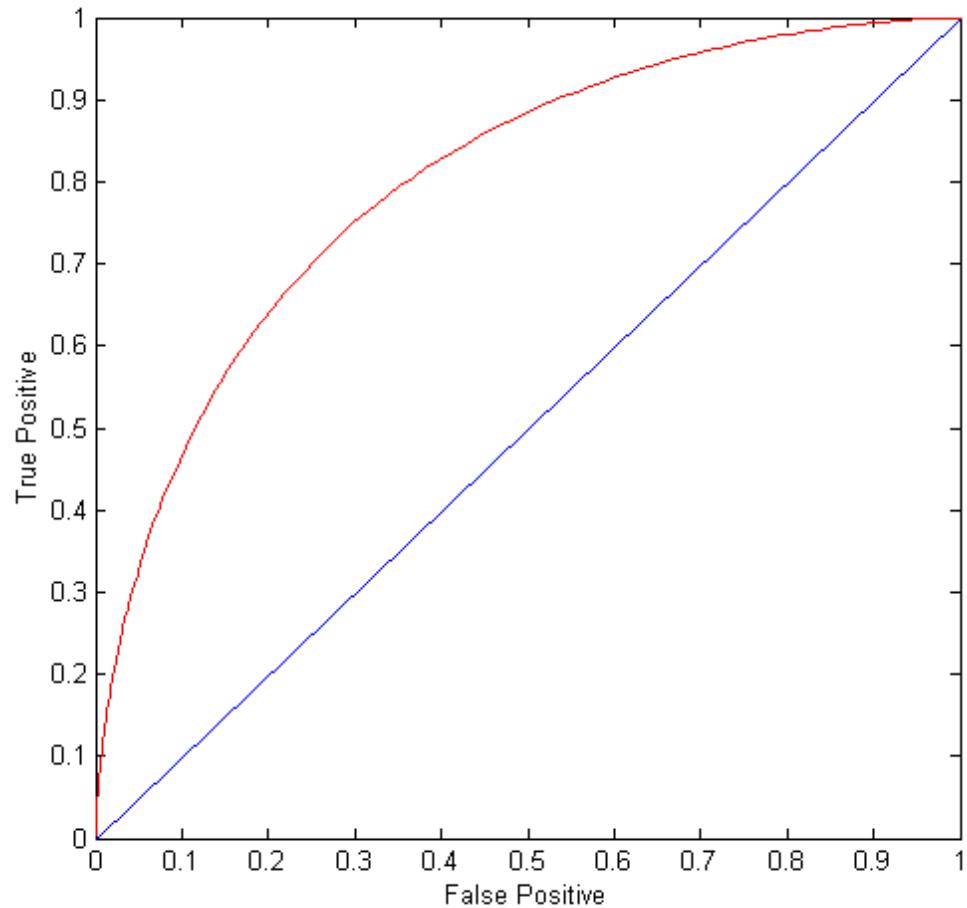
ROC (Receiver Operating Characteristic)

- A graphical approach for displaying trade-off between detection rate and false alarm rate
- Developed in 1950s for signal detection theory to analyze noisy signals
- ROC curve plots TPR against FPR
 - Performance of a model represented as a point in an ROC curve

ROC Curve

(TPR,FPR):

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal
- Diagonal line:
 - Random guessing
 - Below diagonal line:
 - prediction is opposite of the true class



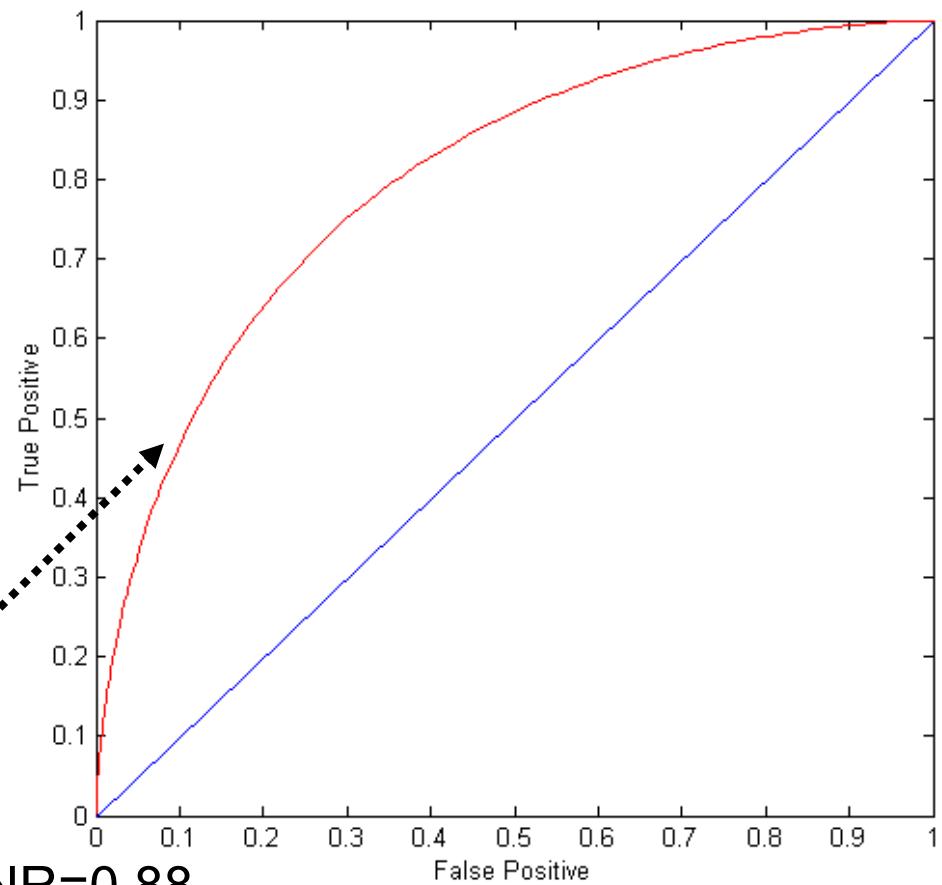
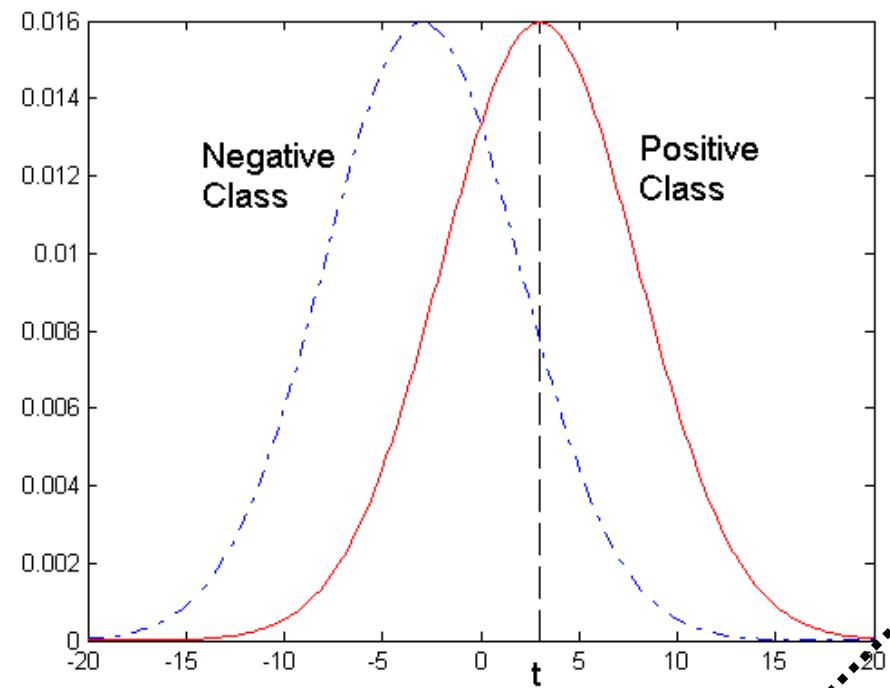
ROC (Receiver Operating Characteristic)



- To draw ROC curve, classifier must produce continuous-valued output
 - Outputs are used to rank test records, from the most likely positive class record to the least likely positive class record
 - By using different thresholds on this value, we can create different variations of the classifier with TPR/FPR tradeoffs
- Many classifiers produce only discrete outputs (i.e., predicted class)
 - How to get continuous-valued outputs?
 - Decision trees, rule-based classifiers, neural networks, Bayesian classifiers, k-nearest neighbors, SVM

ROC Curve Example

- 1-dimensional data set containing 2 classes (positive and negative)
- Any points located at $x > t$ is classified as positive



At threshold t :

$\text{TPR}=0.5$, $\text{FNR}=0.5$, $\text{FPR}=0.12$, $\text{TNR}=0.88$

How to Construct an ROC curve

Instance	Score	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

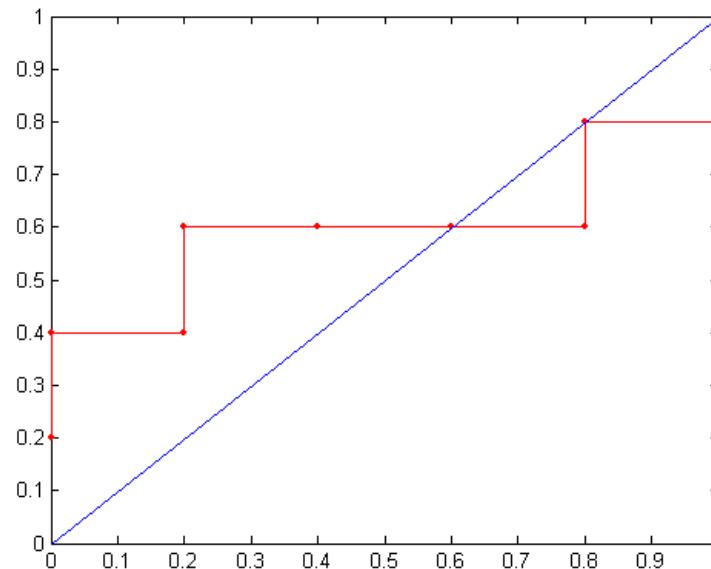
- Use a classifier that produces a continuous-valued score for each instance
 - The more likely it is for the instance to be in the + class, the higher the score
- Sort the instances in decreasing order according to the score
- Apply a threshold at each unique value of the score
- Count the number of TP, FP, TN, FN at each threshold
 - $TPR = TP/(TP+FN)$
 - $FPR = FP/(FP + TN)$

How to construct an ROC curve

Class	+	-	+	-	-	-	+	-	+	+	
Threshold >=	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

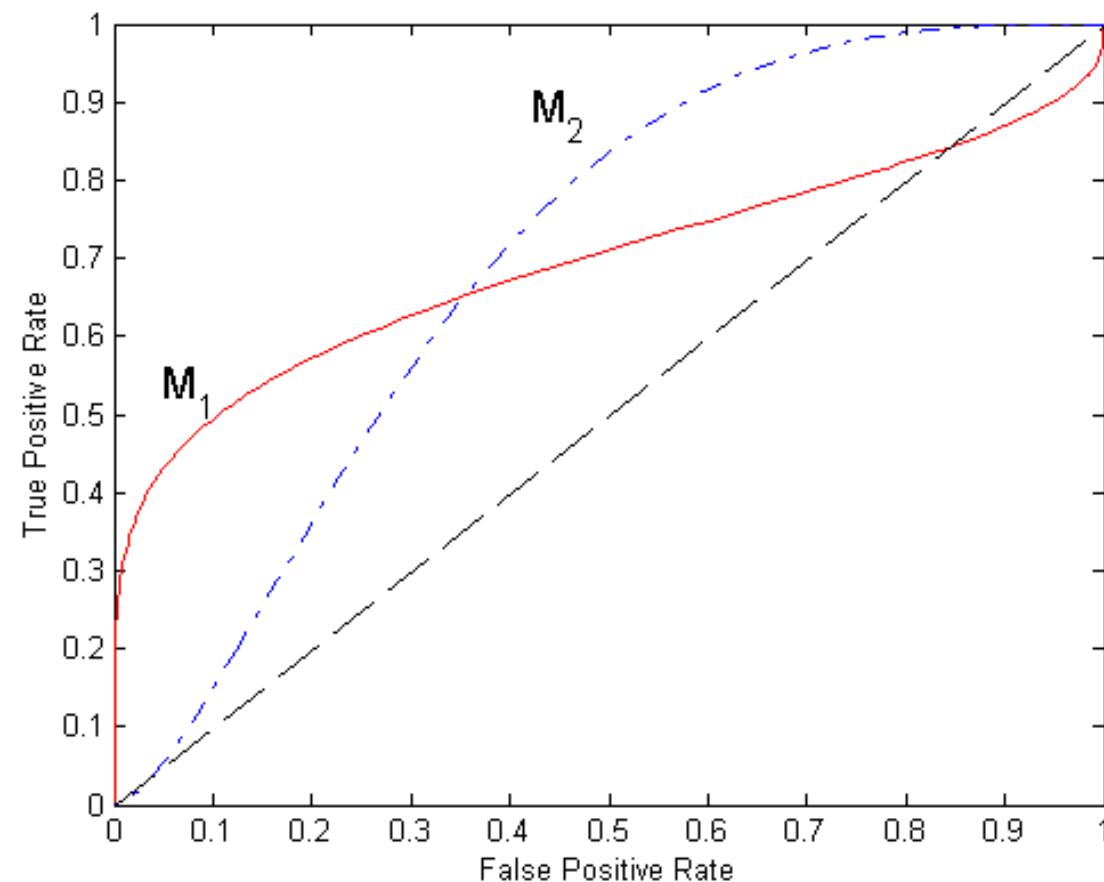
→ →

ROC Curve:



Instance	Score	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

Using ROC for Model Comparison



- No model consistently outperforms the other
 - M_1 is better for small FPR
 - M_2 is better for large FPR
- Area Under the ROC curve (AUC)
 - Ideal: Area = 1
 - Random guess: Area = 0.5

Dealing with Imbalanced Classes - Summary

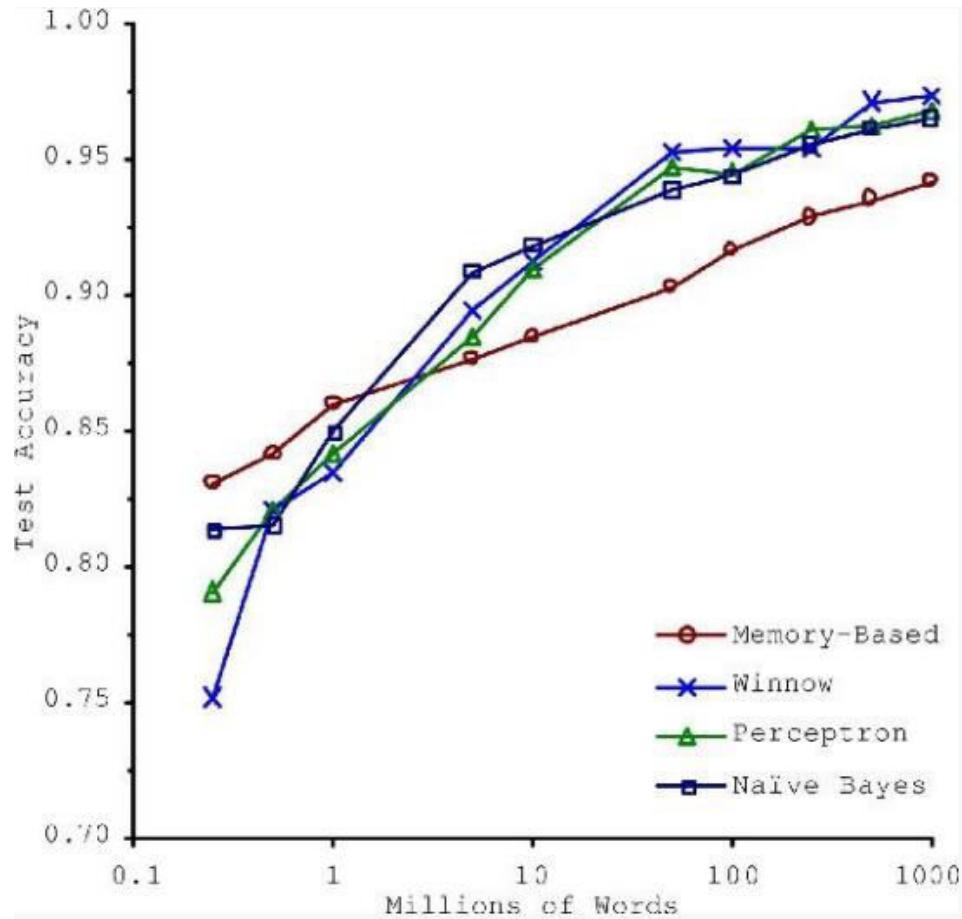
- Many measures exists, but none of them may be ideal in all situations
 - Random classifiers can have high value for many of these measures
 - TPR/FPR provides important information but may not be sufficient by itself in many practical scenarios
 - Given two classifiers, sometimes you can tell that one of them is strictly better than the other
 - C_1 is strictly better than C_2 if C_1 has strictly better TPR and FPR relative to C_2 (or same TPR and better FPR, and vice versa)
 - Even if C_1 is strictly better than C_2 , C_1 's F-value can be worse than C_2 's if they are evaluated on data sets with different imbalances
 - Classifier C_1 can be better or worse than C_2 depending on the scenario at hand (class imbalance, importance of TP vs FP, cost/time tradeoffs)

Challenges of Machine Learning

- Training Data
 - Insufficient
 - Non representative
- Model Selection
 - Overfitting
 - Underfitting
- Validation and Testing

Insufficient Training Data

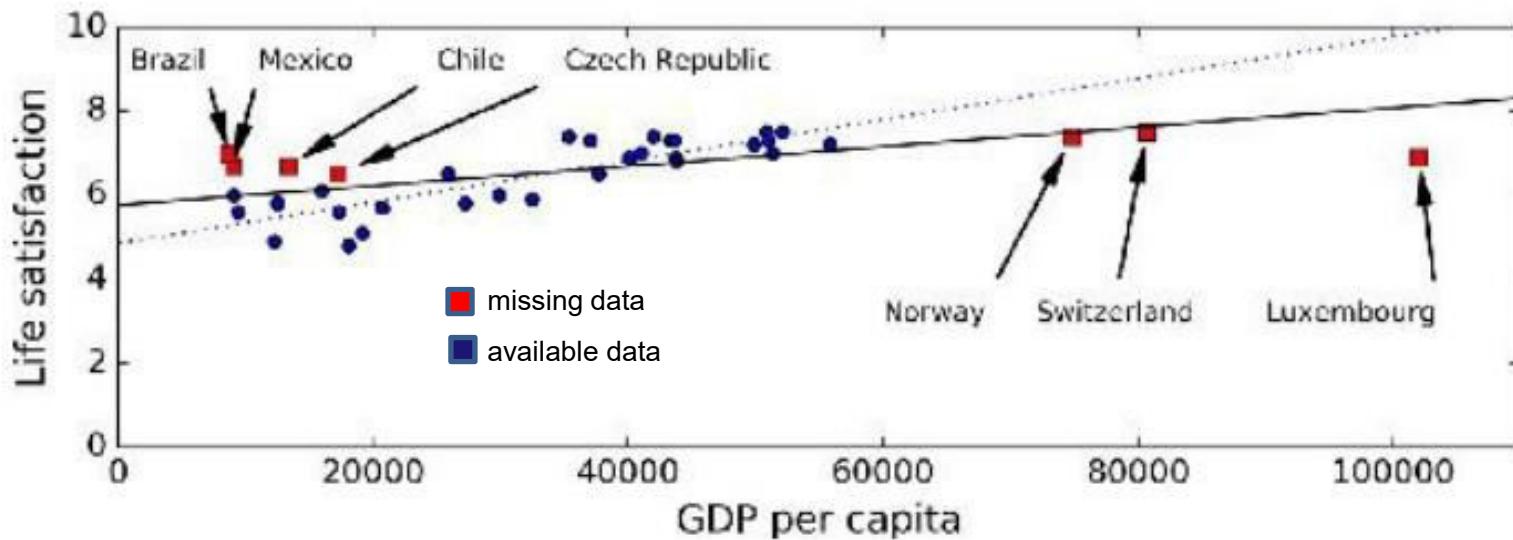
Consider trade-off Between Algorithm development & training data capture



Non-representative Training Data

Training Data be representative of the new cases we want to generalize

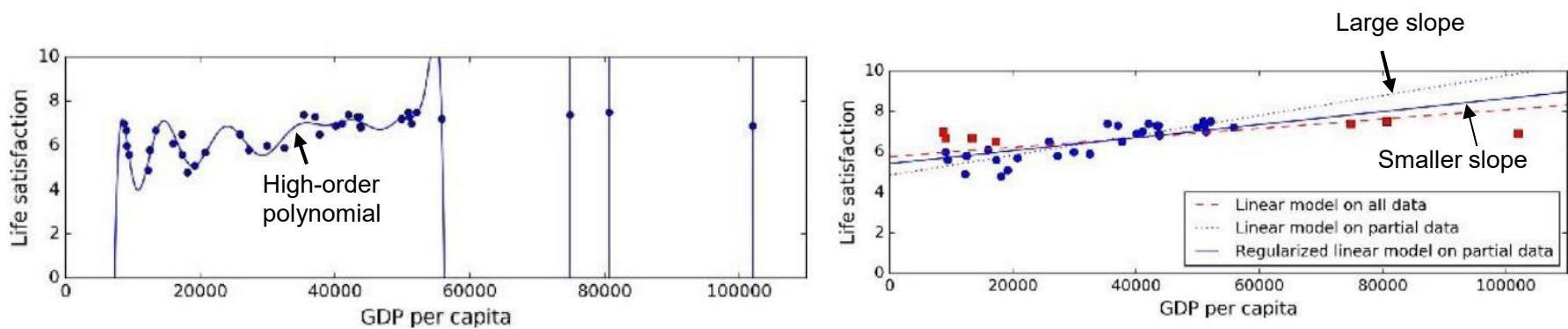
- Small sample size leads to sampling noise
 - Missing data over emphasizes the role of wealth on happiness
- If sampling process is flawed, even large sample size can lead to sampling bias



Model Selection

Overfitting or Underfitting

- Overfitting leads to high performance in training set but performs poorly on new data
 - e.g., a high-degree polynomial life satisfaction model that strongly overfits the training data
 - Small training set or sampling noise can lead to model following the noise than the underlying pattern in the dataset
 - Solution: Regularization to constrain the values of parameters



- Underfitting when the model is too simple to learn the underlying structure in the data
 - Select a more powerful model, with more parameters
 - Feed better features to the learning algorithm
 - Reduce regularization

Choice of Hyperparameters

Modern ML models often use a lot of model parameters

- Known as *hyperparameters*
- Model performance depends on choice of parameters
- Each parameter can assume a number of values
 - Real numbers or *categories*
- Exponential number of hyperparameter combinations possible
- Best model correspond to best cross validation performance over the set of hyperparameter combinations
- Expensive to perform
- Some empirical frameworks available for hyperparameter optimization
 - Grid search
 - Random search
 - Bayesian

Evaluating Predictive Performance of a Model



- Want to estimate the generalization performance, the predictive performance of our model on future (unseen) data.
 - Want to increase the predictive performance by tweaking the learning algorithm and selecting the best performing model from a given hypothesis space.
 - Want to identify the ML algorithm that is best-suited for the problem at hand; thus, we want to compare different algorithms, selecting the best-performing one as well as the best performing model from the algorithm's hypothesis space.
-

Evaluation and Validation

Performance of ML algorithms is statistical / predictive

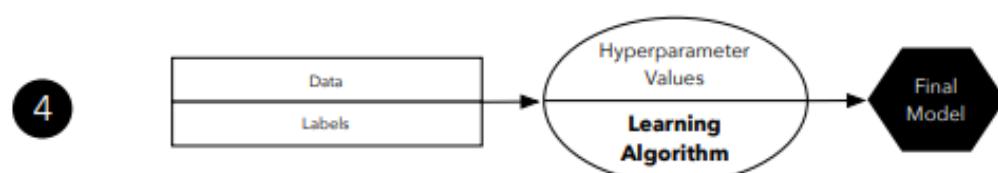
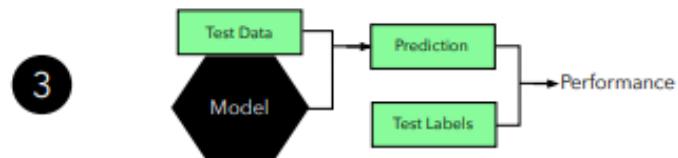
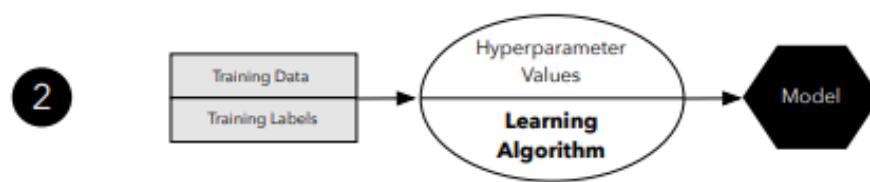
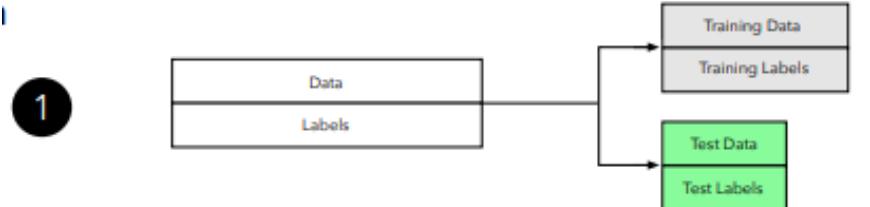
- Good ML algorithms need to work well on test data
 - But test data is often not accessible to the provider of the algorithm
- Common assumption is training data is representative of test data
- Randomly chosen subset of the training data is held out as validation set, aka dev set
- Once ML model is trained, its performance is evaluated on validation data
 - Expectation is ML model working well on validation set will work well on unknown test data
- Typically 20-30% of the data is randomly held out as validation data

Cross Validation

K-fold validation is often performed

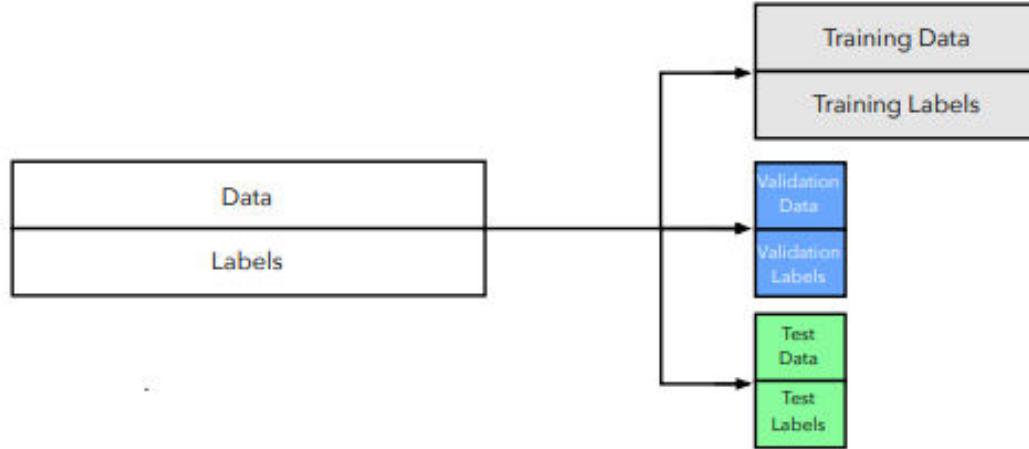
- To reduce the bias of validation set selection process
- Often K is chosen as 10
 - aka 10 fold cross validation
- 10 fold cross validation involves
 - randomly selecting the validation set 10 times
 - model generation with 10 resulting training set
 - Evaluate the performance of each on that validation set
 - averaging the performance over the validation sets

Holdout for Model Evaluation

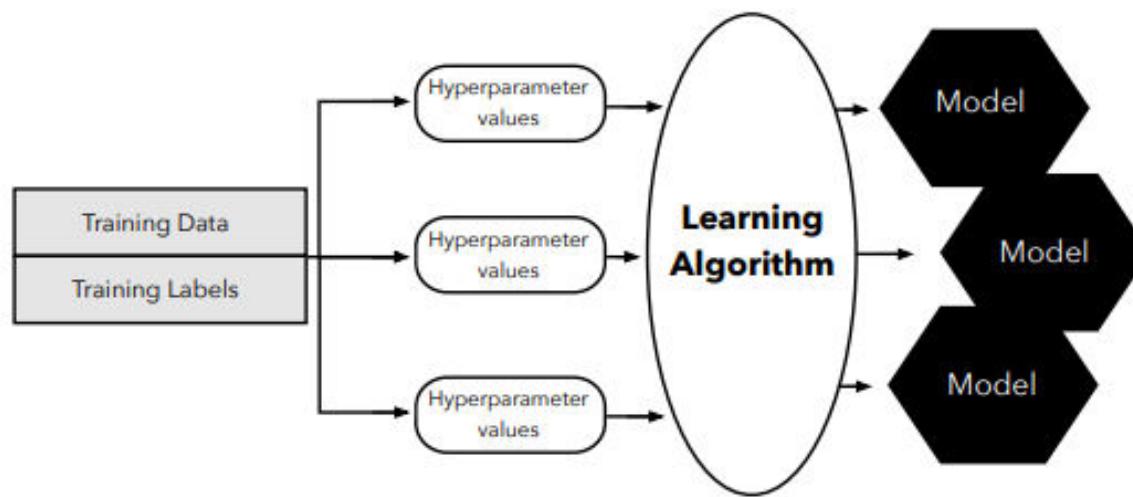


Holdout for Model Selection (1)

1

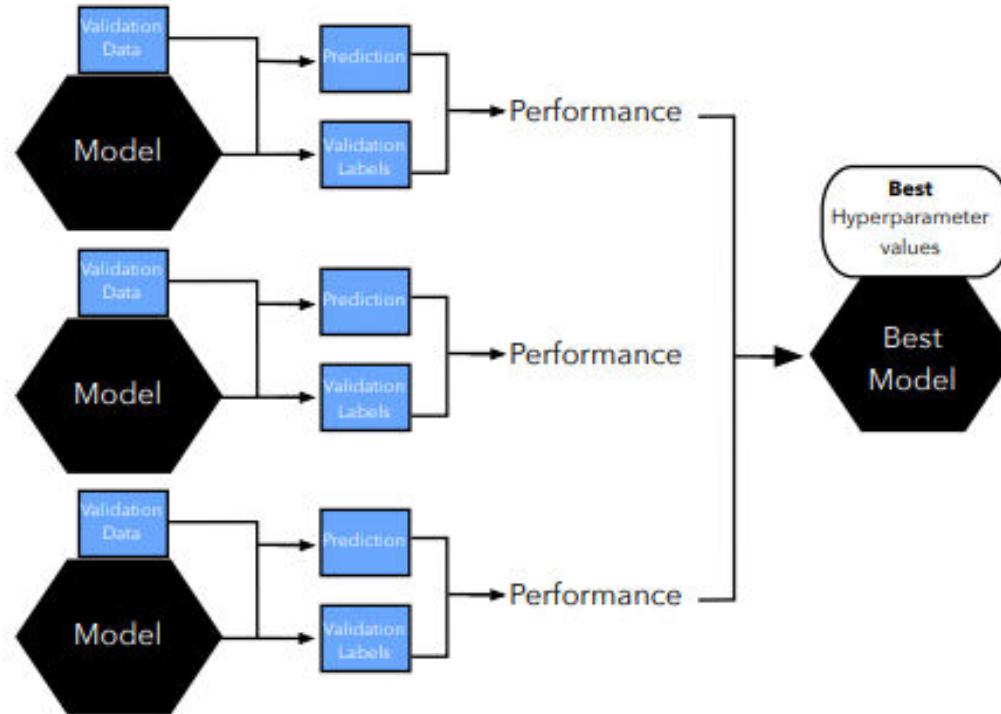


2

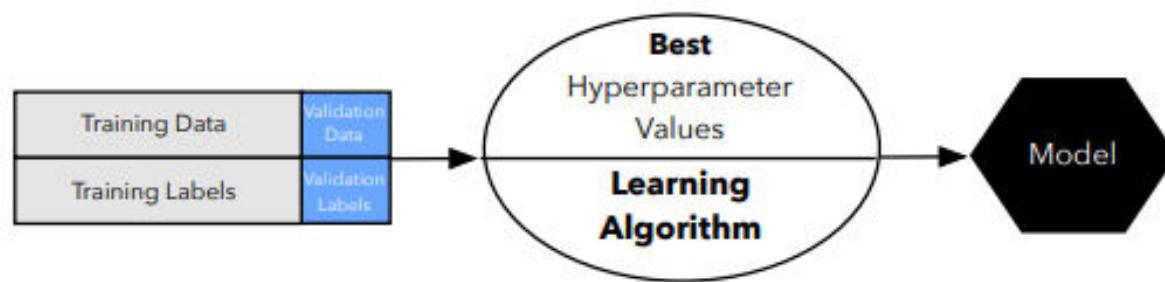


Holdout for Model Selection (2)

3

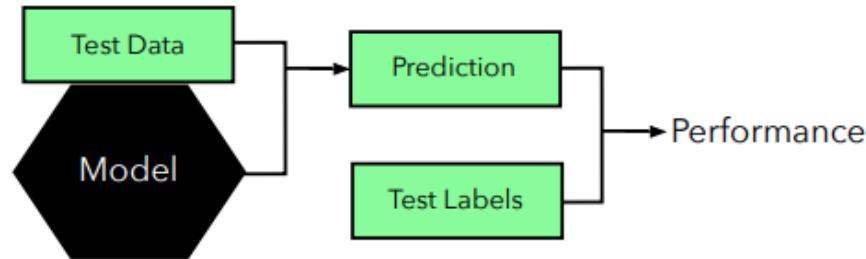


4

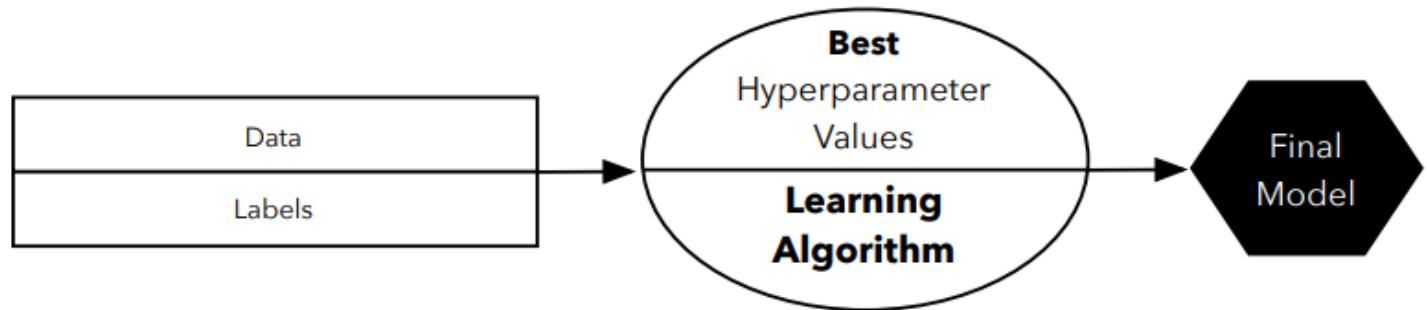


Holdout for Model Selection (3)

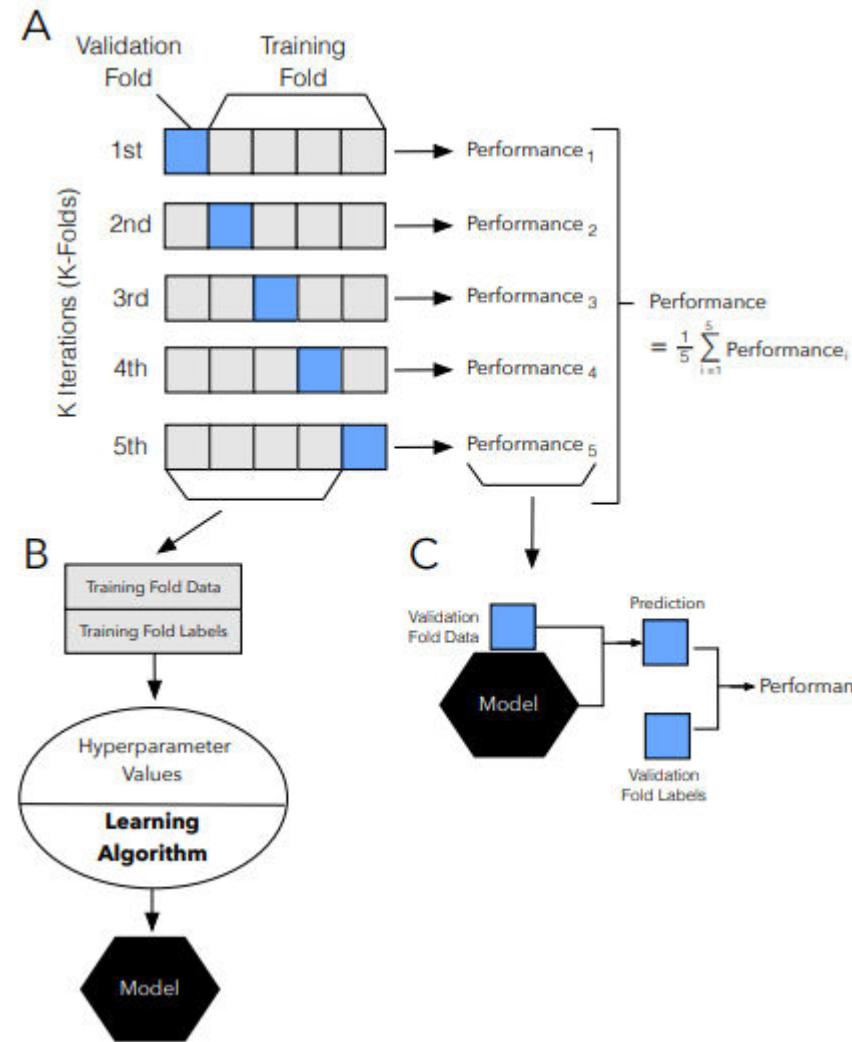
5



6

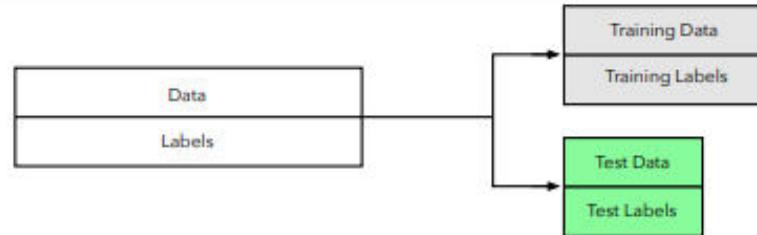


K Fold Cross Validation (CV) for Model Evaluation

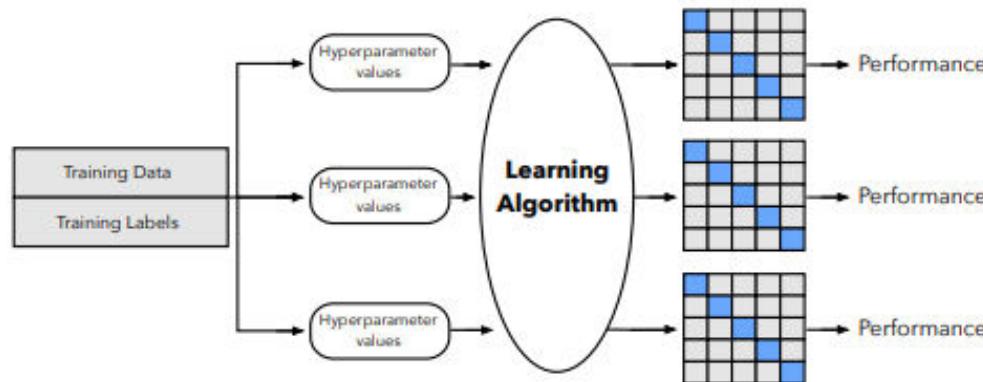


CV for Model Selection (1)

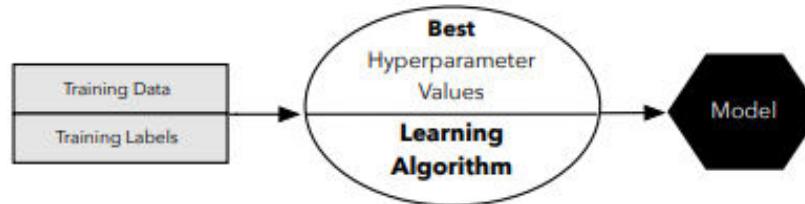
1



2

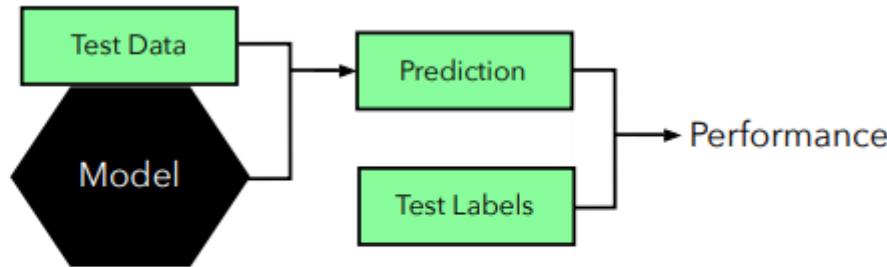


3

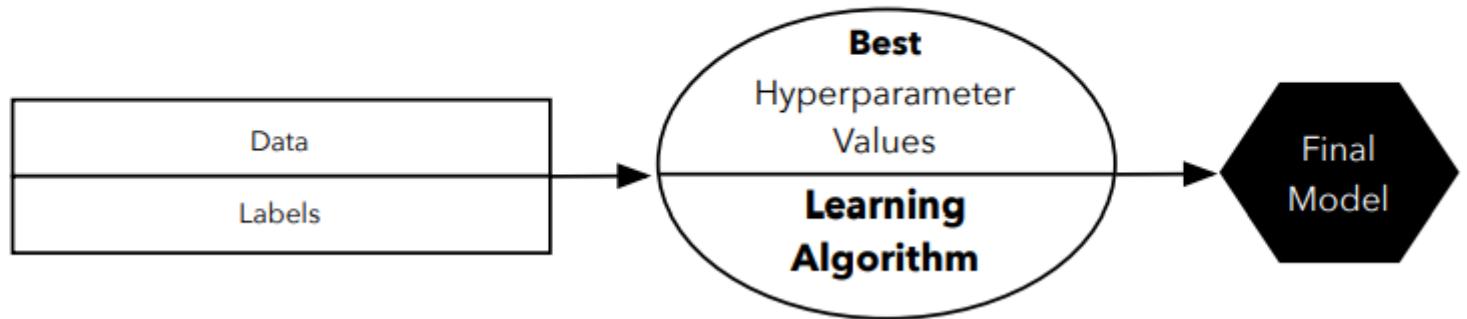


CV for Model Selection (2)

4



5



**END
of
Session 2**



Machine Learning

ZG565



BITS Pilani
Pilani Campus

Dr. Sugata Ghosal, PhD
Sugata.ghosal@pilani.bits-pilani.ac.in



Linear Regression

Session Content

- What is Linear regression
- Approaches to linear regression
- Least Squares Based Solution
- Direct Solution

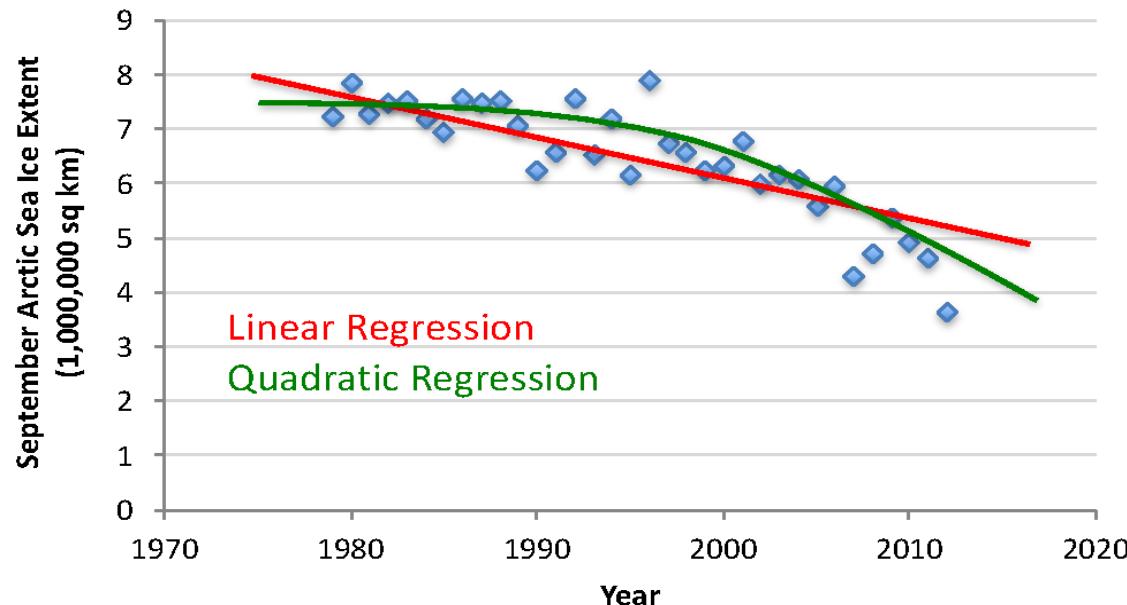
Regression

- Wish to learn a function $f :: X \rightarrow Y$, where predicted output Y is real, given the n real training instances $\{<x^1,y^1> \dots <x^n,y^n>\}$.
- Examples include
 - predict weight from gender, height, age, ...
 - Predict house price from locality, area, income, ...
 - predict Google stock price today from Google, Yahoo, MSFT prices yesterday
 - predict each pixel intensity in robot's current camera image, from previous image and previous action

Least Squares Approach

Given:

- Data $\mathbf{X} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}\}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^d$
- Corresponding labels $\mathbf{y} = \{y^{(1)}, \dots, y^{(n)}\}$ where $y^{(i)} \in \mathbb{R}$



Data from G. Witt. Journal of Statistics Education, Volume 21, Number 1 (2013)

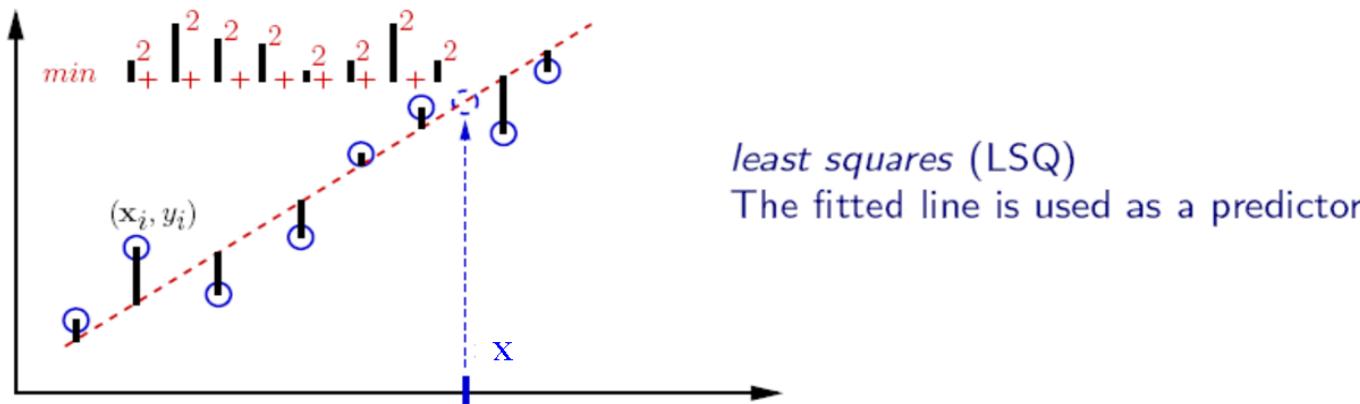
Linear Regression

- Hypothesis:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_d x_d = \sum_{j=0}^d \theta_j x_j = h_{\boldsymbol{\theta}}(\mathbf{x})$$

Assume $x_0 = 1$

- Fit model by minimizing sum of squared errors



Figures are courtesy of Greg Shakhnarovich

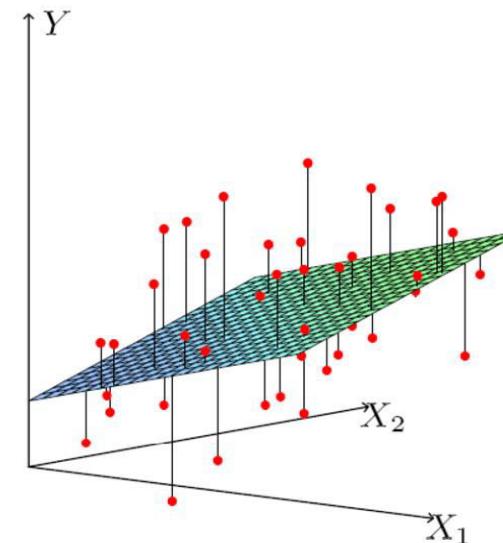
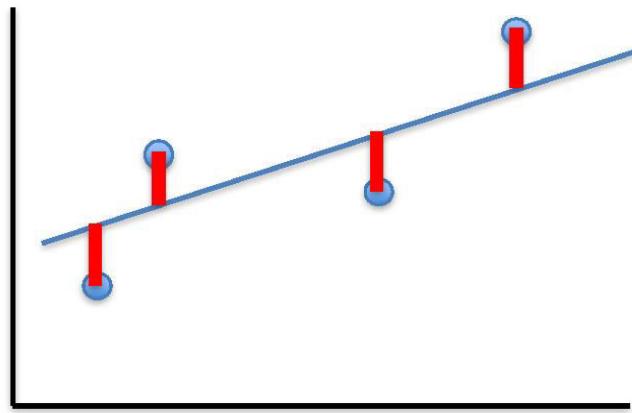
3

Least Squares Linear Regression

- Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$$

- Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$



Least Squares Based Solution

- Benefits of vectorization
 - More compact equations
 - Faster code (using optimized matrix libraries)

- Consider our model:

$$h(\mathbf{x}) = \sum_{j=0}^d \theta_j x_j$$

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \mathbf{x}^\top = [1 \quad x_1 \quad \dots \quad x_d]$$

- Can write the model in vectorized form as $h(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x}$

Least Squares Based Solution

- Consider our model for n instances:

$$h(\mathbf{x}^{(i)}) = \sum_{j=0}^d \theta_j x_j^{(i)}$$

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix}$$

$$\mathbb{R}^{(d+1) \times 1} \qquad \qquad \qquad \mathbb{R}^{n \times (d+1)}$$

- Can write the model in vectorized form as $h_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{X}\boldsymbol{\theta}$

Least Squares Based Solution

- For the linear regression cost function:

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$$

$$= \frac{1}{2n} \sum_{i=1}^n \left(\boldsymbol{\theta}^\top \mathbf{x}^{(i)} - y^{(i)} \right)^2$$

$$= \frac{1}{2n} \underbrace{(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^\top}_{\mathbb{R}^{1 \times n}} \underbrace{(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})}_{\mathbb{R}^{n \times 1}}$$

$\mathbb{R}^{n \times (d+1)}$
 $\mathbb{R}^{(d+1) \times 1}$

Let:

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

Least Squares Based Solution

- Solve for optimal θ analytically
 - Notice that the solution is when $\frac{\partial}{\partial \theta} J(\theta) = 0$

- Derivation:

$$\begin{aligned}
 J(\theta) &= \frac{1}{2n} (\mathbf{X}\theta - \mathbf{y})^\top (\mathbf{X}\theta - \mathbf{y}) \\
 &\propto \theta^\top \mathbf{X}^\top \mathbf{X} \theta - \boxed{\mathbf{y}^\top \mathbf{X} \theta} - \boxed{\theta^\top \mathbf{X}^\top \mathbf{y}} + \mathbf{y}^\top \mathbf{y} \\
 &\propto \theta^\top \mathbf{X}^\top \mathbf{X} \theta - 2\theta^\top \mathbf{X}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y}
 \end{aligned}$$

1 x 1

Take derivative and set equal to 0, then solve for θ :

$$\frac{\partial}{\partial \theta} (\theta^\top \mathbf{X}^\top \mathbf{X} \theta - 2\theta^\top \mathbf{X}^\top \mathbf{y} + \cancel{\mathbf{y}^\top \mathbf{y}}) = 0$$

$$(\mathbf{X}^\top \mathbf{X})\theta - \mathbf{X}^\top \mathbf{y} = 0$$

$$(\mathbf{X}^\top \mathbf{X})\theta = \mathbf{X}^\top \mathbf{y}$$

Closed Form Solution:

$$\theta = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

Least Squares Based Solution

- Can obtain θ by simply plugging X and y into

$$\theta = (X^T X)^{-1} X^T y$$

$$X = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \quad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

- If $X^T X$ is not invertible (i.e., singular), may need to:
 - Use pseudo-inverse instead of the inverse
 - In python, `numpy.linalg.pinv(a)`
 - Remove redundant (not linearly independent) features
 - Remove extra features to ensure that $d \leq n$

Intuition Behind Cost Function

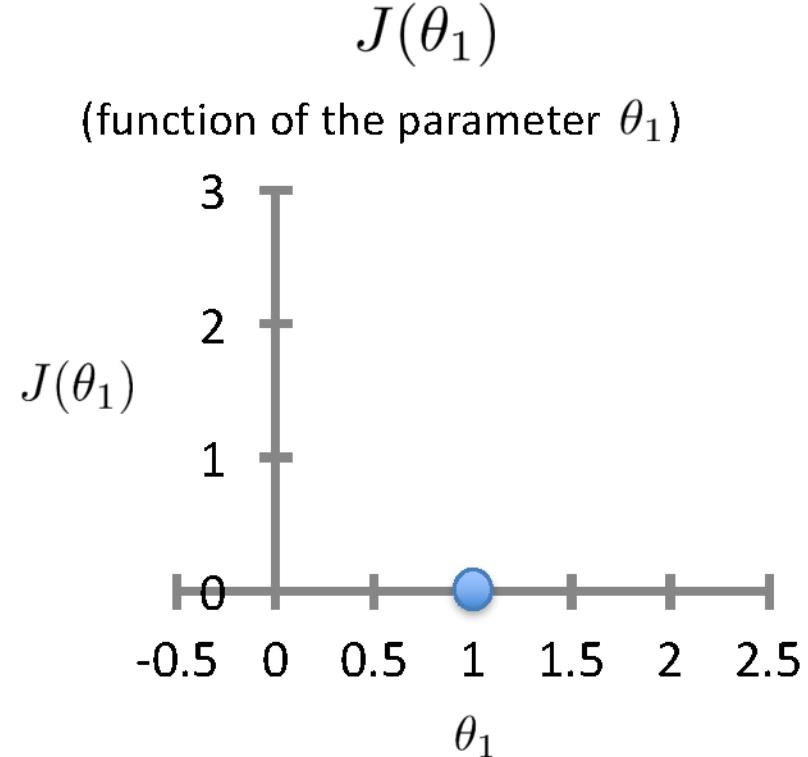
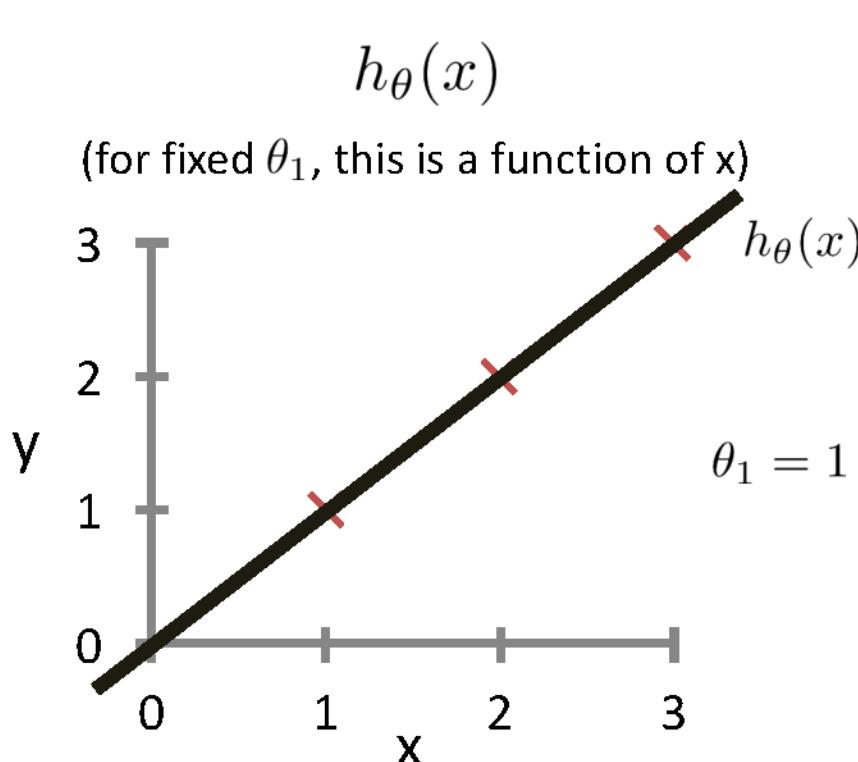
$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$$

For insight on $J()$, let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

Intuition Behind Cost Function

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2$$

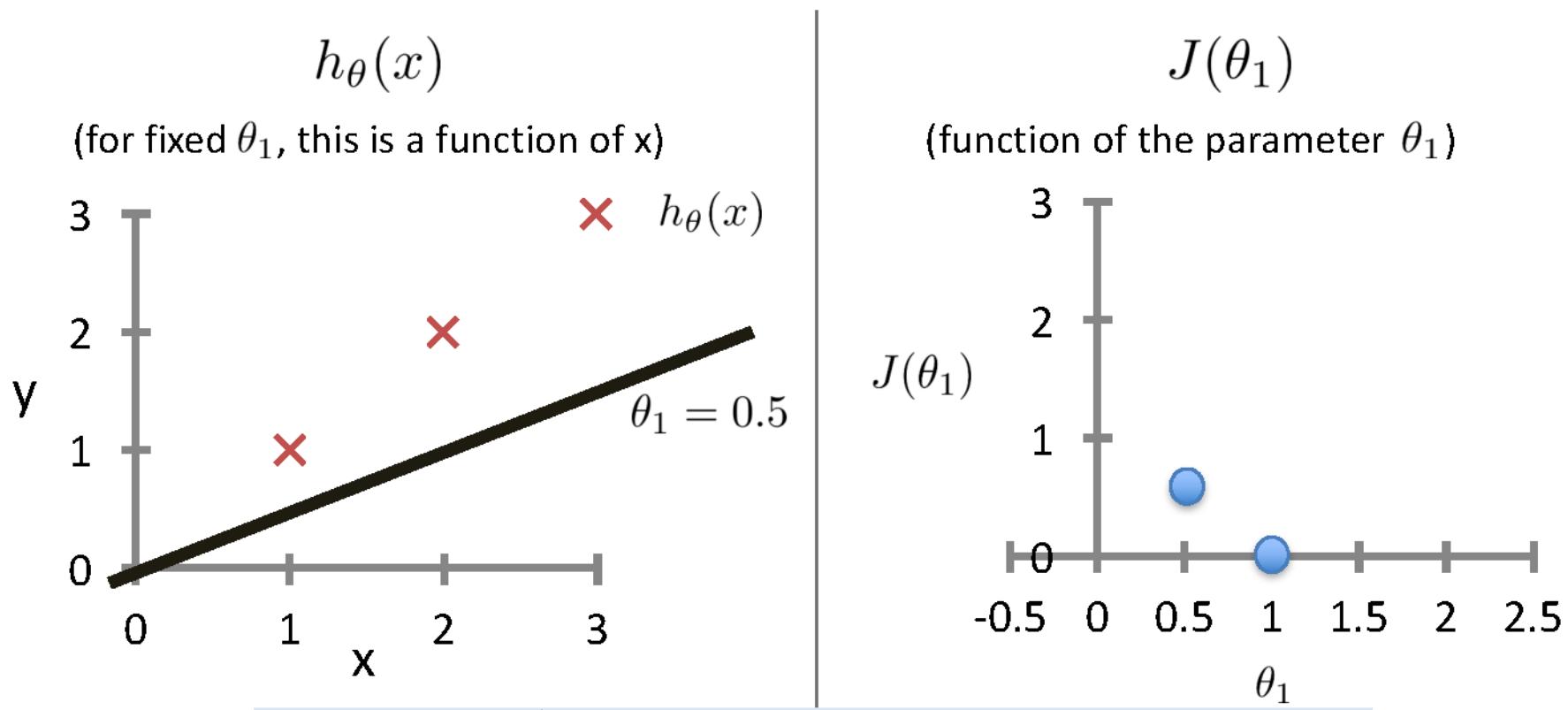
For insight on $J()$, let's assume $x \in \mathbb{R}$ so $\theta = [\theta_0, \theta_1]$



Intuition Behind Cost Function

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2$$

For insight on $J()$, let's assume $x \in \mathbb{R}$ so $\theta = [\theta_0, \theta_1]$



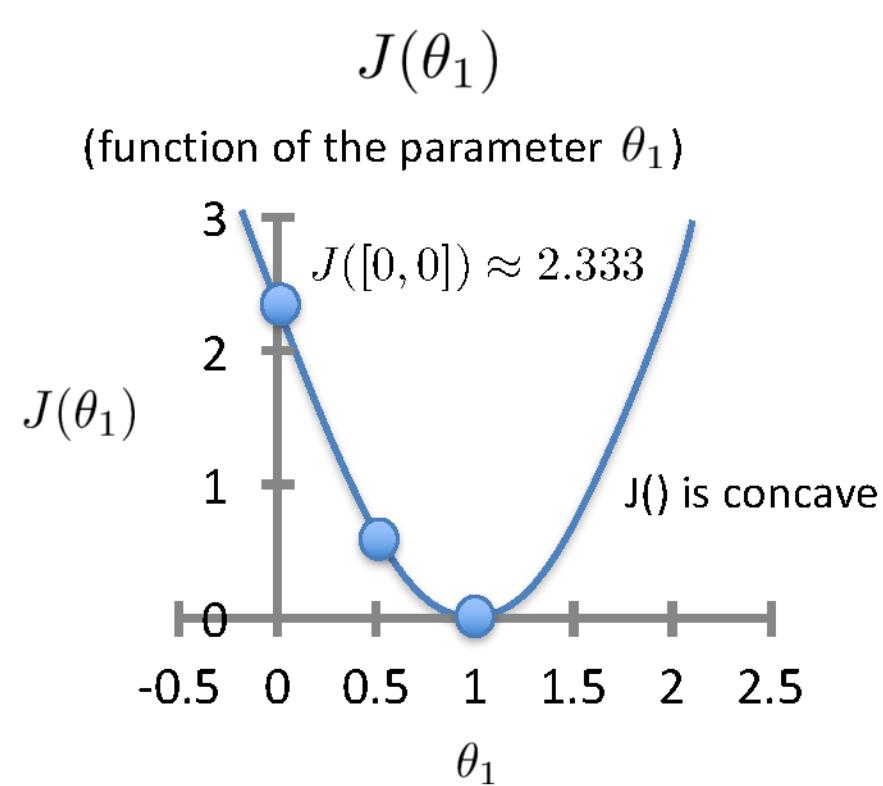
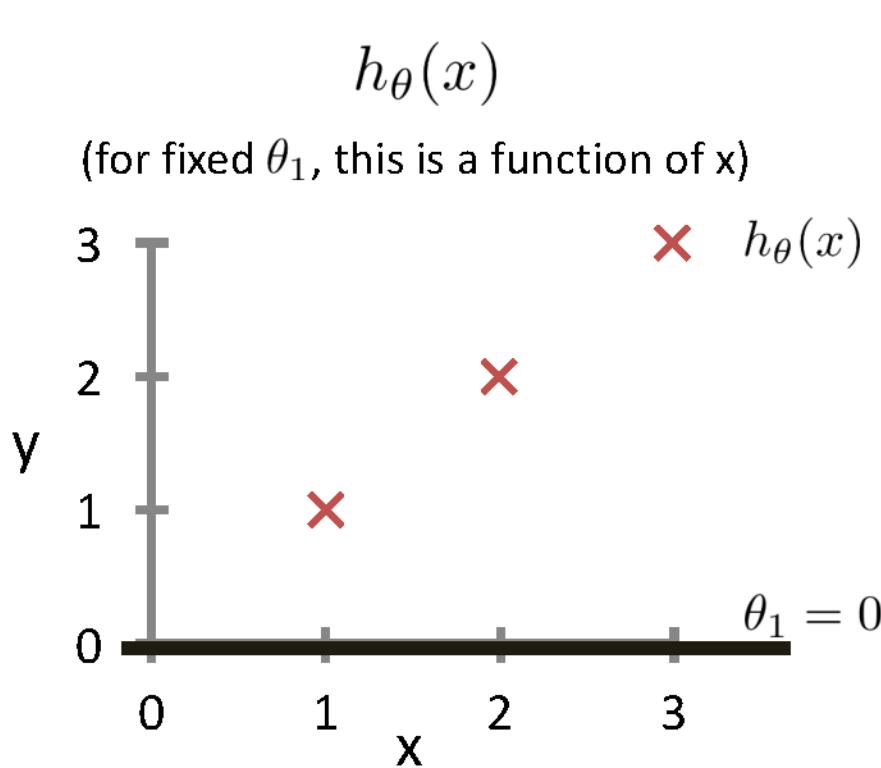
Based on example
by Andrew Ng

$$J([0, 0.5]) = \frac{1}{2 \times 3} [(0.5 - 1)^2 + (1 - 2)^2 + (1.5 - 3)^2] \approx 0.58$$

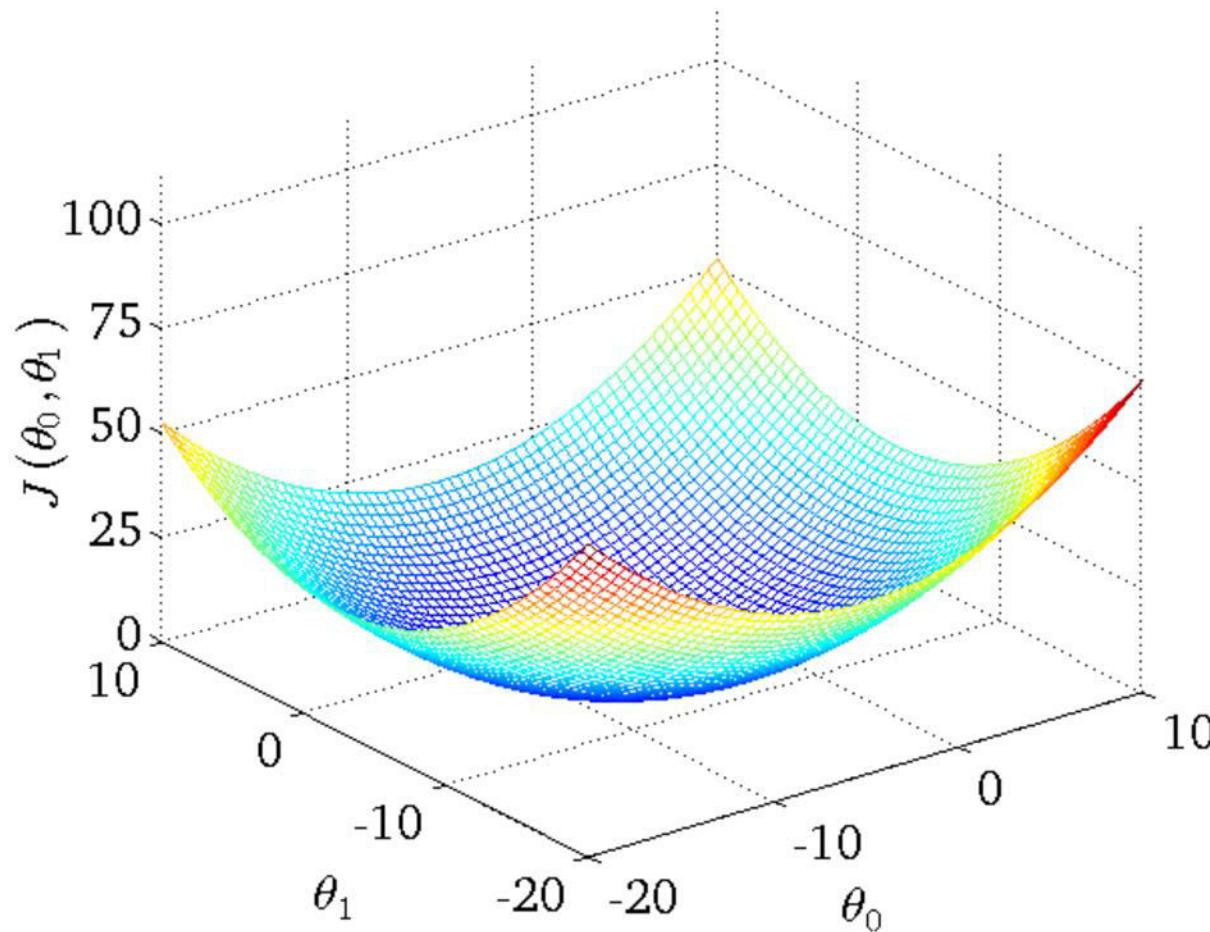
Intuition Behind Cost Function

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2$$

For insight on $J()$, let's assume $x \in \mathbb{R}$ so $\theta = [\theta_0, \theta_1]$



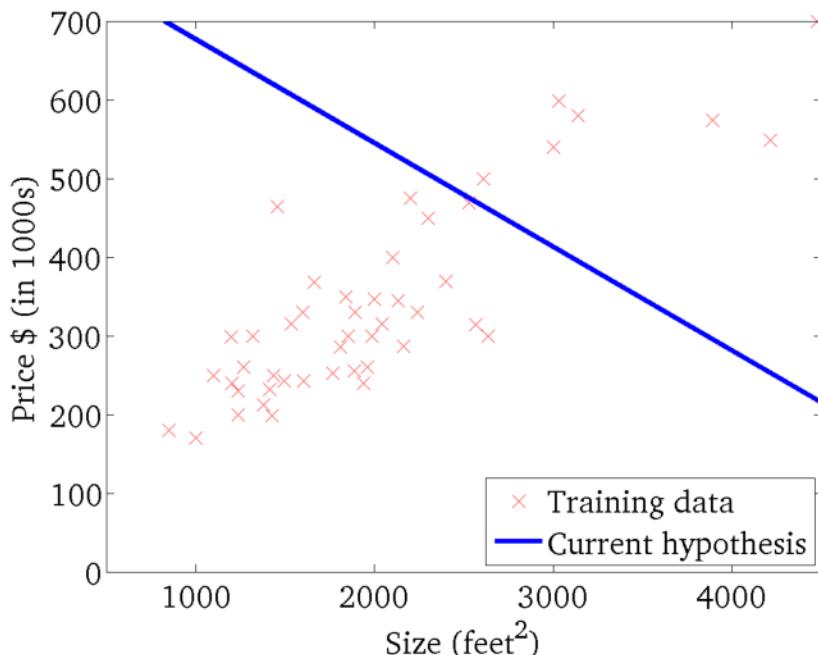
Intuition Behind Cost Function



Intuition Behind Cost Function

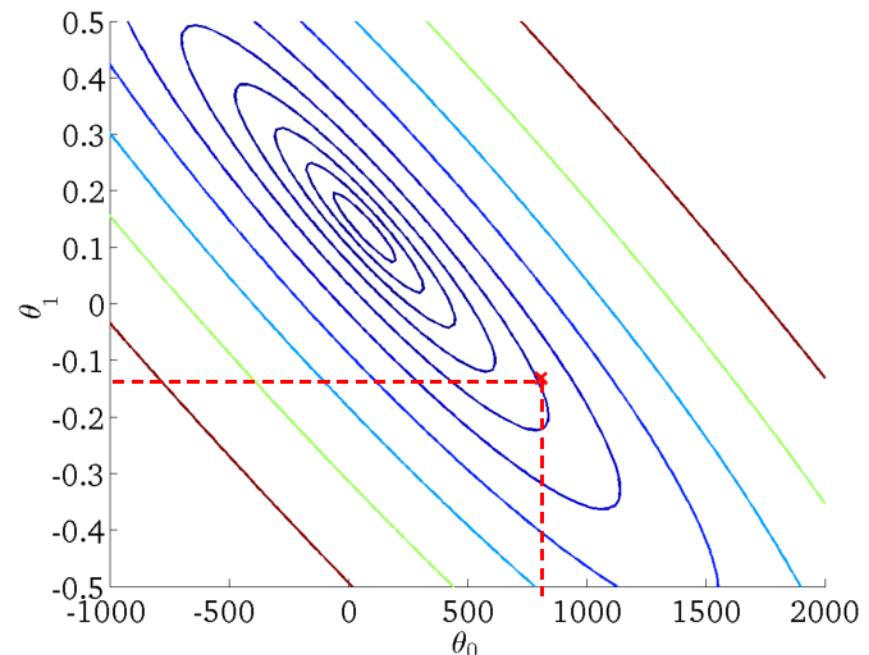
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



$$J(\theta_0, \theta_1)$$

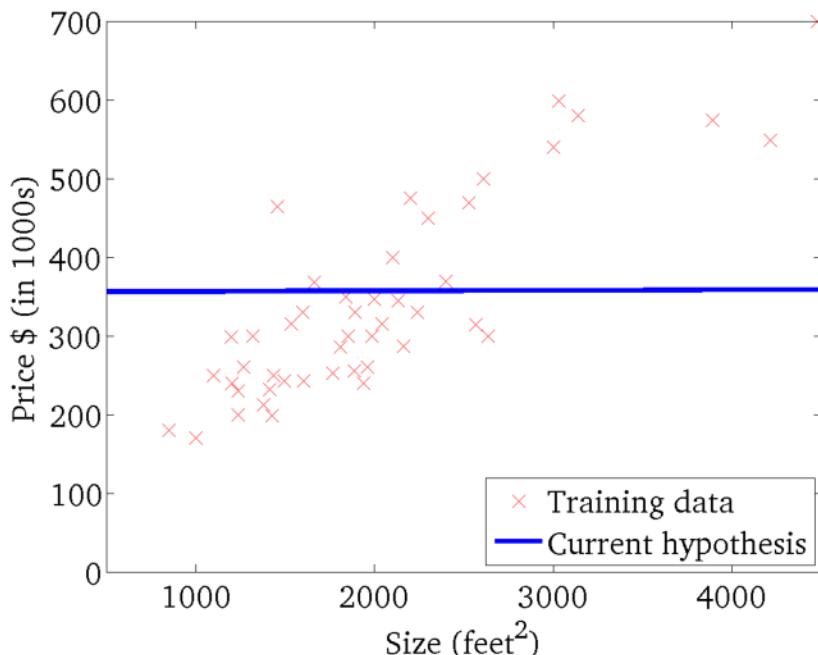
(function of the parameters θ_0, θ_1)



Intuition Behind Cost Function

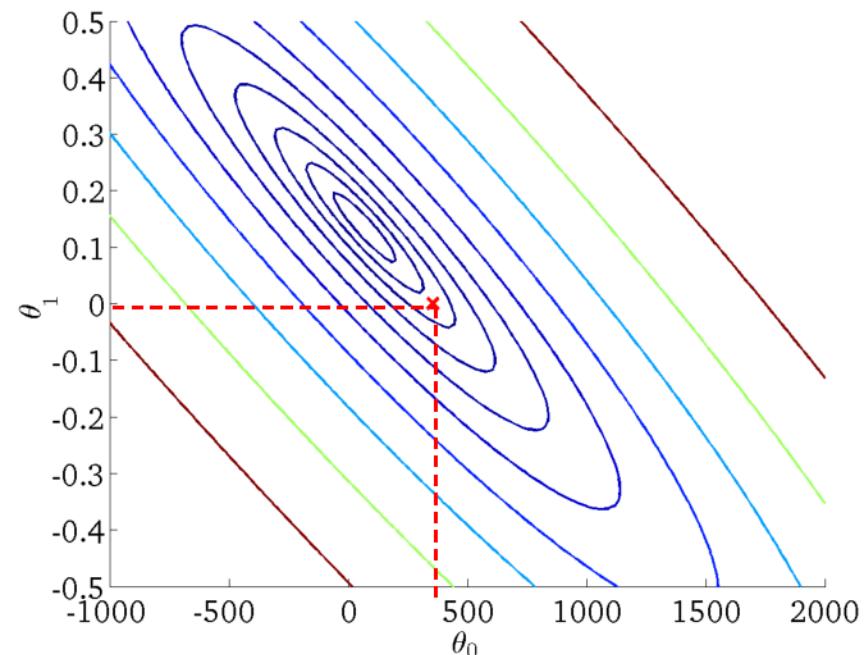
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



$$J(\theta_0, \theta_1)$$

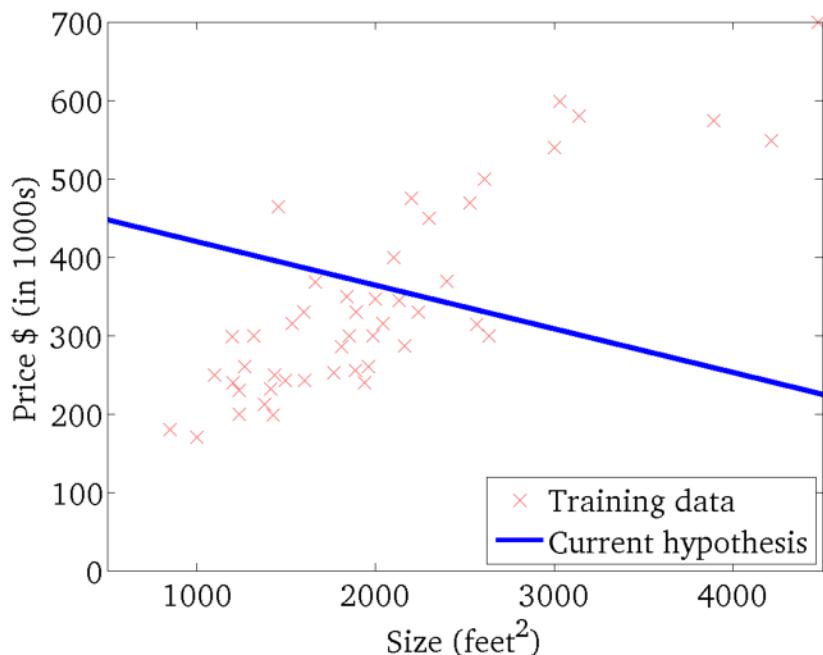
(function of the parameters θ_0, θ_1)



Intuition Behind Cost Function

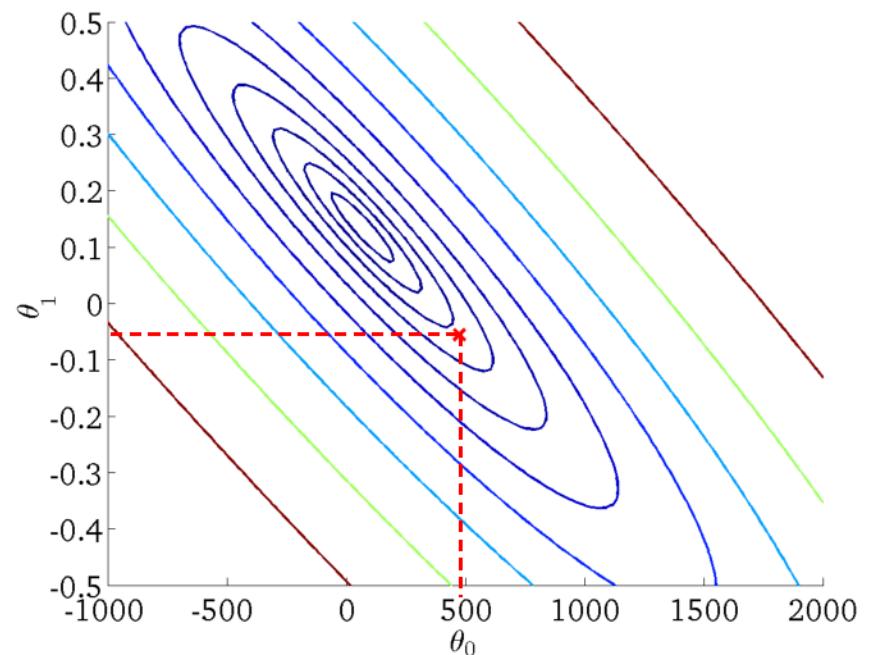
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



$$J(\theta_0, \theta_1)$$

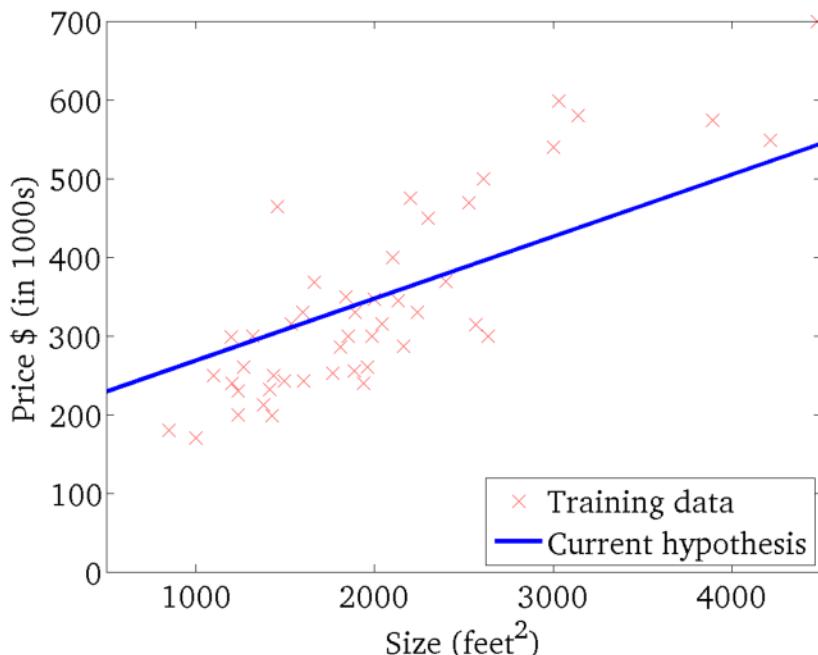
(function of the parameters θ_0, θ_1)



Intuition Behind Cost Function

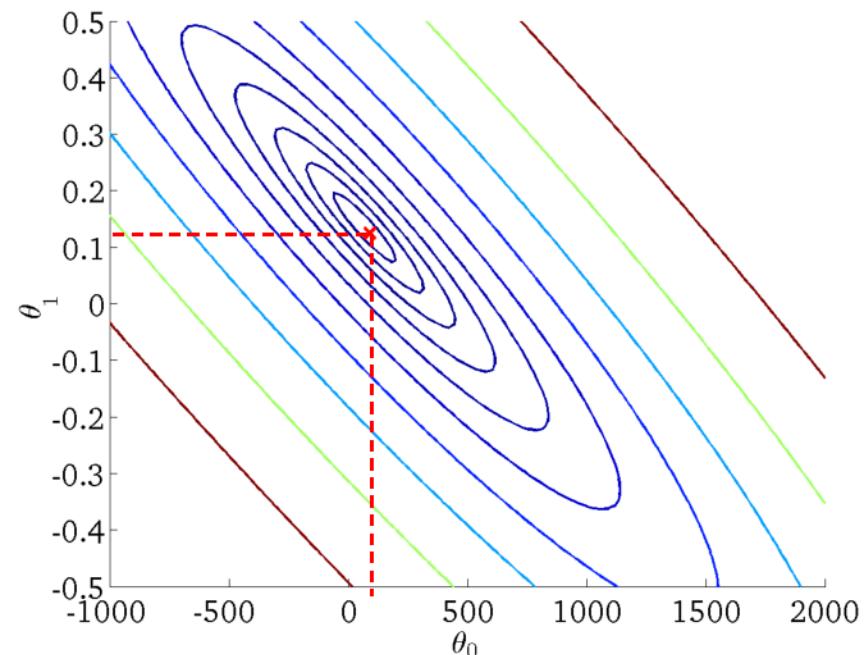
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



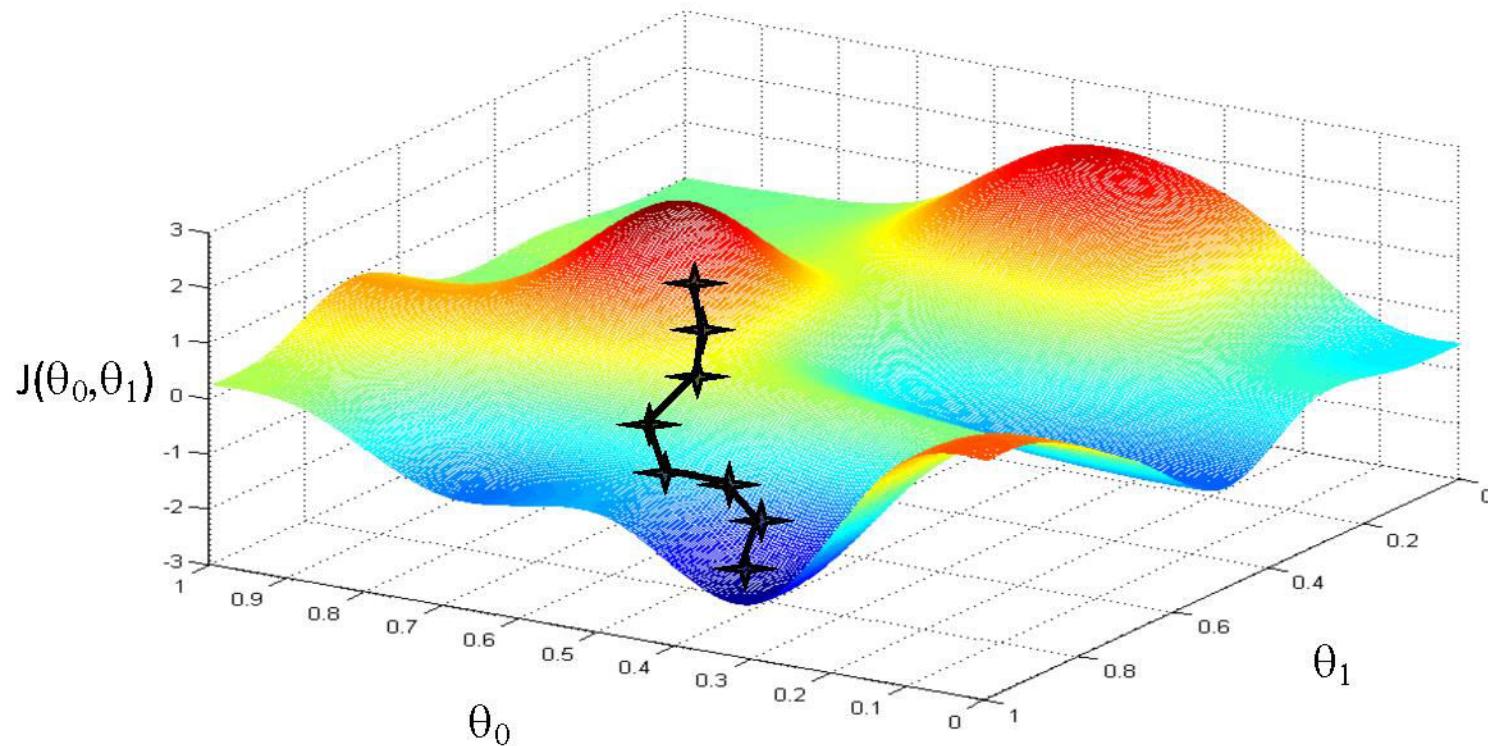
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



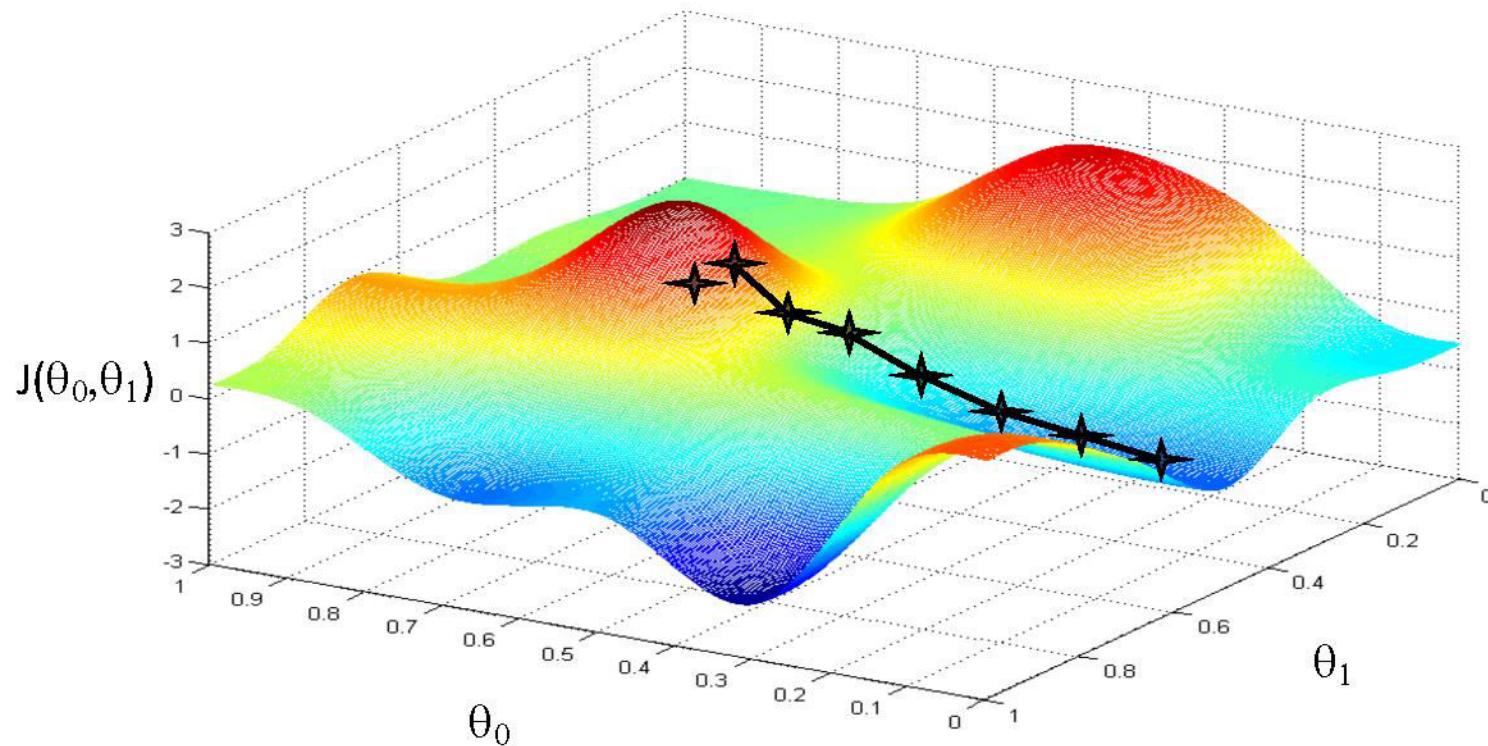
Basic Search Procedure

- Choose initial value for θ
- Until we reach a minimum:
 - Choose a new value for θ to reduce $J(\theta)$



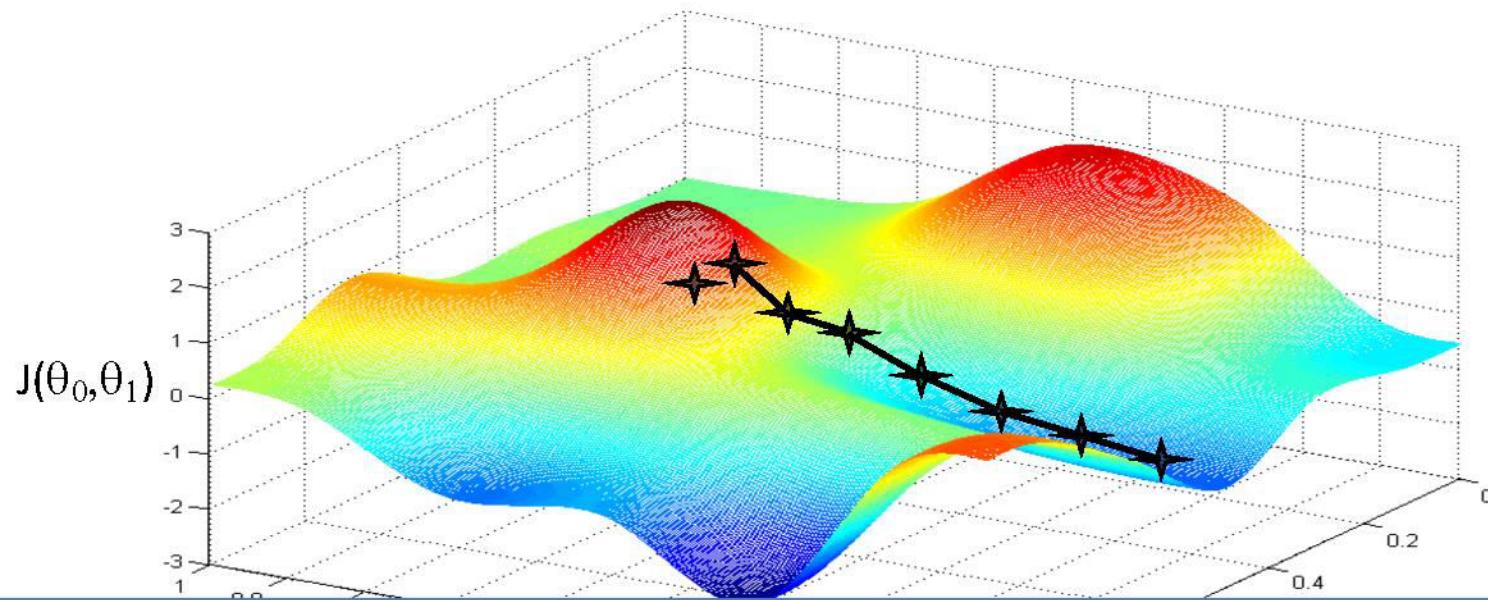
Basic Search Procedure

- Choose initial value for θ
- Until we reach a minimum:
 - Choose a new value for θ to reduce $J(\theta)$



Basic Search Procedure

- Choose initial value for θ
- Until we reach a minimum:
 - Choose a new value for θ to reduce $J(\theta)$



Since the least squares objective function is convex (concave),
we don't need to worry about local minima

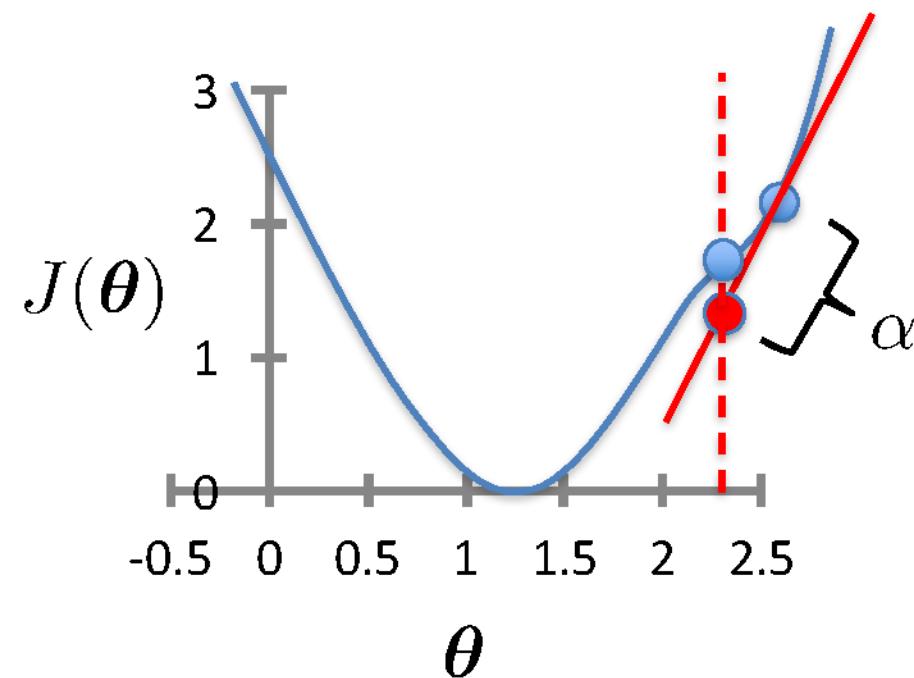
Gradient Descent

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

simultaneous update
for $j = 0 \dots d$

learning rate (small)
e.g., $\alpha = 0.05$



Gradient Descent

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

simultaneous update
for $j = 0 \dots d$

For Linear Regression:

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta} (\mathbf{x}^{(i)}) - y^{(i)} \right)^2$$

Gradient Descent

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

simultaneous update
for $j = 0 \dots d$

For Linear Regression:

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta} (\mathbf{x}^{(i)}) - y^{(i)} \right)^2 \\ &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right)^2\end{aligned}$$

Gradient Descent

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

simultaneous update
for $j = 0 \dots d$

For Linear Regression:

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 \\ &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^n \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) \times \frac{\partial}{\partial \theta_j} \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right)\end{aligned}$$

Gradient Descent

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

simultaneous update
for $j = 0 \dots d$

For Linear Regression:

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 \\ &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^n \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) \times \frac{\partial}{\partial \theta_j} \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) \\ &= \frac{1}{n} \sum_{i=1}^n \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) x_j^{(i)}\end{aligned}$$

Gradient Descent for Linear Regression

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

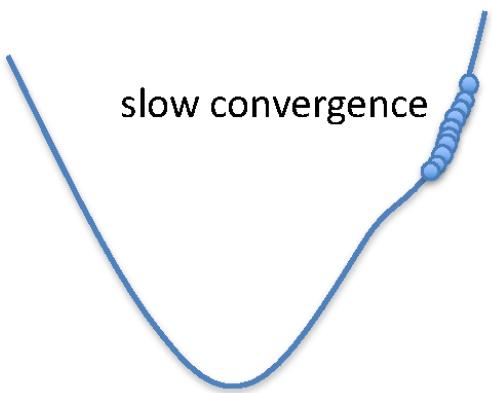
simultaneous
update
for $j = 0 \dots d$

- To achieve simultaneous update
 - At the start of each GD iteration, compute $h_{\theta}(\mathbf{x}^{(i)})$
 - Use this stored value in the update step loop
- Assume convergence when $\|\theta_{new} - \theta_{old}\|_2 < \epsilon$

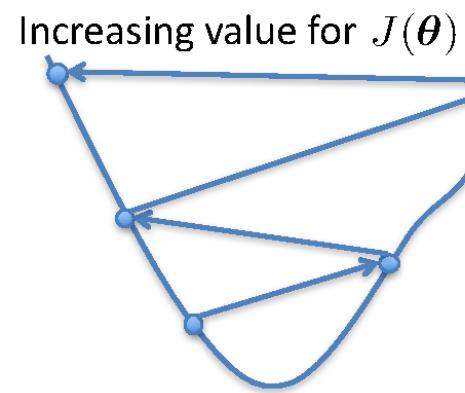
L₂ norm: $\|\mathbf{v}\|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \dots + v_{|v|}^2}$

Choosing Step Size (*learning rate*)

α too small



α too large

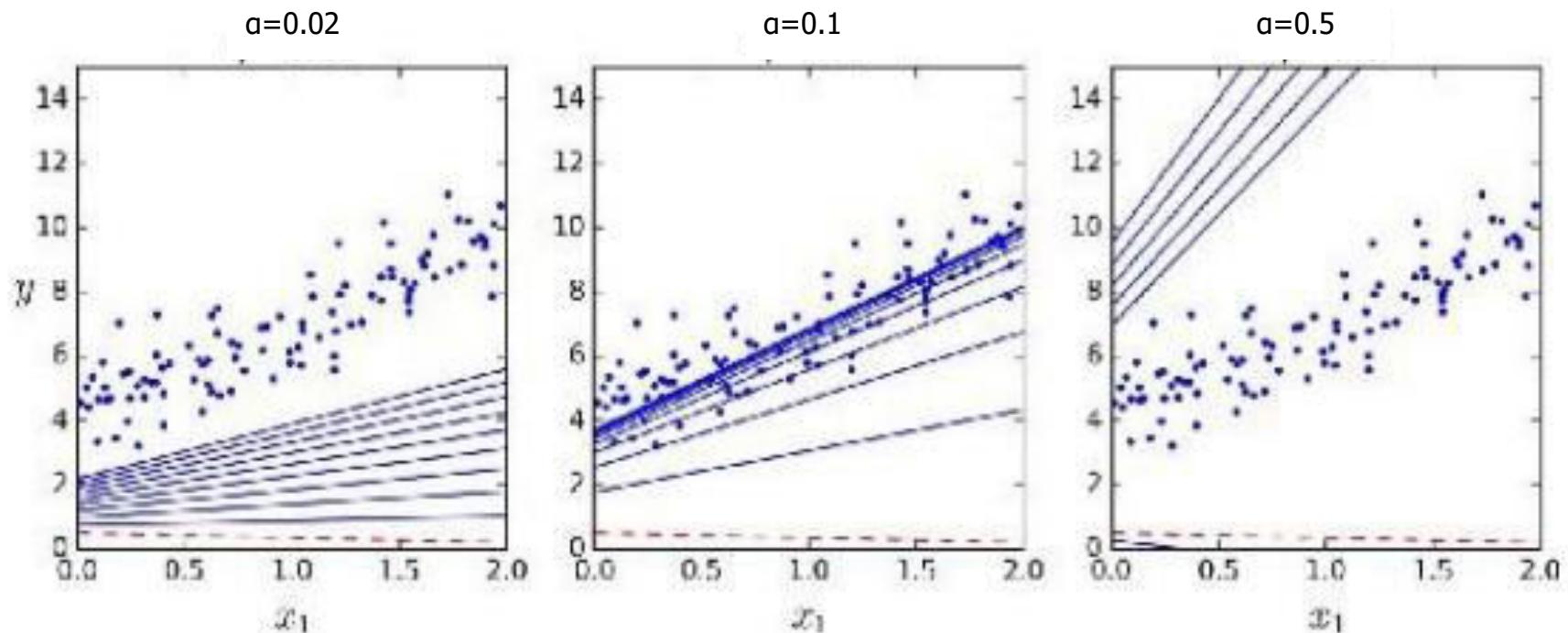


- May overshoot the minimum
- May fail to converge
- May even diverge

To see if gradient descent is working, print out $J(\theta)$ each iteration

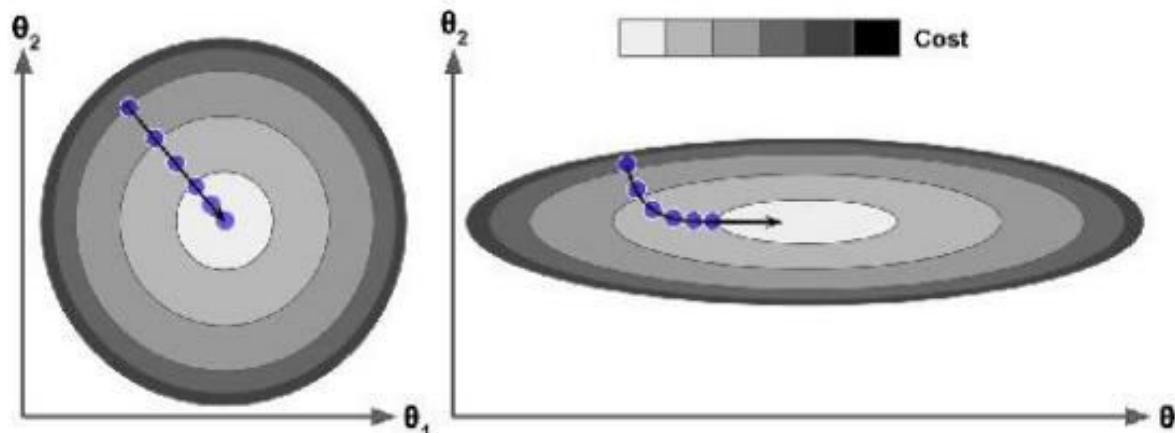
- The value should decrease at each iteration
- If it doesn't, adjust α

Impact of Learning Rate



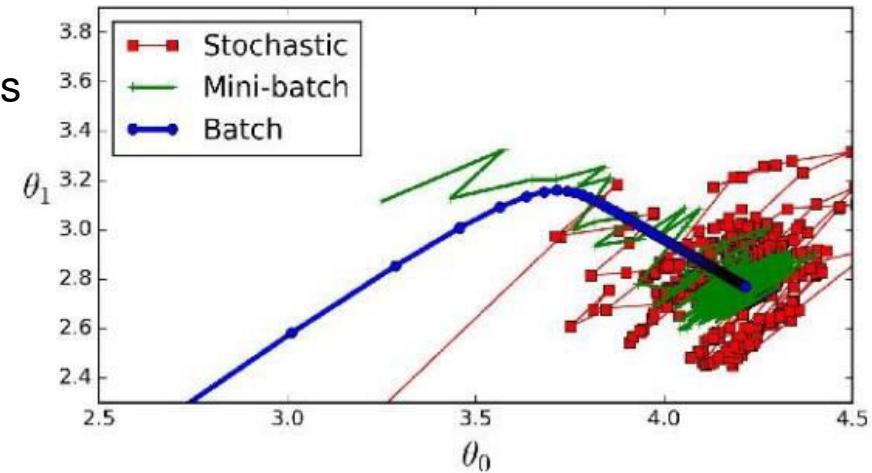
Feature Normalization

- Gradient Descent can be slow when feature scales are widely different
- Normalization of feature values for same variance needed for faster convergence



Impact Due To Batch Size

- Choice of batch size impacts the rate of convergence of gradient descent (GD)
- In Batch GD, entire training set is used to calculate the training error in each iteration/epoch and gradient is calculated and used for weight updates
 - Converges in least number of iterations, i.e., rate of convergence is highest [$O(1/\text{iterations})$]
 - Computation requirement per iteration is highest
 - Memory requirement is also highest
- In Stochastic gradient descent, a **randomly** selected training instance is used
 - Converges in highest number of iterations, i.e., rate of convergence is slowest [$\sim O(1/\sqrt{\text{iterations}})$]
 - Computation requirement per iteration is lowest
 - Memory requirement is also lowest
- In mini batch GD, a subset of training data of size, say 64, 128, 256 is used
 - very efficient implementation possible



Closed Form Solution Vs. Gradient Descent



- | Gradient Descent | Closed Form Solution |
|--|---|
| <ul style="list-style-type: none">• Requires multiple iterations• Need to choose α• Works well when n is large• Can support incremental learning | <ul style="list-style-type: none">• Non-iterative• No need for α• Slow if n is large<ul style="list-style-type: none">– Computing $(X^T X)^{-1}$ is roughly $O(n^3)$ |

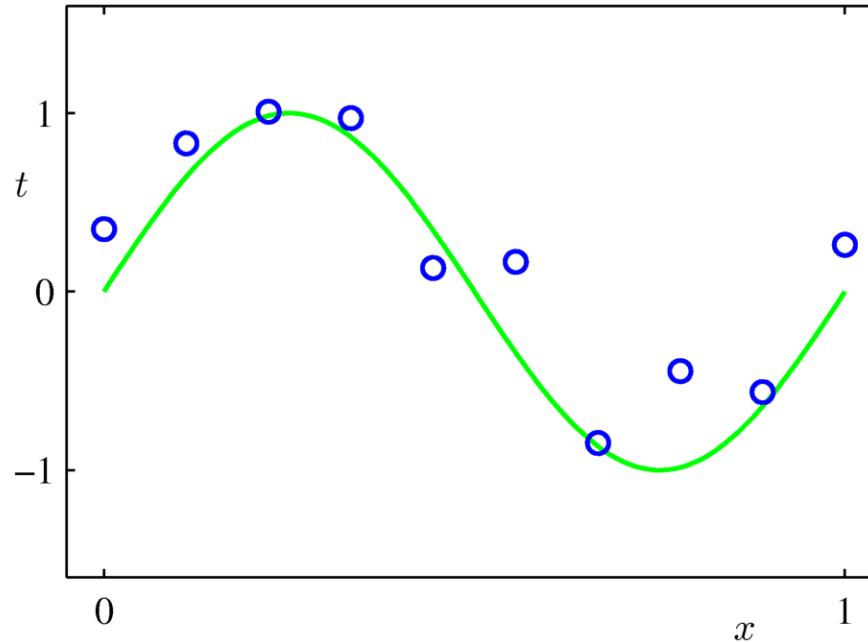
Closed Form Solution

Extending to More Complex Model

- The inputs **X** for linear regression can be:
 - Original quantitative inputs
 - Transformation of quantitative inputs
 - e.g. log, exp, square root, square, etc.
 - Polynomial transformation
 - example: $y = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2 + \beta_3 \cdot x^3$
 - Basis expansions
 - Dummy coding of categorical inputs
 - Interactions between variables
 - example: $x_3 = x_1 \cdot x_2$

This allows use of linear regression techniques
to fit non-linear datasets.

Fitting a Polynomial Curve



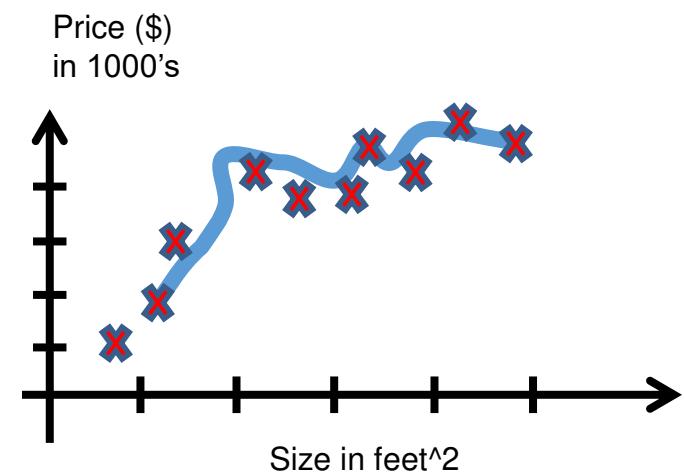
$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_p x^p = \sum_{j=0}^p \theta_j x^j = \sum_{j=0}^p \theta_j z_j$$
$$z_j = x^j$$

Segment Content

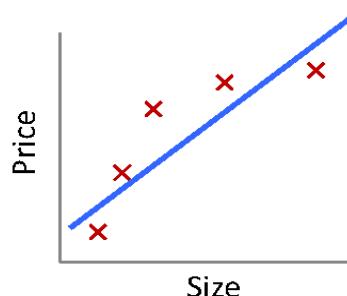
- Overfitting of the training data
- Effect of Training data size on overfitting
- Regularization
 - Ridge
 - Lasso
 - Early Stopping

Addressing overfitting

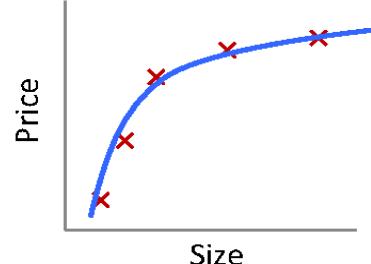
- x_1 = size of house
- x_2 = no. of bedrooms
- x_3 = no. of floors
- x_4 = age of house
- x_5 = average income in neighborhood
- x_6 = kitchen size
- :
- x_{100}



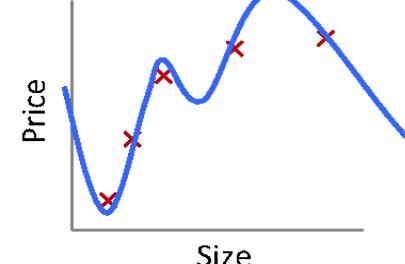
Quality of Fit



Underfitting
(high bias)



Correct fit



Overfitting
(high variance)

Overfitting:

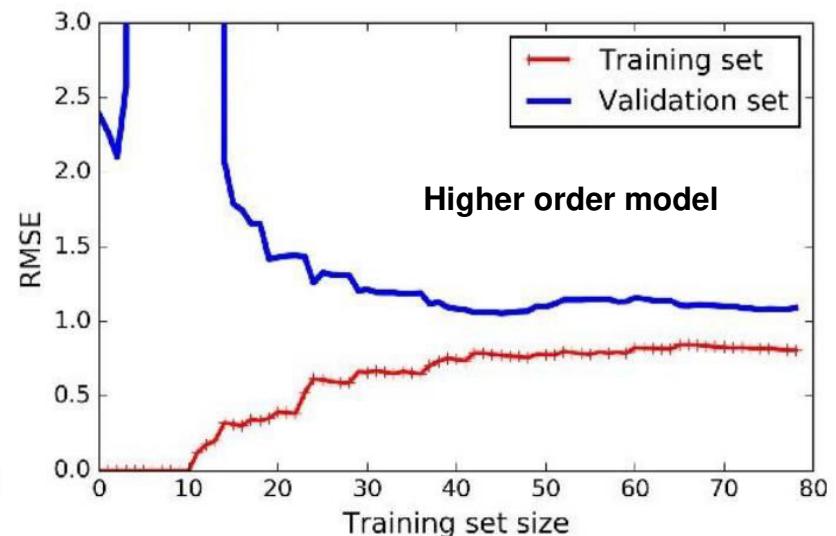
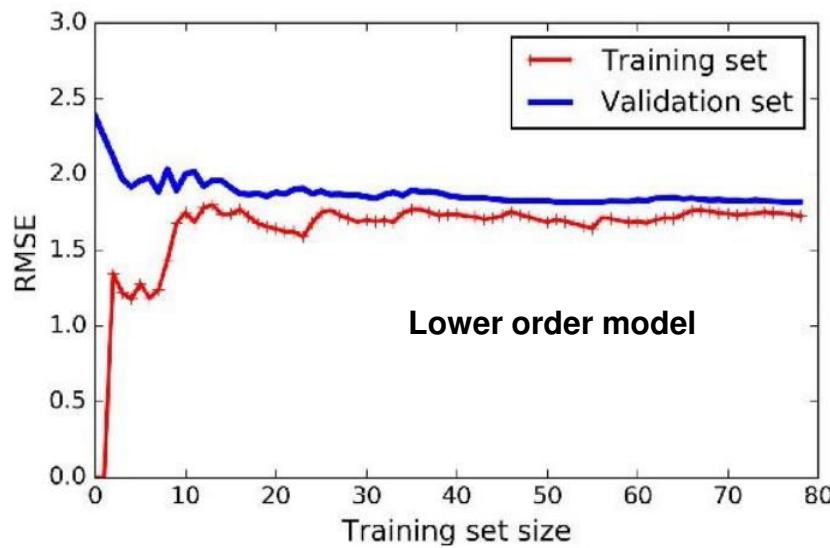
- The learned hypothesis may fit the training set very well ($J(\theta) \approx 0$)
- ...but fails to generalize to new examples

Addressing overfitting

- Reduce number of features.
 - Manually select which features to keep.
 - Model selection algorithm
- Regularization.
 - Keep all the features, but reduce magnitude/values of parameters θ_j .
 - Works well when we have a lot of features, each/many of which contributes a bit to predicting y .

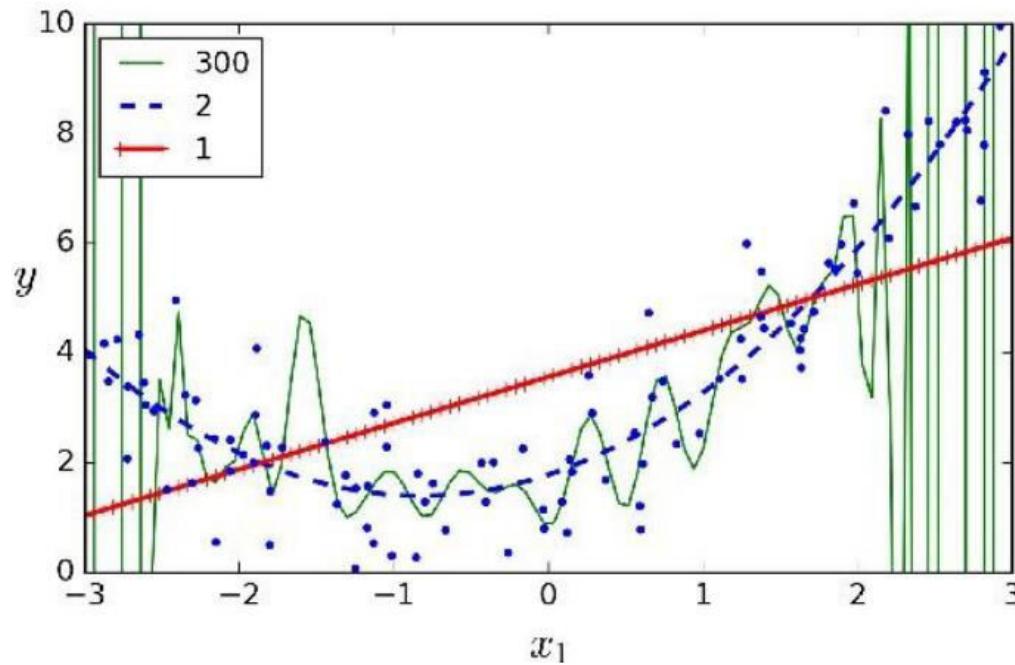
Effect of Training Size on Overfitting

- Size of training dataset needs to be large to prevent when higher order model is used.



Polynomial Fitting can lead to Overfitting

- Underlying target function is quadratic
- Linear model results in underfitting with large bias
- Polynomial of order 300 results in a large variance



Regularization

- A method for automatically controlling the complexity of the learned hypothesis
- **Idea:** penalize for large values of θ_j
 - Can incorporate into the cost function
 - Works well when we have a lot of features, each that contributes a bit to predicting the label
- Can also address overfitting by eliminating features (either manually or via model selection)

Ridge Regularization

- Linear regression objective function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$$

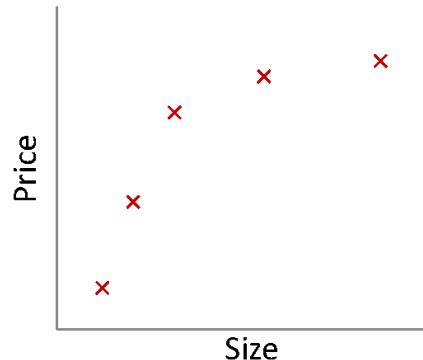

The diagram shows the objective function $J(\boldsymbol{\theta})$ as a sum of two terms. A blue bracket under the first term $\frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$ is labeled "model fit to data". A blue bracket under the second term $\frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$ is labeled "regularization".

- λ is the regularization parameter ($\lambda \geq 0$)
- No regularization on θ_0 !

Understanding Regularization

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$$

- What happens if we set λ to be huge (e.g., 10^{10})?



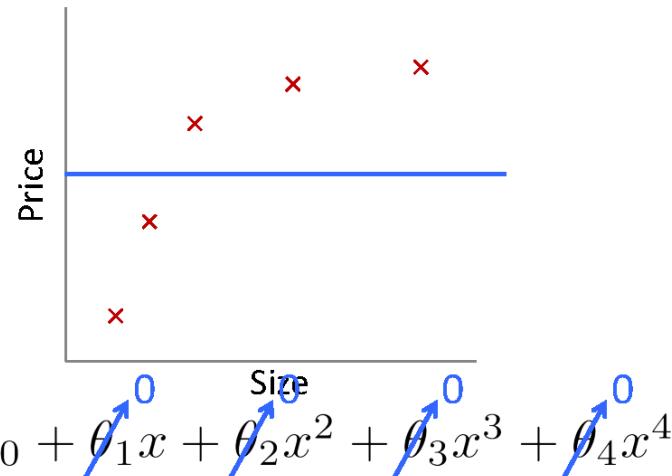
$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Based on example by Andrew Ng

Understanding Regularization

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$$

- What happens if we set λ to be huge (e.g., 10^{10})?



Based on example by Andrew Ng

Ridge regression

- Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$$

- Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- Gradient update:

$$\begin{aligned} \frac{\partial}{\partial \theta_0} J(\boldsymbol{\theta}) & \quad \theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right) \\ \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) & \quad \theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} - \lambda \theta_j \end{aligned}$$

regularization

Ridge Regression

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$$

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)$$

$$\theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} - \lambda \theta_j$$

- We can rewrite the gradient step as:

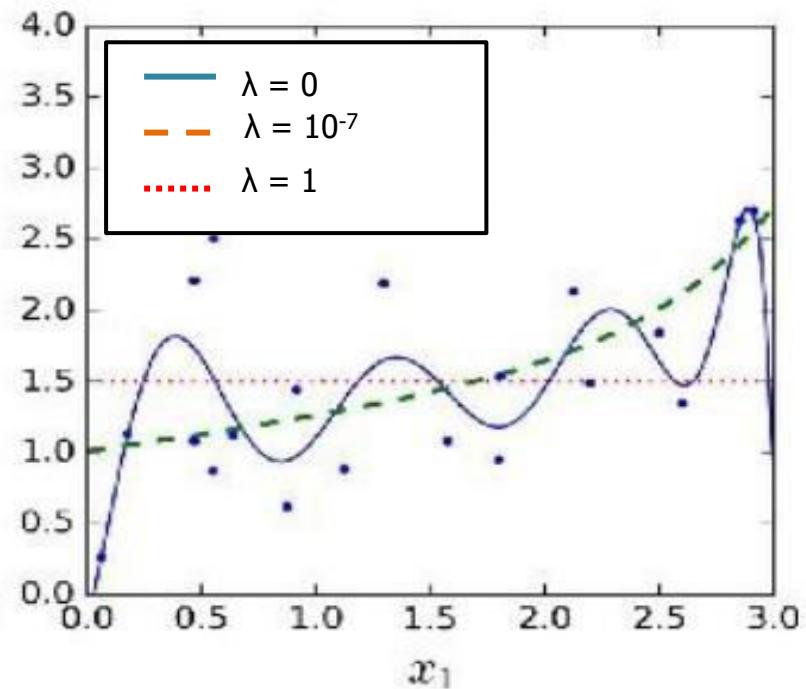
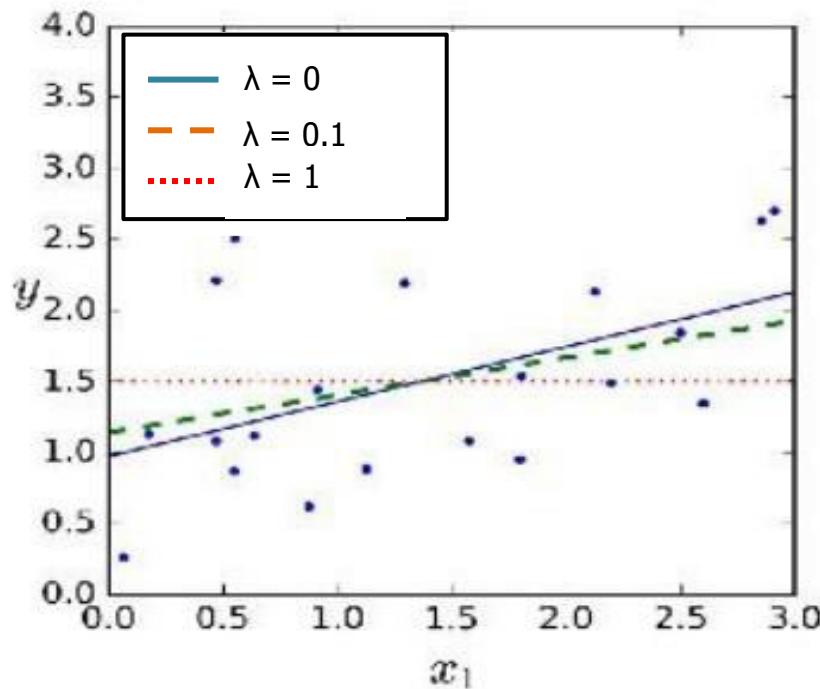
$$\theta_j \leftarrow \theta_j (1 - \alpha \lambda) - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

Lasso Regularization

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^d |\theta_j|$$

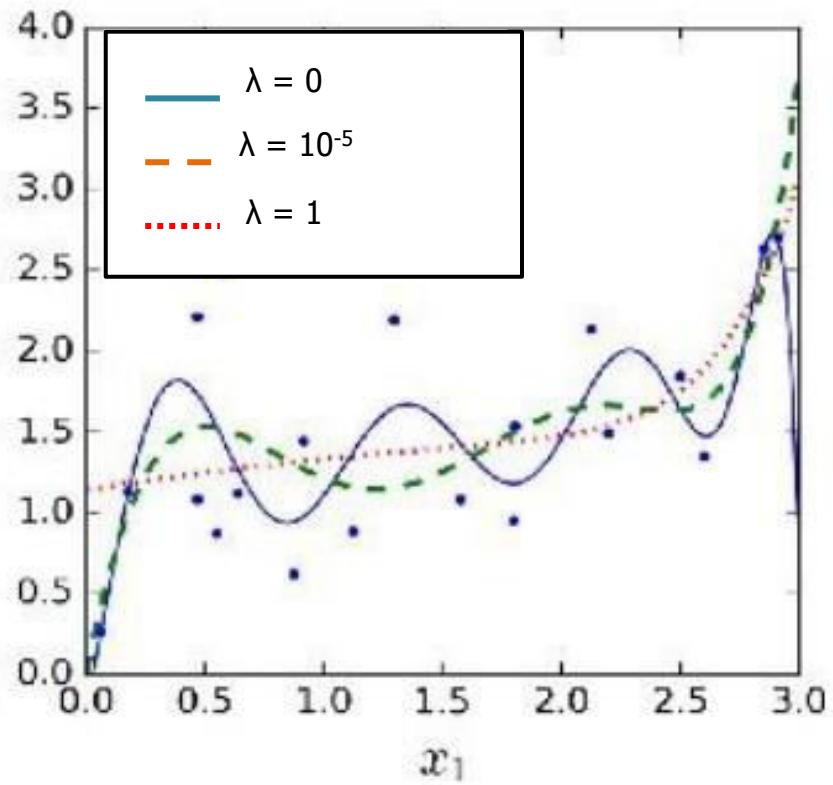
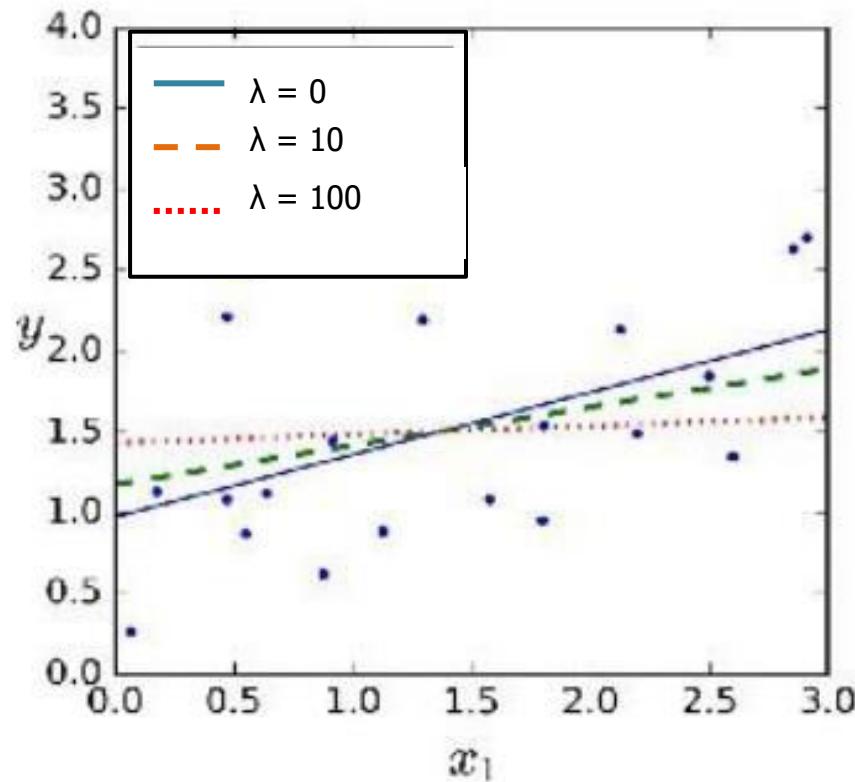
$$\theta_j = \theta_j - \frac{\alpha}{n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right) \mathbf{x}_j^{(i)} - \alpha \lambda \text{sign}(\theta_j)$$

Ridge Vs Lasso Regularization



Lasso

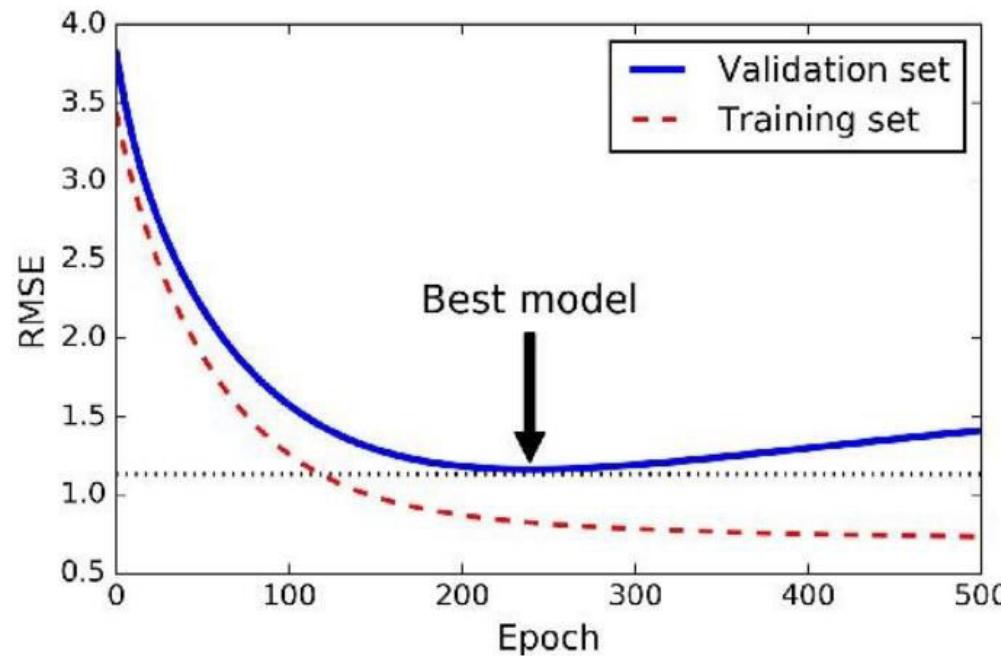
Ridge Vs Lasso Regularization



Ridge

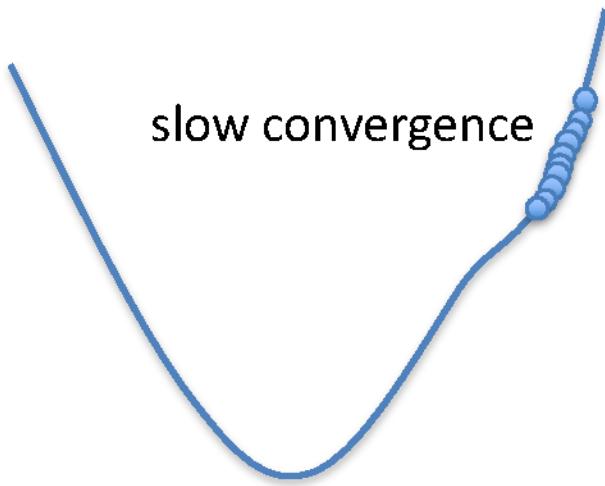
Early Stopping

- Do Not Over train to prevent overfitting
- Stop training once error on the validation set starts showing an upward trend, even if the error on the training set keeps decreasing

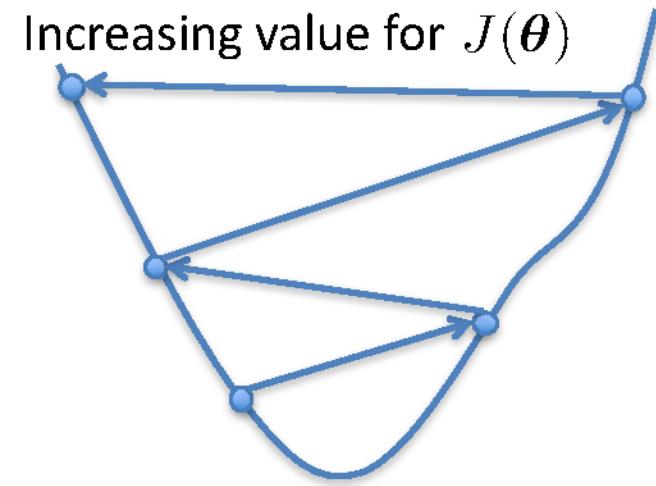


Choosing α

α too small



α too large



- May overshoot the minimum
- May fail to converge
- May even diverge

To see if gradient descent is working, print out $J(\theta)$ each iteration

- The value should decrease at each iteration
- If it doesn't, adjust α

Learning Model Parameters – Closed Form Solution (using vectorization)

Vectorization

- Benefits of vectorization
 - More compact equations
 - Faster code (using optimized matrix libraries)

- Consider our model:

$$h(\mathbf{x}) = \sum_{j=0}^d \theta_j x_j$$

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \mathbf{x}^\top = [1 \quad x_1 \quad \dots \quad x_d]$$

- Can write the model in vectorized form as $h(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$

Vectorization

- Consider our model for n instances:

$$h(\mathbf{x}^{(i)}) = \sum_{j=0}^d \theta_j x_j^{(i)}$$

- Let

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix}$$
$$\mathbb{R}^{(d+1) \times 1} \qquad \qquad \qquad \mathbb{R}^{n \times (d+1)}$$

- Can write the model in vectorized form as $h_{\theta}(\mathbf{x}) = \mathbf{X}\theta$

Vectorization

- For the linear regression cost function:

$$\begin{aligned} J(\boldsymbol{\theta}) &= \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 \\ &= \frac{1}{2n} \sum_{i=1}^n \left(\boldsymbol{\theta}^\top \mathbf{x}^{(i)} - y^{(i)} \right)^2 \\ &= \frac{1}{2n} \underbrace{(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^\top}_{\mathbb{R}^{1 \times n}} \underbrace{(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})}_{\mathbb{R}^{n \times 1}} \end{aligned}$$

$\mathbb{R}^{n \times (d+1)}$
 $\mathbb{R}^{(d+1) \times 1}$

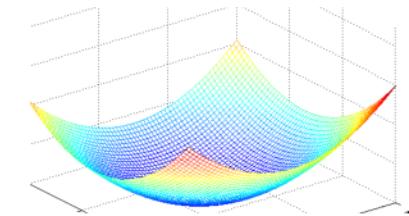
Let:

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

Closed Form Solution

- Instead of using GD, solve for optimal θ analytically
 - Notice that the solution is when $\frac{\partial}{\partial \theta} J(\theta) = 0$
- Derivation:

$$\begin{aligned}\mathcal{J}(\theta) &= \frac{1}{2n} (\mathbf{X}\theta - \mathbf{y})^\top (\mathbf{X}\theta - \mathbf{y}) \\ &\propto \theta^\top \mathbf{X}^\top \mathbf{X} \theta - \boxed{\mathbf{y}^\top \mathbf{X} \theta} - \boxed{\theta^\top \mathbf{X}^\top \mathbf{y}} + \mathbf{y}^\top \mathbf{y} \\ &\propto \theta^\top \mathbf{X}^\top \mathbf{X} \theta - 2\theta^\top \mathbf{X}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y}\end{aligned}$$



Take derivative and set equal to 0, then solve for θ :

$$\frac{\partial}{\partial \theta} (\theta^\top \mathbf{X}^\top \mathbf{X} \theta - 2\theta^\top \mathbf{X}^\top \mathbf{y} + \cancel{\mathbf{y}^\top \mathbf{y}}) = 0$$

$$(\mathbf{X}^\top \mathbf{X})\theta - \mathbf{X}^\top \mathbf{y} = 0$$

$$(\mathbf{X}^\top \mathbf{X})\theta = \mathbf{X}^\top \mathbf{y}$$

Closed Form Solution:

$$\theta = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

Closed Form Solution

- Can obtain θ by simply plugging X and y into

$$\theta = (X^T X)^{-1} X^T y$$

$$X = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \quad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

- If $X^T X$ is not invertible (i.e., singular), may need to:
 - Use pseudo-inverse instead of the inverse
 - In python, `numpy.linalg.pinv(a)`
 - Remove redundant (not linearly independent) features
 - Remove extra features to ensure that $d \leq n$

Gradient Descent vs Closed Form

Gradient Descent

- Requires multiple iterations
- Need to choose α
- Works well when n is large
- Can support incremental learning

Closed Form Solution

- Non-iterative
- No need for α
- Slow if n is large
 - Computing $(X^T X)^{-1}$ is roughly $O(n^3)$

Extending Linear Regression to More Complex Models

- The inputs \mathbf{X} for linear regression can be:
 - Original quantitative inputs
 - Transformation of quantitative inputs
 - e.g. log, exp, square root, square, etc.
 - Polynomial transformation
 - example: $y = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2 + \beta_3 \cdot x^3$
 - Basis expansions
 - Dummy coding of categorical inputs
 - Interactions between variables
 - example: $x_3 = x_1 \cdot x_2$

This allows use of linear regression techniques to fit non-linear datasets.

Linear Basis Function Models

Linear Basis Function

- Simplest linear model for regression is one that involves a linear combination of the input variables

$$y(x, w) = w_0 + w_1 x_1 + \dots + w_D x_D$$

- Extend the class of models by considering linear combinations of fixed nonlinear functions of the input variables, of the form

$$y(x, w) = w_0 + \sum_{j=1}^{M-1} w_j \varphi_j(x)$$

- where $\varphi_j(x)$ are known as basis functions.
- By denoting the maximum value of the index j by M - 1, the total number of parameters in this model will be M.

Linear Basis Function

- Convenient to define an additional dummy ‘basis function’ $\phi_0(x)=1$. So,

$$y(x, w) = \sum_{j=1}^{M-1} w_j \varphi_j(x) = \mathbf{w}^\top \boldsymbol{\Phi}_j(x)$$

where $w = (w_0, \dots, w_{M-1})^T$ and $\boldsymbol{\Phi} = (\phi_0, \phi_1, \dots, \phi_n)$

- If the original variables comprise the vector x , then the features can be expressed in terms of the basis functions $\{\phi_j(x)\}$

Linear Basis Function Models

- Generally,

$$h_{\theta}(\mathbf{x}) = \sum_{j=0}^d \theta_j \phi_j(\mathbf{x})$$

basis function

- Typically, $\phi_0(\mathbf{x}) = 1$ so that θ_0 acts as a bias
- In the simplest case, we use linear basis functions :

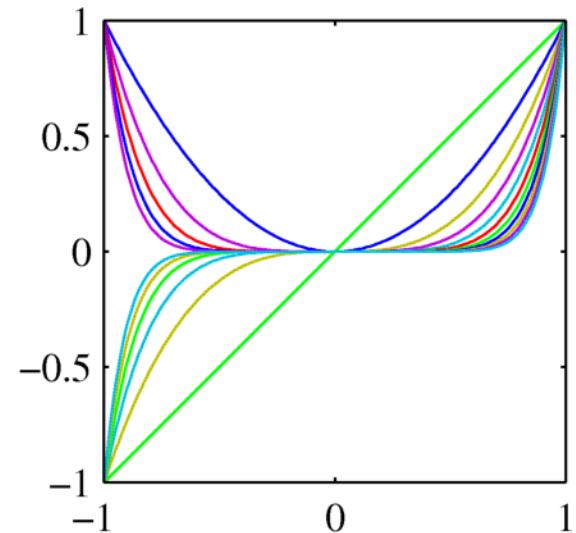
$$\phi_j(\mathbf{x}) = x_j$$

Linear Basis Function Models

- Polynomial basis functions:

$$\phi_j(x) = x^j$$

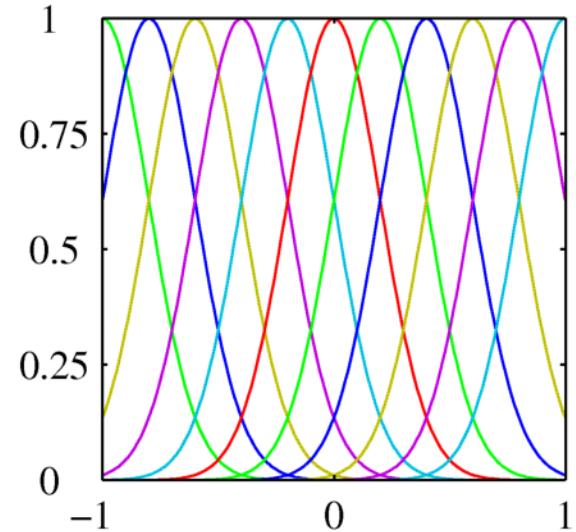
- These are global; a small change in x affects all basis functions



- Gaussian basis functions:

$$\phi_j(x) = \exp \left\{ -\frac{(x - \mu_j)^2}{2s^2} \right\}$$

- These are local; a small change in x only affect nearby basis functions. μ_j and s control location and scale (width).



Linear Basis Function Models

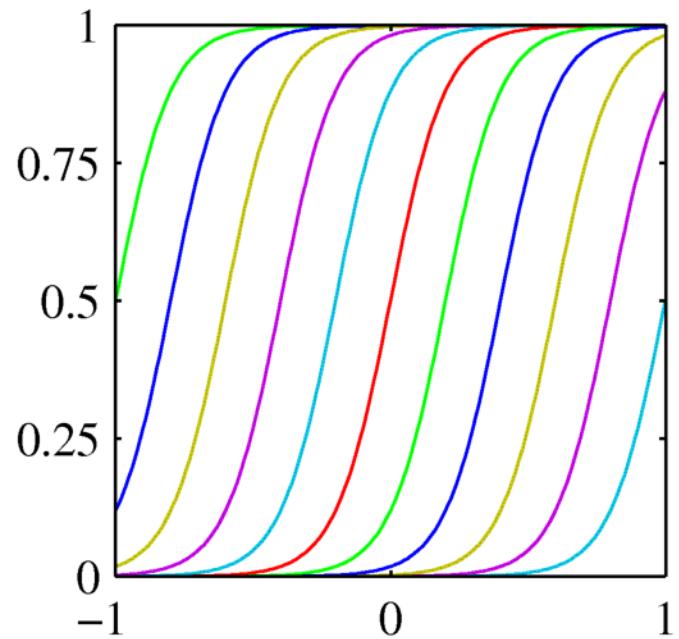
- Sigmoidal basis functions:

$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right)$$

where

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

- These are also local; a small change in x only affects nearby basis functions. μ_j and s control location and scale (slope).



Linear Basis Function Models

- Basic Linear Model:

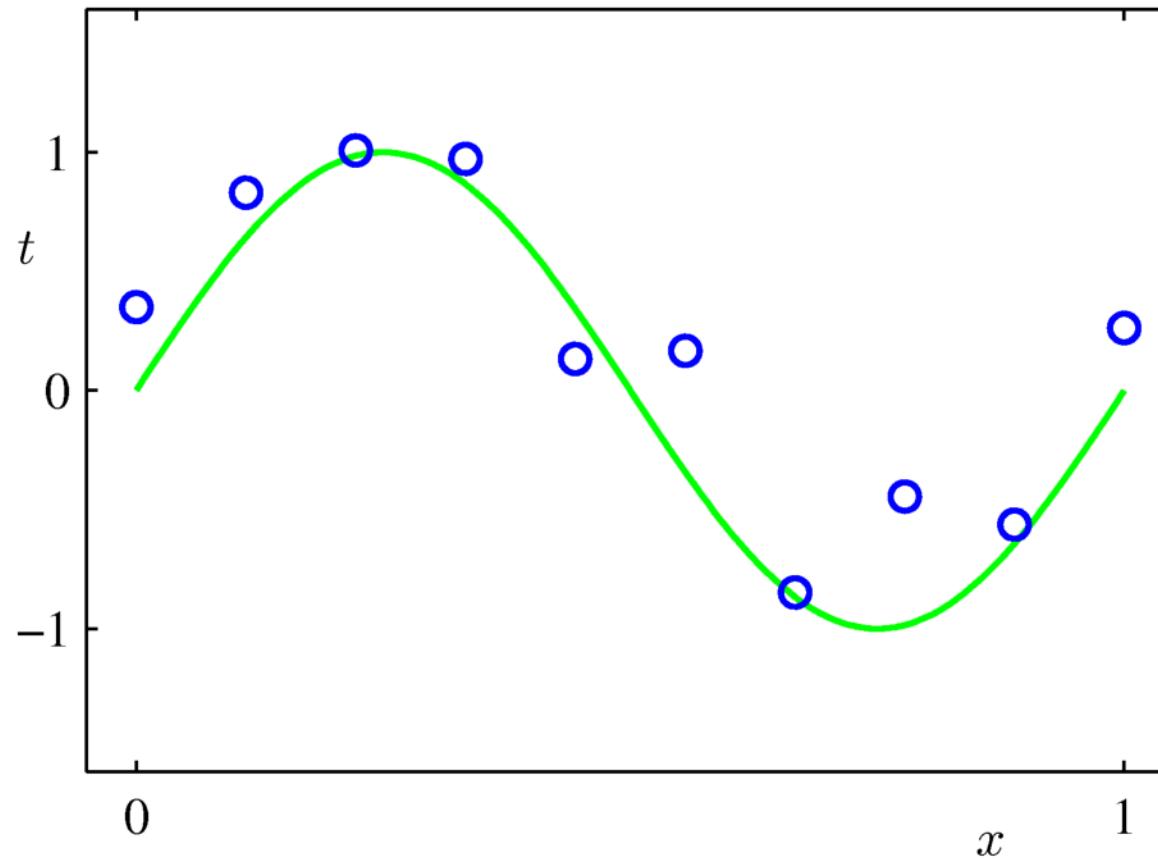
$$h_{\theta}(\mathbf{x}) = \sum_{j=0}^d \theta_j x_j$$

- Generalized Linear Model:

$$h_{\theta}(\mathbf{x}) = \sum_{j=0}^d \theta_j \phi_j(\mathbf{x})$$

- Once we have replaced the data by the outputs of the basis functions, fitting the generalized model is exactly the same problem as fitting the basic model
 - Unless we use the kernel trick – more on that when we cover support vector machines
 - Therefore, there is no point in cluttering the math with basis functions

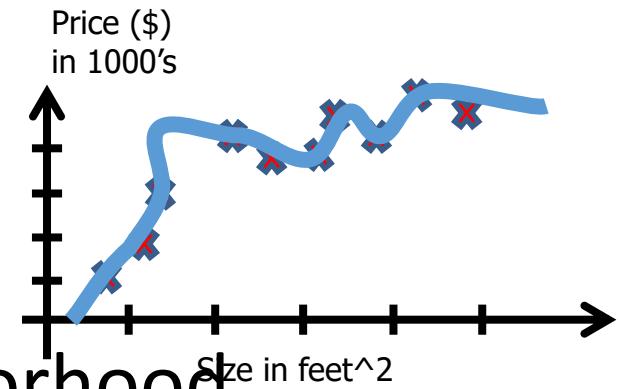
Example of Fitting a Polynomial Curve with a Linear Model



$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_p x^p = \sum_{j=0}^p \theta_j x^j$$

Addressing overfitting

- x_1 = size of house
- x_2 = no. of bedrooms
- x_3 = no. of floors
- x_4 = age of house
- x_5 = average income in neighborhood
- x_6 = kitchen size
- :
- x_{100}



Slide credit: Andrew Ng

Addressing overfitting

- **1. Reduce number of features.**
 - Manually select which features to keep.
 - Model selection algorithm

- **2. Regularization.**
 - Keep all the features, but reduce magnitude/values of parameters θ_j .
 - Works well when we have a lot of features, each of which contributes a bit to predicting y .

Slide credit: Andrew Ng

Regularization

- A method for automatically controlling the complexity of the learned hypothesis
- **Idea:** penalize for large values of θ_j
 - Can incorporate into the cost function
 - Works well when we have a lot of features, each that contributes a bit to predicting the label
- Can also address overfitting by eliminating features (either manually or via model selection)

Regularization

- Linear regression objective function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$$


The equation represents the cost function for linear regression. It consists of two parts: a sum of squared differences between the predicted values $h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$ and the actual values $y^{(i)}$ for all data points i , scaled by $\frac{1}{2n}$; and a regularization term, which is a sum of the squares of all model parameters θ_j for j from 1 to d , scaled by $\frac{\lambda}{2}$. The regularization parameter λ controls the trade-off between fitting the data well and keeping the model parameters small.

- λ is the regularization parameter ($\lambda \geq 0$)
- No regularization on θ_0 !

Understanding Regularization

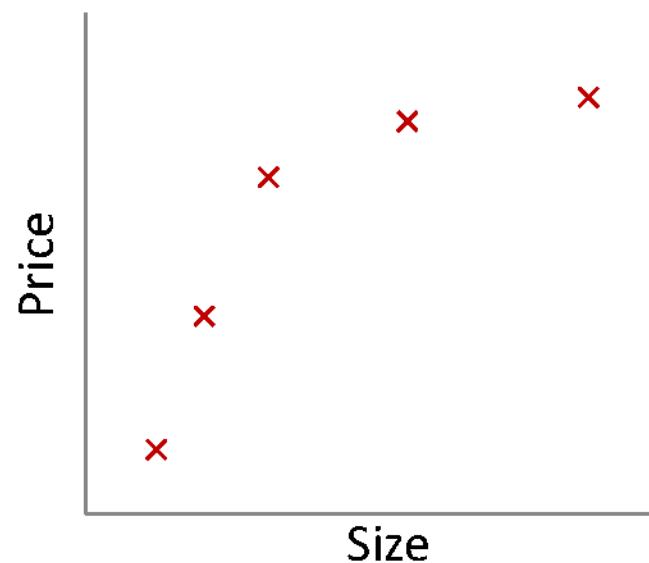
$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$$

- Note that $\sum_{j=1}^d \theta_j^2 = \|\boldsymbol{\theta}_{1:d}\|_2^2$
 - This is the magnitude of the feature coefficient vector!
- We can also think of this as:
$$\sum_{j=1}^d (\theta_j - 0)^2 = \|\boldsymbol{\theta}_{1:d} - \vec{0}\|_2^2$$
 - L₂ regularization pulls coefficients toward 0

Understanding Regularization

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$$

- What happens if we set λ to be huge (e.g., 10^{10})?

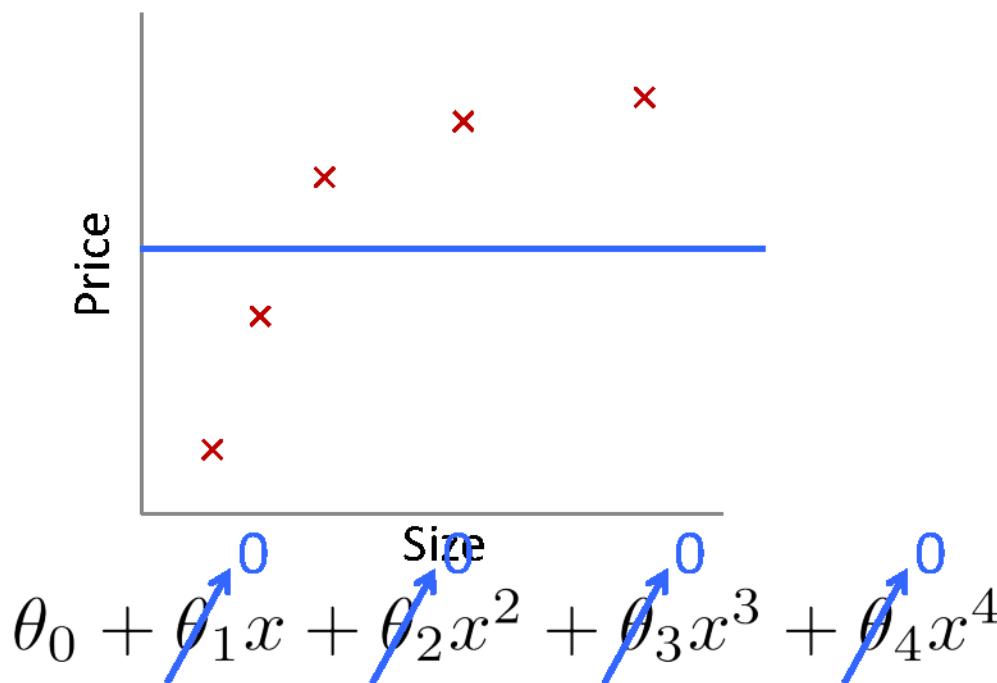


$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Understanding Regularization

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$$

- What happens if we set λ to be huge (e.g., 10^{10})?



Regularized Linear Regression

- Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$$

- Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

- Gradient update:

$$\frac{\partial}{\partial \theta_0} J(\boldsymbol{\theta})$$

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)$$

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

$$\theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} - \lambda \theta_j$$

Regularized Linear Regression

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^d \theta_j^2$$

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)$$

$$\theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} - \lambda \theta_j$$

- We can rewrite the gradient step as:

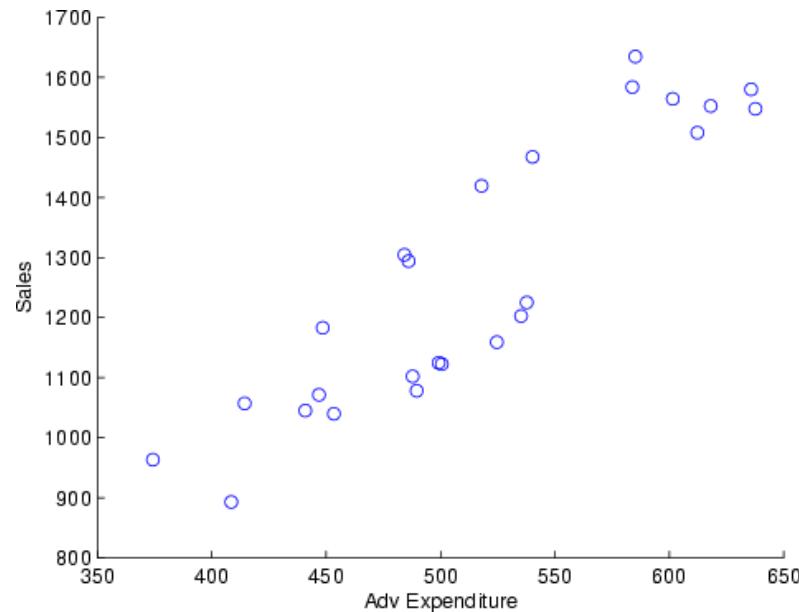
$$\theta_j \leftarrow \theta_j (1 - \alpha \lambda) - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

Skewness or Kurtosis

- Skewness is the measure of symmetry or asymmetry of data distribution.
- A distribution or data set is said to be symmetric if it looks the same to the left and right points of the center.
- Kurtosis is the characteristic of being flat or peaked. It is a measure of whether data is heavy-tailed or light-tailed in a normal distribution
- histogram is an effective way to show both the skewness and kurtosis of a data set because you can easily spot if something is wrong with your data. A probability plot is also a great tool because a normal distribution would just follow the straight line
- best way to fix it is to perform a log transform of the same data, with the intent to reduce the skewness
- you can also conduct a [boxcox transform. scipy does all the hardwork for you.](#)
- apply a square root transformation via *Numpy*, by calling the `sqrt()` function

Regression analysis

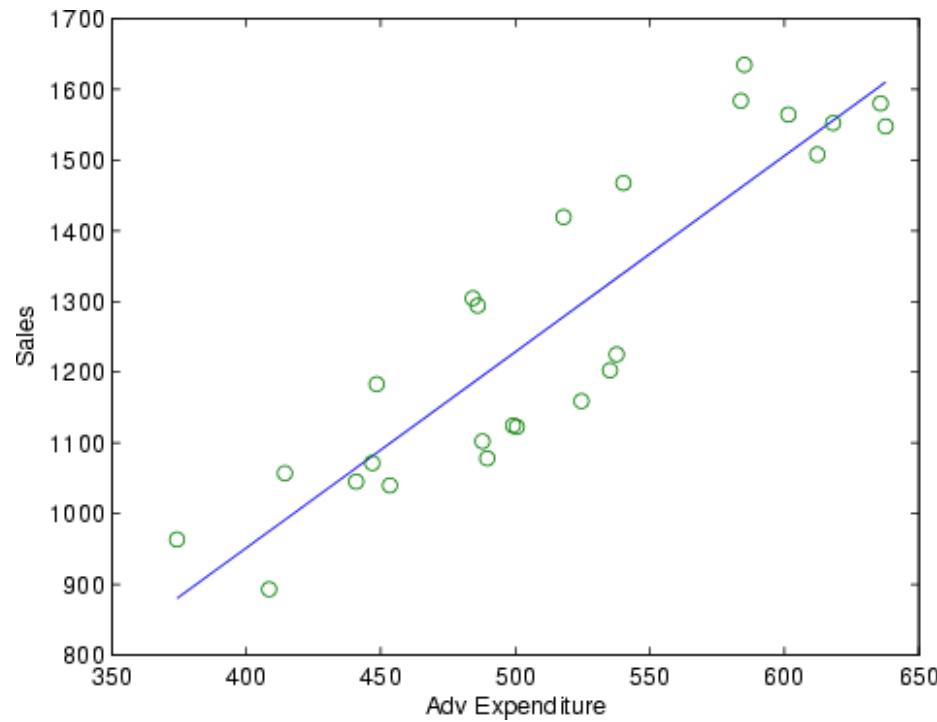
- Step 1: graphical display of data — scatter plot: sales vs. advertisement cost



- calculate correlation

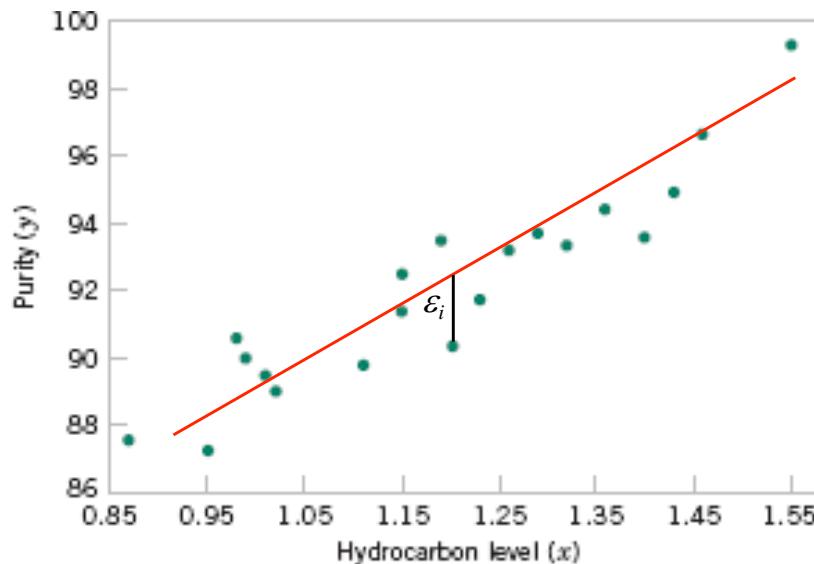
$$\hat{\rho} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \times \sum_{i=1}^n (y_i - \bar{y})^2}} \quad -1 \leq \hat{\rho} \leq 1$$

- Step 2: find the relationship or association between Sales and Advertisement Cost — Regression



Simple linear regression

Based on the scatter diagram, it is probably reasonable to assume that the mean of the random variable Y is related to X by the following **simple linear regression model**:



Response Regressor or Predictor

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i \quad i = 1, 2, \dots, n$$
 Intercept Slope Random error

$$\varepsilon_i \sim N(0, \sigma^2)$$

where the slope and intercept of the line are called **regression coefficients**.

- The case of simple linear regression considers a single regressor or predictor x and a dependent or response variable Y.

Regression coefficients

$$S_{xx} = \sum_{i=1}^n (x_i - \bar{x})^2 = \sum_{i=1}^n x_i^2 - \frac{\left(\sum_{i=1}^n x_i\right)^2}{n}$$

$$S_{xy} = \sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x}) = \sum_{i=1}^n x_i y_i - \frac{\left(\sum_{i=1}^n x_i\right)\left(\sum_{i=1}^n y_i\right)}{n}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

$$\hat{\beta}_1 = \frac{S_{xy}}{S_{xx}}$$

} Fitted (estimated)
regression model

Caveat: regression relationships are valid only for values of the regressor variable within the range of the original data. Be careful with extrapolation.

Estimation of variance

- Using the fitted model, we can estimate value of the response variable for given predictor

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

- Residuals: $r_i = y_i - \hat{y}_i$
- Our model: $Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i, i = 1, \dots, n, \text{Var}(\varepsilon_i) = \sigma^2$
- Unbiased estimator (MSE: Mean Square Error)

$$\hat{\sigma}^2 = MSE = \frac{\sum_{i=1}^n r_i^2}{n-2}$$

Punchline

- the coefficients

$$\hat{\beta}_1 \text{ and } \hat{\beta}_0$$

and both calculated from data, and they are subject to error.

- if the true model is $y = \beta_1 x + \beta_0$, $\hat{\beta}_1$ and $\hat{\beta}_0$ are point estimators for the true coefficients
- we can talk about the ``accuracy'' of $\hat{\beta}_1$ and $\hat{\beta}_0$

Assessing linear regression model

- Test hypothesis about true slope and intercept

$$\beta_1 = ?, \quad \beta_0 = ?$$

- Construct confidence intervals

$$\beta_1 \in [\hat{\beta}_1 - a, \hat{\beta}_1 + a] \quad \beta_0 \in [\hat{\beta}_0 - b, \hat{\beta}_0 + b] \quad \text{with probability } 1 - \alpha$$

- Assume the errors are normally distributed

$$\varepsilon_i \sim N(0, \sigma^2)$$

Properties of Regression Estimators

slope parameter β_1

$$E(\hat{\beta}_1) = \beta_1$$

intercept parameter β_0

$$E(\hat{\beta}_0) = \beta_0$$

$$V(\hat{\beta}_1) = \frac{\sigma^2}{S_{xx}}$$

$$V(\hat{\beta}_0) = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}} \right]$$

unbiased estimator

$$S_{xx} = \sum_{i=1}^n (x_i - \bar{x})^2 = \sum_{i=1}^n x_i^2 - \frac{\left(\sum_{i=1}^n x_i \right)^2}{n}$$

unbiased estimator

Standard errors of coefficients



- We can replace σ^2 with its estimator $\hat{\sigma}^2$...

$$\hat{\sigma}^2 = MSE = \frac{\sum_{i=1}^n r_i^2}{n-2}$$

$$r_i = y_i - \hat{y}_i \quad \hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

- Using results from previous page, estimate the

$$se(\hat{\beta}_1) = \sqrt{\frac{\hat{\sigma}^2}{S_{xx}}} \quad \text{and} \quad se(\hat{\beta}_0) = \sqrt{\hat{\sigma}^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}} \right]}$$

Hypothesis test in simple linear regression

- we wish to test the hypothesis whether the slope equals a constant $\beta_{1,0}$

$$H_0: \beta_1 = \beta_{1,0}$$

$$H_1: \beta \neq \beta_{1,0}$$

- e.g. relate **ads** to **sales**, we are interested in study whether or not increase a \$ on ads will increase $\$ \beta_{1,0}$ in sales?
- sale = $\beta_{1,0}$ ads +constant?



A related and important question...

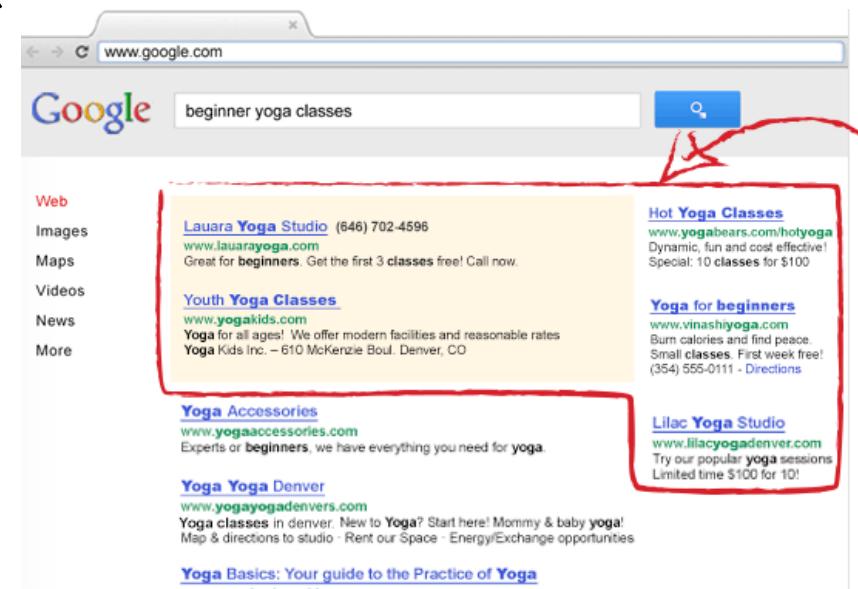
- whether or not the slope is zero ?

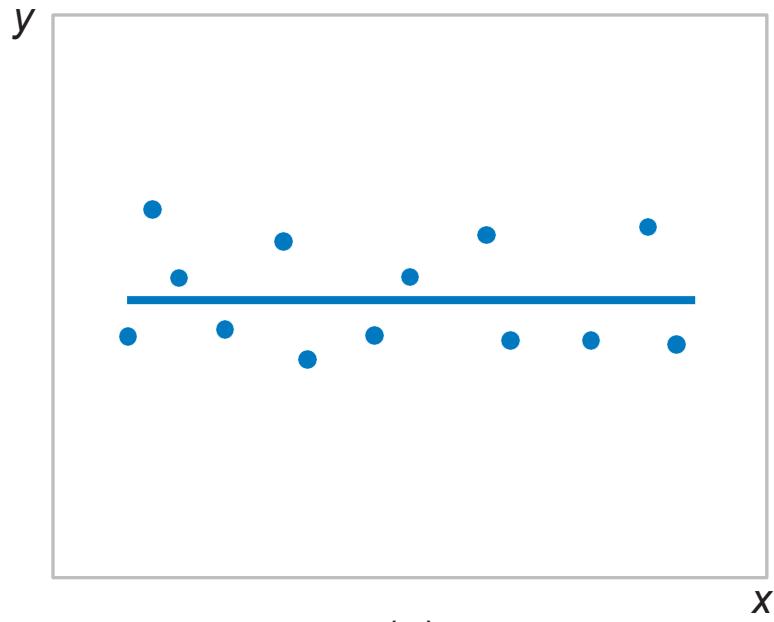
$$H_0: \beta_1 = 0$$

$$H_1: \beta_1 \neq 0$$

- if $\beta_1 = 0$, that means Y does not depend on X, i.e.,
- Y and X are **independent**
- In the advertisement example, does **ads increase sales?** or no effect?

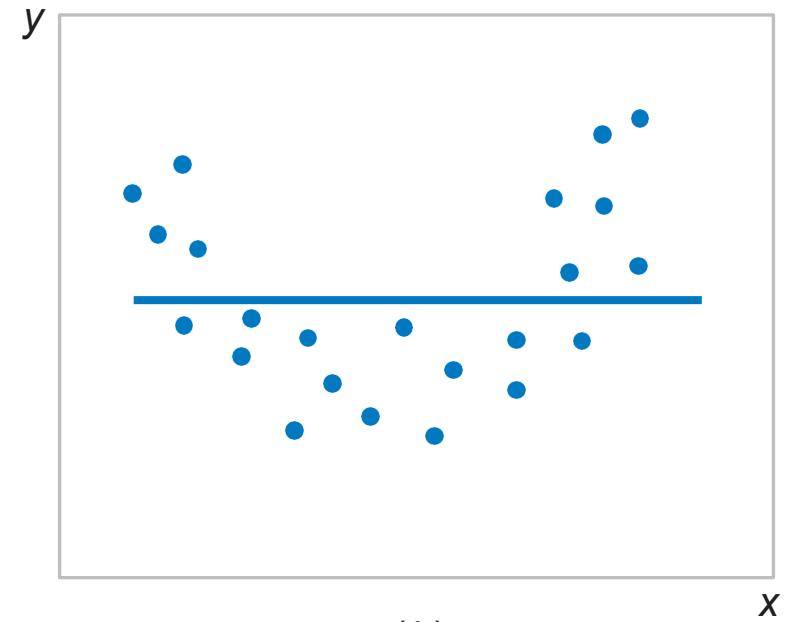
Significance of regression





(a)

- H_0 not rejected



(b)

- H_0 rejected

Use t-test for slope

Under H_0

slope parameter β_1

$$E(\hat{\beta}_1) = \beta_{1,0}$$

$$V(\hat{\beta}_1) = \frac{\sigma^2}{S_{xx}}$$

$$\hat{\beta}_1 \sim N\left(\beta_{1,0}, \frac{\sigma^2}{S_{xx}} \right)$$

- Under H_0 , test statistic

$$T_0 = \frac{\hat{\beta}_1 - \beta_{1,0}}{\sqrt{\hat{\sigma}^2 / S_{xx}}}$$

~ t distribution with
n-2 degree of freedom

- Reject H_0 if

$$|t_0| > t_{\alpha/2, n-2}$$

(two-sided test)

Example: oxygen purity tests of coefficients

- Consider the test

$$H_0: \beta_1 = 0$$

$$H_1: \beta_1 \neq 0$$

$$\hat{\beta}_1 = 14.947 \quad n = 20,$$

$$S_{xx} = 0.68088, \quad \sigma^2 = 1.18$$

- Calculate the test statistic

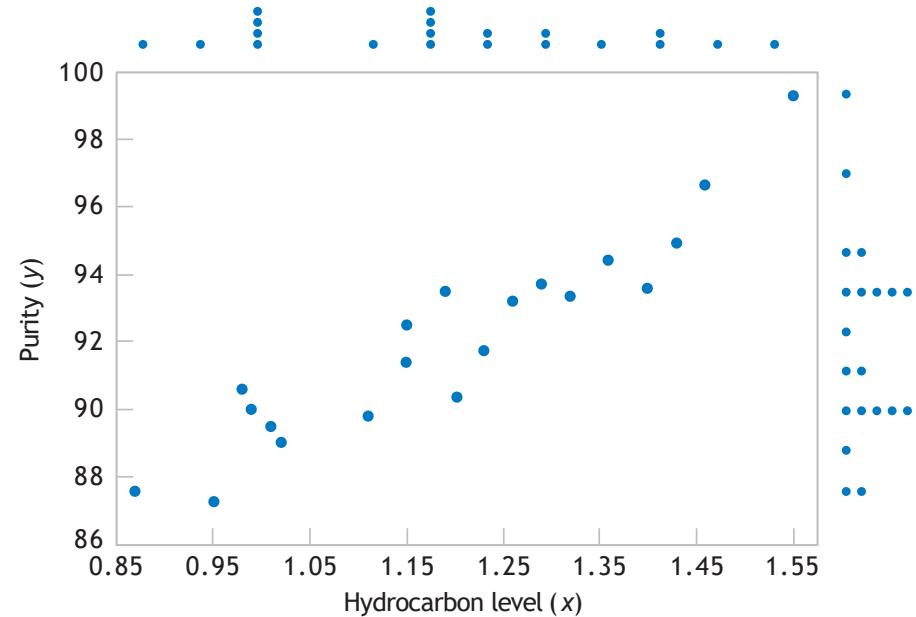


Figure 11-1

Scatter diagram of oxygen purity versus hydrocarbon level from Table 11-1.

- Threshold $t_{\alpha/2, n-2} = t_{0.005, 18} = 2.88$
- Reject H_0 since $|t_0| > t_{\alpha/2, n-2}$

Use t-test for intercept

- Use a similar form of test

$$H_0: \beta_0 = \beta_{0,0}$$

$$H_1: \beta_0 \neq \beta_{0,0}$$

- **Test statistic** $T_0 = \frac{\hat{\beta}_0 - \beta_{0,0}}{\sqrt{\hat{\sigma}^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}} \right]}} = \frac{\hat{\beta}_0 - \beta_{0,0}}{se(\hat{\beta}_0)}$

Under H_0 , $T_0 \sim t$ distribution with $n-2$ degree of freedom

- Reject H_0 if $|t_0| > t_{\alpha/2, n-2}$

Confidence interval

- we can obtain confidence interval estimates of slope and intercept
- width of confidence interval is a measure of the overall quality of the regression

		true parameter
slope	intercept	
$T_0 = \frac{\hat{\beta}_1 - \beta_{1,0}}{\sqrt{\hat{\sigma}^2 / S_{xx}}}$	$T_0 = \frac{\hat{\beta}_0 - \beta_{0,0}}{\sqrt{\hat{\sigma}^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}} \right]}}$	
$\sim t$ distribution with $n-2$ degree of freedom	$\sim t$ distribution with $n-2$ degree of freedom	0

Confidence intervals

a $100(1 - \alpha)\%$ confidence interval on the slope β_1

$$\hat{\beta}_1 - t_{\alpha/2, n-2} \sqrt{\frac{\hat{\sigma}^2}{S_{xx}}} \leq \beta_1 \leq \hat{\beta}_1 + t_{\alpha/2, n-2} \sqrt{\frac{\hat{\sigma}^2}{S_{xx}}}$$

a $100(1 - \alpha)\%$ confidence interval on the intercept β_0

$$\begin{aligned} \hat{\beta}_0 - t_{\alpha/2, n-2} \sqrt{\hat{\sigma}^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}} \right]} \\ \leq \beta_0 \leq \hat{\beta}_0 + t_{\alpha/2, n-2} \sqrt{\hat{\sigma}^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}} \right]} \end{aligned}$$

Example: oxygen purity tests of coefficients

find a 95% confidence interval on the slope ($\alpha = 0.05$)

$$\hat{\beta}_1 = 14.947, S_{xx} = 0.68088, \text{ and } \hat{\sigma}^2 = 1.18$$

$$\hat{\beta}_1 - t_{0.025,18} \sqrt{\frac{\hat{\sigma}^2}{S_{xx}}} \leq \beta_1 \leq \hat{\beta}_1 + t_{0.025,18} \sqrt{\frac{\hat{\sigma}^2}{S_{xx}}}$$

$$14.947 - 2.101 \sqrt{\frac{1.18}{0.68088}} \leq \beta_1 \leq 14.947 + 2.101 \sqrt{\frac{1.18}{0.68088}}$$

$$12.181 \leq \beta_1 \leq 17.713$$

The confidence interval does not include 0, so enough evidence saying there is enough correlation between X and Y.

Example: house selling price and annual taxes

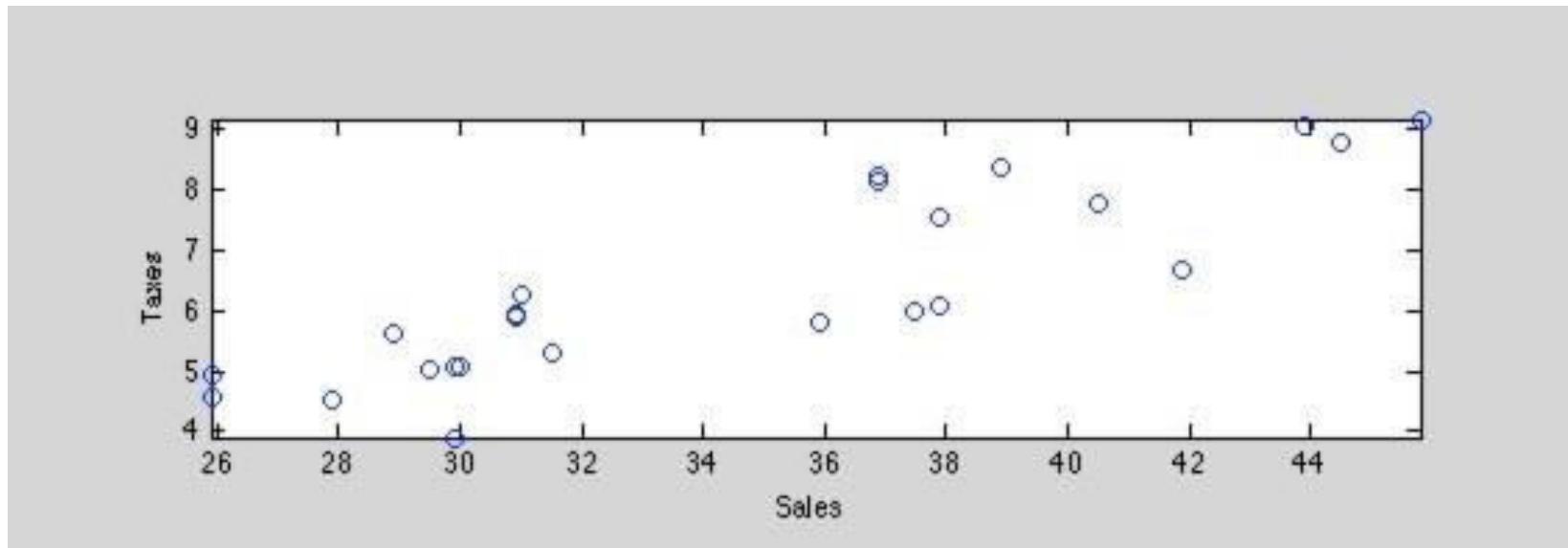
Sale Price/1000	Taxes (Local, School), County/1000	Sale Price/1000	Taxes (Local, School), County/1000
25.9	4.9176	30.0	5.0500
29.5	5.0208	36.9	8.2464
27.9	4.5429	41.9	6.6969
25.9	4.5573	40.5	7.7841
29.9	5.0597	43.9	9.0384
29.9	3.8910	37.5	5.9894
30.9	5.8980	37.9	7.5422
28.9	5.6039	44.5	8.7951
35.9	5.8282	37.9	6.0831
31.5	5.3003	38.9	8.3607
31.0	6.2712	36.9	8.1400
30.9	5.9592	45.8	9.1416



Independent variable X: SalePrice

Dependent variable Y: Taxes

- qualitative analysis



Calculate correlation

$$\hat{\rho} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \times \sum_{i=1}^n (y_i - \bar{y})^2}} = 0.8760$$

Independent variable Y: SalePrice

Dependent variable X: Taxes

$$n = 24 \quad \bar{x} = 34.6125 \quad \bar{y} = 6.4049$$

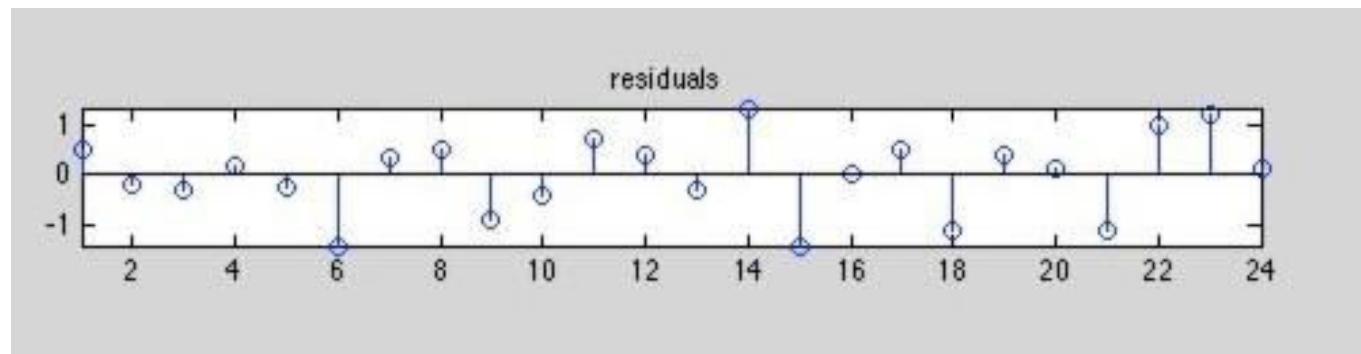
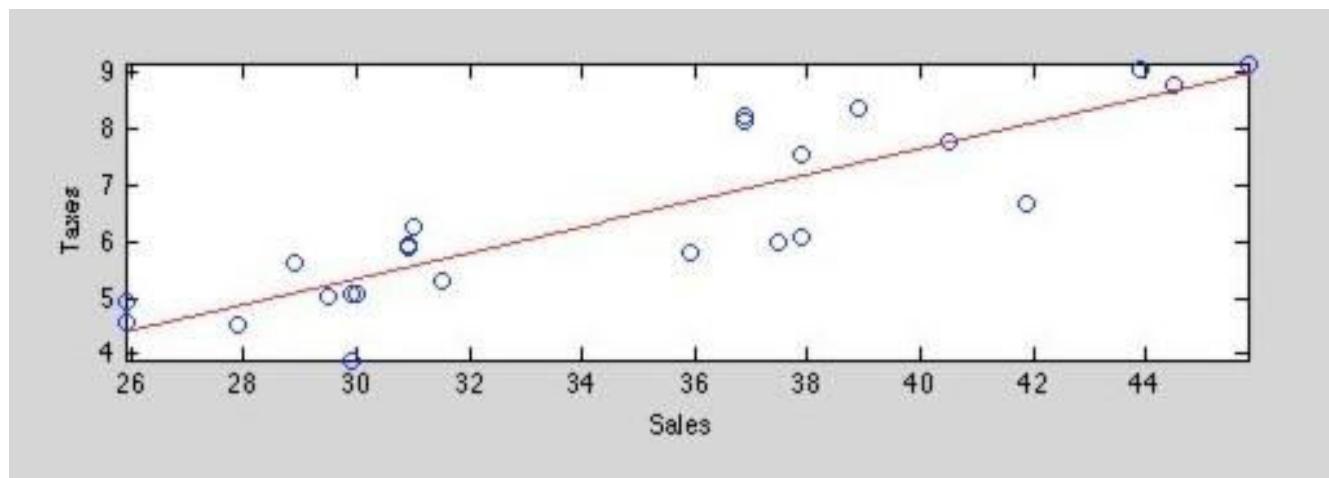
$$S_{xx} = \sum_{i=1}^n (x_i - \bar{x})^2 = 829.0462$$

$$S_{xy} = \sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x}) = 191.3612$$

$$\hat{\beta}_1 = \frac{S_{xy}}{S_{xx}} = \frac{191.3612}{829.0462} = 0.2308$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} = 6.4049 - 0.2308 \times 34.6125 = -1.5837$$

Fitted simple linear regression model $\hat{y} = -1.5837 + 0.2308x$



$$\sum_{i=1}^n r_i^2$$

- residuals: $\hat{\sigma}^2 = MSE = \frac{\sum_{i=1}^n r_i^2}{n-2} = 0.6088$

- standard error of regression coefficients
-

$$se(\hat{\beta}_1) = \sqrt{\frac{\hat{\sigma}^2}{S_{xx}}} = \sqrt{\frac{0.6088}{829.0462}} = 0.0271$$

$$se(\hat{\beta}_0) = \sqrt{\hat{\sigma}^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}} \right]} = \sqrt{0.6088 \left[\frac{1}{24} + \frac{34.6125^2}{829.0462} \right]} = 0.9514$$

- test
-

Test $H_0: \beta_1 = 0$ using the t -test; use $\alpha = 0.05$

- calculate test statistics

$$t_0 = \frac{\hat{\beta}_1}{\sqrt{\hat{\sigma}^2/S_{xx}}} = \frac{\hat{\beta}_1}{se(\hat{\beta}_1)} = \frac{0.2308}{0.0271} = 8.5166$$

- threshold

$$t_{\alpha/2, n-2} = t_{0.0025, 22} = 3.119$$

- value of test statistic is greater than threshold
-  reject H_0

- construct confidence interval for slope parameter

$$\hat{\beta}_1 - t_{\alpha/2, n-2} \sqrt{\frac{\hat{\sigma}^2}{S_{xx}}} \leq \beta_1 \leq \hat{\beta}_1 + t_{\alpha/2, n-2} \sqrt{\frac{\hat{\sigma}^2}{S_{xx}}}$$

$$t_{\alpha/2, n-2} = t_{0.0025, 22} = 3.119$$

$$0.2308 - 3.119 \times 0.0271 \leq \beta_1 \leq 0.2308 + 3.119 \times 0.0271$$

$$0.14631 \leq \beta_1 \leq 0.3153$$



BITS Pilani
Pilani Campus

Machine Learning ZG565

Dr. Sugata Ghosal
sugata.ghosal@pilani.bits-pilani.ac.in



Lecture No. – 5 | Discriminative Classifiers

Date – 10/12/2022

Time – 4:15 PM to 6:15 PM

Session Content

- Linear Regression
 - Overfitting, Regularization, Performance ..
- Types of Classifiers
- Logistic regression
 - Logistic function
 - Log-loss function and gradient descent
 - Regularization
 - Multi-class classification

Classification

- Given a collection of records (training set)
 - Each record is characterized by a tuple (x, y) , where x is the attribute (feature) set and y is the class label
 - x aka attribute, predictor, independent variable, input
 - Y aka class, response, dependent variable, output
- Task
 - Learn a model or function that maps each attribute set x into one of the predefined class labels y

Task	Attribute set, x	Class label, y
Categorizing email messages	Features extracted from email message header and content	spam or non-spam
Identifying tumor cells	Features extracted from x-rays or MRI scans	malignant or benign cells
Cataloging galaxies	Features extracted from telescope images	Elliptical, spiral, or irregular-shaped galaxies

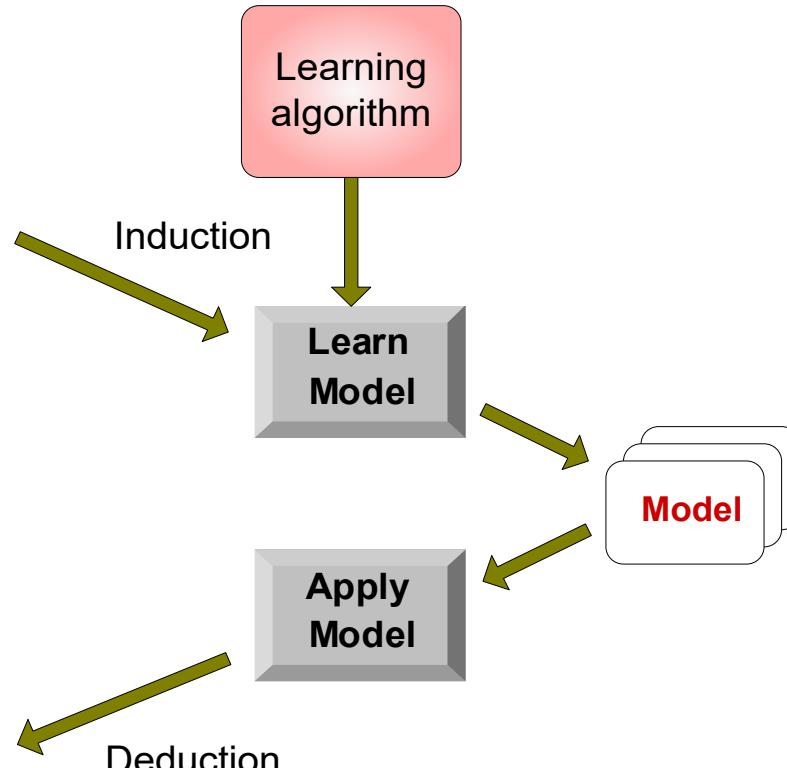
General Approach for Building Classification Model

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Types of Classifiers

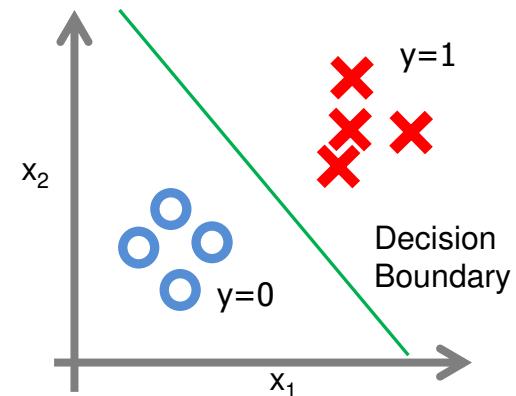
Linear Classifier

- Classes are separated by a linear decision surface (e.g., straight line in 2-dimensional feature/attribute space)
 - If for a given record, linear combination of features x_i is ≥ 0 , i.e.,

$$w_0 + \sum_i w_i x_i \geq 0$$

it belongs to one class (say, $y = 1$), else it belongs to the other class (say, $y=0$ or -1)

- w_i s are learned during the training (induction) phase of the classifier.
- Learnt w_i s are applied to a test record during the deduction / inferencing phase.
- In nonlinear classification, classes are separated by a non-linear surface



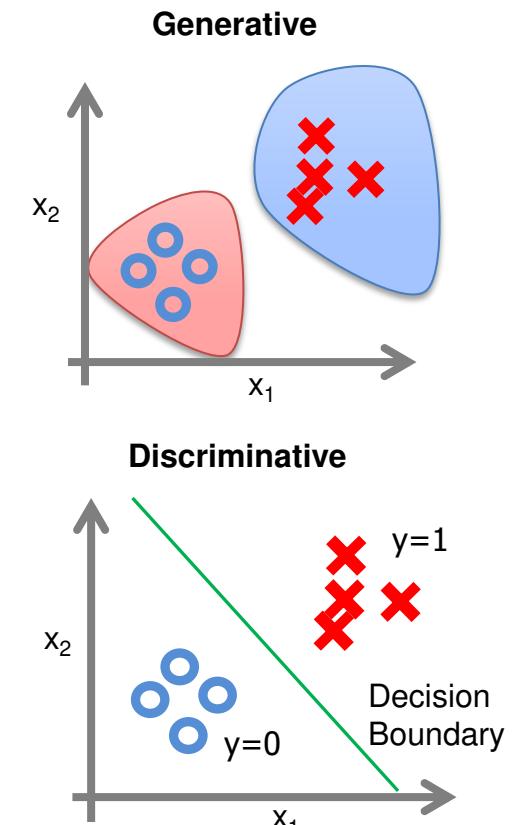
Generative Vs. Discriminative Models

- Generative Model

- Class-conditional probability distribution of attribute/feature set and prior probability of classes are learnt during the training phase
- Given these learnt probabilities, during inferencing phase, probability of a test record belonging to different classes are calculated and compared.
- Can result in linear or nonlinear decision surface

- Discriminative Model

- Given a training set, a function f is learnt that directly maps an attribute/feature vector x to the output class ($y=1$ or $0/-1$)
- A linear function f results in linear decision surface



Logistic Regression

Logistic Regression

Idea:

- Given data X and associated binary (0/1) class label Y , Logistic Regression tries to learn a discriminant function $P(Y|X)$
 - If $Y = 1$, $P(Y|X) = 1$ else $P(Y|X) = 0$

Logistic Regression versus linear regression



- **Linear Regression** could help us predict the student's test score on a scale of 0 - 100. Linear regression predictions are continuous (numbers in a range).
 - **Logistic Regression** could help use predict whether the student passed or failed. Logistic regression predictions are discrete (only specific values or categories are allowed). We can also view probability scores underlying the model's classifications.
-

Linear Regression versus Logistic Regression

Classification requires discrete values:

$$y = 0 \text{ or } 1$$

For linear Regression output values:

$h_{\theta}(x)$ can be much > 1 or much < 0

Logistic Regression: $0 \leq h_{\theta}(x) \leq 1$

Sigmoid/Logistic Function

- Sigmoid/logistic function takes a real value as input and outputs another value between 0 and 1
- That framework is called logistic regression
 - Logistic: A special mathematical sigmoid function it uses
 - Regression: Combines a weight vector with observations to create an answer

$$h_{\theta}(x) = g(\theta^T x)$$

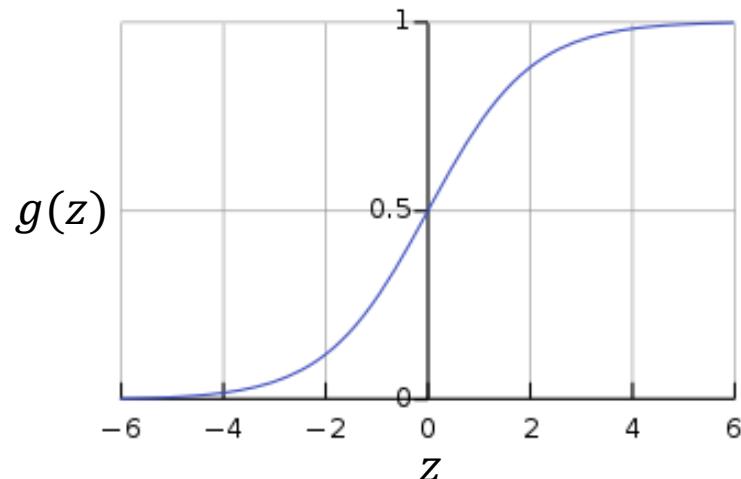
Hypothesis representation

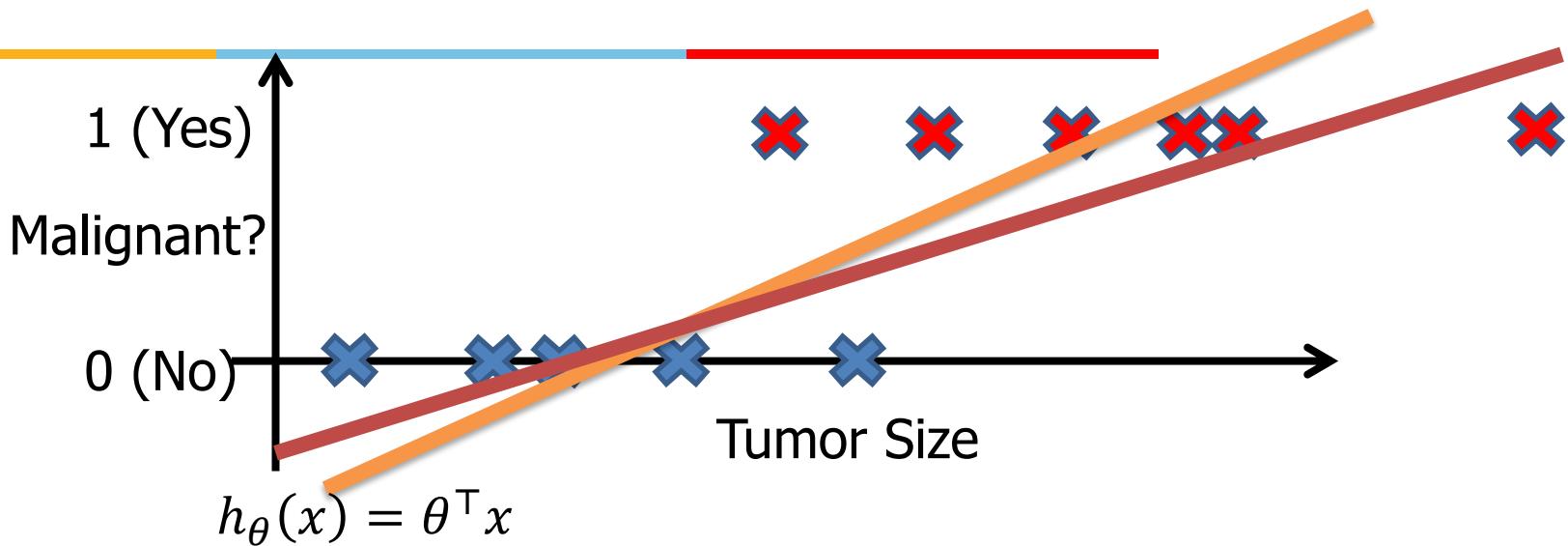
- Want $0 \leq h_\theta(x) \leq 1$
- $h_\theta(x) = g(\theta^\top x)$,

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

- Sigmoid function
- Logistic function

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}$$





- Threshold classifier output $h_\theta(x)$ at 0.5
 - If $h_\theta(x) \geq 0.5$, predict “ $y = 1$ ”
 - If $h_\theta(x) < 0.5$, predict “ $y = 0$ ”

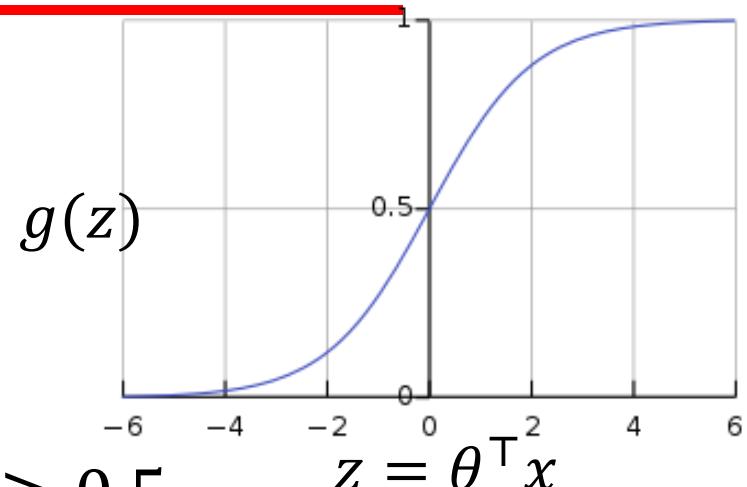
Interpretation of hypothesis output

- $h_{\theta}(x)$ = estimated probability that $y = 1$ on input x
- Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$
- $h_{\theta}(x) = 0.7$
- Tell patient that 70% chance of tumor being malignant

Logistic regression

$$h_{\theta}(x) = g(\theta^T x)$$

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



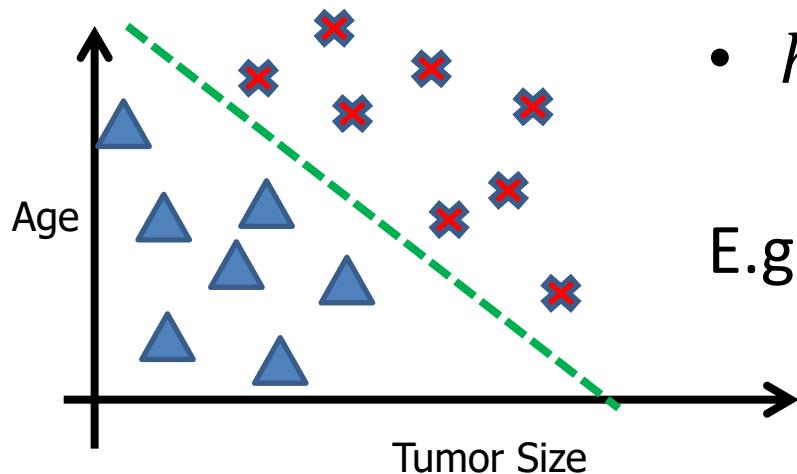
Suppose predict “y = 1” if $h_{\theta}(x) \geq 0.5$

$$z = \theta^T x \geq 0$$

predict “y = 0” if $h_{\theta}(x) < 0.5$

$$z = \theta^T x < 0$$

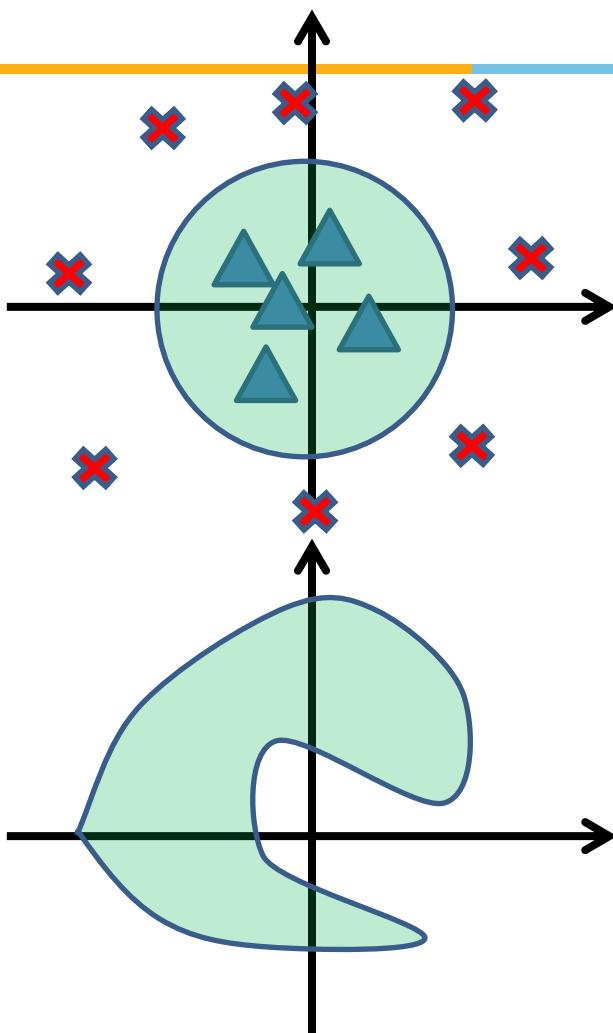
Decision boundary



- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

- E.g., $\theta_0 = -3, \theta_1 = 1, \theta_2 = 1$

- Predict " $y = 1$ " if $-3 + x_1 + x_2 \geq 0$



- $$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

E.g., $\theta_0 = -1, \theta_1 = 0, \theta_2 = 0, \theta_3 = 1, \theta_4 = 1$
- Predict “ $y = 1$ ” if $-1 + x_1^2 + x_2^2 \geq 0$
- $$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots)$$

Learning model parameters

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

m examples

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \quad x_0 = 1, y \in \{0, 1\}$$

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

How to choose parameters
(feature weights) θ ?

Cost function for Linear Regression

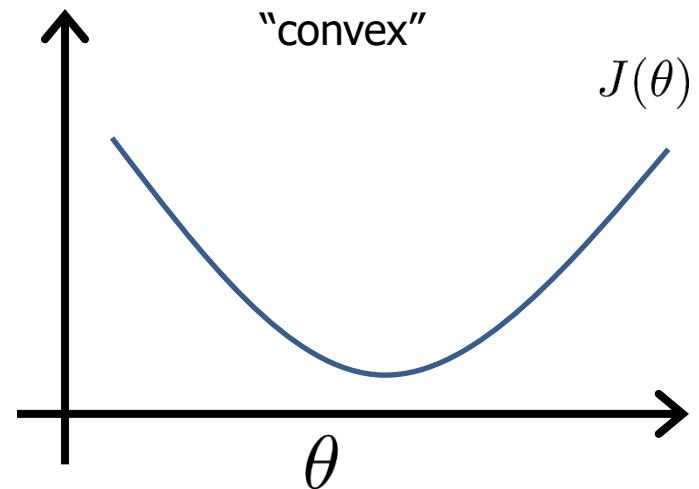
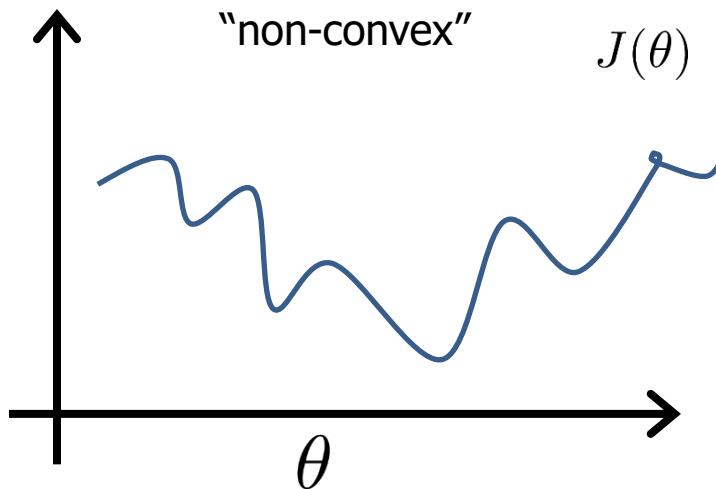
$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_\theta(x^{(i)}), y)$$

$$\text{Cost}(h_\theta(x), y) = \frac{1}{2} (h_\theta(x) - y)^2$$

MSE Cost Function

Linear regression: $J(\theta) = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$

$$\text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$$

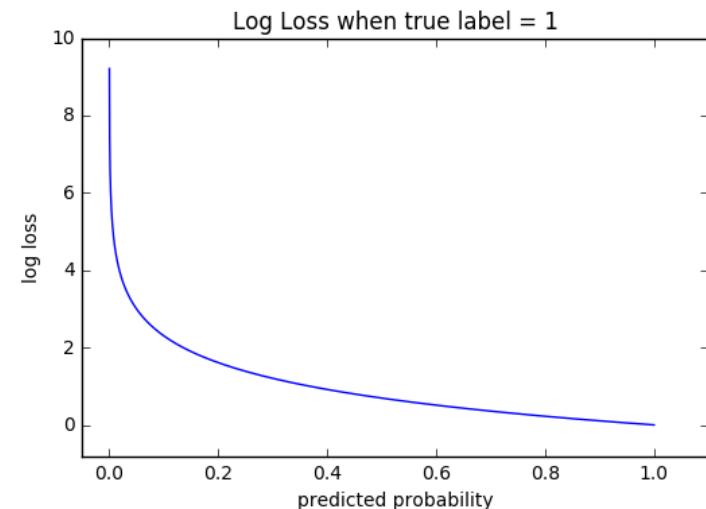


Error (Cost) Function

- Our prediction function is non-linear (due to sigmoid transform)
 - Squaring this prediction as we do in MSE results in a non-convex function with many local minima.
 - If our cost function has many local minimums, gradient descent may not find the optimal global minimum.
 - So instead of Mean Squared Error, we use a error/cost function called Cross-Entropy, also known as Log Loss.
-

Cross Entropy

- Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1.
- Cross-entropy loss increases as the predicted probability diverges from the actual label.
- So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value.
- A perfect model would have a log loss of 0.
- Cross-entropy loss can be divided into two separate cost functions:
one for $y=1$ and
one for $y=0$.



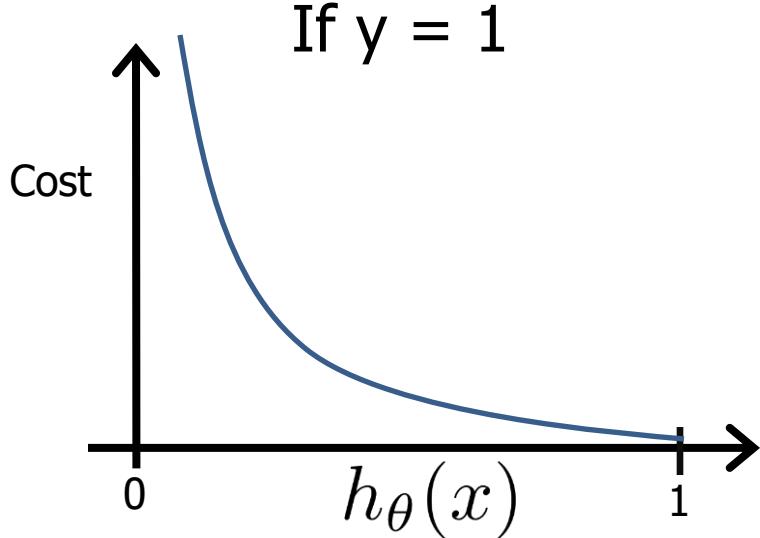
Logistic regression cost function (cross entropy)

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

If $y = 1$

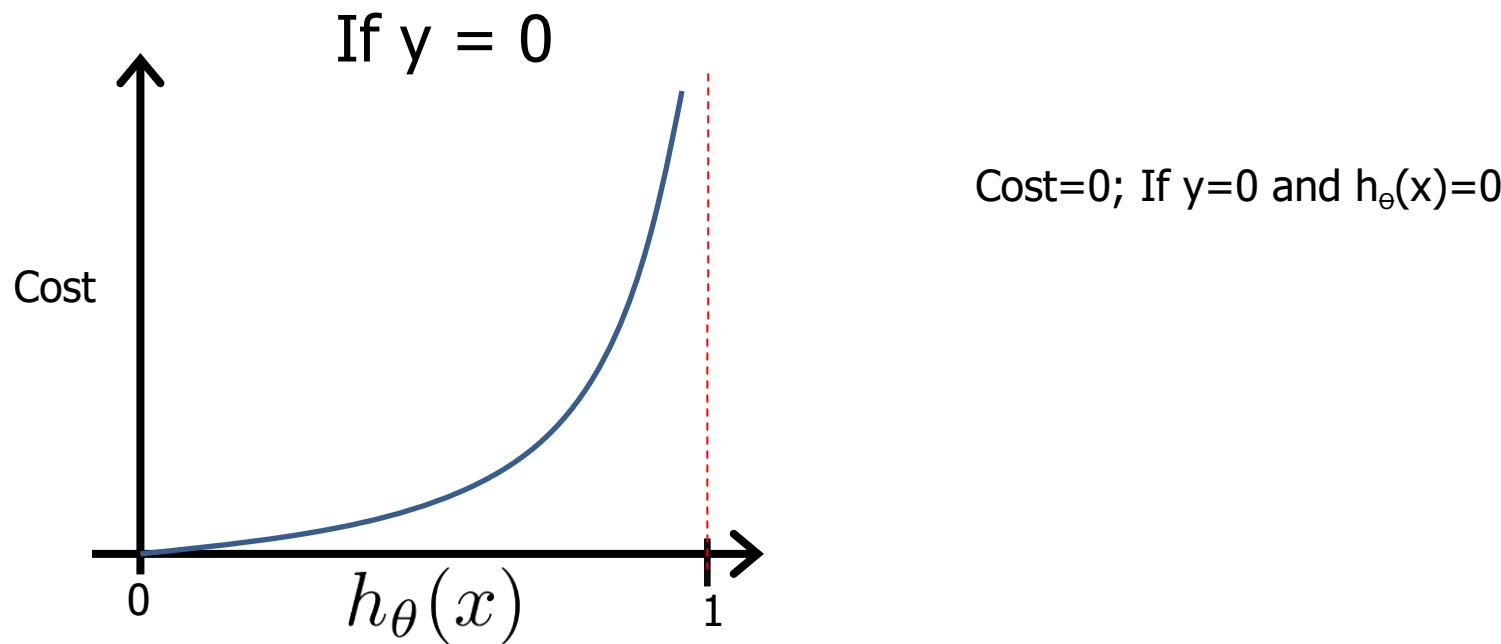
$\text{Cost} = 0$ if $y = 1, h_\theta(x) = 1$
 But as $h_\theta(x) \rightarrow 0$
 $\text{Cost} \rightarrow \infty$

Captures intuition that if $h_\theta(x) = 0$,
 (predict $P(y = 1|x; \theta) = 0$), but $y = 1$,
 we'll penalize learning algorithm by a very
 large cost.



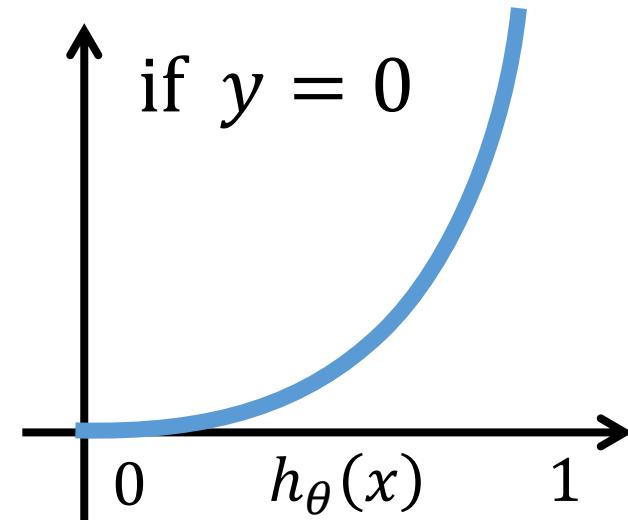
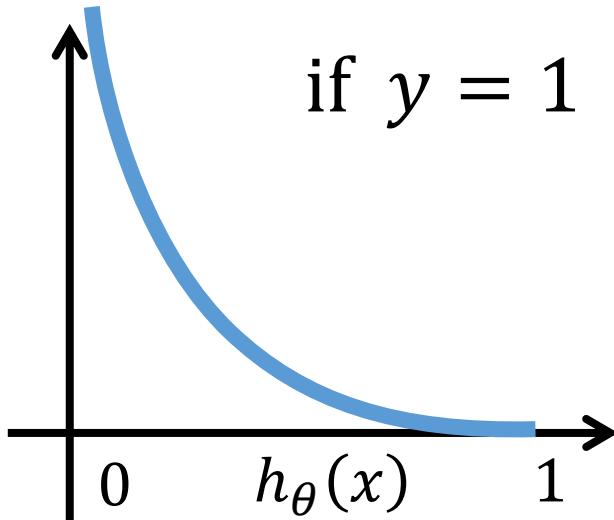
Logistic regression cost function

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



Cost function for Logistic Regression

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



Logistic regression cost function

- $\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$
- $\text{Cost}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x))$
- If $y = 1$: $\text{Cost}(h_\theta(x), y) = -\log(h_\theta(x))$
- If $y = 0$: $\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x))$



Cost function

$$\begin{aligned} J(\theta) &= \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_\theta(x^{(i)}), y^{(i)}) \\ &= -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)})) \right] \end{aligned}$$

To fit parameters θ : [Apply Gradient Descent Algorithm](#)

$$\min_{\theta} J(\theta)$$

To make a prediction given new x :

$$\text{Output } h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

Gradient descent

$$J(\theta)$$

$$= -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

Goal: $\min_{\theta} J(\theta)$

Good news: Convex function!
Bad news: No analytical solution

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

(Simultaneously update all θ_i)

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Gradient descent for **Linear Regression**

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$h_\theta(x) = \theta^\top x$$

}

Gradient descent for **Logistic Regression**

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}$$

}

Regularization

- Without regularization

$$w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))$$

- With L2 regularization

$$w_i \leftarrow w_i - \eta \lambda w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))$$

- λ is called a "regularization" term
 - helps reduce overfitting
 - keep weights nearer to zero
 - used very frequently in Logistic Regression

Logistic regression more generally

- Logistic regression when Y not boolean (but still discrete-valued).
- Now $y \in \{y_1 \dots y_R\}$: learn $R-1$ sets of weights

for $k < R$ $P(Y = y_k | X) = \frac{\exp(w_{k0} + \sum_{i=1}^n w_{ki}X_i)}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^n w_{ji}X_i)}$

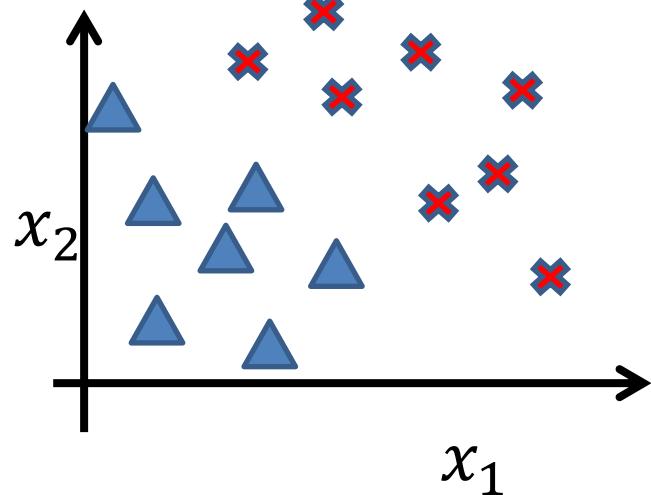
for $k = R$ $P(Y = y_R | X) = \frac{1}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^n w_{ji}X_i)}$

Multi-class classification

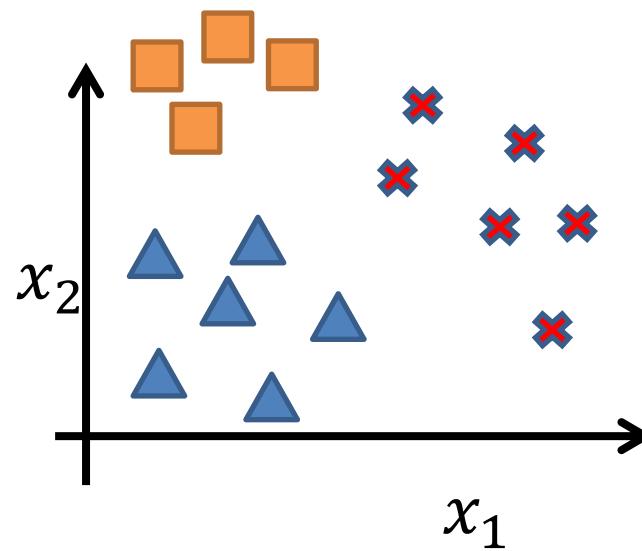
- Email foldering/tagging: Work, Friends, Family, Hobby
- Medical diagrams: Not ill, Cold, Flu
- Weather: Sunny, Cloudy, Rain, Snow

Multi-class classification

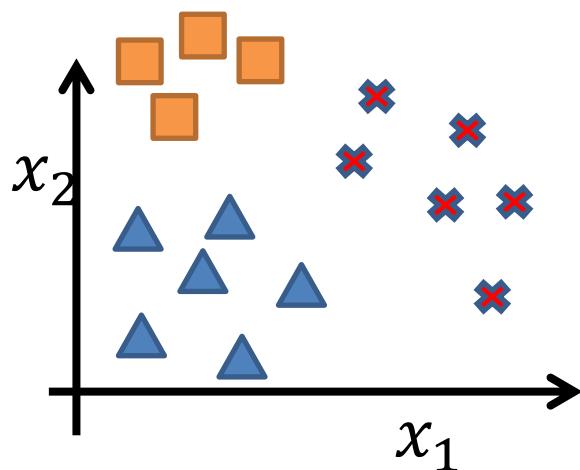
Binary classification



Multiclass classification

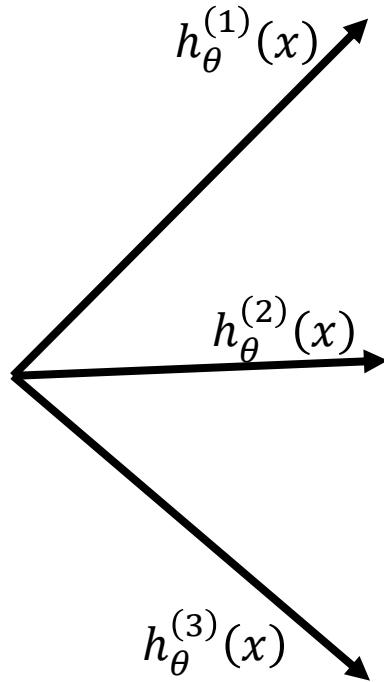


One-vs-all (one-vs-rest)

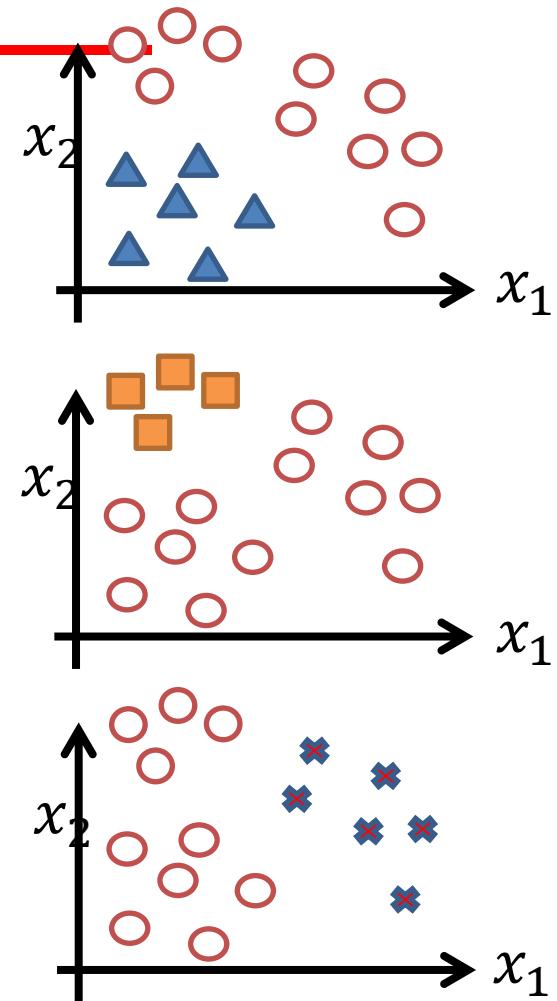


Class 1: 
 Class 2: 
 Class 3: 

$$h_{\theta}^{(i)}(x) = P(y = i|x; \theta) \quad (i = 1, 2, 3)$$



A diagram illustrating the one-vs-all strategy. Three parallel decision boundaries are shown originating from a common point on the left. The top boundary is labeled $h_{\theta}^{(1)}(x)$, the middle boundary is labeled $h_{\theta}^{(2)}(x)$, and the bottom boundary is labeled $h_{\theta}^{(3)}(x)$.



One-vs-all

- Train a logistic regression classifier $h_{\theta}^{(i)}(x)$ for each class i to predict the probability that $y = i$
- Given a new input x , pick the class i that maximizes

$$\max_i h_{\theta}^{(i)}(x)$$

Logistic regression (Classification)

- **Model**

$$h_{\theta}(x) = P(Y = 1|X_1, X_2, \dots, X_n) = \frac{1}{1+e^{-\theta^T x}}$$

- **Cost function**

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)}) \quad \text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$

- **Learning**

Gradient descent: Repeat $\{\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}\}$

- **Inference**

$$\hat{Y} = h_{\theta}(x^{\text{test}}) = \frac{1}{1 + e^{-\theta^T x^{\text{test}}}}$$

How does logistic regression handle missing values?



- Replace missing values with column averages (i.e. replace missing values in feature 1 with the average for feature 1).
 - Replace missing values with column medians.
 - Impute missing values using the other features.
 - Remove records that are missing features.
 - Use a machine learning technique that uses classification trees, e.g. Decision tree
-

Logistic Regression Applications

- **Credit Card Fraud** : Predicting if a given credit card transaction is fraud or not
- **Health** : Predicting if a given mass of tissue is benign or malignant
- **Marketing** : Predicting if a given user will buy an insurance product or not
- **Banking** : Predicting if a customer will default on a loan.

Class Imbalance Problem

- Find needle in haystack
- Lots of classification problems where the classes are skewed (more records from one class than another)
 - Credit card fraud
 - Intrusion detection
 - Defective products in manufacturing assembly line

Approaches to solve Class imbalance problem

- Up-sample minority class
 - randomly duplicating observations from a minority class
- Down-sample majority class
 - removing random observations.
- Generate Synthetic Samples
 - new samples based on the distances between the point and its nearest neighbors
- Change the performance metric
 - Use Recall, Precision or ROC curves instead of accuracy
- Try different algorithms
 - Some algorithms as Support Vector Machines and Tree-Based algorithms are better to work with imbalanced classes.

References

- Tom M. Mitchell
Generative and discriminative classifiers: Naïve Bayes and Logistic Regression
<http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf>
 - Andrew Ng, Michael Jordan
On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes
<http://papers.nips.cc/paper/2020-on-discriminative-vs-generative-classifiers-a-comparison-of-logistic-regression-and-naive-bayes.pdf>
-

References

- <http://www.cs.cmu.edu/~tom/NewChapters.html>
- <http://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf>
- https://medium.com/@sangha_deb/naive-bayes-vs-logistic-regression-a319b07a5d4c
- <https://www.youtube.com/watch?v=-la3q9d7AKQ>
- <http://www.datasciencesmachinelearning.com/2018/11/handling-outliers-in-python.html>

Interpretability

- <https://christophm.github.io/interpretable-ml-book/logistic.html>



BITS Pilani
Pilani Campus

Machine Learning

ZG565

Dr. Sugata Ghosal
sugata.ghosal@pilani.bits-pilani.ac.in



Lecture No. – 5 | Discriminative Classifiers

Date – 10/12/2022

Time – 4:15 PM to 6:15 PM

Session Content

- Linear Regression
 - Overfitting, Regularization, Performance ..
- Types of Classifiers
- Logistic regression
 - Logistic function
 - Log-loss function and gradient descent
 - Regularization
 - Multi-class classification

Classification

- Given a collection of records (training set)
 - Each record is characterized by a tuple (x, y) , where x is the attribute (feature) set and y is the class label
 - x aka attribute, predictor, independent variable, input
 - Y aka class, response, dependent variable, output
- Task
 - Learn a model or function that maps each attribute set x into one of the predefined class labels y

Task	Attribute set, x	Class label, y
Categorizing email messages	Features extracted from email message header and content	spam or non-spam
Identifying tumor cells	Features extracted from x-rays or MRI scans	malignant or benign cells
Cataloging galaxies	Features extracted from telescope images	Elliptical, spiral, or irregular-shaped galaxies

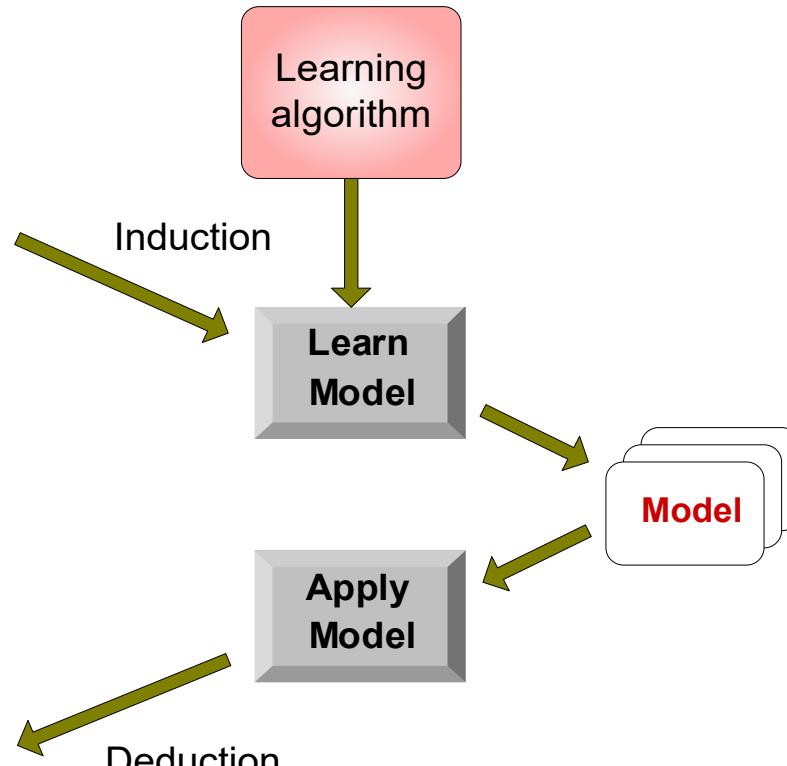
General Approach for Building Classification Model

<i>Tid</i>	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

<i>Tid</i>	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Types of Classifiers

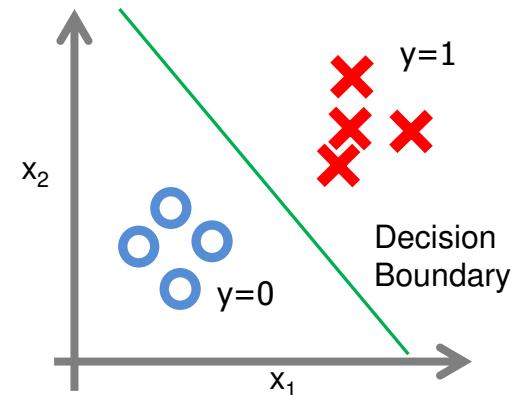
Linear Classifier

- Classes are separated by a linear decision surface (e.g., straight line in 2-dimensional feature/attribute space)
 - If for a given record, linear combination of features x_i is ≥ 0 , i.e.,

$$w_0 + \sum_i w_i x_i \geq 0$$

it belongs to one class (say, $y = 1$), else it belongs to the other class (say, $y=0$ or -1)

- w_i s are learned during the training (induction) phase of the classifier.
- Learnt w_i s are applied to a test record during the deduction / inferencing phase.
- In nonlinear classification, classes are separated by a non-linear surface



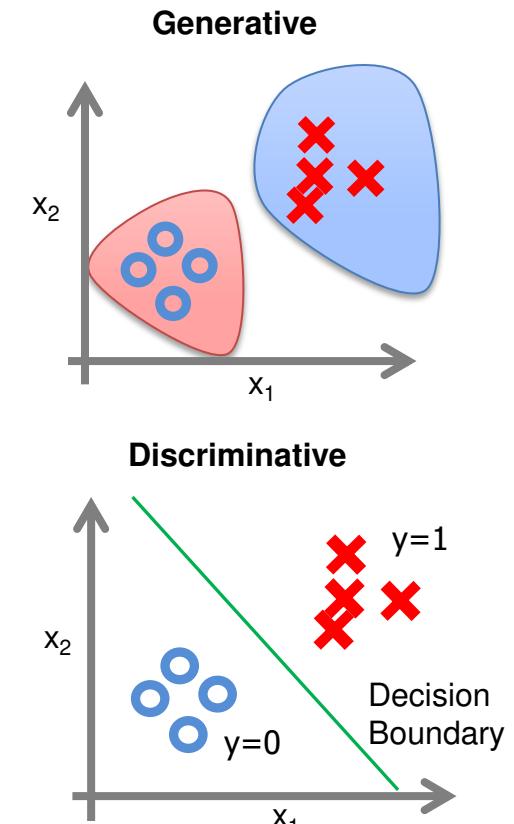
Generative Vs. Discriminative Models

- Generative Model

- Class-conditional probability distribution of attribute/feature set and prior probability of classes are learnt during the training phase
- Given these learnt probabilities, during inferencing phase, probability of a test record belonging to different classes are calculated and compared.
- Can result in linear or nonlinear decision surface

- Discriminative Model

- Given a training set, a function f is learnt that directly maps an attribute/feature vector x to the output class ($y=1$ or $0/-1$)
- A linear function f results in linear decision surface



Logistic Regression

Logistic Regression

Idea:

- Given data X and associated binary (0/1) class label Y , Logistic Regression tries to learn a discriminant function $P(Y|X)$
 - If $Y = 1$, $P(Y|X) = 1$ else $P(Y|X) = 0$

Logistic Regression versus linear regression



- **Linear Regression** could help us predict the student's test score on a scale of 0 - 100. Linear regression predictions are continuous (numbers in a range).
 - **Logistic Regression** could help use predict whether the student passed or failed. Logistic regression predictions are discrete (only specific values or categories are allowed). We can also view probability scores underlying the model's classifications.
-

Linear Regression versus Logistic Regression

Classification requires discrete values:

$$y = 0 \text{ or } 1$$

For linear Regression output values:

$h_{\theta}(x)$ can be much > 1 or much < 0

Logistic Regression: $0 \leq h_{\theta}(x) \leq 1$

Sigmoid/Logistic Function

- Sigmoid/logistic function takes a real value as input and outputs another value between 0 and 1
- That framework is called logistic regression
 - Logistic: A special mathematical sigmoid function it uses
 - Regression: Combines a weight vector with observations to create an answer

$$h_{\theta}(x) = g(\theta^T x)$$

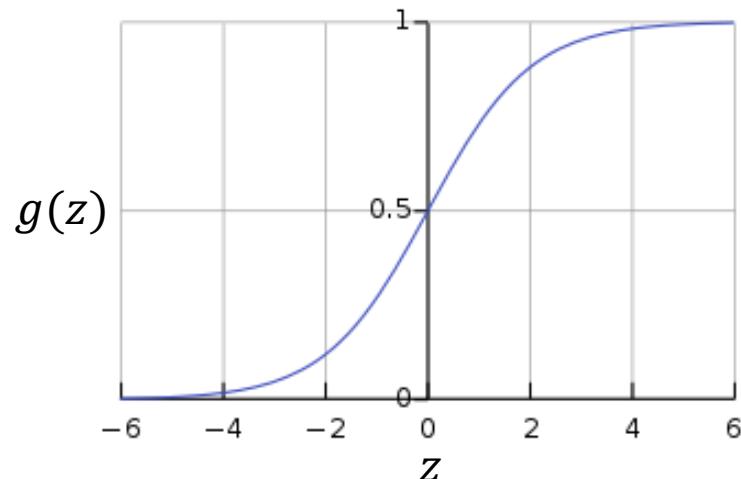
Hypothesis representation

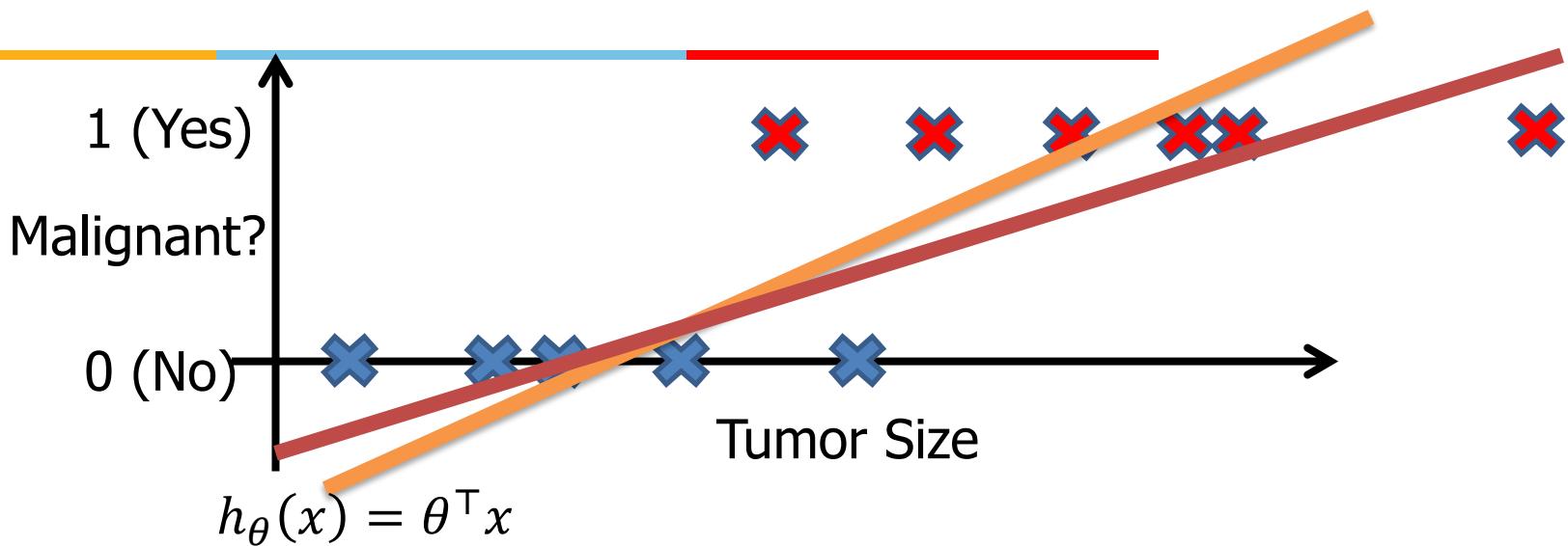
- Want $0 \leq h_\theta(x) \leq 1$
- $h_\theta(x) = g(\theta^\top x)$,

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

- Sigmoid function
- Logistic function

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}$$





- Threshold classifier output $h_\theta(x)$ at 0.5
 - If $h_\theta(x) \geq 0.5$, predict “ $y = 1$ ”
 - If $h_\theta(x) < 0.5$, predict “ $y = 0$ ”

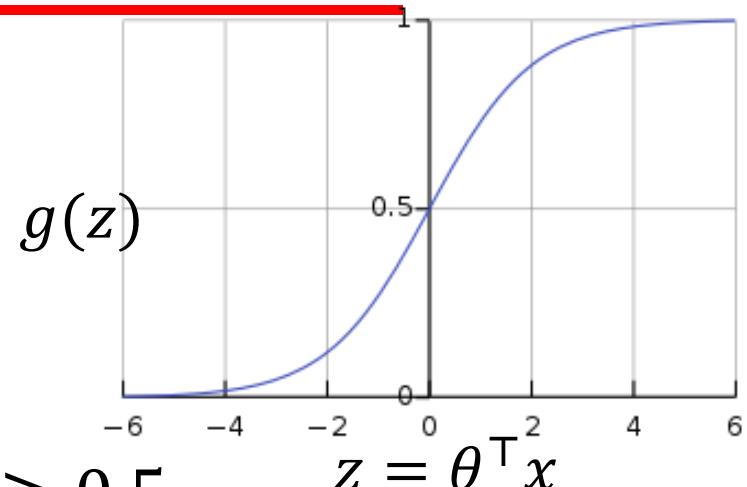
Interpretation of hypothesis output

- $h_{\theta}(x)$ = estimated probability that $y = 1$ on input x
- Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$
- $h_{\theta}(x) = 0.7$
- Tell patient that 70% chance of tumor being malignant

Logistic regression

$$h_{\theta}(x) = g(\theta^T x)$$

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



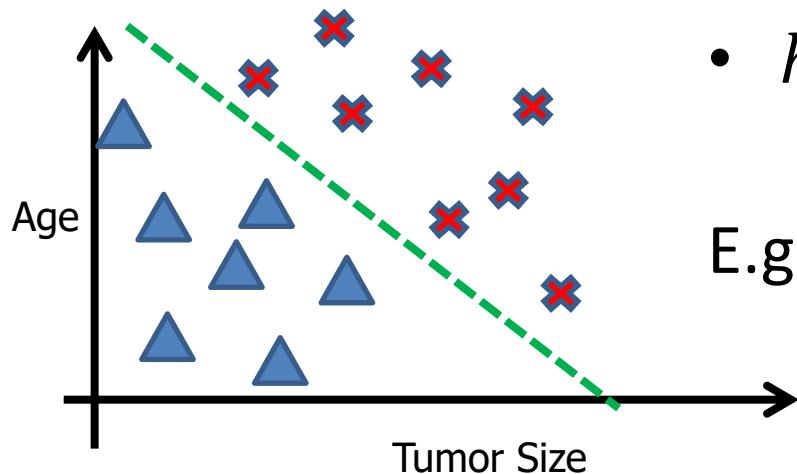
Suppose predict “y = 1” if $h_{\theta}(x) \geq 0.5$

$$z = \theta^T x \geq 0$$

predict “y = 0” if $h_{\theta}(x) < 0.5$

$$z = \theta^T x < 0$$

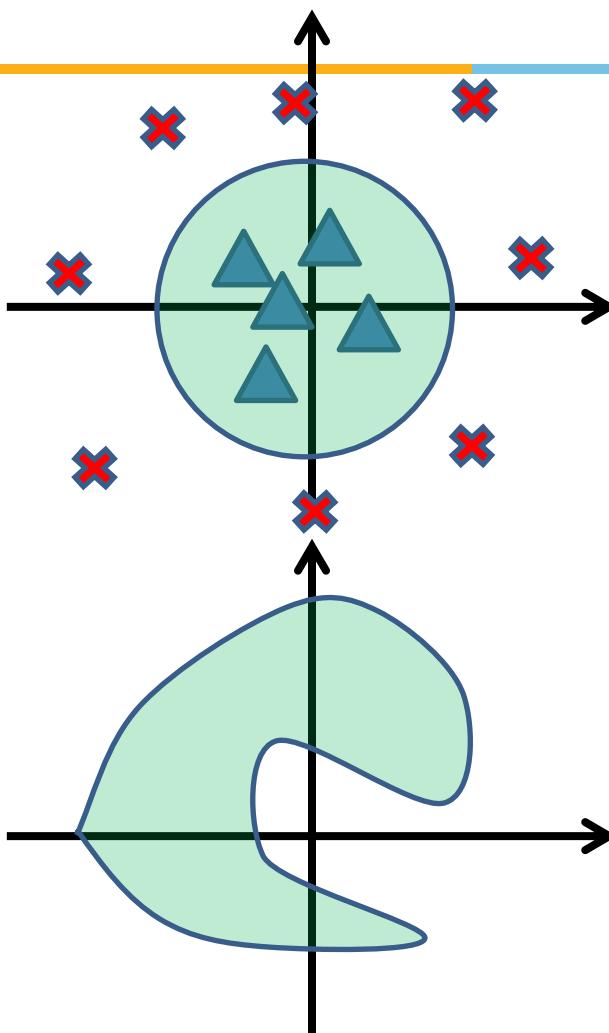
Decision boundary



- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

E.g., $\theta_0 = -3, \theta_1 = 1, \theta_2 = 1$

- Predict " $y = 1$ " if $-3 + x_1 + x_2 \geq 0$



- $$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

E.g., $\theta_0 = -1, \theta_1 = 0, \theta_2 = 0, \theta_3 = 1, \theta_4 = 1$
- Predict “ $y = 1$ ” if $-1 + x_1^2 + x_2^2 \geq 0$
- $$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots)$$

Learning model parameters

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

m examples

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \quad x_0 = 1, y \in \{0, 1\}$$

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

How to choose parameters
(feature weights) θ ?

Cost function for Linear Regression

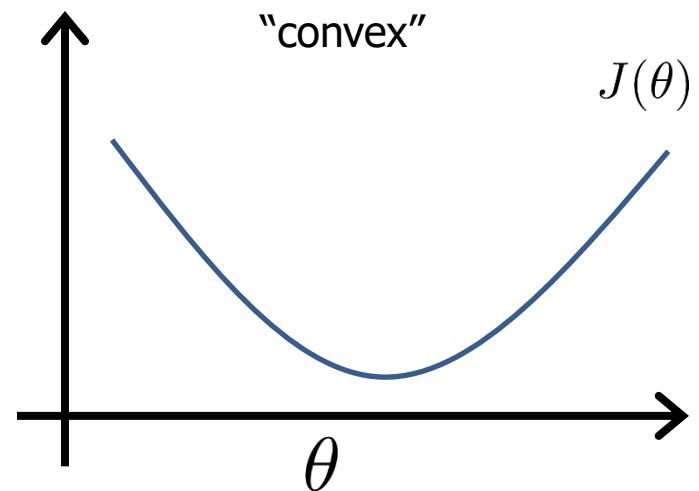
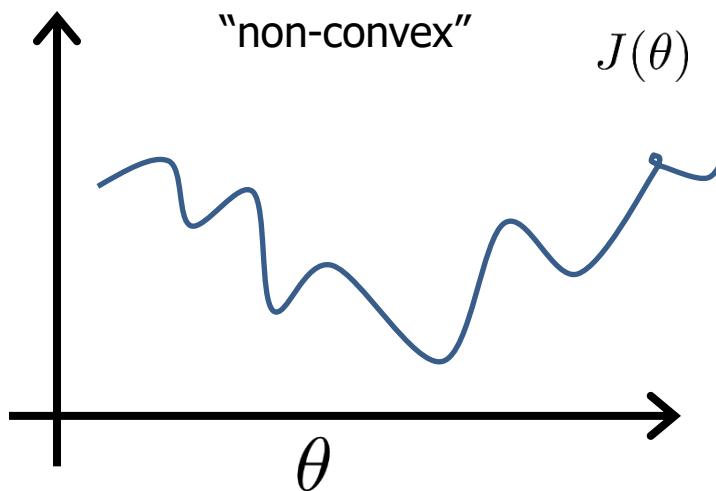
$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_\theta(x^{(i)}), y)$$

$$\text{Cost}(h_\theta(x), y) = \frac{1}{2} (h_\theta(x) - y)^2$$

MSE Cost Function

Linear regression: $J(\theta) = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$

$$\text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$$

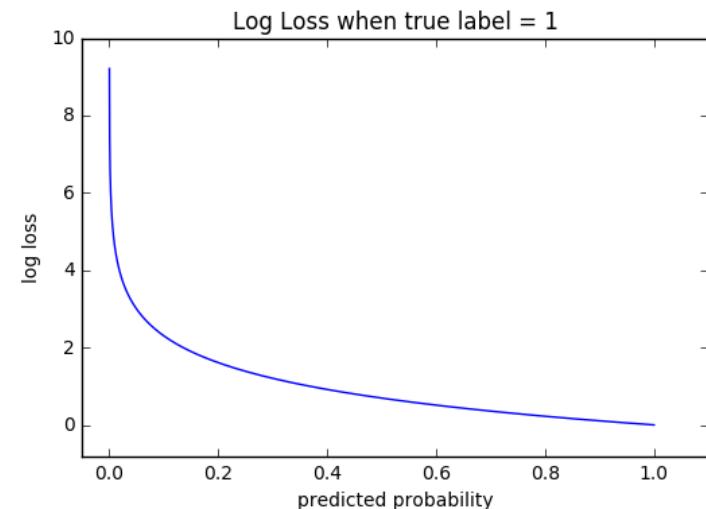


Error (Cost) Function

- Our prediction function is non-linear (due to sigmoid transform)
 - Squaring this prediction as we do in MSE results in a non-convex function with many local minima.
 - If our cost function has many local minimums, gradient descent may not find the optimal global minimum.
 - So instead of Mean Squared Error, we use a error/cost function called Cross-Entropy, also known as Log Loss.
-

Cross Entropy

- Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1.
- Cross-entropy loss increases as the predicted probability diverges from the actual label.
- So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value.
- A perfect model would have a log loss of 0.
- Cross-entropy loss can be divided into two separate cost functions:
one for $y=1$ and
one for $y=0$.



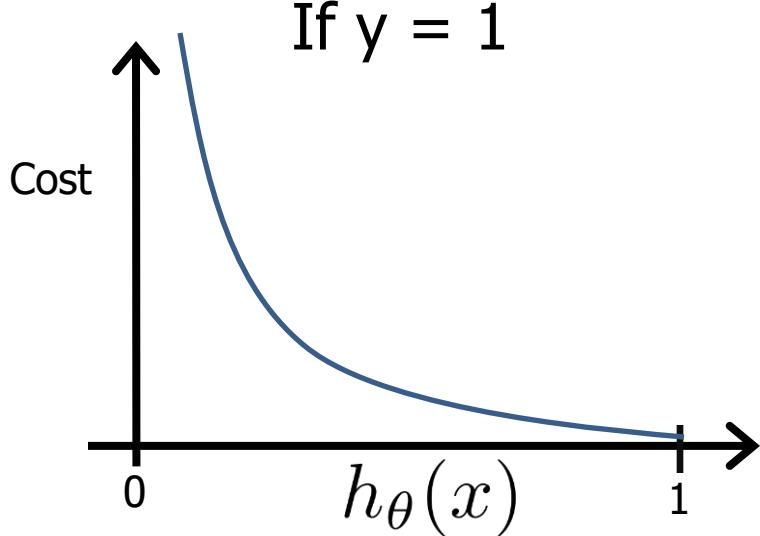
Logistic regression cost function (cross entropy)

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

If $y = 1$

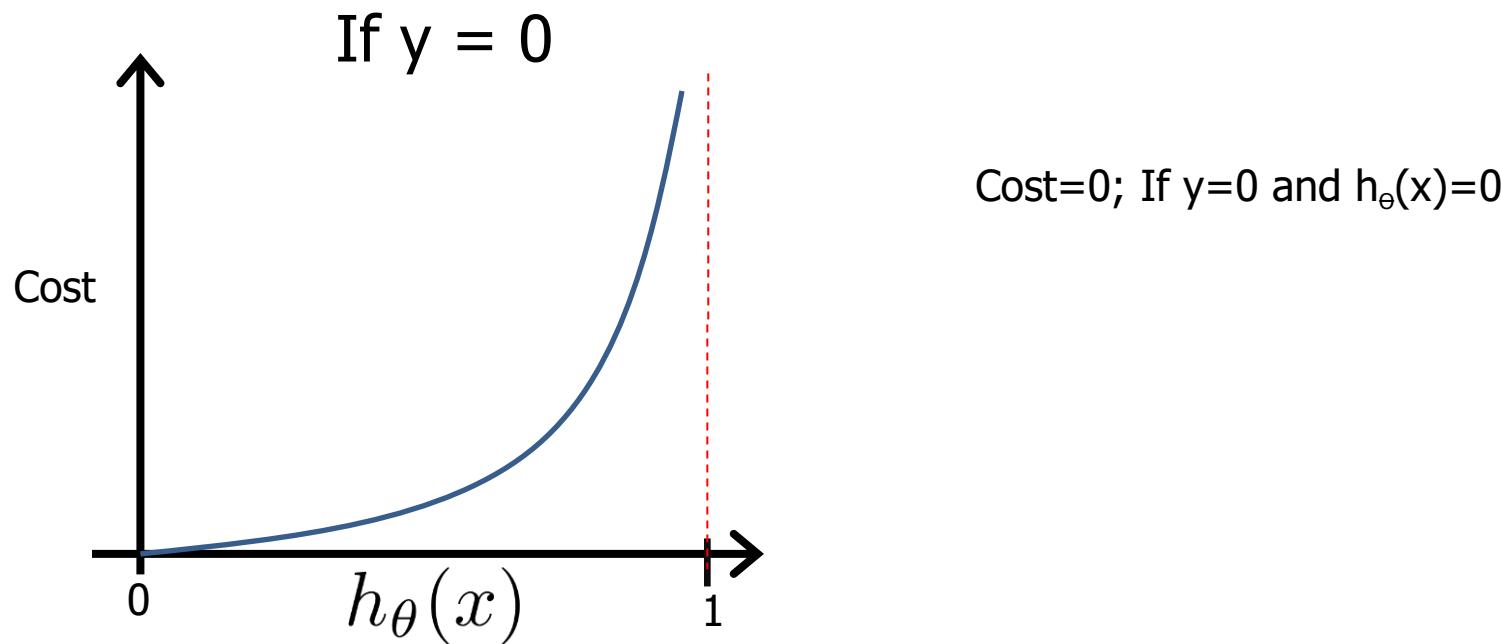
$\text{Cost} = 0$ if $y = 1, h_\theta(x) = 1$
 But as $h_\theta(x) \rightarrow 0$
 $\text{Cost} \rightarrow \infty$

Captures intuition that if $h_\theta(x) = 0$,
 (predict $P(y = 1|x; \theta) = 0$), but $y = 1$,
 we'll penalize learning algorithm by a very
 large cost.



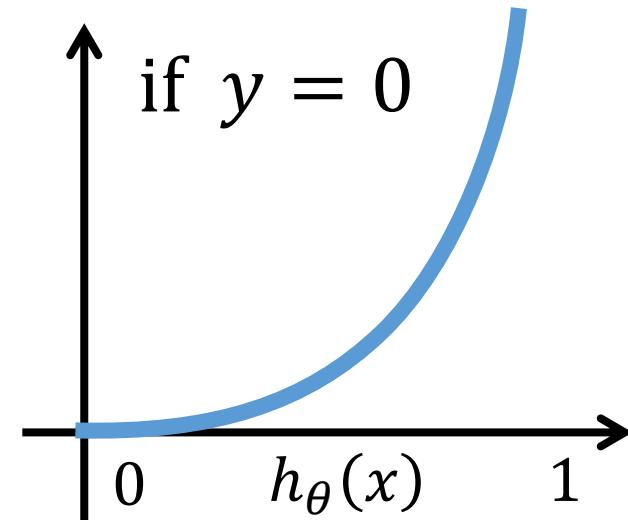
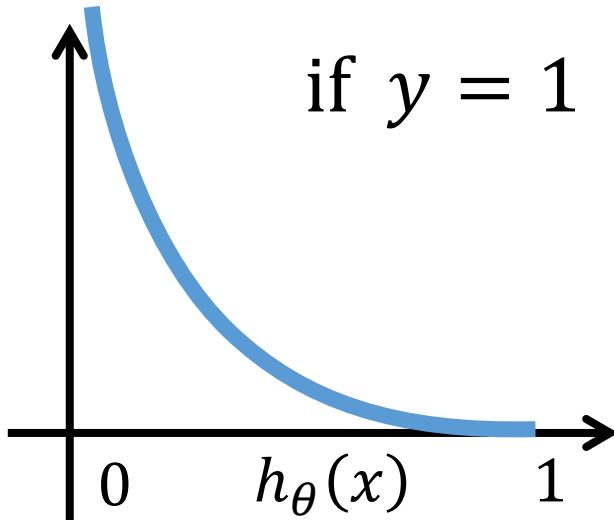
Logistic regression cost function

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



Cost function for Logistic Regression

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



Logistic regression cost function

- $\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$
- $\text{Cost}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x))$
- If $y = 1$: $\text{Cost}(h_\theta(x), y) = -\log(h_\theta(x))$
- If $y = 0$: $\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x))$



Cost function

$$\begin{aligned} J(\theta) &= \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_\theta(x^{(i)}), y^{(i)}) \\ &= -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)})) \right] \end{aligned}$$

To fit parameters θ : [Apply Gradient Descent Algorithm](#)

$$\min_{\theta} J(\theta)$$

To make a prediction given new x :

$$\text{Output } h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

Gradient descent

$$J(\theta)$$

$$= -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

Goal: $\min_{\theta} J(\theta)$

Good news: Convex function!
Bad news: No analytical solution

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

(Simultaneously update all θ_i)

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Gradient descent for **Linear Regression**

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$h_\theta(x) = \theta^\top x$$

}

Gradient descent for **Logistic Regression**

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}$$

}

Regularization

- Without regularization

$$w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))$$

- With L2 regularization

$$w_i \leftarrow w_i - \eta \lambda w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))$$

- λ is called a "regularization" term
 - helps reduce overfitting
 - keep weights nearer to zero
 - used very frequently in Logistic Regression

Logistic regression more generally

- Logistic regression when Y not boolean (but still discrete-valued).
- Now $y \in \{y_1 \dots y_R\}$: learn $R-1$ sets of weights

for $k < R$ $P(Y = y_k | X) = \frac{\exp(w_{k0} + \sum_{i=1}^n w_{ki}X_i)}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^n w_{ji}X_i)}$

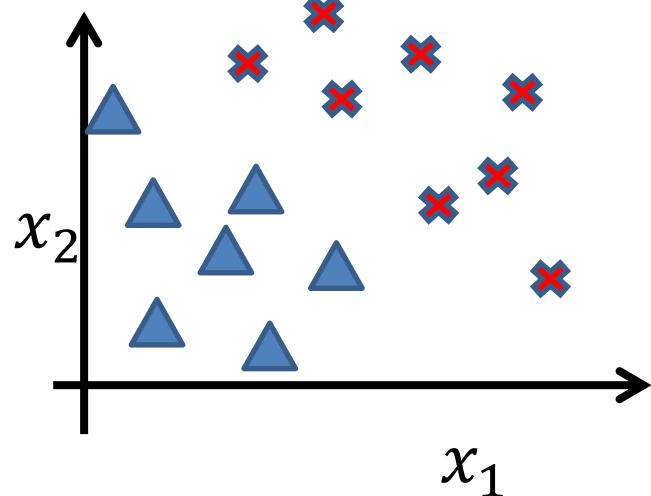
for $k=R$ $P(Y = y_R | X) = \frac{1}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^n w_{ji}X_i)}$

Multi-class classification

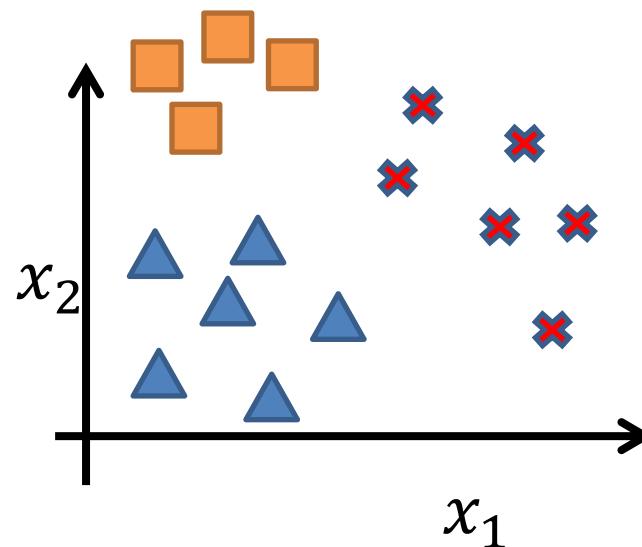
- Email foldering/tagging: Work, Friends, Family, Hobby
- Medical diagrams: Not ill, Cold, Flu
- Weather: Sunny, Cloudy, Rain, Snow

Multi-class classification

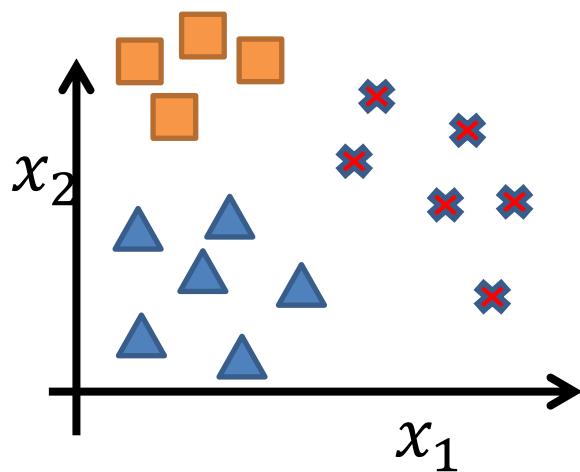
Binary classification



Multiclass classification

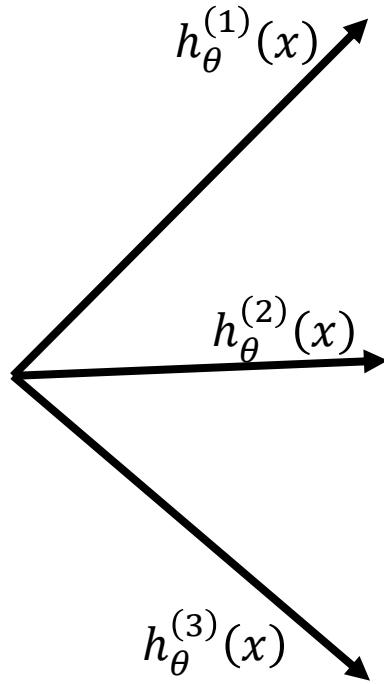


One-vs-all (one-vs-rest)

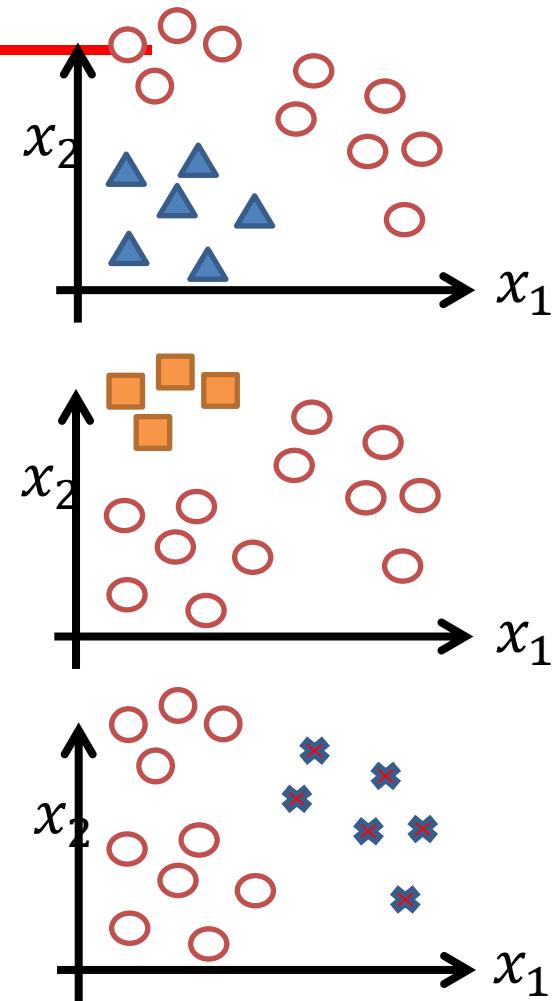


Class 1: 
 Class 2: 
 Class 3: 

$$h_{\theta}^{(i)}(x) = P(y = i|x; \theta) \quad (i = 1, 2, 3)$$



A diagram illustrating the one-vs-all strategy. Three parallel decision boundaries are shown originating from a common point on the left. The top boundary is labeled $h_{\theta}^{(1)}(x)$, the middle boundary is labeled $h_{\theta}^{(2)}(x)$, and the bottom boundary is labeled $h_{\theta}^{(3)}(x)$.



One-vs-all

- Train a logistic regression classifier $h_{\theta}^{(i)}(x)$ for each class i to predict the probability that $y = i$
- Given a new input x , pick the class i that maximizes

$$\max_i h_{\theta}^{(i)}(x)$$

Logistic regression (Classification)

- **Model**

$$h_{\theta}(x) = P(Y = 1|X_1, X_2, \dots, X_n) = \frac{1}{1+e^{-\theta^T x}}$$

- **Cost function**

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)}) \quad \text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$

- **Learning**

Gradient descent: Repeat $\{\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}\}$

- **Inference**

$$\hat{Y} = h_{\theta}(x^{\text{test}}) = \frac{1}{1 + e^{-\theta^T x^{\text{test}}}}$$

How does logistic regression handle missing values?

- Replace missing values with column averages (i.e. replace missing values in feature 1 with the average for feature 1).
- Replace missing values with column medians.
- Impute missing values using the other features.
- Remove records that are missing features.
- Use a machine learning technique that uses classification trees, e.g. Decision tree

Logistic Regression Applications

- **Credit Card Fraud** : Predicting if a given credit card transaction is fraud or not
- **Health** : Predicting if a given mass of tissue is benign or malignant
- **Marketing** : Predicting if a given user will buy an insurance product or not
- **Banking** : Predicting if a customer will default on a loan.

Class Imbalance Problem

- Find needle in haystack
- Lots of classification problems where the classes are skewed (more records from one class than another)
 - Credit card fraud
 - Intrusion detection
 - Defective products in manufacturing assembly line

Approaches to solve Class imbalance problem

- Up-sample minority class
 - randomly duplicating observations from a minority class
- Down-sample majority class
 - removing random observations.
- Generate Synthetic Samples
 - new samples based on the distances between the point and its nearest neighbors
- Change the performance metric
 - Use Recall, Precision or ROC curves instead of accuracy
- Try different algorithms
 - Some algorithms as Support Vector Machines and Tree-Based algorithms are better to work with imbalanced classes.

References

- Tom M. Mitchell
Generative and discriminative classifiers: Naïve Bayes and Logistic Regression
<http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf>
 - Andrew Ng, Michael Jordan
On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes
<http://papers.nips.cc/paper/2020-on-discriminative-vs-generative-classifiers-a-comparison-of-logistic-regression-and-naive-bayes.pdf>
-

References

- <http://www.cs.cmu.edu/~tom/NewChapters.html>
- <http://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf>
- https://medium.com/@sangha_deb/naive-bayes-vs-logistic-regression-a319b07a5d4c
- <https://www.youtube.com/watch?v=-la3q9d7AKQ>
- <http://www.datasciencesmachinelearning.com/2018/11/handling-outliers-in-python.html>

Interpretability

- <https://christophm.github.io/interpretable-ml-book/logistic.html>



BITS Pilani
Pilani Campus

Machine Learning **ZG565**

Dr. Sugata Ghosal

sugata.ghosal@pilani.bits-pilani.ac.in



Lecture No. – 7 | Decision Tree

Date – 17/12/2022

Time: 4:15 PM – 6:15 PM

Session Content

- Decision Tree
- Handling overfitting
- Continuous values
- Missing Values

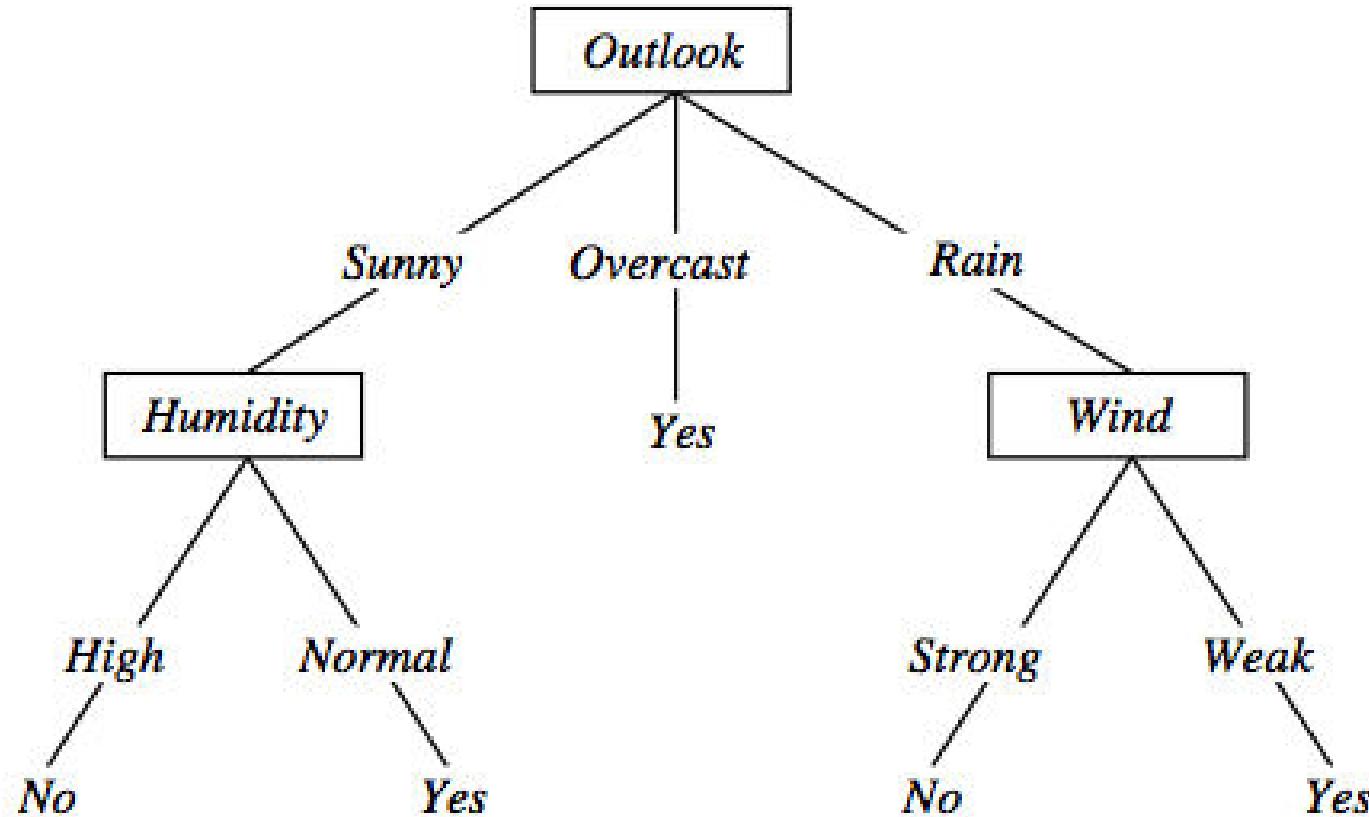
Decision trees

- Decision Trees is one of the most widely used and practical methods of classification
 - Method for approximating discrete-valued functions
 - Learned functions are represented as decision trees (or if-then-else rules)
 - Expressive hypotheses space
-

Decision Tree

- Advantages:
 - Inexpensive to construct
 - Extremely fast at classifying unknown records
 - Easy to interpret for small-sized trees
 - Can easily handle redundant or irrelevant attributes (unless the attributes are interacting)
- Disadvantages:
 - Space of possible decision trees is exponentially large.
Greedy approaches are often unable to find the best tree.
 - Does not take into account interactions between attributes
 - Each decision boundary involves only a single attribute

Decision tree representation (PlayTennis)



$\langle \text{Outlook}=\text{Sunny}, \text{Temp}=\text{Hot}, \text{Humidity}=\text{High}, \text{Wind}=\text{Strong} \rangle \quad \text{No}$

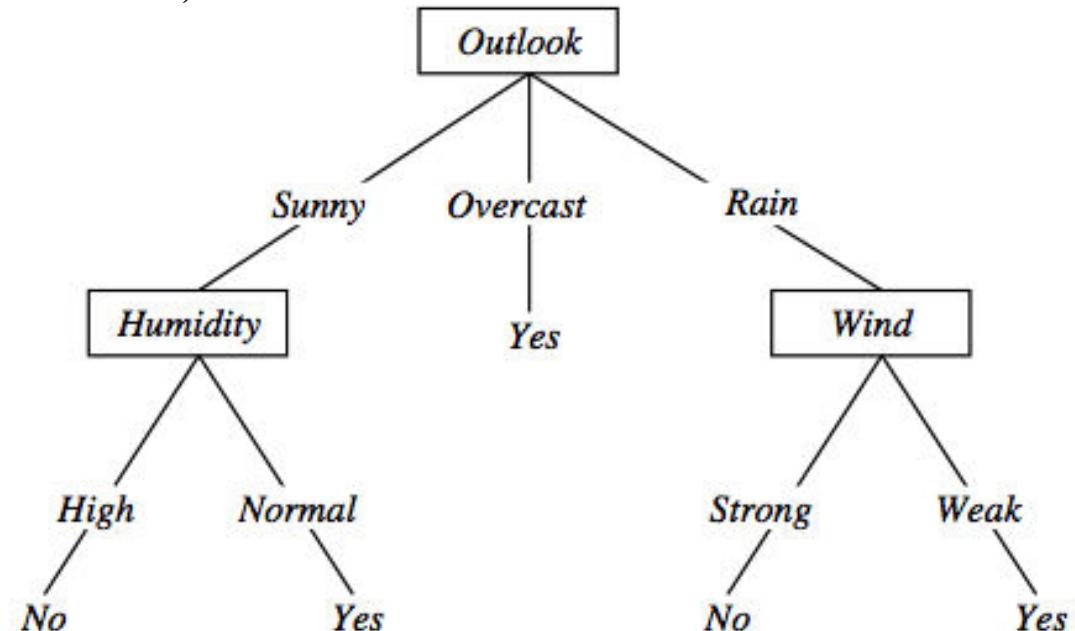
Decision trees expressivity

- Decision trees represent a disjunction of conjunctions on constraints on the value of attributes:

$(Outlook = Sunny \wedge Humidity = Normal) \vee$

$(Outlook = Overcast) \vee$

$(Outlook = Rain \wedge Wind = Weak)$



Measure of Information

- The amount of information (surprise element) conveyed by a message is inversely proportional to its probability of occurrence. That is

$$I_k \propto \frac{1}{p_k}$$

- The mathematical operator satisfies above properties is the logarithmic operator.

$$I_k = \log_r \frac{1}{p_k} \text{ units}$$

Entropy

- Entropy of discrete random variable $X=\{x_1, x_2 \dots x_n\}$
$$H(X) = E[I(X)] = E[-\log(P(X))].$$

; since: $\log_2(1/P(\text{event})) = -\log_2 P(\text{event})$
- As uncertainty increases, entropy increases
- Entropy across all values

$$H(X) = - \sum_{i=1}^n P(x_i) \log_b P(x_i)$$

Entropy in general

- Entropy measures the amount of information in a random variable

$$H(X) = -p_+ \log_2 p_+ - p_- \log_2 p_- \quad X = \{+, -\}$$

for binary classification [two-valued random variable]

$$H(X) = - \sum_{i=1}^c p_i \log_2 p_i = \sum_{i=1}^c p_i \log_2 1/p_i \quad X = \{i, \dots, c\}$$

for classification in c classes

Entropy in binary classification

- Entropy measures the *impurity* of a collection of examples. It depends from the distribution of the random variable p .
 - S is a collection of training examples
 - p_+ the proportion of positive examples in S
 - p_- the proportion of negative examples in S

$$\text{Entropy}(S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_- \quad [0 \log_2 0 = 0]$$

$$\text{Entropy}([14+, 0-]) = -14/14 \log_2 (14/14) - 0 \log_2 (0) = 0$$

$$\text{Entropy}([9+, 5-]) = -9/14 \log_2 (9/14) - 5/14 \log_2 (5/14) = 0.94$$

$$\begin{aligned} \text{Entropy}([7+, 7-]) &= -7/14 \log_2 (7/14) - 7/14 \log_2 (7/14) \\ &= 1/2 + 1/2 = 1 \end{aligned} \quad [\log_2 1/2 = -1]$$

Note: the log of a number < 1 is negative, $0 \leq p \leq 1$, $0 \leq \text{entropy} \leq 1$

- <https://www.easycalculation.com/log-base2-calculator.php>

Information gain as entropy reduction

- *Information gain* is the *expected* reduction in entropy caused by partitioning the examples on an attribute.
- The higher the information gain the more effective the attribute in classifying training data.
- Expected reduction in entropy knowing A

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$Values(A)$ possible values for A

S_v subset of S for which A has value v

Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example: Information gain

- Let

- $Values(Wind) = \{ Weak, Strong \}$
- $S = [9+, 5-]$
- $S_{Weak} = [6+, 2-]$
- $S_{Strong} = [3+, 3-]$

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- Information gain due to knowing Wind:

$$\begin{aligned}
 Gain(S, Wind) &= Entropy(S) - 8/14 \text{ Entropy}(S_{Weak}) - 6/14 \text{ Entropy}(S_{Strong}) \\
 &= 0.94 - 8/14 \times 0.811 - 6/14 \times 1.00 \\
 &= 0.048
 \end{aligned}$$

Example

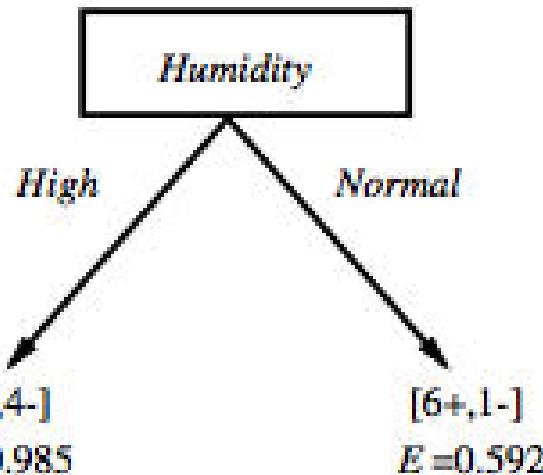
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Which attribute is the best classifier?

Which attribute is the best classifier?

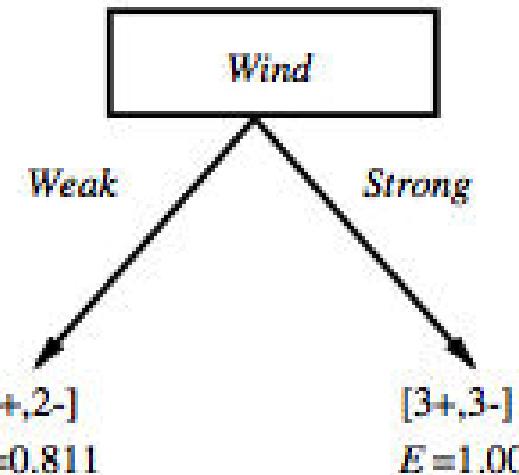
$S: [9+, 5-]$

$E = 0.940$



$S: [9+, 5-]$

$E = 0.940$



Gain (S, Humidity)

$$\begin{aligned}
 &= .940 - (7/14).985 - (7/14).592 \\
 &= .151
 \end{aligned}$$

Gain (S, Wind)

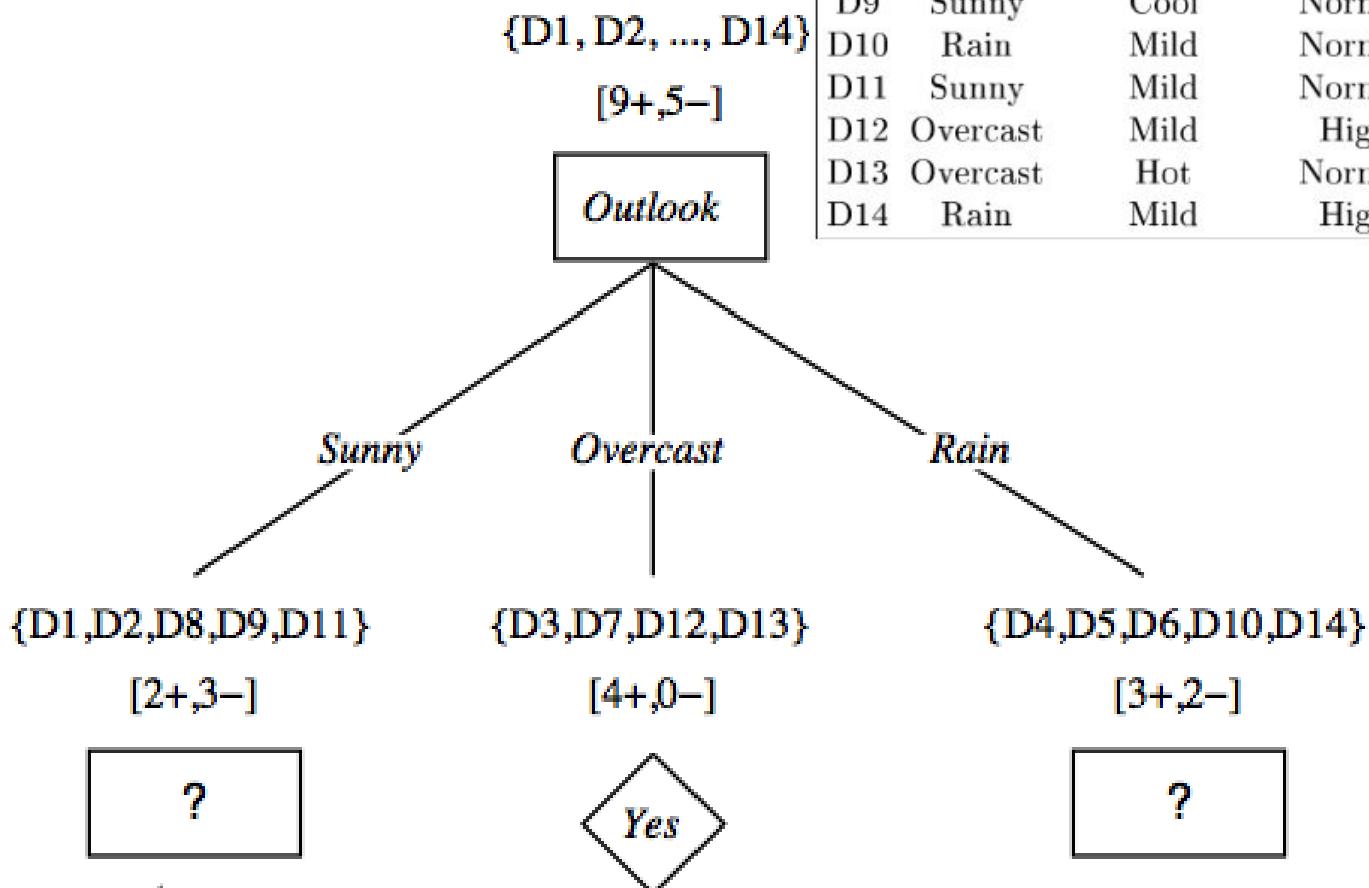
$$\begin{aligned}
 &= .940 - (3/14).811 - (6/14)1.0 \\
 &= .048
 \end{aligned}$$

First step: which attribute to test at the root?

- Which attribute should be tested at the root?
 - $Gain(S, Outlook) = 0.246$
 - $Gain(S, Humidity) = 0.151$
 - $Gain(S, Wind) = 0.084$
 - $Gain(S, Temperature) = 0.029$
- *Outlook* provides the best prediction for the target
- Lets grow the tree:
 - add to the tree a successor for each possible value of *Outlook*
 - partition the training samples according to the value of *Outlook*

After first step

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



Second step

- Working on *Outlook=Sunny* node:

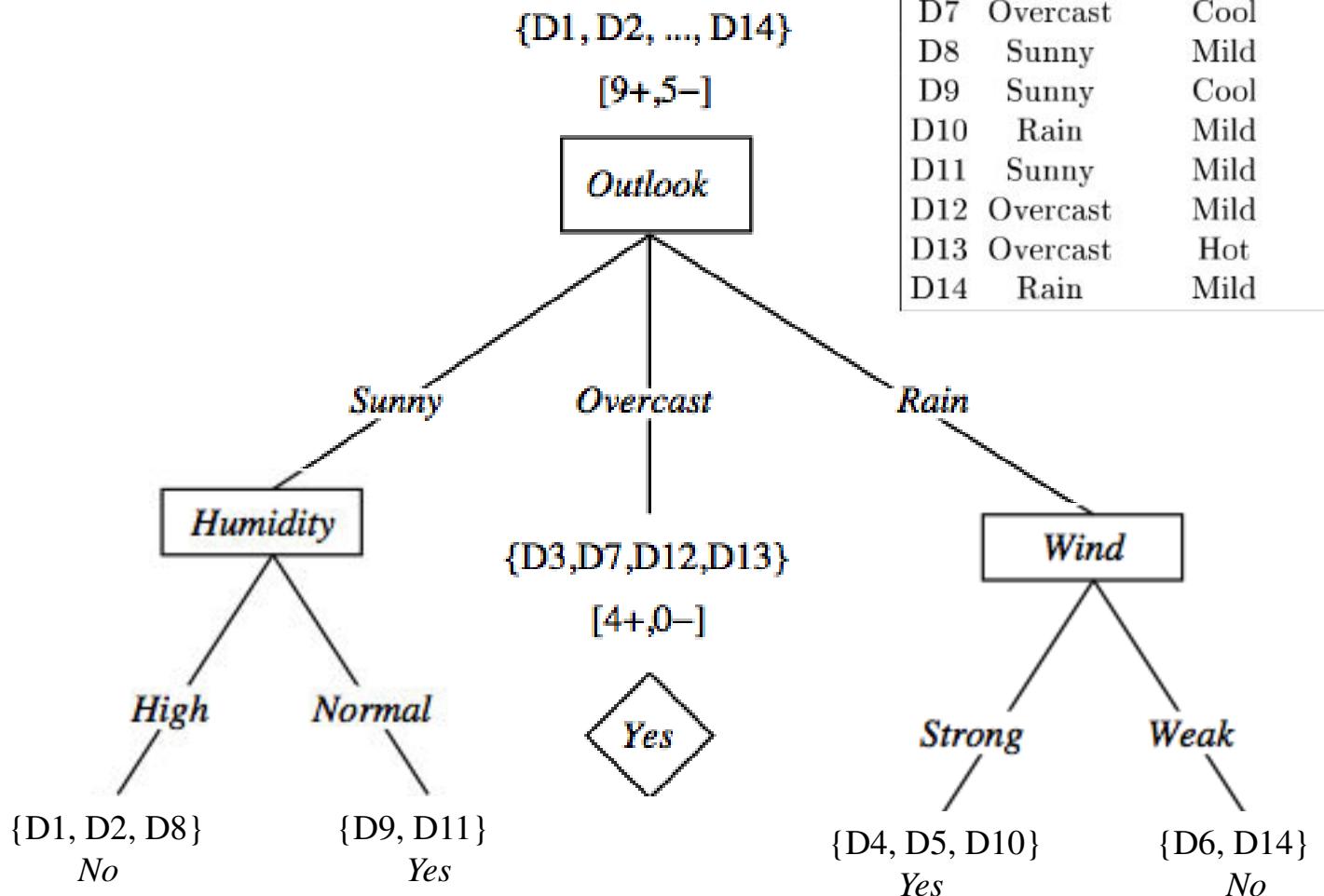
$$Gain(S_{Sunny}, \text{Humidity}) = 0.970 - 3/5 \times 0.0 - 2/5 \times 0.0 = 0.970$$

$$Gain(S_{Sunny}, \text{Wind}) = 0.970 - 2/5 \times 1.0 - 3.5 \times 0.918 = 0.019$$

$$Gain(S_{Sunny}, \text{Temp.}) = 0.970 - 2/5 \times 0.0 - 2/5 \times 1.0 - 1/5 \times 0.0 = 0.570$$

- *Humidity* provides the best prediction for the target
- Lets grow the tree:
 - add to the tree a successor for each possible value of *Humidity*
 - partition the training samples according to the value of *Humidity*

Second and third steps



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

ID3: algorithm

$\text{ID3}(X, T, \text{Attrs})$

X : training examples:

T : target attribute (e.g. *PlayTennis*),

Attrs : other attributes, initially all attributes

Create *Root* node

If all X 's are +, *return* *Root* with class +

If all X 's are -, *return* *Root* with class -

If Attrs is empty *return* *Root* with class most common value of T in X

else

$A \leftarrow$ best attribute; decision attribute for *Root* $\leftarrow A$

For each possible value v_i of A :

- add a new branch below *Root*, for test $A = v_i$

- $X_i \leftarrow$ subset of X with $A = v_i$

- *If* X_i is empty *then* add a new leaf with class the most common value of T in X

- *else* add the subtree generated by $\text{ID3}(X_i, T, \text{Attrs} - \{A\})$

return *Root*

Prefer shorter hypotheses: Occam's razor

- Why prefer shorter hypotheses?
- Arguments in favor:
 - There are fewer short hypotheses than long ones
 - If a short hypothesis fits data unlikely to be a coincidence
 - Elegance and aesthetics
- Arguments against:
 - Not every short hypothesis is a reasonable one.
- Occam's razor says that when presented with competing hypotheses that make the same predictions, one should select the solution which is simple"

Issues in decision trees learning

- Overfitting
 - Reduced error pruning
 - Rule post-pruning
- Extensions
 - Continuous valued attributes
 - Handling training examples with missing attribute values

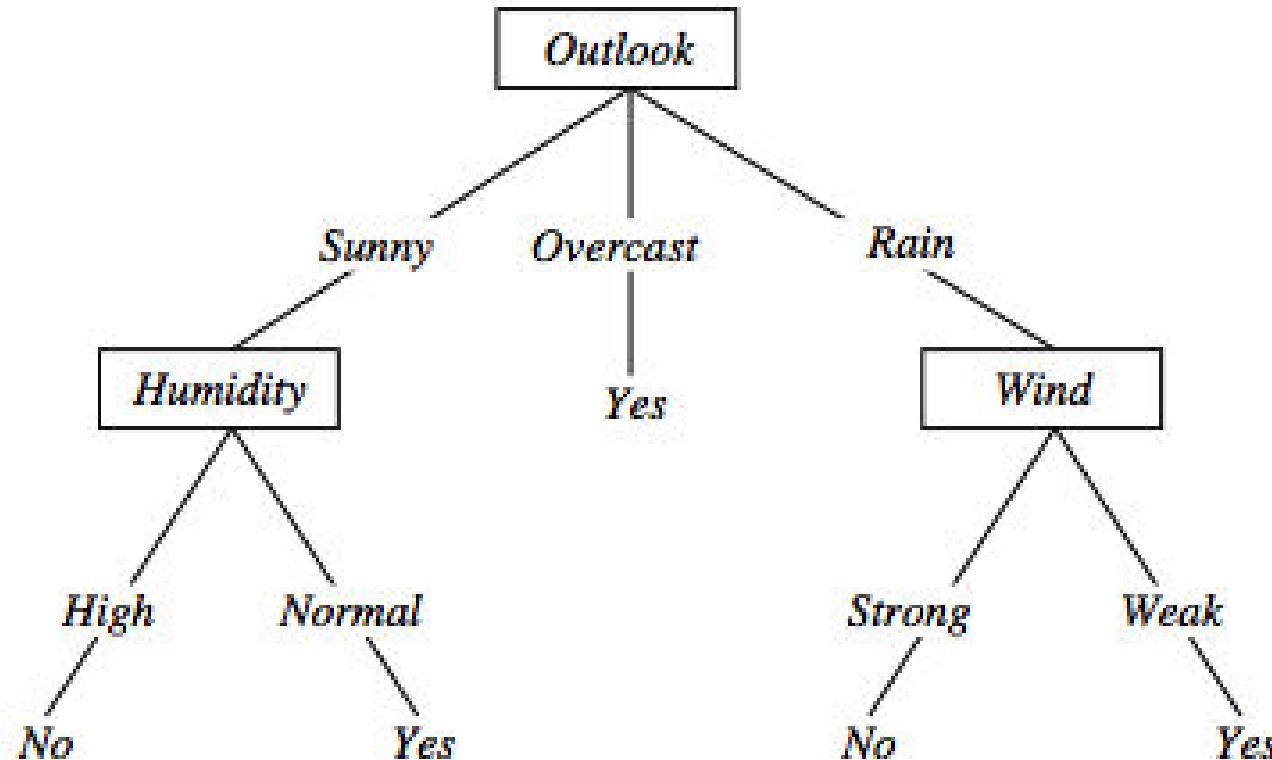
Overfitting: definition

- **overfitting** is "the production of an analysis that corresponds too closely or exactly to a particular set of data"
- Building trees that “adapt too much” to the training examples may lead to “overfitting”.
- May therefore fail to fit additional data or predict future observations reliably
- **overfitted model** is a statistical model that contains more parameters than can be justified by the data

Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No
D15	Sunny	Hot	Normal	Strong	No

Overfitting in decision trees



$\langle Outlook=Sunny, Temp=Hot, Humidity=Normal, Wind=Strong, PlayTennis=No \rangle$

New noisy example causes splitting of second leaf node.

Avoid overfitting in Decision Trees

- Two strategies:
 1. Stop growing the tree earlier than perfect classification
 2. Allow the tree to *overfit* the data, and then *post-prune* the tree
- Training and validation set
 - split the training in two parts (training and validation) and use validation to assess the utility of *post-pruning*
 - *Reduced error pruning*
 - *Rule post pruning*

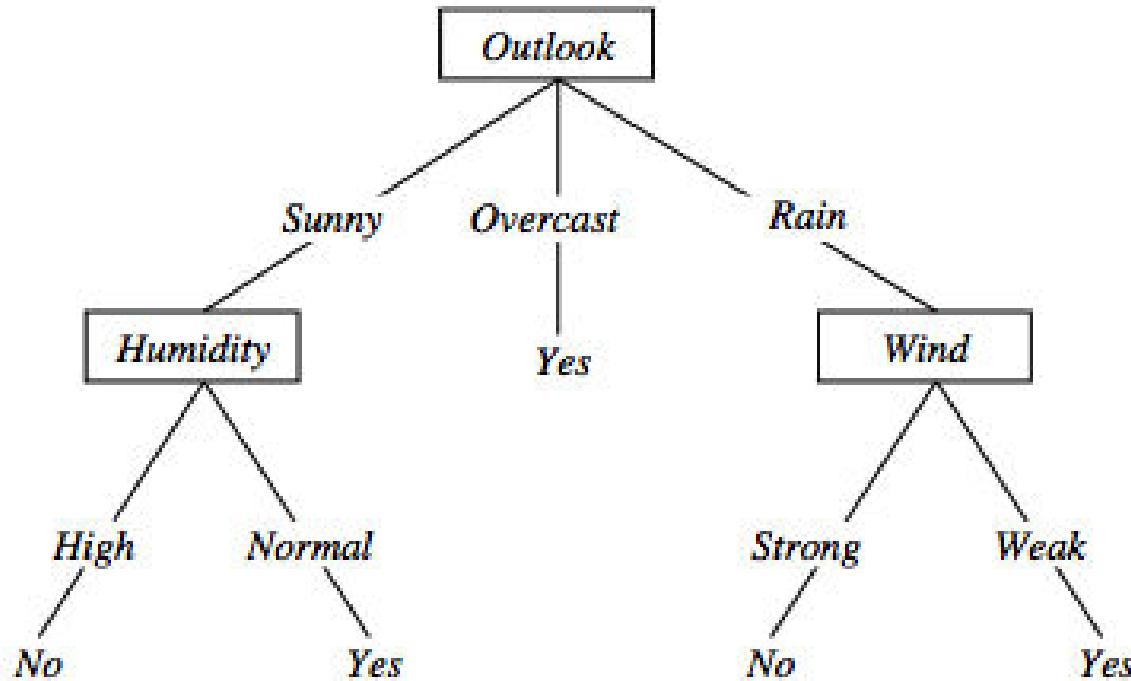
Reduced-error pruning

- Each node is a candidate for pruning
- *Pruning* consists in removing a subtree rooted in a node: the node becomes a leaf and is assigned the most common classification
- Nodes are removed only if the resulting tree performs no worse **on the validation set**.
- Nodes are pruned iteratively: at each iteration the node whose removal most increases accuracy on the validation set is pruned.
- Pruning stops when no pruning increases accuracy

Rule post-pruning

1. Create the decision tree from the training set
 2. Convert the tree into an equivalent set of rules
 - Each path corresponds to a rule
 - Each node along a path corresponds to a pre-condition
 - Each leaf classification to the post-condition
 3. Prune (generalize) each rule by removing those preconditions whose removal improves accuracy ...
 - ... over validation set
 4. Sort the rules in estimated order of accuracy, and consider them in sequence when classifying new instances
-

Converting to rules



$$(Outlook=Sunny) \wedge (Humidity=High) \Rightarrow (PlayTennis=No)$$

Rule Post-Pruning

- Convert tree to rules (one for each path from root to a leaf)
- For each antecedent in a rule, remove it if error rate on validation set does not decrease
- Sort final rule set by accuracy

Outlook=sunny ^ humidity=high -> No
Outlook=sunny ^ humidity=normal -> Yes
Outlook=overcast -> Yes
Outlook=rain ^ wind=strong -> No
Outlook=rain ^ wind=weak -> Yes

Compare first rule to:

Outlook=sunny->No
Humidity=high->No

Calculate accuracy of 3 rules based on validation set and pick best version.

Why converting to rules?

- Each distinct path produces a different rule: a condition removal may be based on a local (contextual) criterion. Node pruning is global and affects all the rules
 - Provides flexibility of not removing entire node
 - In rule form, tests are not ordered and there is no book-keeping involved when conditions (nodes) are removed
 - Converting to rules improves readability for humans
-

Dealing with continuous-valued attributes

- Given a continuous-valued attribute A , dynamically create a new attribute A_c
$$A_c = \text{True if } A < c, \text{ False otherwise}$$
- How to determine threshold value c ?
- Example. *Temperature* in the *PlayTennis* example
 - Sort the examples according to *Temperature*

<i>Temperature</i>	40	48		60	72	80		90
<i>PlayTennis</i>	No	No	54	Yes	Yes	Yes	85	No
 - Determine candidate thresholds by averaging consecutive values where there is a change in classification: $(48+60)/2=54$ and $(80+90)/2=85$

Problems with information gain

- Natural bias of information gain: it favors attributes with many possible values.
- Consider the attribute *Date* in the *PlayTennis* example.
 - *Date* would have the highest information gain since it perfectly separates the training data.
 - It would be selected at the root resulting in a very broad tree
 - Very good on the training, this tree would perform poorly in predicting unknown instances. Overfitting.
- The problem is that the partition is too specific, too many small classes are generated.
- We need to look at alternative measures ...

An alternative measure: gain ratio

$$SplitInformation(S, A) \equiv - \sum_{i=1}^c \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

- S_i are the sets obtained by partitioning on value i of A
- $SplitInformation$ measures the entropy of S with respect to the values of A . The more uniformly dispersed the data the higher it is.

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

- $GainRatio$ penalizes attributes that split examples in many small classes such as *Date*. Let $|S|=n$, *Date* splits examples in n classes
 - $SplitInformation(S, Date) = -[(1/n \log_2 1/n) + \dots + (1/n \log_2 1/n)] = -\log_2 1/n = \log_2 n$
- Compare with A , which splits data in two even classes:
 - $SplitInformation(S, A) = -[(1/2 \log_2 1/2) + (1/2 \log_2 1/2)] = -[-1/2 - 1/2] = 1$

Handling missing values training data



- How to cope with the problem that the value of some attribute may be missing?
 - The strategy: use other examples to guess attribute
 1. Assign the value that is most common among the training examples at the node
 2. Assign a probability to each value, based on frequencies, and assign values to missing attribute, according to this probability distribution
-

Applications

Suited for following classification problems:

- Applications whose Instances are represented by attribute-value pairs.
- The target function has discrete output values
- Disjunctive descriptions may be required
- The training data may contain missing attribute values

Real world applications

- Biomedical applications
- Manufacturing
- Banking sector
- Make-Buy decisions

Good References

Decision Tree

- https://www.youtube.com/watch?v=eKD5gxPPeY0&list=PLBv09BD7ez_4temBw7vLA19p3tdQH6FYO&index=1

Overfitting

- https://www.youtube.com/watch?time_continue=1&v=t56Nid85Thg
- <https://www.youtube.com/watch?v=y6SpA2Wuyt8>

Random Forest

- <https://www.stat.berkeley.edu/~breiman/RandomForests/>



BITS Pilani
Pilani Campus

Instance-based Learning

Dr. Sugata Ghosal

sugata.ghosal@pilani.bits-pilani.ac.in



Text Book(s)

- | | |
|----|--|
| T1 | Christopher Bishop: Pattern Recognition and Machine Learning, Springer International Edition |
| T2 | Tom M. Mitchell: Machine Learning, The McGraw-Hill Companies, Inc.. |

These slides are prepared by the instructor, with grateful acknowledgement of Prof. Tom Mitchell, Prof. Burges, Prof. Andrew Moore and many others who made their course materials freely available online.

Topics to be covered

- Instance based learning
 - K-Nearest Neighbour Learning
 - Locally Weighted Regression (LWR) Learning
 - Radial Basis Functions
-

k-Nearest Neighbor Classifier

- Nearest Neighbour classifier is an instance based classifier
- ‘lazy learning’, as learning is postponed until a new instance is encountered
- Constructs a local approximation to the target function, applicable in the neighbourhood of new instance
- Suitable in cases where target function is complex over the entire input space, but easily describable in local approximations
- Real world applications found in recommendation systems (amazon).
- Caveat is the high cost of classification, which happens at the time of processing rather than before hand (there’s no training phase)

Instance Based Learning

Key idea: just store all training examples $\langle x_i, f(x_i) \rangle$

Nearest neighbor:

- Given query instance x_q , first locate nearest training example x_n , then estimate $f^*(x_q) = f(x_n)$

K-nearest neighbor:

- Given x_q , take vote among its k nearest neighbors (if discrete-valued target function)
- Take mean of f values of k nearest neighbors (if real-valued) $f^*(x_q) = \sum_{i=1}^k f(x_i)/k$

When to Consider Nearest Neighbors

- Instances map to points in \mathbb{R}^N
- Less than 20 attributes per instance
- Lots of training data

Advantages:

- Training is very fast
- Learn complex target functions
- Do not lose information

Disadvantages:

- Slow at query time
- Easily fooled by irrelevant attributes

k-Nearest Neighbor Classifier

- Considers all instances as members of n-dimensional space
- Nearest neighbours of an instance is determined based on Euclidean distance
- Distance between two n-dimensional instances x_i and x_j is given by:

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^n (a_r(x_i) - a_r(x_j))^2}$$

- For nearest neighbour classifier, target function can be discrete or continuous

Distance Measures : Special Cases of Minkowski

- $h = 1$: Manhattan (city block, L₁ norm) distance

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{ip} - x_{jp}|$$

- $h = 2$: (L₂ norm) Euclidean distance

$$d(i, j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \dots + |x_{ip} - x_{jp}|^2)}$$

- $h \rightarrow \infty$. “supremum” (L_{max} norm, L _{∞} norm) distance.
 - This is the maximum difference between any component (attribute) of the vectors

$$d(i, j) = \lim_{h \rightarrow \infty} \left(\sum_{f=1}^p |x_{if} - x_{jf}|^h \right)^{\frac{1}{h}} = \max_f^p |x_{if} - x_{jf}|$$

Standardizing Numeric Data

- X: raw score to be standardized, μ : mean of the population, σ : standard deviation
- the distance between the raw score and the population mean in units of the standard deviation
- negative when the raw score is below the mean, “+” when above

Where

$$z = \frac{x - \mu}{\sigma}$$

Multiple Features

$$y_n = w_0 + w_1 f_{n1} + w_2 f_{n2} + w_3 f_{n3} + w_4 f_{n4}$$

Size (feet ²) f1	Number of bedrooms f2	Number of floors f3	Age of home (years) f4	Price (\$1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...

Feature scaling

Feature scaling of features x_i consists of rescaling the range of features to scale the range in $[0, 1]$ or $[-1, 1]$ (Do not apply to $x_0 = 1$)

E.g. $x_1 = \frac{\text{size} - 1000}{2000}$

Average value of x_1
Maximum value of x_1 – min value of x_1

$$x_2 = \frac{\#\text{bedrooms} - 2}{5}$$

Discrete and Continuous-valued function

- ➊ discrete-valued target function:

- ➊ $f : \mathbb{R}^n \rightarrow V$ where V is the finite set $\{v_1, v_2, \dots, v_s\}$
- ➊ the target function value is the most common value among the k nearest training examples

$$\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^k \delta(v, f(x_i))$$

where $\delta(a, b) = (a == b)$

- ➋ continuous-valued target function:

- ➋ algorithm has to calculate the mean value instead of the most common value
- ➋ $f : \mathbb{R}^n \rightarrow \mathbb{R}$

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^k f(x_i)}{k}$$

k-Nearest Neighbor Classifier

Training algorithm:

- For each training example $(x, f(x))$, add the example to the list *training_examples*

Classification algorithm:

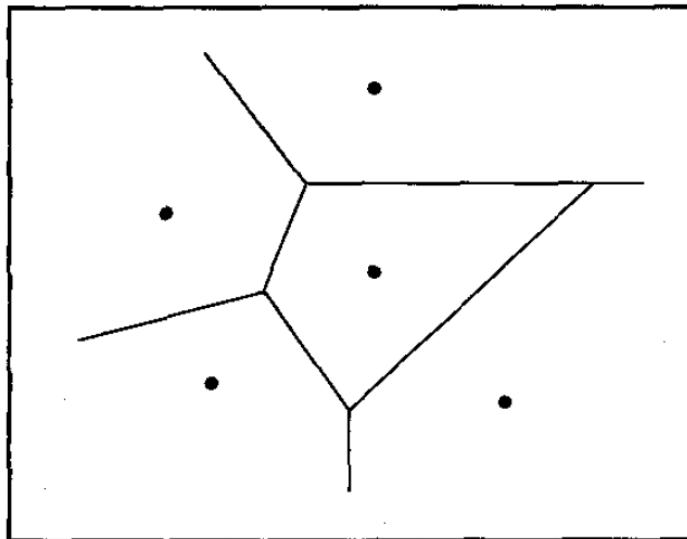
- Given a query instance x_q to be classified,
 - Let $x_1 \dots x_k$ denote the k instances from *training_examples* that are nearest to x_q
 - Return

$$\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^k \delta(v, f(x_i))$$

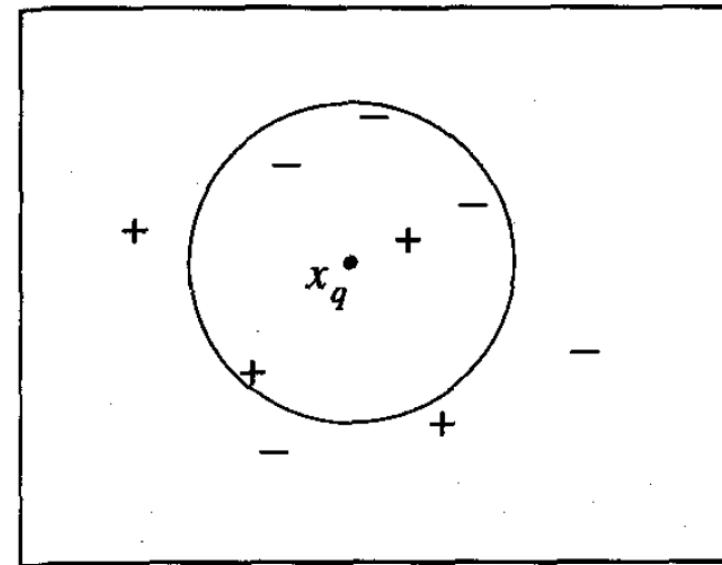
where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.

* It can be used for Regression as well.

k-NN examples



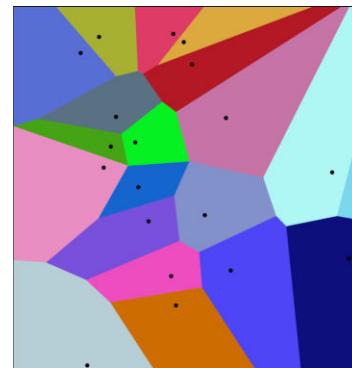
K=1



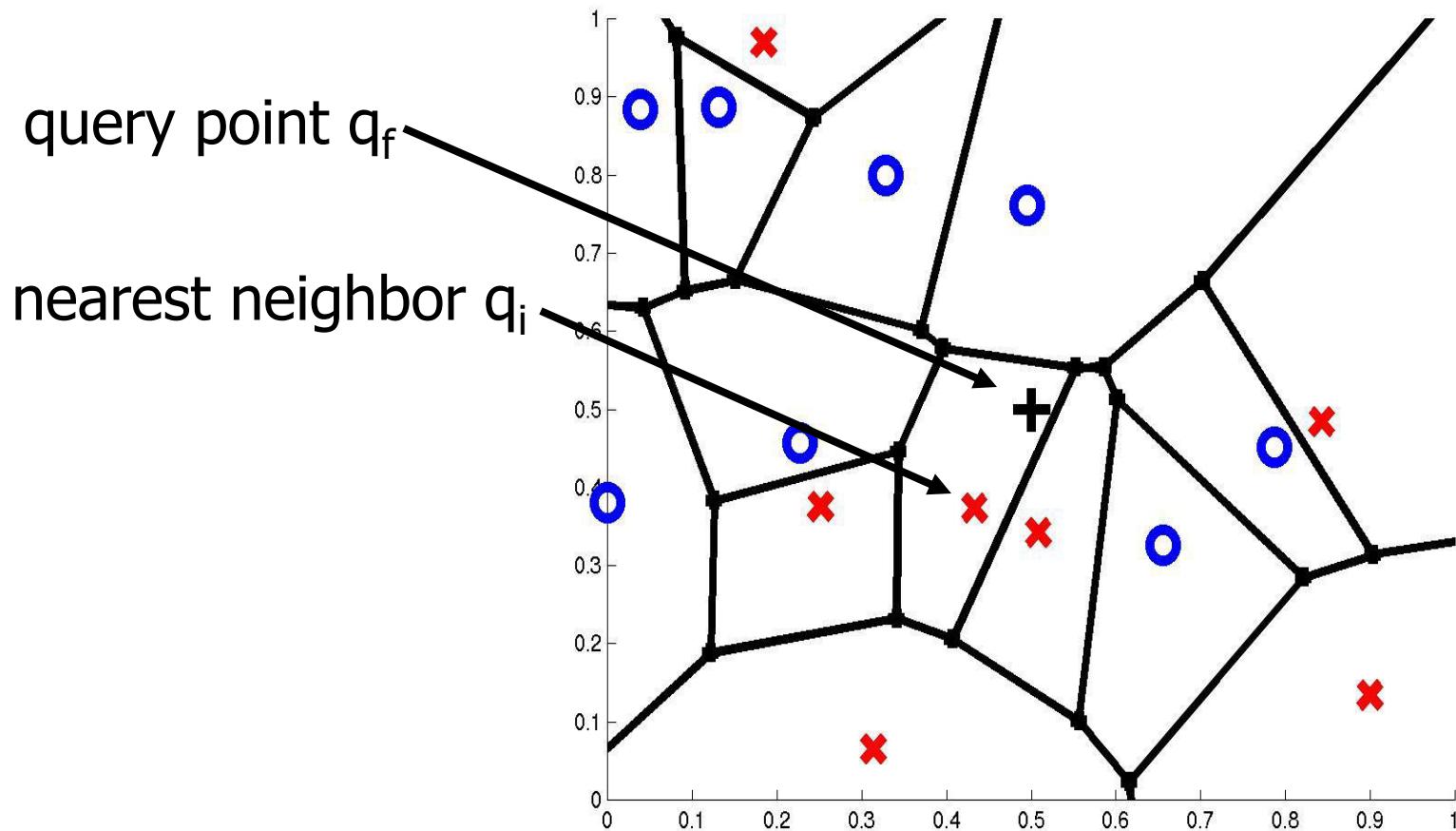
K=5

Voronoi Diagram

- It is a partition of a plane into regions close to each of a given set of objects.



Voronoi Diagram

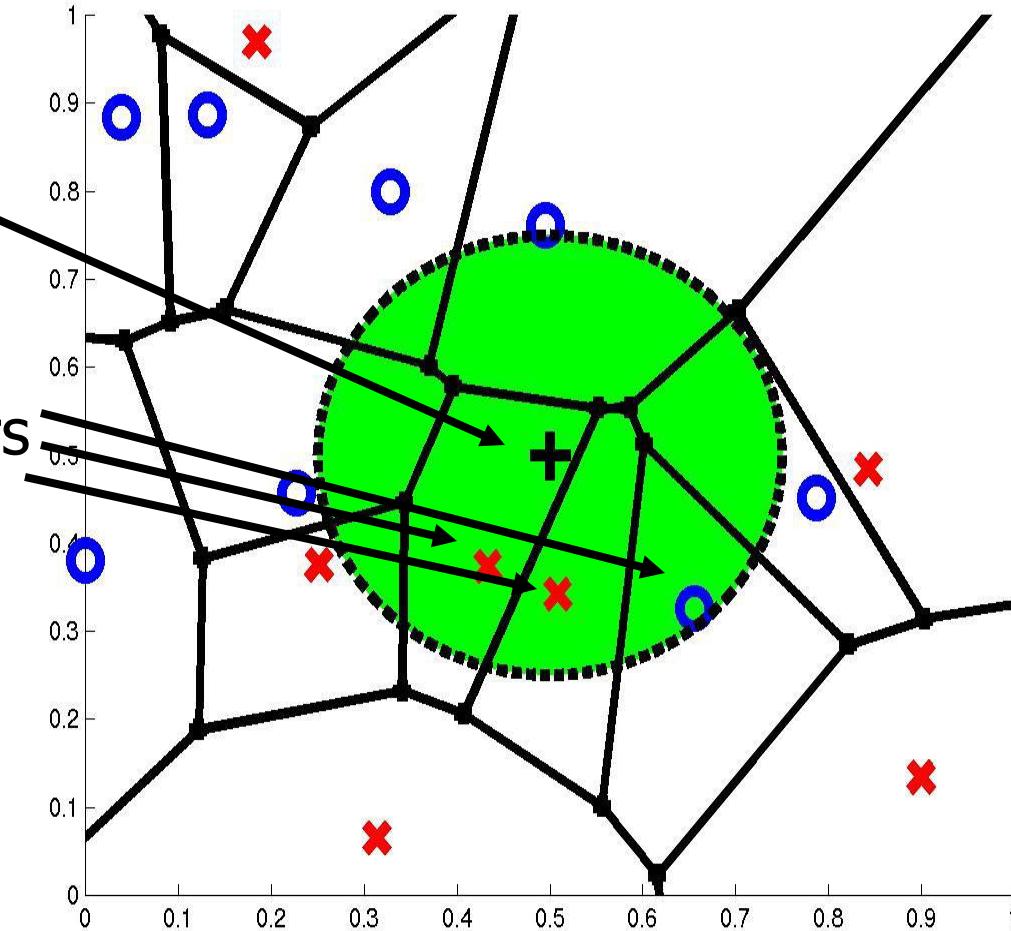


3-Nearest Neighbors

query point q_f

3 nearest neighbors

$2x, 1o$

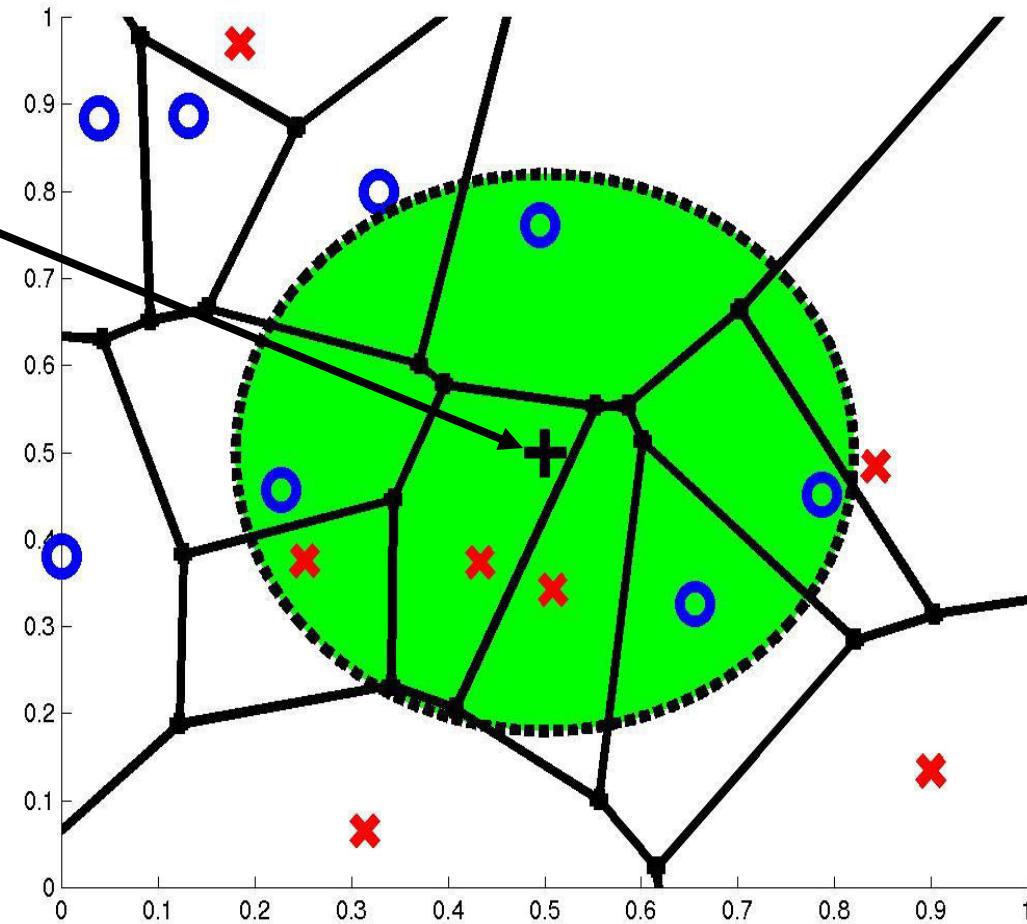


7-Nearest Neighbors

query point q_f

7 nearest neighbors

$3x, 4o$

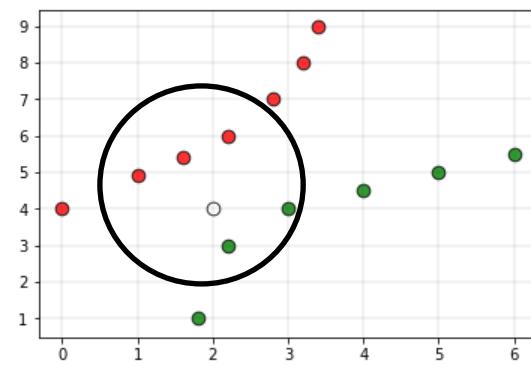


Various issues that affect the performance of kNN:

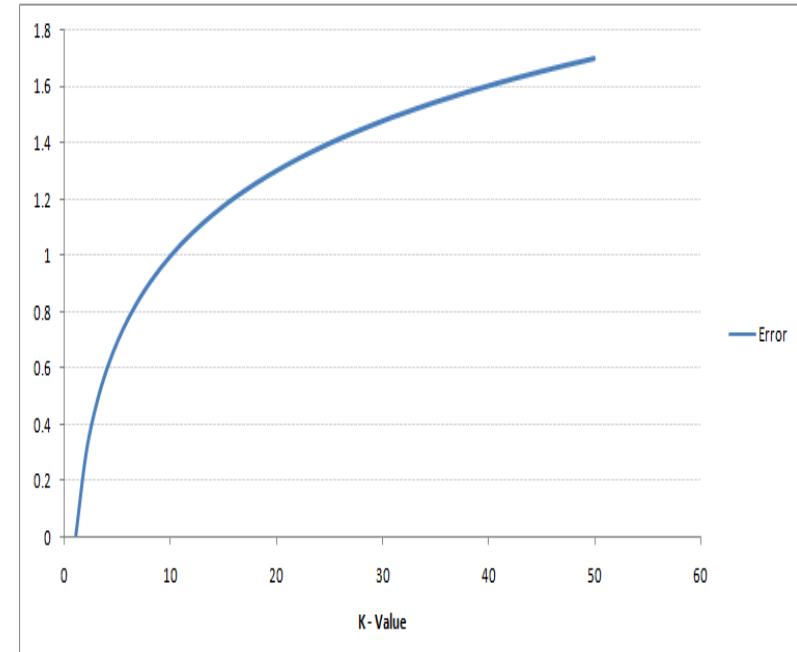
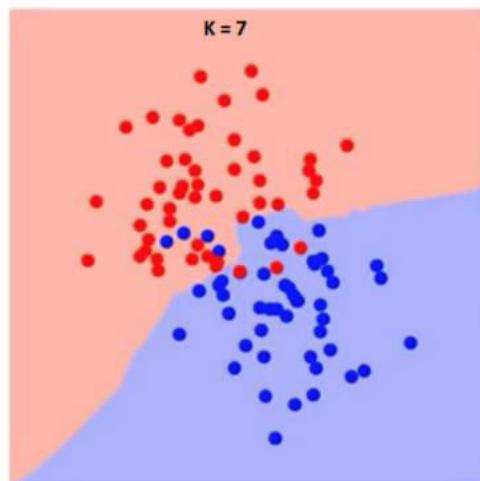
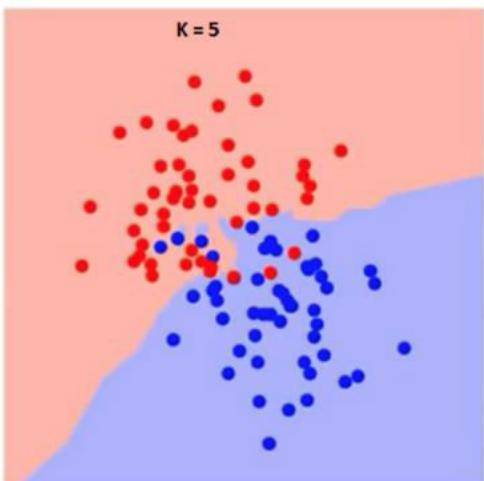
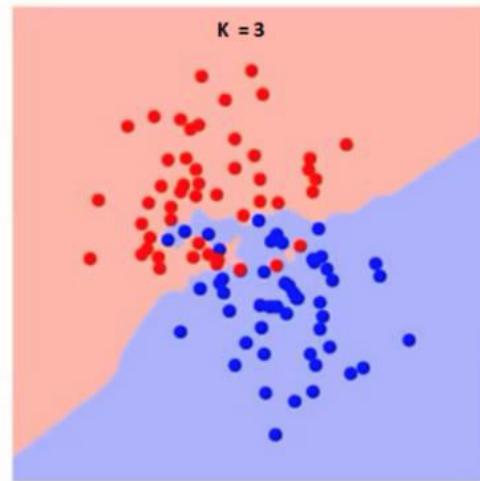
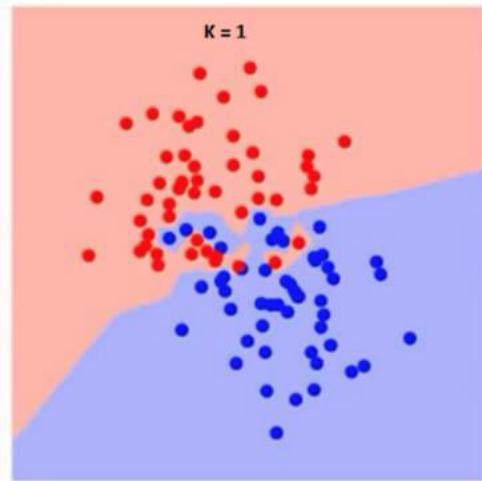
Performance of a classifier largely depends on the of the hyperparameter k

- Choosing smaller values for K, noise can have a higher influence on the result.
- Larger values of k are computationally expensive

Assigning the class labels can be tricky. For example, in the below case, for (k=5) the point is closer to ‘green’ classification, but gets classified as ‘red’ due to higher red votes/majority voting to ‘red’

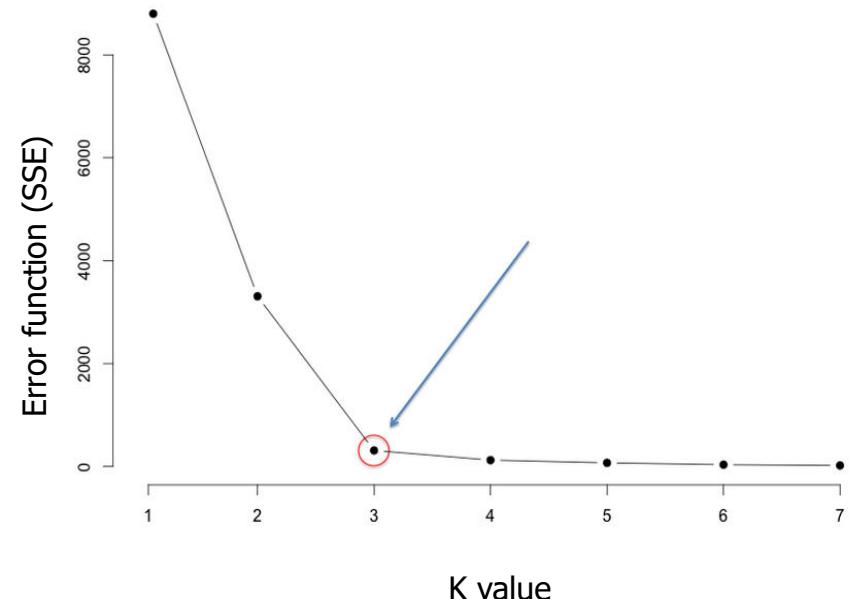


Finding optimal k for kNN classifiers



Finding K - Elbow method

- Compute sum of squares error (SSE) or any other error function for varying values of K (1 to a reasonable X) and plot against K
- In the plot, the elbow (see pic) gives the value of K beyond which the error function plot almost flattens
- As K approaches the total number of instances in the set, error function drops down to '0', but beyond optimal K, the model becomes too generic



Distance weighted nearest neighbor

- ➊ contribution of each of the k nearest neighbors is weighted accorded to their distance to x_q
 - discrete-valued target functions

$$\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^k w_i \delta(v, f(x_i))$$

where $w_i \equiv \frac{1}{d(x_q, x_i)^2}$ and $\hat{f}(x_q) = f(x_i)$ if $x_q = x_i$

- ➋ continuous-valued target function:

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^k w_i f(x_i)}{\sum_{i=1}^k w_i}$$

Distance Weighted k-NN

Give more weight to neighbors closer to the query point

- $f^*(x_q) = \sum_{i=1}^k w_i f(x_i) / \sum_{i=1}^k w_i ;$
- $w_i = K(d(x_q, x_i))$

and

- $d(x_q, x_i)$ is the distance between x_q and x_i

Variation: Instead of only k-nearest neighbors use all training examples (Shepard's method)

Distance Weighted Average

Weighting the data

$$- f^*(x_q) = \sum_i f(x_i) K(d(x_i, x_q)) / \sum_i K(d(x_i, x_q))$$

Relevance of a data point ($x_i, f(x_i)$) is measured by calculating the distance $d(x_i, x_q)$ between the query x_q and the input vector x_i .

Weighting the error criterion

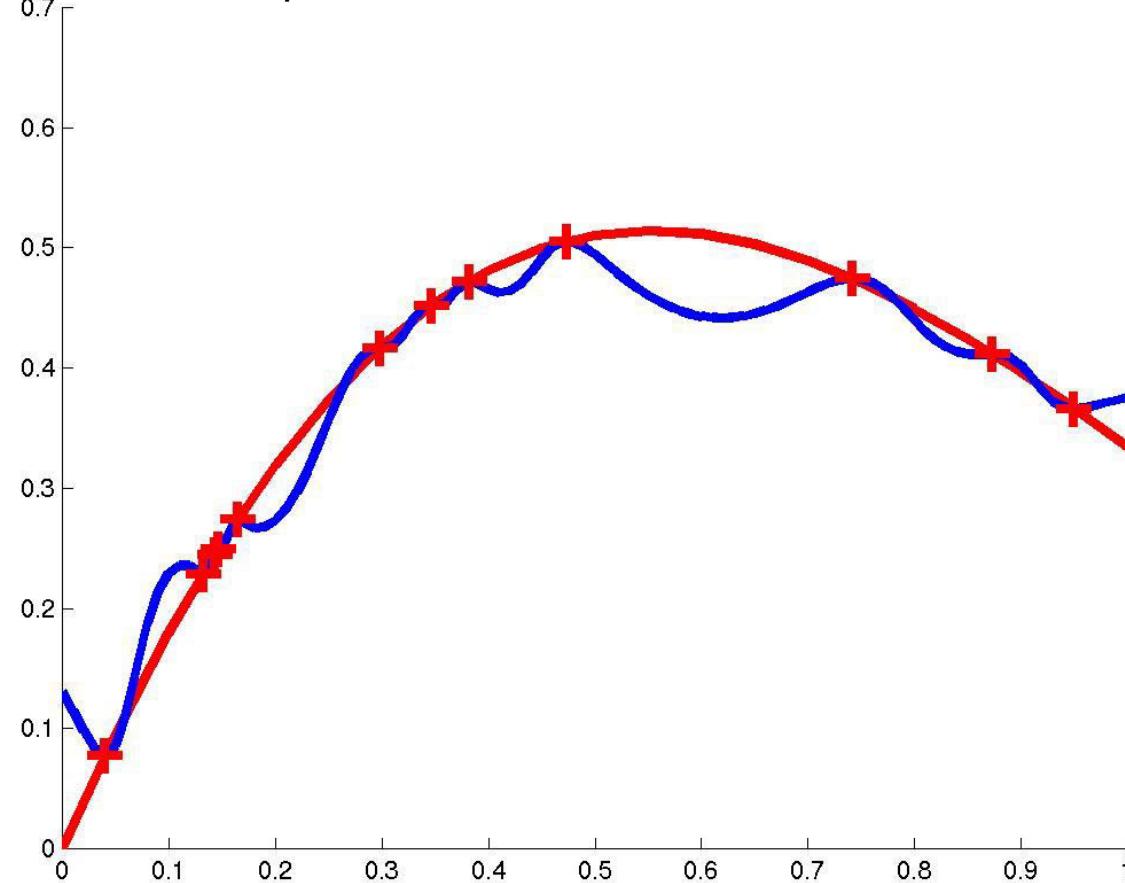
$$- E(x_q) = \sum_i (f^*(x_q) - f(x_i))^2 K(d(x_i, x_q))$$

Best estimate $f^*(x_q)$ will minimize the cost $E(x_q)$, therefore

$$\partial E(x_q) / \partial f^*(x_q) = 0$$

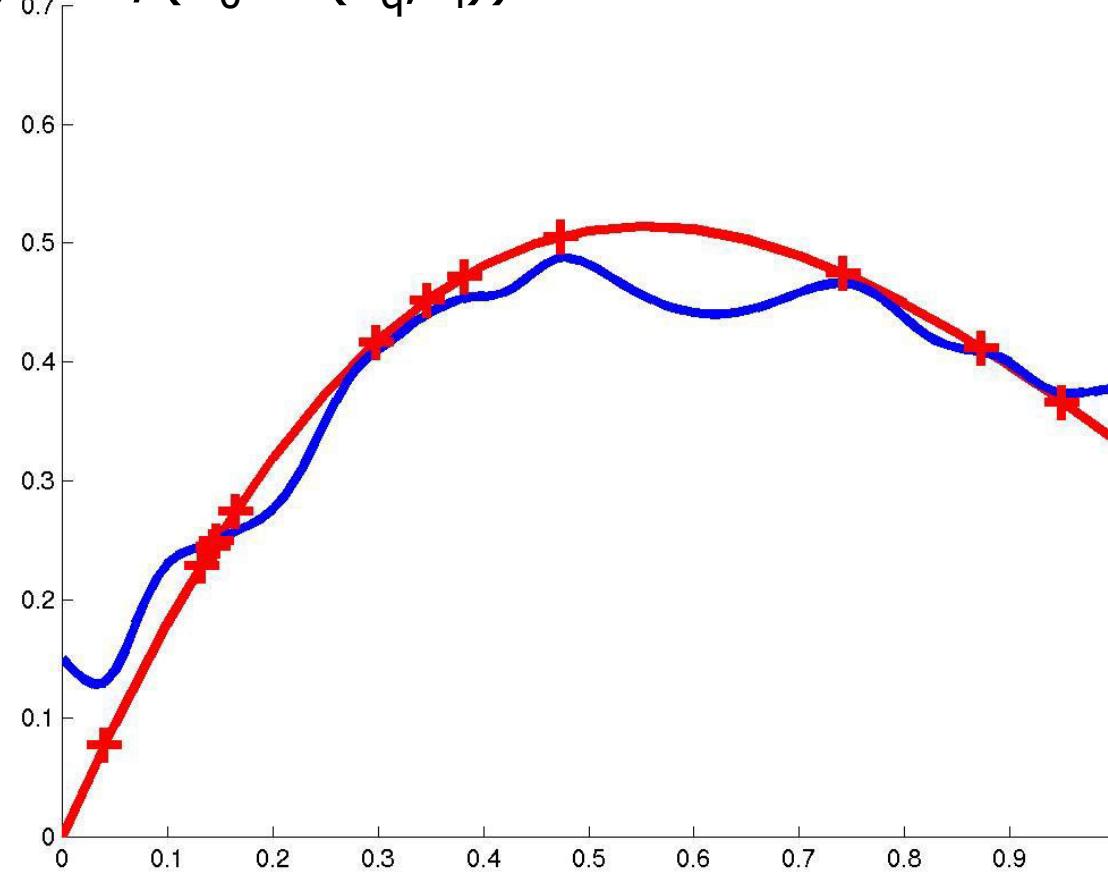
Distance Weighted NN

$$K(d(x_q, x_i)) = 1/d(x_q, x_i)^2$$



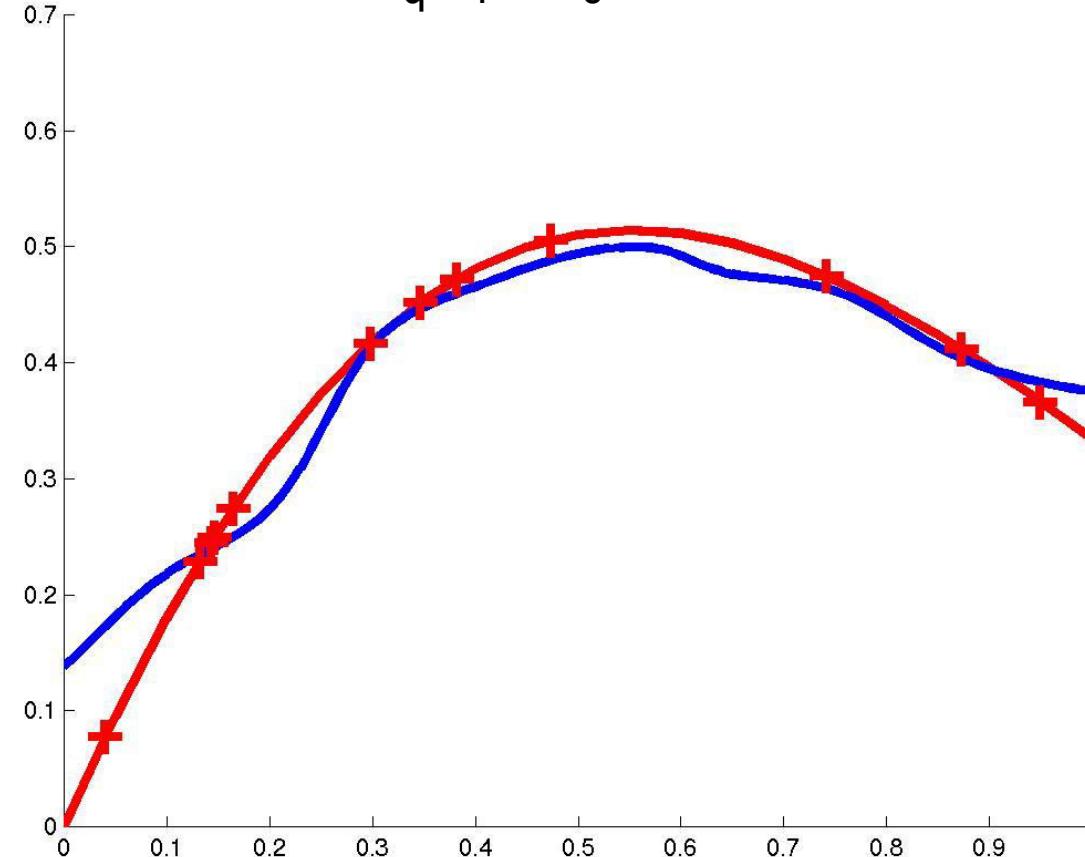
Distance Weighted NN

$$K(d(x_q, x_i)) = 1/(d_0 + d(x_q, x_i))^2$$



Distance Weighted NN

$K(d(x_q, x_i)) = \exp(-(d(x_q, x_i)/\sigma_0)^2)$ – **Gaussian weighted**



Curse of Dimensionality

Imagine instances described by 20 attributes but only a few are relevant to target function

Curse of dimensionality: nearest neighbor is easily misled when instance space is high-dimensional

One approach:

Stretch j -th axis by weight z_j , where z_1, \dots, z_n chosen to minimize prediction error

Use cross-validation to automatically choose weights

z_1, \dots, z_n

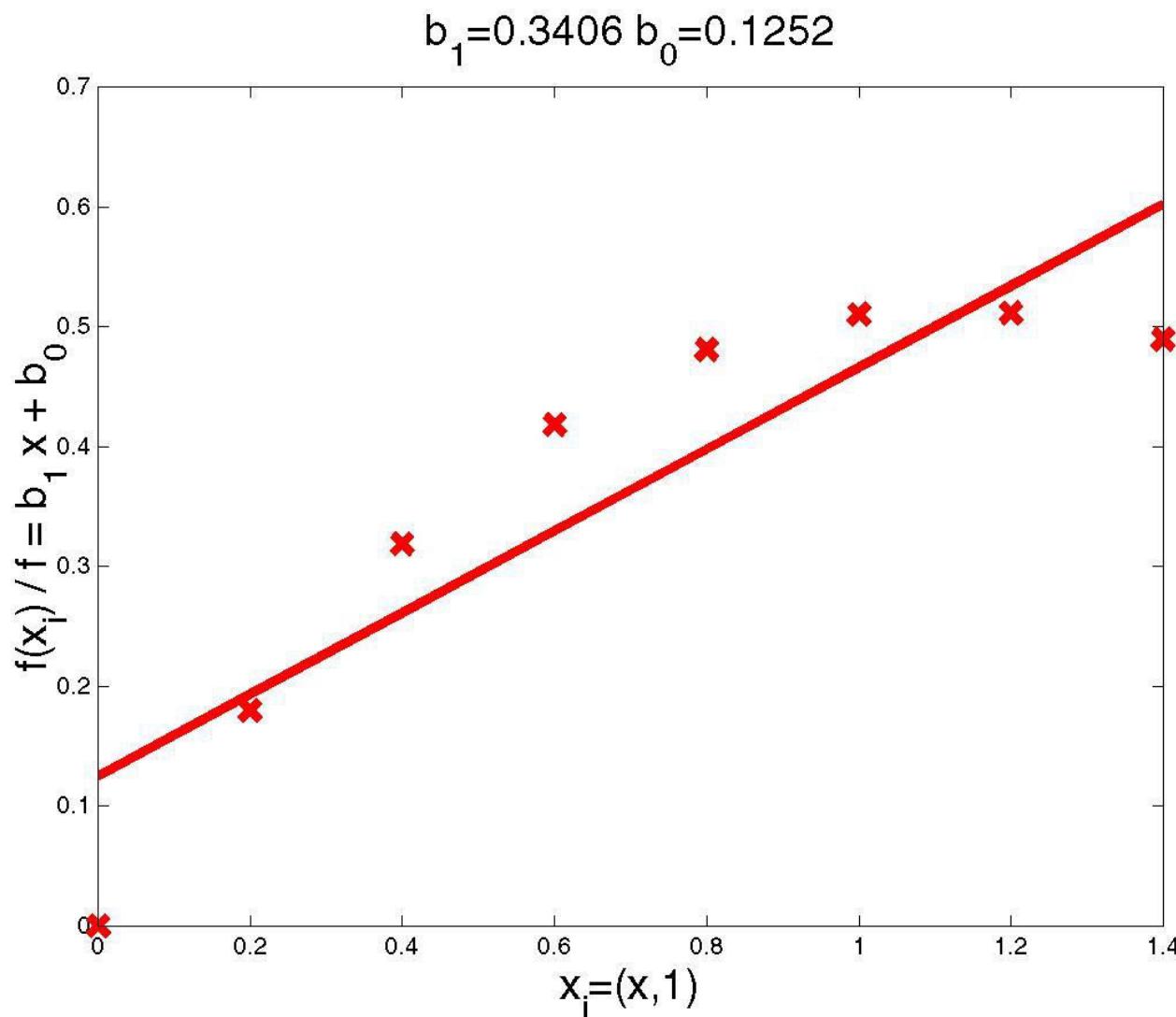
Note setting z_j to zero eliminates this dimension altogether (feature subset selection)

Locally Weighted Regression

Locally Weighted Regression

- Locally – Function approximated based on data near query point
- Weighted – Contribution by each training example is weighted by its distance from query point
- Regression- Approximates real-valued target function

Linear Regression Example



Locally weighted regression

- ➊ a note on terminology:
 - ➊ *Regression* means approximating a real-valued target function
 - ➋ *Residual* is the error $\hat{f}(x) - f(x)$ in approximating the target function
 - ➋ *Kernel function* is the function of distance that is used to determine the weight of each training example. In other words, the kernel function is the function K such that $w_i = K(d(x_i, x_q))$
 - ➋ nearest neighbor approaches can be thought of as approximating the target function at the single query point x_q
 - ⌋ locally weighted regression is a generalization to this approach, because it constructs an explicit approximation of f over a local region surrounding x_q
-

Locally weighted regression

- target function is approximated using a **linear function**
$$\hat{f}(x) = w_0 + w_1 a_1(x) + \dots + w_n a_n(x)$$
 - methods like **gradient descent** can be used to calculate the coefficients w_0, w_1, \dots, w_n to minimize the error in fitting such linear functions
 - ANNs require a global approximation to the target function
 - here, just a local approximation is needed
- ⇒ the error function has to be redefined
-

Locally weighted regression

- possibilities to redefine the error criterion E

1. Minimize the squared error over just the k nearest neighbors

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest neighbors}} (f(x) - \hat{f}(x))^2$$

2. Minimize the squared error over the entire set D , while weighting the error of each training example by some decreasing function K of its distance from x_q

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 \cdot K(d(x_q, x))$$

3. Combine 1 and 2

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest neighbors}} (f(x) - \hat{f}(x))^2 \cdot K(d(x_q, x))$$

Locally weighted regression

- ➊ choice of the error criterion
 - E_2 is the most esthetically criterion, because it allows every training example to have impact on the classification of x_q
 - however, computational effort grows with the number of training examples
 - E_3 is a good approximation to E_2 with constant effort

$$\Delta w_j = \eta \sum_{x \in k \text{ nearest neighbors}} K(d(x_q, x))(f(x) - \hat{f}(x))a_j$$

- ➋ Remarks on locally weighted linear regression:
 - in most cases, constant, linear or quadratic functions are used
 - costs for fitting more complex functions are prohibitively high
 - simple approximations are good enough over a sufficiently small subregion of X

Design Issues in Local Regression

- Local model order (constant, linear, quadratic)
- Distance function d
 - feature scaling: $d(x,q) = (\sum_{j=1}^d m_j (x_j - q_j)^2)^{1/2}$
 - irrelevant dimensions $m_j = 0$
- kernel function K

See paper by Atkeson [1996] "Locally Weighted Learning"

Radial Basis Function

Radial Basis Function

- closely related to distance-weighted regression and to ANNs
- learned hypotheses have the form

$$\hat{f}(x) = w_0 + \sum_{u=1}^k w_u \cdot K_u(d(x_u, x))$$

where

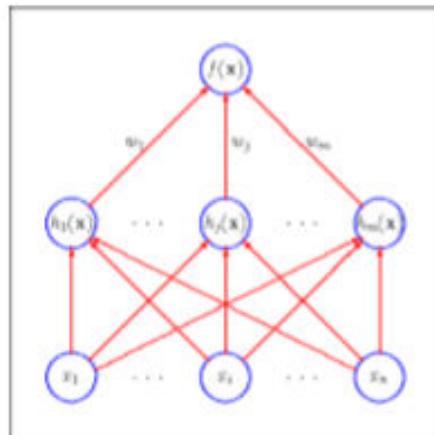
- each x_u is an instance from X and
 - $K_u(d(x_u, x))$ decreases as $d(x_u, x)$ increases and
 - k is a user-provided constant
-
- though $\hat{f}(x)$ is a global approximation to $f(x)$, the contribution of each of the K_u terms is localized to a region nearby the point x_u

Radial Basis Function

- it is common to choose each function $K_u(d(x_u, x))$ to be a Gaussian function centered at x_u with some variance σ^2

$$K_u(d(x_u, x)) = e^{-\frac{1}{2\sigma_u^2} d^2(x_u, x)}$$

- the function of $\hat{f}(x)$ can be viewed as describing a two-layer network where the first layer of units computes the various $K_u(d(x_u, x))$ values and the second layer a linear combination of the results



Training Radial Basis Function Networks

How to choose the center x_n for each Kernel function K_n ?

- scatter uniformly across instance space
- use distribution of training instances (clustering)

How to train the weights?

- Choose mean x_n and variance σ_n for each K_n
 - say, by using non-linear optimization or EM
- Hold K_n fixed and use local linear regression to compute the optimal weights w_n

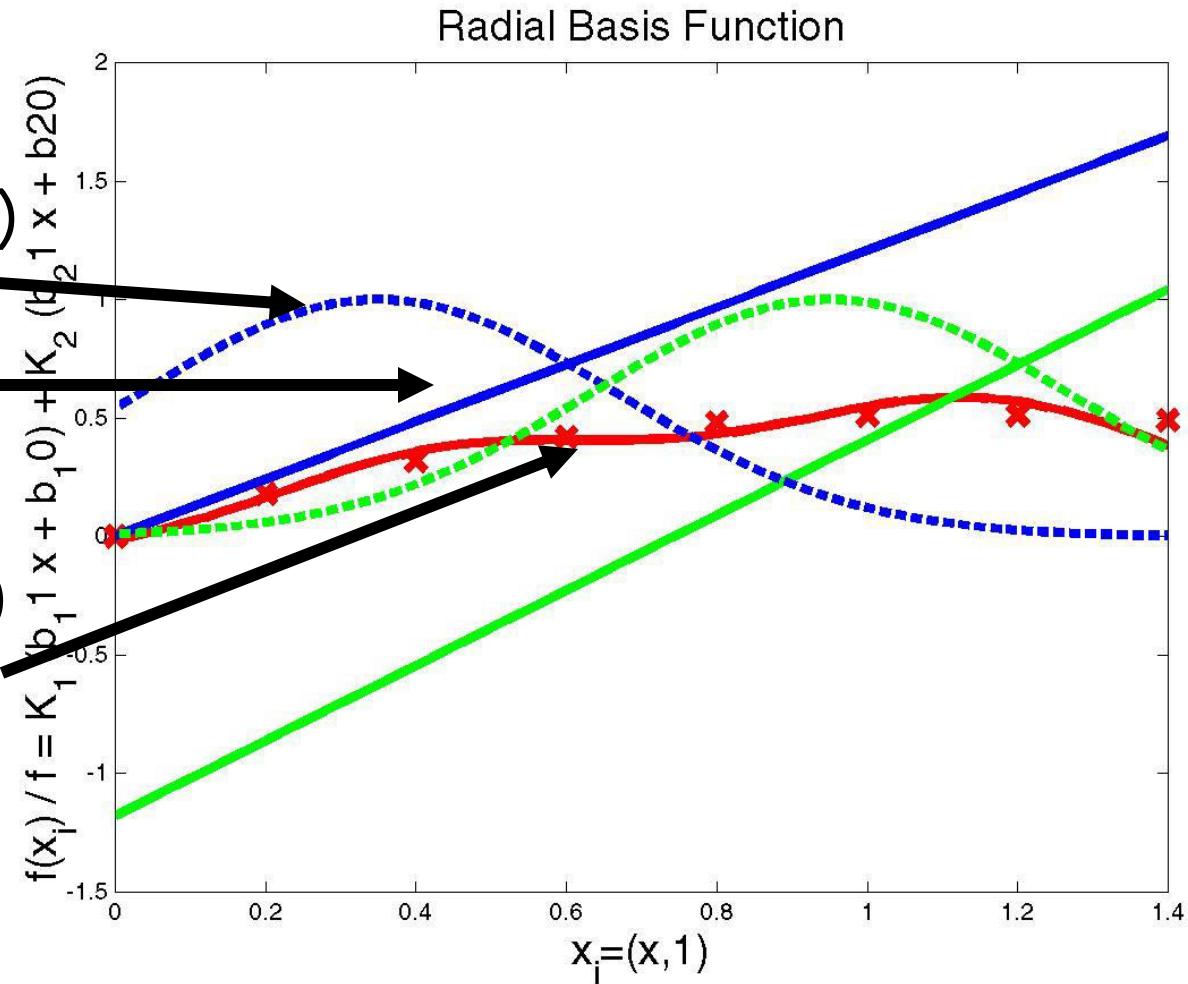
Radial Basis Network Example

$$K_1(d(x_1, x)) =$$

$$\exp(-1/2 d(x_1, x)^2 / \sigma^2)$$

$$w_1 x + w_0$$

$$\hat{f}(x) = K_1(w_1 x + w_0) + K_2(w_3 x + w_2)$$



Lazy and Eager Learning

Lazy: wait for query before generalizing

- k-nearest neighbors, weighted linear regression

Eager: generalize before seeing query

- Radial basis function networks, decision trees, back-propagation
- Eager learner must create global approximation

Lazy learner can create local approximations

If they use the same hypothesis space, lazy can represent more complex functions ($H=$ linear functions)

Literature & Software

- T. Mitchell, “Machine Learning”, chapter 8, “Instance-Based Learning”
- “Locally Weighted Learning”, Christopher Atkeson, Andrew Moore, Stefan Schaal
- R. Duda et al., “Pattern recognition”, chapter 4 “Non-Parametric Techniques”

Netlab toolbox

- k-nearest neighbor classification
- Radial basis function networks

Thank You



BITS Pilani
Pilani Campus

Machine Learning

ZG565

Dr. Sugata Ghosal

sugata.ghosal@pilani.bits-pilani.ac.in



Lecture No. – 8 | Problem Solving
Date – 31/12/2022
Time: 4:15 PM – 6:15 PM

Session Content

- Radial Basis Network
 - Bias-Variance Decomposition
 - Minimum Description Length
 - Decision Theory

 - Problem Solving
-

Radial Basis Function

Radial Basis Function

- closely related to distance-weighted regression and to ANNs
- learned hypotheses have the form

$$\hat{f}(x) = w_0 + \sum_{u=1}^k w_u \cdot K_u(d(x_u, x))$$

where

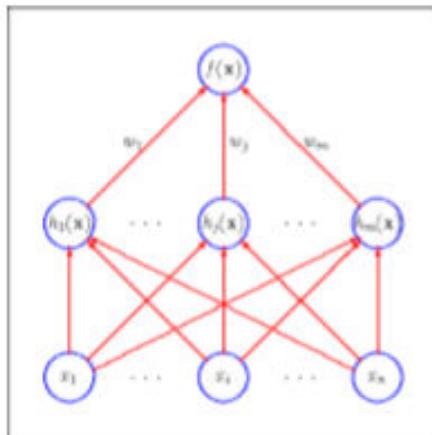
- each x_u is an instance from X and
 - $K_u(d(x_u, x))$ decreases as $d(x_u, x)$ increases and
 - k is a user-provided constant
-
- though $\hat{f}(x)$ is a global approximation to $f(x)$, the contribution of each of the K_u terms is localized to a region nearby the point x_u

Radial Basis Function

- it is common to choose each function $K_u(d(x_u, x))$ to be a Gaussian function centered at x_u with some variance σ^2

$$K_u(d(x_u, x)) = e^{-\frac{1}{2\sigma_u^2} d^2(x_u, x)}$$

- the function of $\hat{f}(x)$ can be viewed as describing a two-layer network where the first layer of units computes the various $K_u(d(x_u, x))$ values and the second layer a linear combination of the results



Training Radial Basis Function Networks

How to choose the center x_n for each Kernel function K_n ?

- scatter uniformly across instance space
- use distribution of training instances (clustering)

How to train the weights?

- Choose mean x_n and variance σ_n for each K_n
 - say, by using non-linear optimization or EM
- Hold K_n fixed and use local linear regression to compute the optimal weights w_n

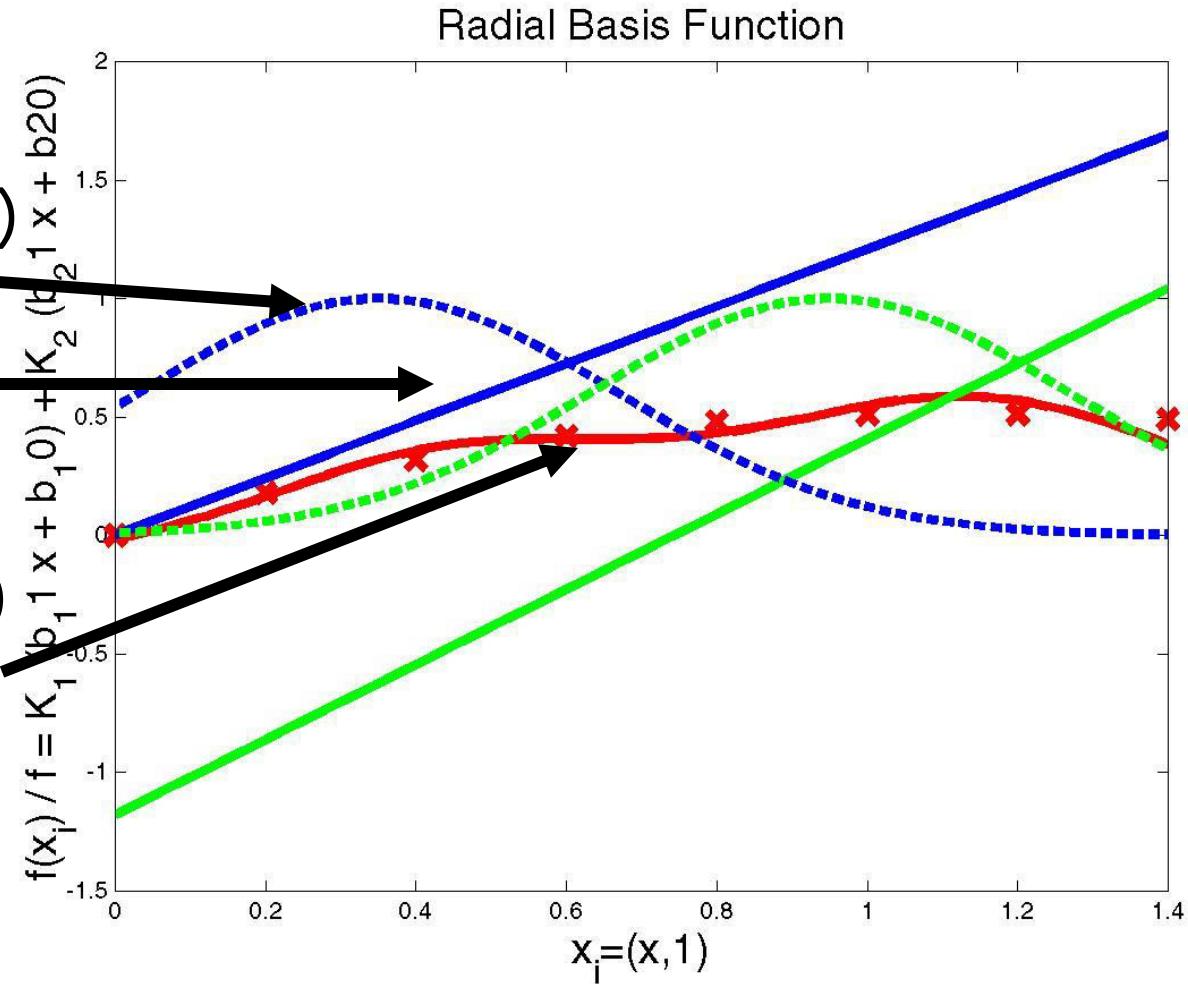
Radial Basis Network Example

$$K_1(d(x_1, x)) =$$

$$\exp(-1/2 d(x_1, x)^2 / \sigma^2)$$

$$w_1 x + w_0$$

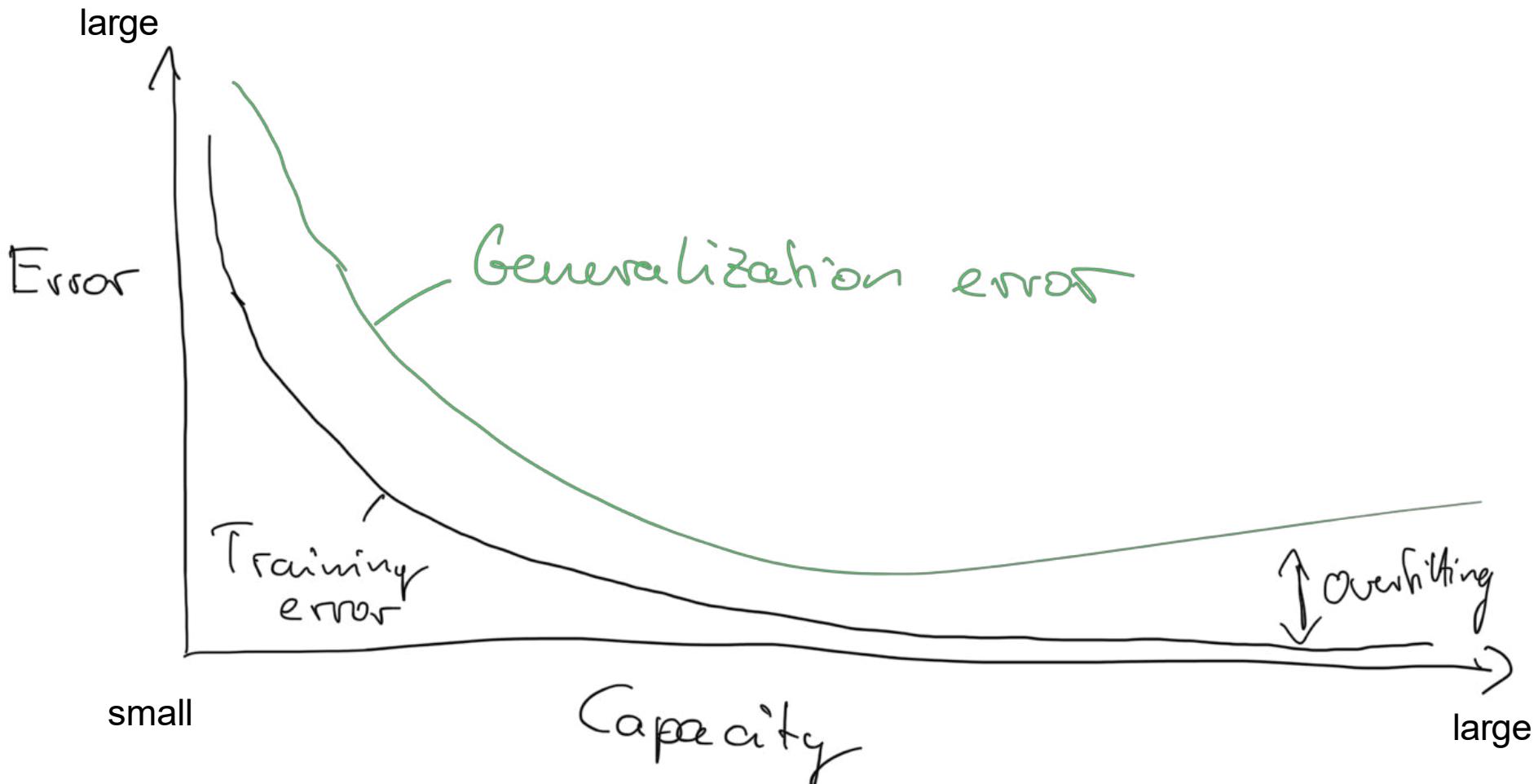
$$\hat{f}(x) = K_1(w_1 x + w_0) + K_2(w_3 x + w_2)$$



Bias-Variance Decomposition and Bias-Variance Trade-off

(and how it related to overfitting and underfitting)

Overfitting and Underfitting



Bias-Variance Decomposition



- Decomposition of the loss into bias and variance help us understand learning algorithms, concepts are related to underfitting and overfitting
- Helps explain why ensemble methods (post midsem) might perform better than single models

Bias-Variance Intuition



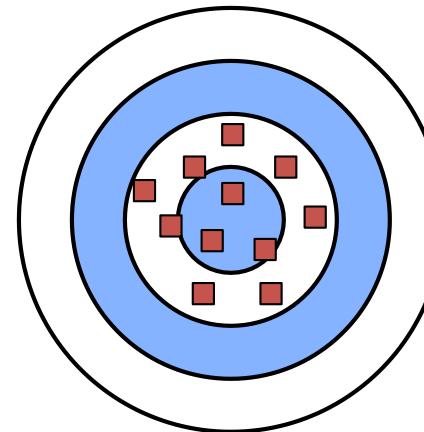
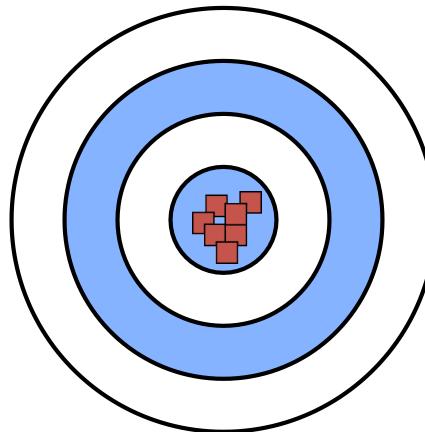
Low Variance

(Precise)

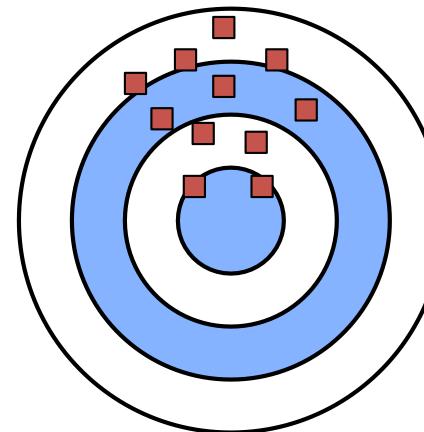
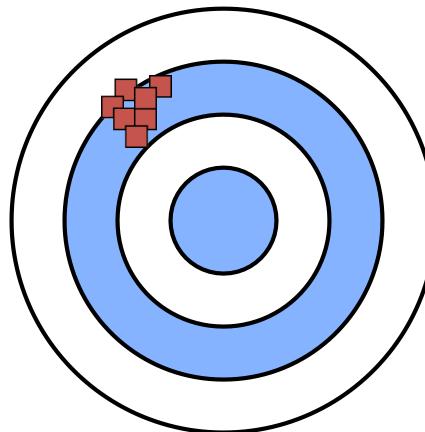
High Variance

(Not Precise)

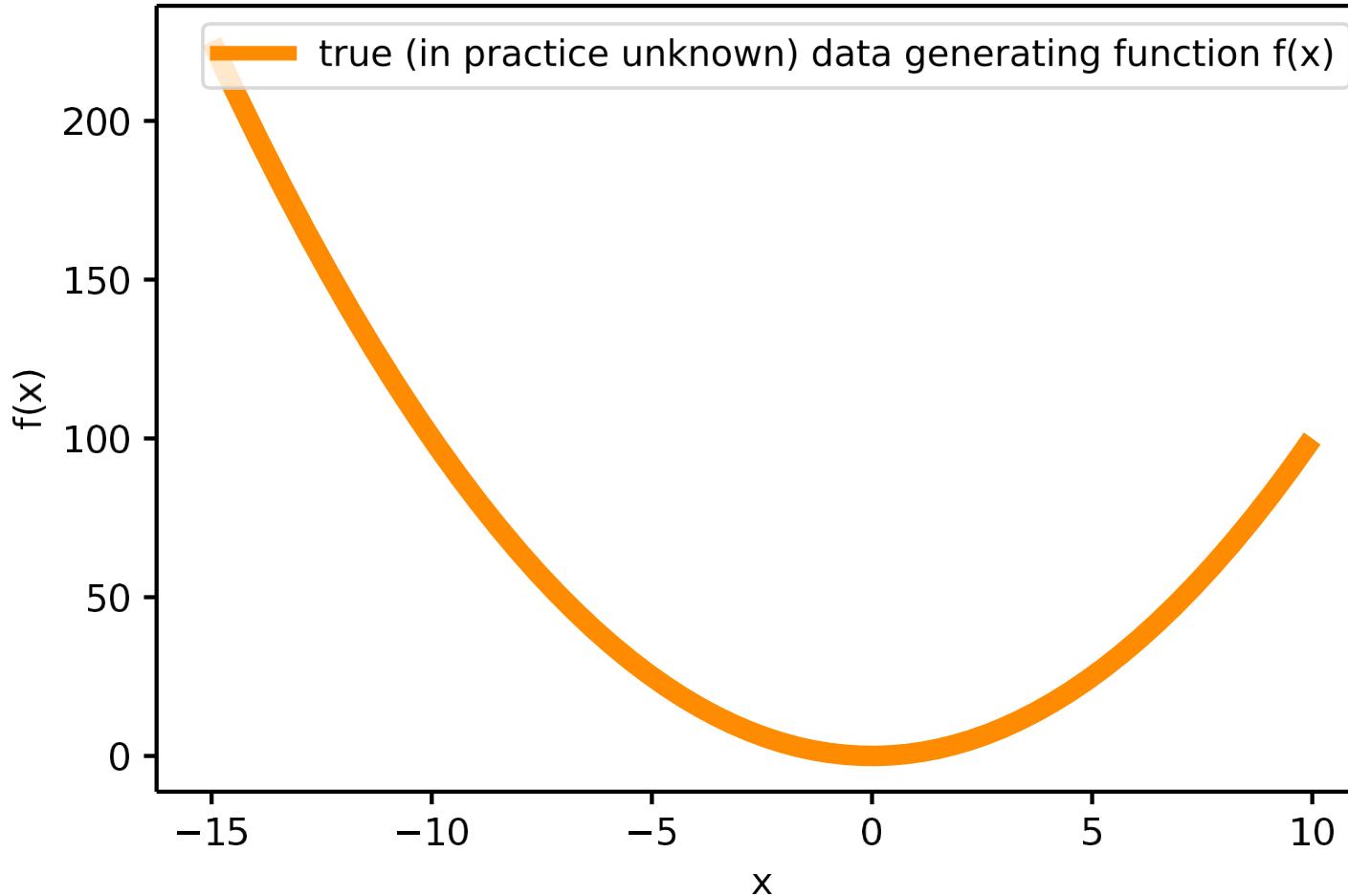
Low Bias
(Accurate)

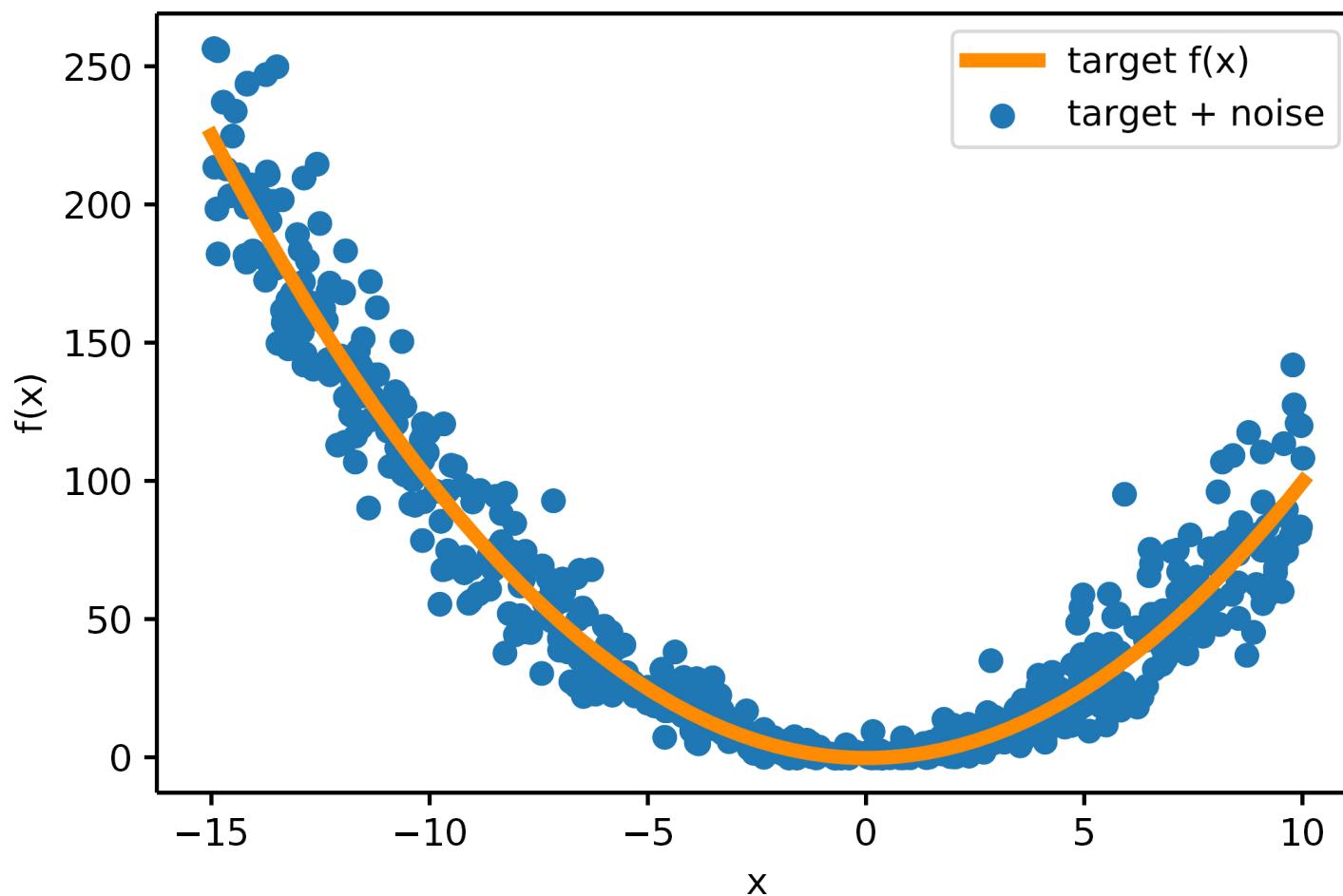


High Bias
(Not Accurate)

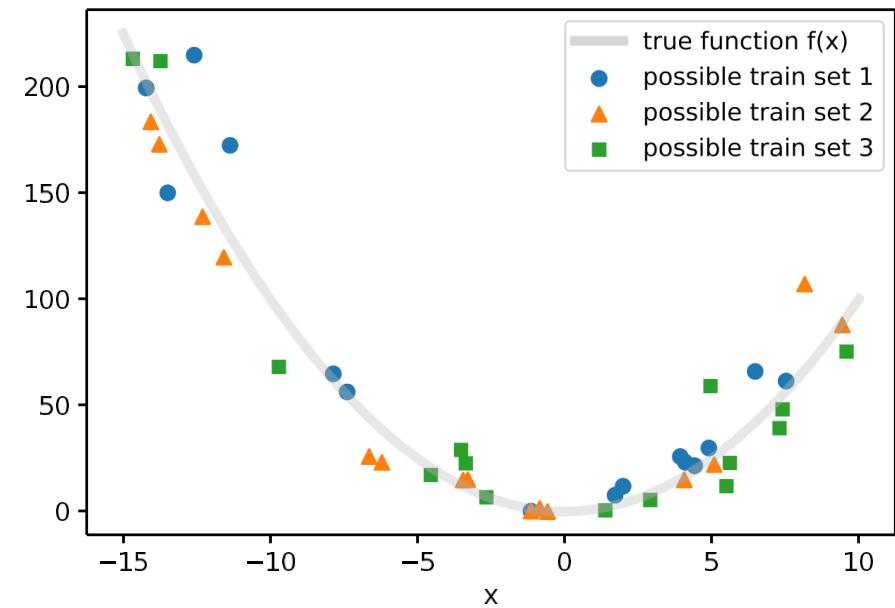
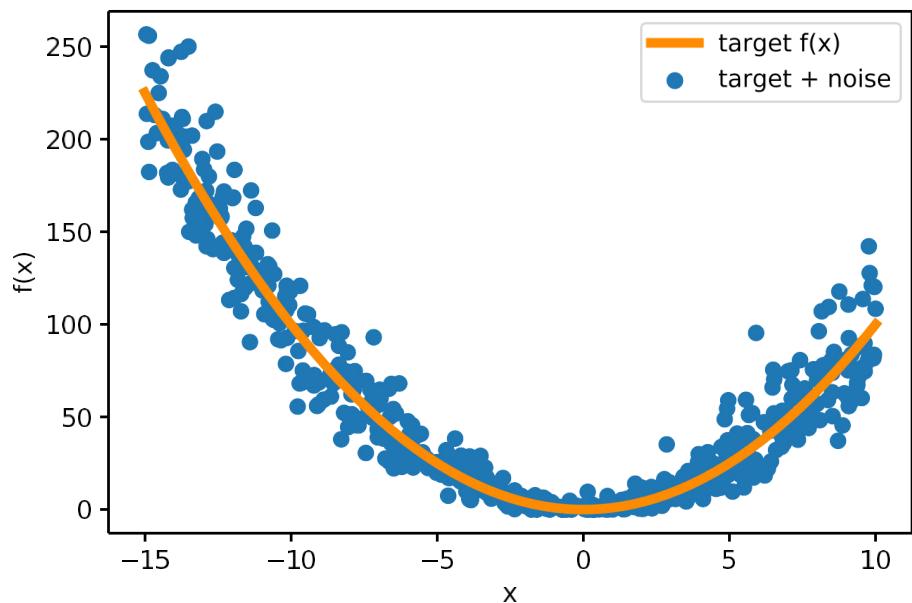


Bias-Variance Intuition

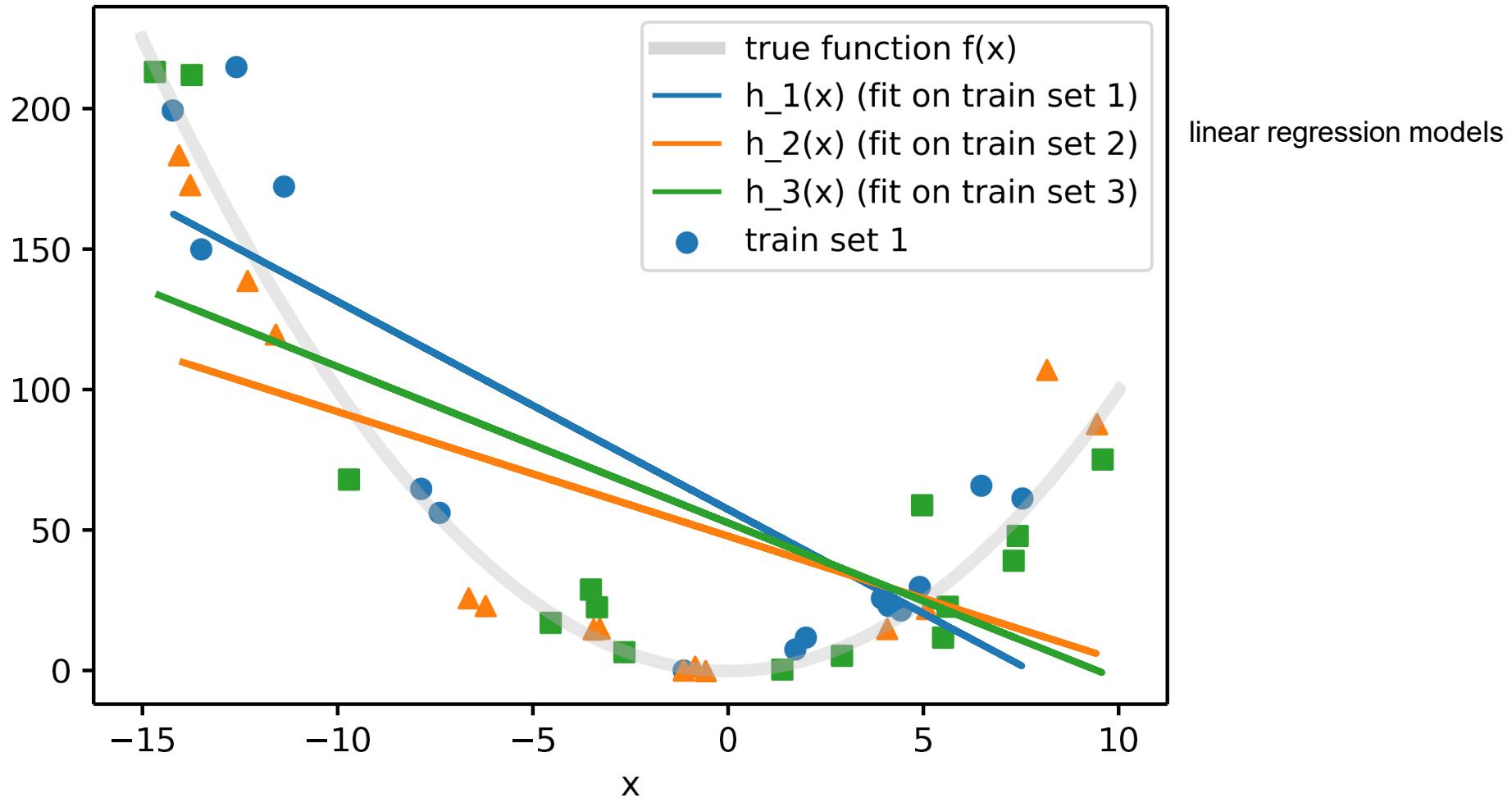




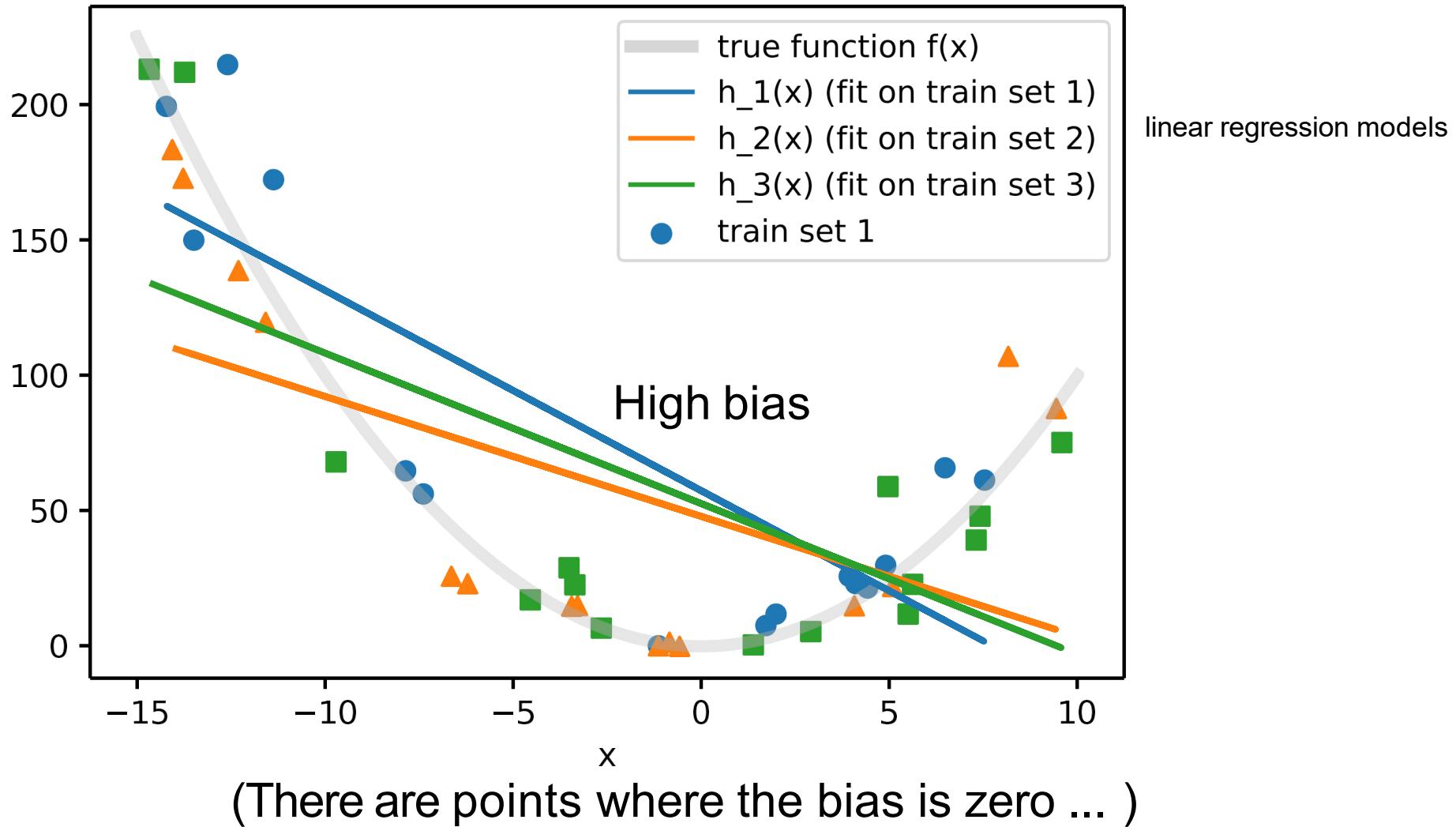
Bias-Variance Intuition



Bias-Variance Intuition



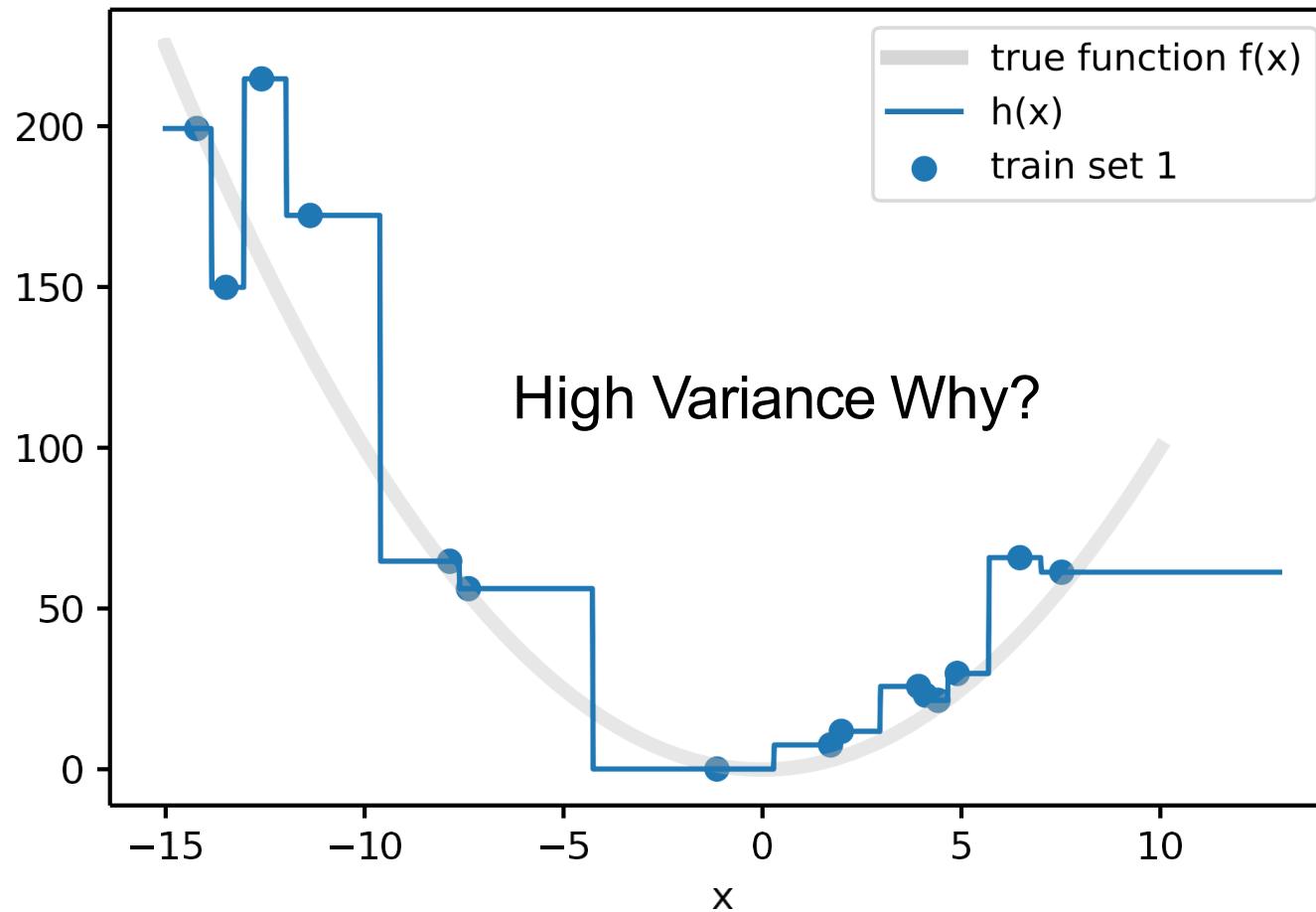
Bias-Variance Intuition



Bias-Variance Intuition



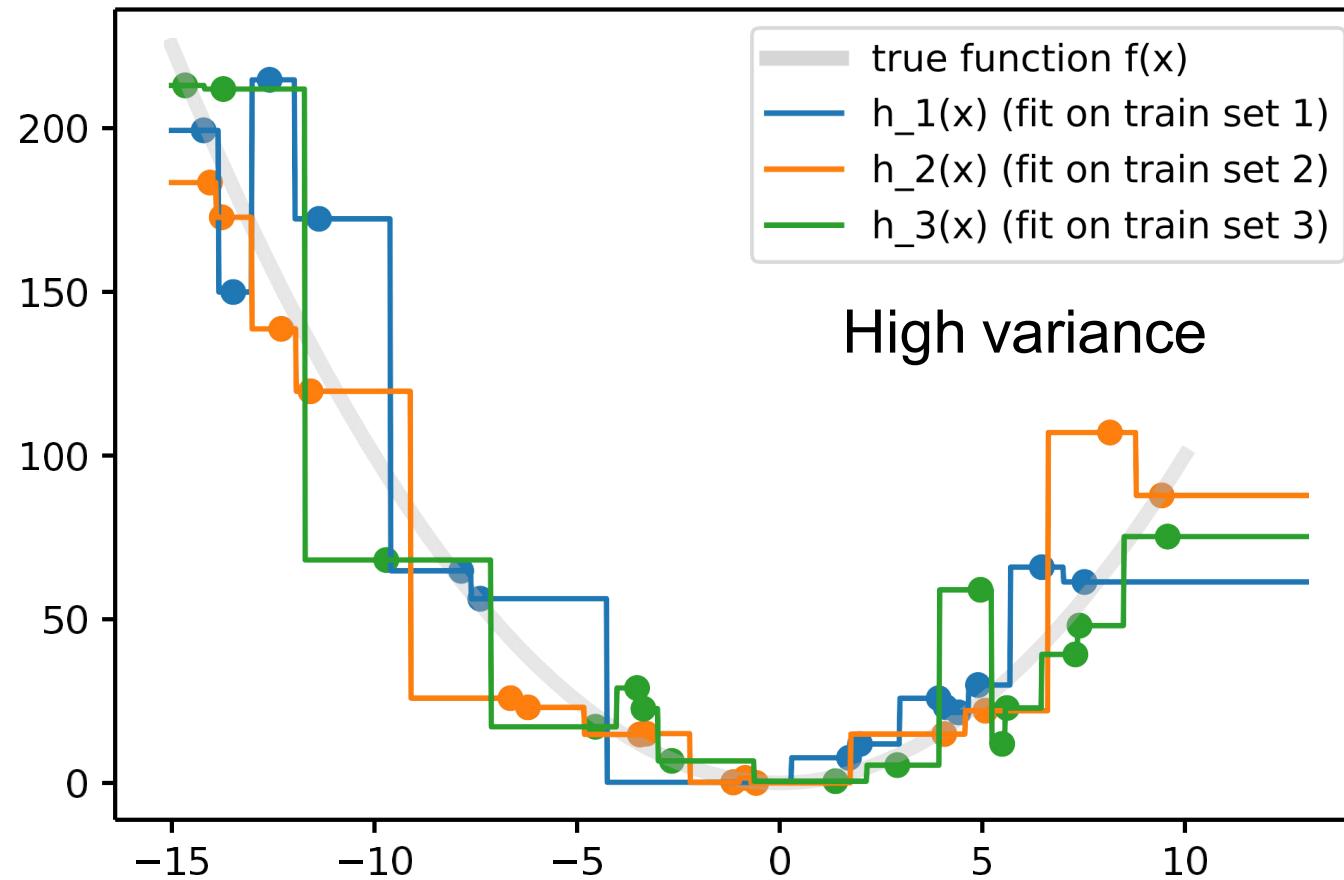
(here, an unpruned decision tree is fitted)



Bias-Variance Intuition



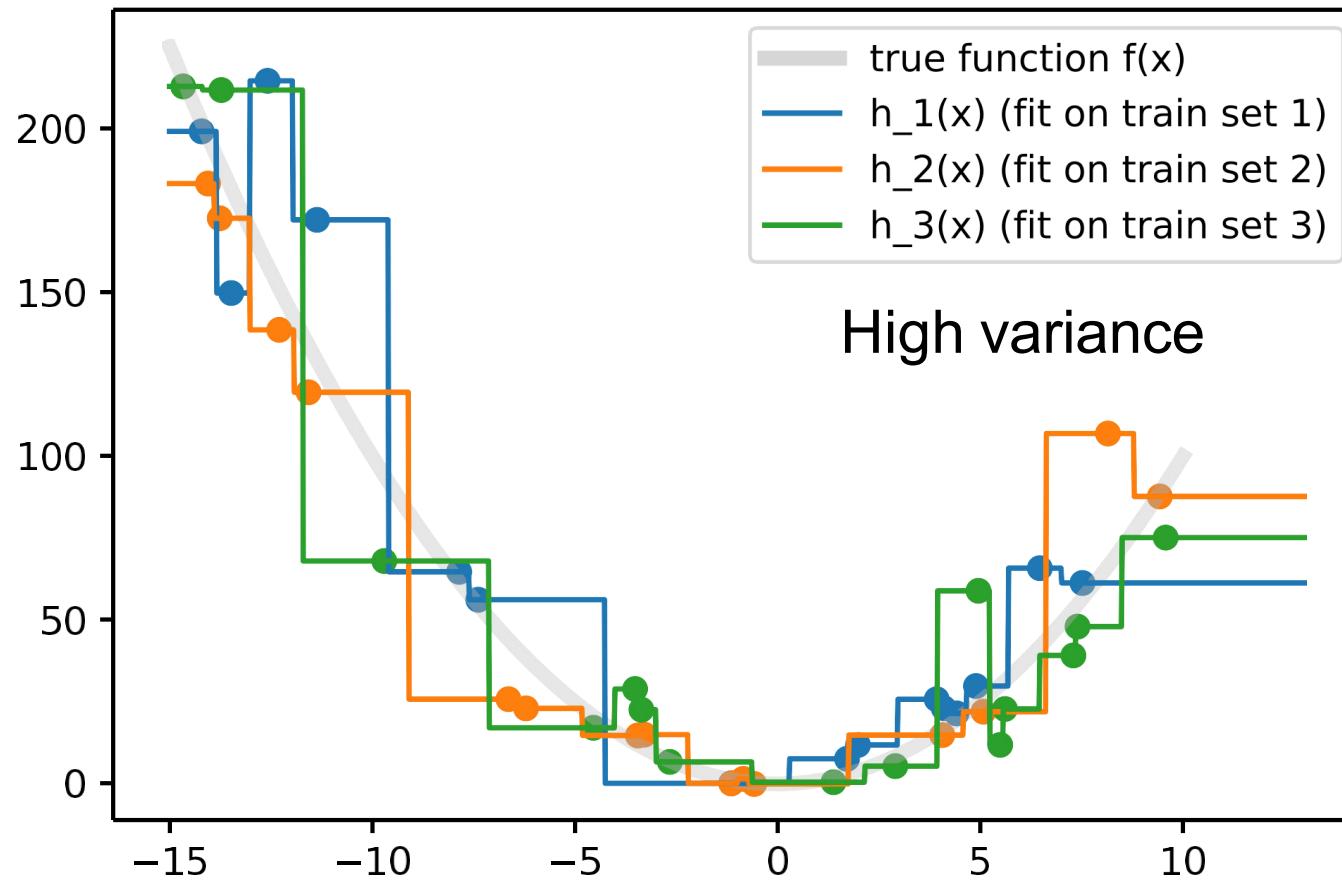
Suppose we have multiple training sets



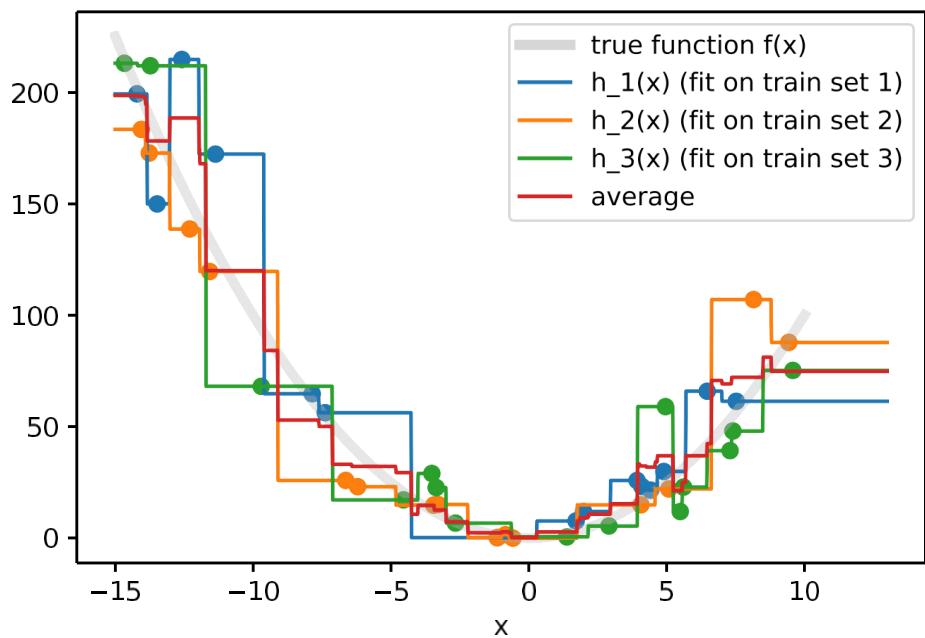
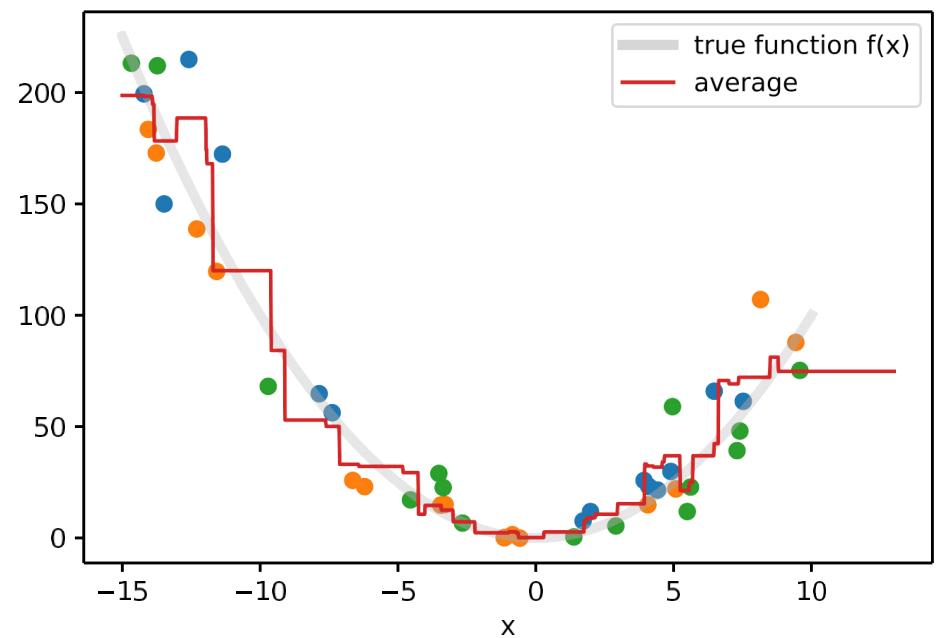
Bias-Variance Intuition



What happens if we take the average?



Bias-Variance Intuition



Terminology



Point estimator $\hat{\theta}$ of some parameter

(could also be a function, e.g., the hypothesis is
an estimator of some target function)

Terminology



Point estimator $\hat{\theta}$ of some parameter

(could also be a function, e.g., the hypothesis is
an estimator of some target function)

$$\text{Bias} = E[\hat{\theta}] - \theta$$

Terminology



General Definition

$$\text{Bias}[\hat{\theta}] = E[\hat{\theta}] - \theta$$

$$\text{Var}[\hat{\theta}] = E[\hat{\theta}^2] - (E[\hat{\theta}])^2$$

$$\text{Var}[\hat{\theta}] = E[(E[\hat{\theta}] - \hat{\theta})^2]$$

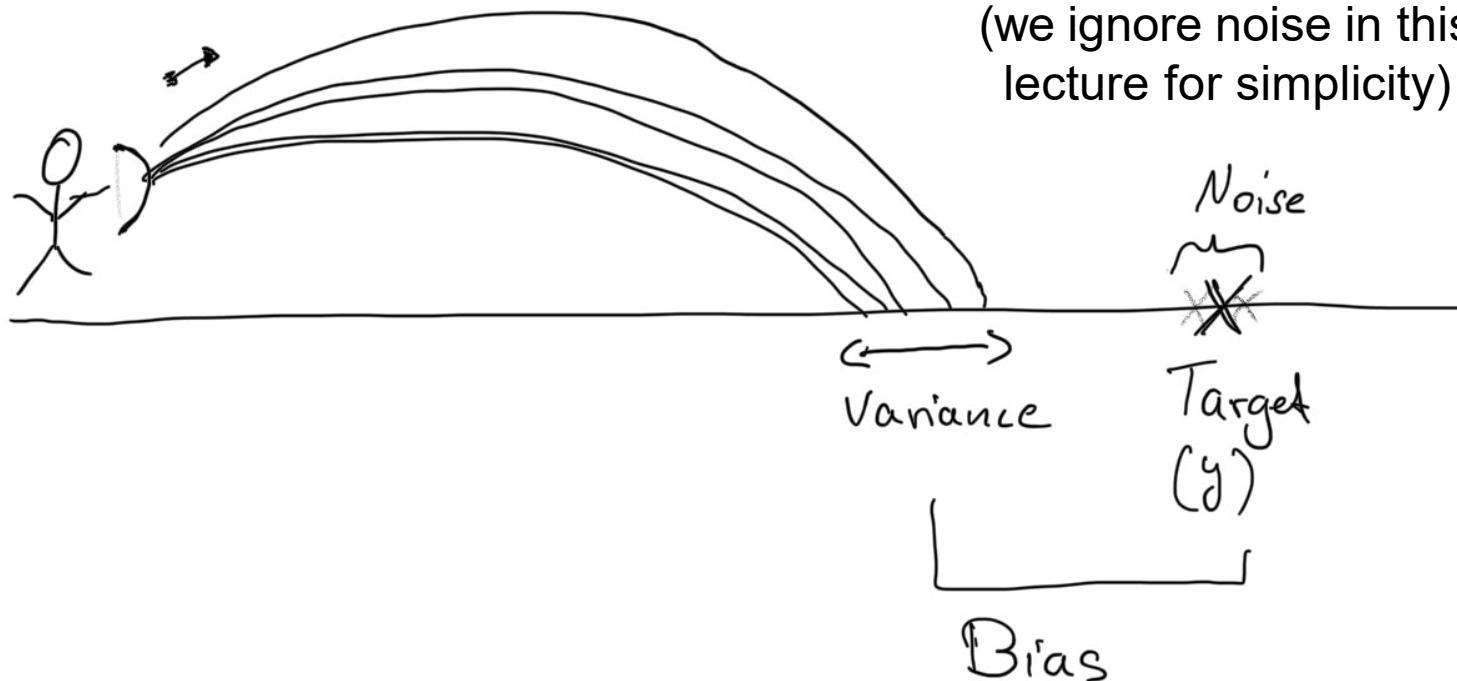
Terminology



$$\text{Bias}[\hat{\theta}] = E[\hat{\theta}] - \theta$$

$$\text{Var}[\hat{\theta}] = E[(E[\hat{\theta}] - \hat{\theta})^2]$$

Intuition



Terminology

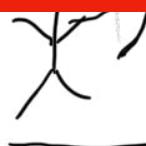


$$\text{Bias}[\hat{\theta}] = E[\hat{\theta}] - \theta$$

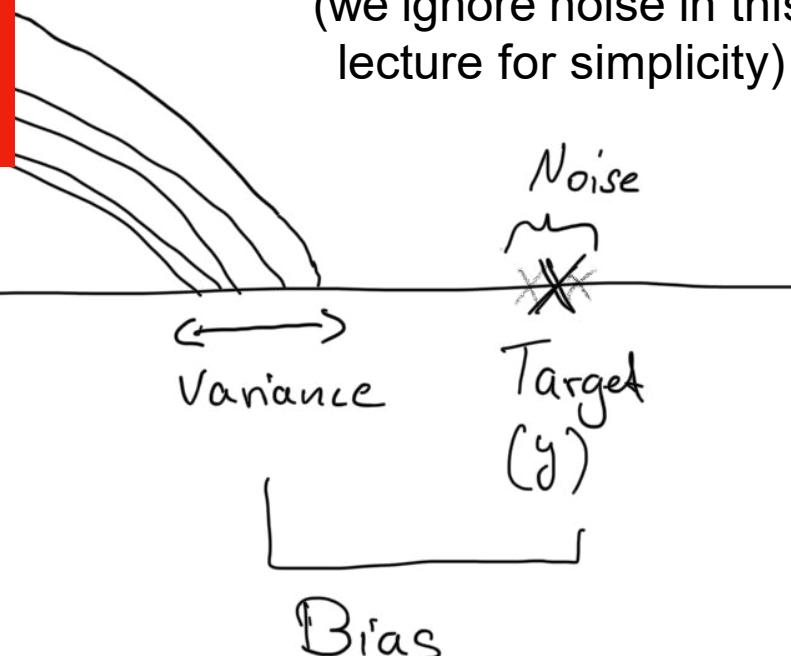
$$\text{Var}[\hat{\theta}] = E[(E[\hat{\theta}] - \hat{\theta})^2]$$

Bias is the difference between the average estimator from different training samples and the true value.

(The expectation is over the training sets.)



(we ignore noise in this lecture for simplicity)



Terminology



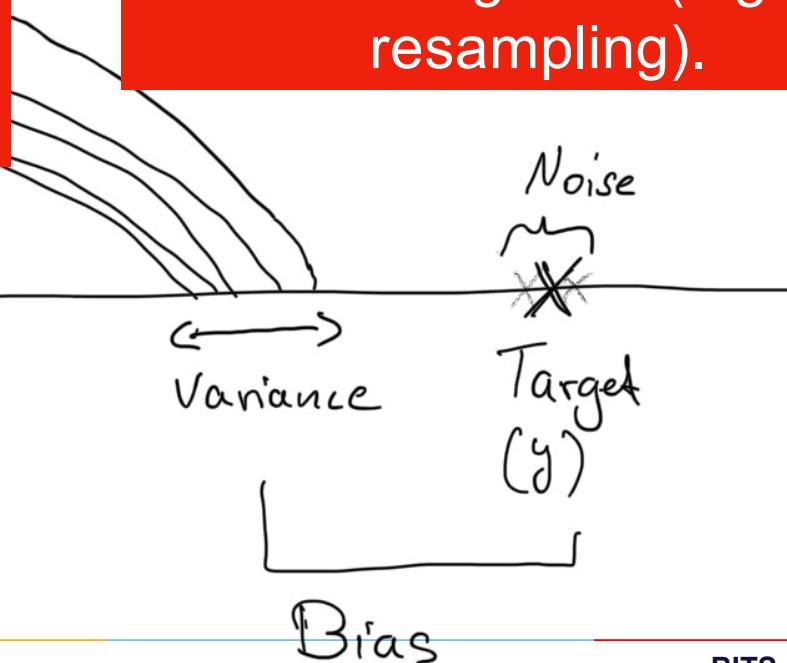
$$\text{Bias}[\hat{\theta}] = E[\hat{\theta}] - \theta$$

Bias is the difference between the average estimator from different training samples and the true value.
(The expectation is over the training sets.)



$$\text{Var}[\hat{\theta}] = E[(E[\hat{\theta}] - \hat{\theta})^2]$$

The variance provides an estimate of how much the estimate varies as we vary the training data (e.g., by resampling).



Bias-Variance Decomposition

Loss = Bias + Variance + Noise

Bias-Variance of the Squared Error

$$\text{Bias}[\hat{\theta}] = E[\hat{\theta}] - \theta$$

$$\text{Var}[\hat{\theta}] = E[\hat{\theta}^2] - (E[\hat{\theta}])^2$$

$$\text{Var}[\hat{\theta}] = E[(E[\hat{\theta}] - \hat{\theta})^2]$$

for simplicity, we ignore
the noise term

"ML Notation" for Squared Error Loss

$y = f(x)$ target

$\hat{y} = \hat{f}(x) = h(x)$ prediction

$S = (y - \hat{y})^2$ squared error

(Next slides: the expectation is over the training data, i.e,
the average estimator from different training samples)

Bias-Variance of the Squared Error

"ML Notation" for

$y = f(x)$ target

$\hat{y} = \hat{f}(x) = h(x)$ prediction

Squared Error Loss $S = (y - \hat{y})^2$ squared error

$$S = (y - \hat{y})^2$$

$$(y - \hat{y})^2 = (y - E[y] + E[y] - \hat{y})^2$$

$$= (y - E[y])^2 + (E[y] - \hat{y})^2 - 2(y - E[y])(E[y] - \hat{y})$$

Bias-Variance of the Squared Error

$$S = (y - \hat{y})^2$$

$$\begin{aligned}
 (y - \hat{y})^2 &= (y - E[y] + E[y] - \hat{y})^2 && \text{???} \\
 &= (y - E[y])^2 + (E[y] - \hat{y})^2 - 2(y - E[y])(E[y] - \hat{y})
 \end{aligned}$$

$$\begin{aligned}
 E[2(y - E[y])(E[y] - \hat{y})] &= 2E[(y - E[y])(E[y] - \hat{y})] \\
 &= 2(y - E[y])E[(E[y] - \hat{y})] \\
 &= 2(y - E[y])(E[E[y]] - E[\hat{y}]) \\
 &= 2(y - E[y])(E[y] - E[\hat{y}]) \\
 &= 0
 \end{aligned}$$

Bias-Variance of the Squared Error

$$S = (y - \hat{y})^2$$

$$\begin{aligned}(y - \hat{y})^2 &= (y - E[y] + E[y] - \hat{y})^2 \\ &= (y - E[y])^2 + (E[y] - \hat{y})^2\end{aligned}$$

$$E[S] = E[(y - \hat{y})^2]$$

$$E[(y - \hat{y})^2] = (y - E[y])^2 + E[(E[y] - \hat{y})^2]$$

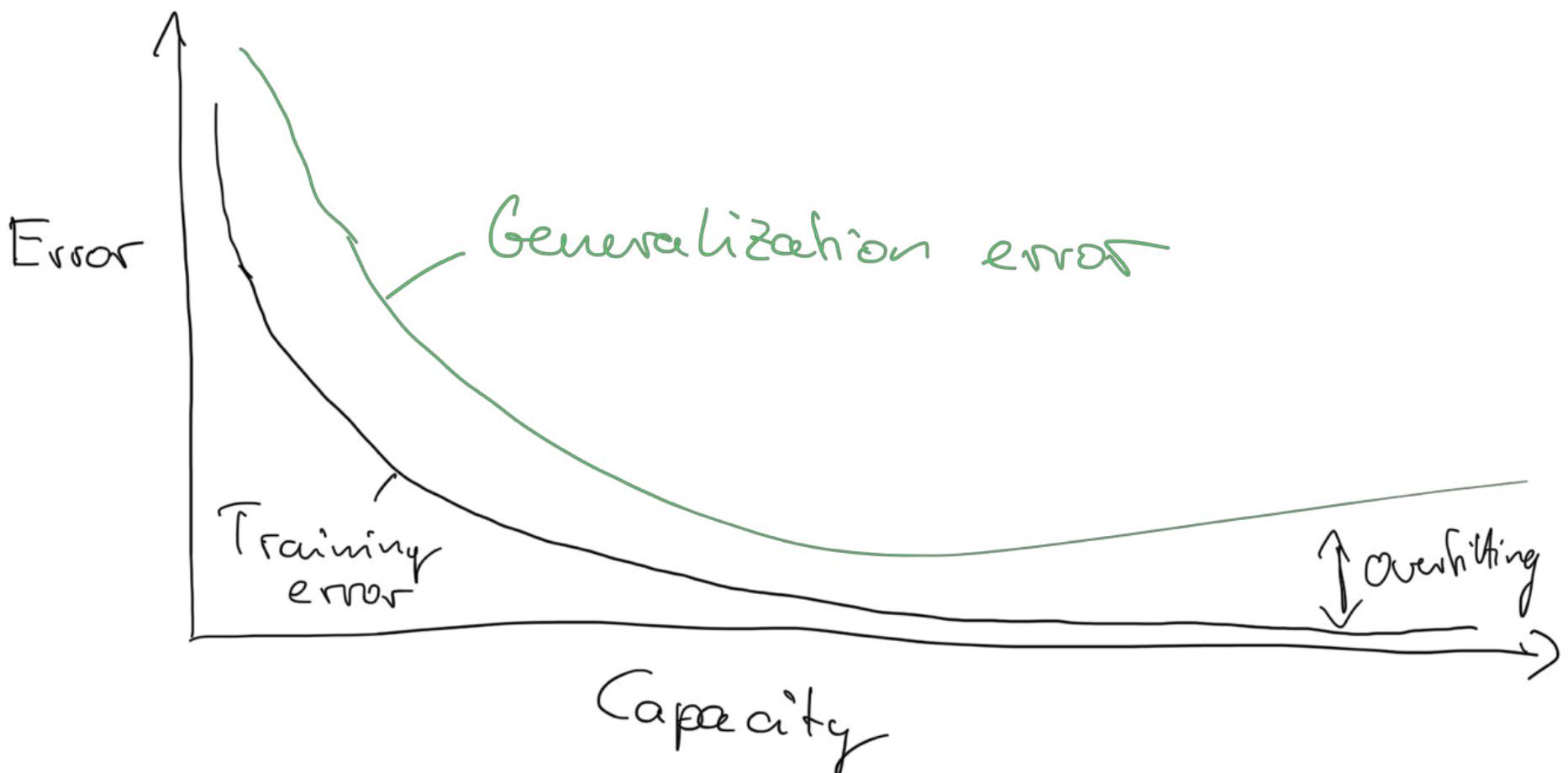
$$= \text{Bias}^2 + \text{Var}$$

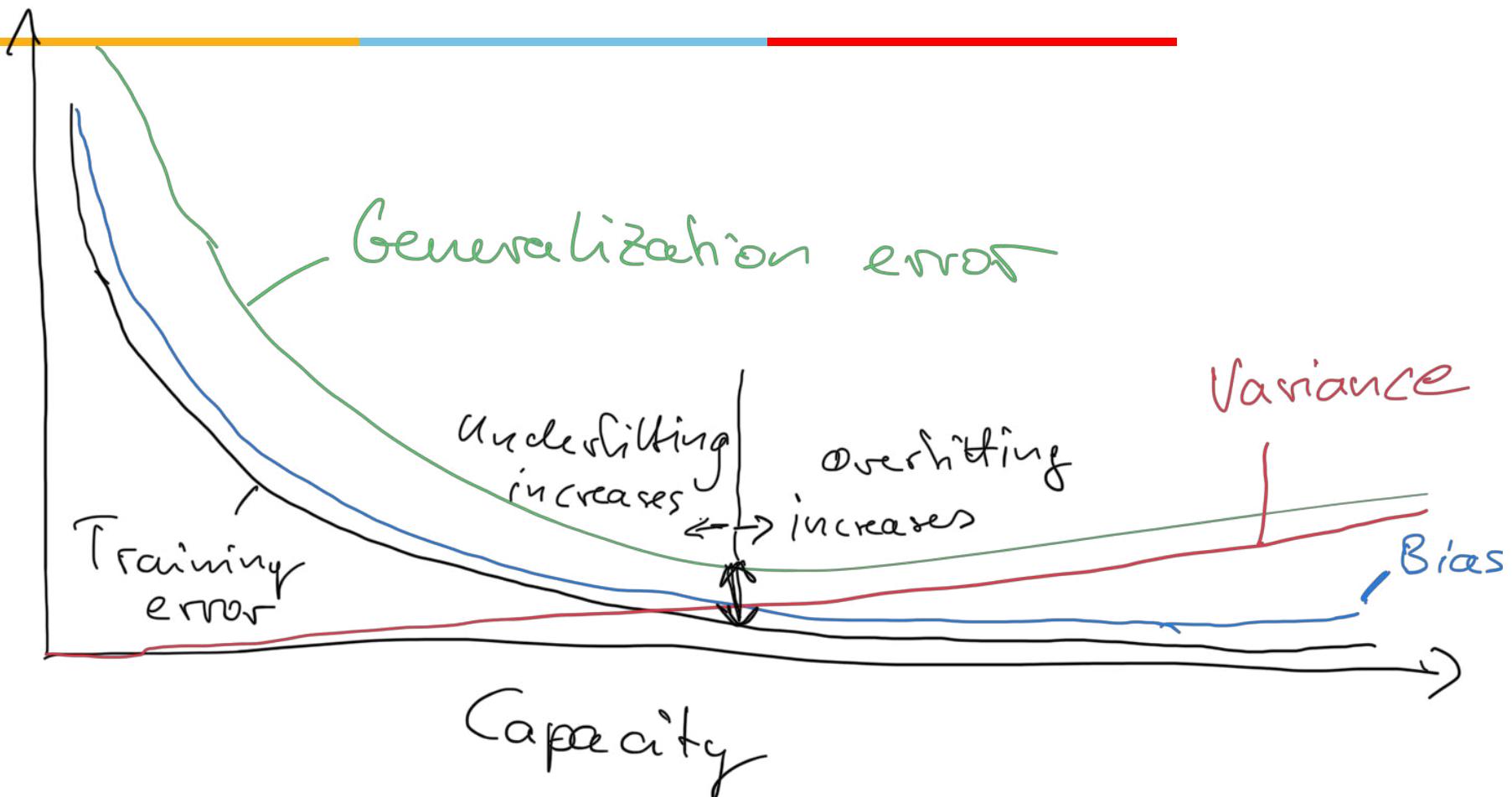
$$\text{Bias}[\hat{\theta}] = E[\hat{\theta}] - \theta$$

$$\text{Var}[\hat{\theta}] = E[\hat{\theta}^2] - (E[\hat{\theta}])^2$$

$$\text{Var}[\hat{\theta}] = E[(E[\hat{\theta}] - \hat{\theta})^2]$$

Now, how is this related to overfitting and underfitting?





Minimum Description Length Principle

Occam's razor: prefer the shortest hypothesis

MDL: prefer the hypothesis h that minimizes

$$h_{MDL} = \operatorname{argmin}_{h \in H} L_{C_1}(h) + L_{C_2}(D|h)$$

where $L_C(x)$ is the description length of x under encoding C

Example: H = decision trees hypothesis, D = training data labels

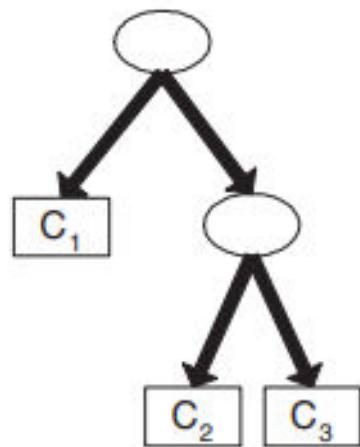
- $L_{C_1}(h)$ is # bits to describe tree h
- $L_{C_2}(D|h)$ is # bits to describe D given h
 - Note $L_{C_2}(D|h) = 0$ if examples classified perfectly by h . Need only describe exceptions
- Hence h_{MDL} trades off tree size for training errors

Minimum Description Length Principle

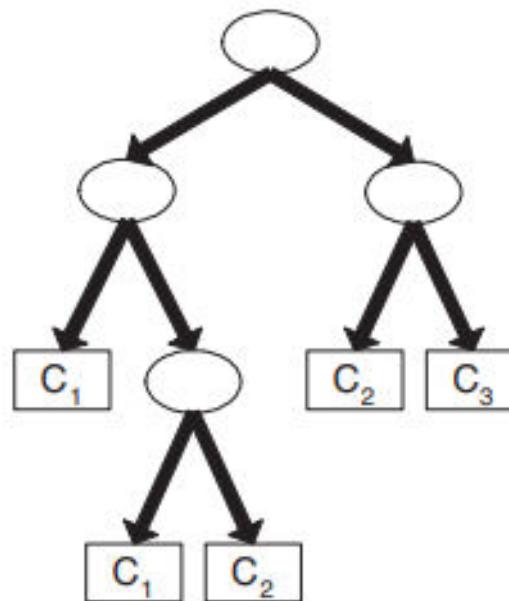
- MDL principle provides a way of trading off hypothesis complexity for the number of errors committed by the hypothesis.
- May select a shorter hypothesis that makes a few errors over a longer hypothesis that perfectly classifies the training data.
- Provides one method for dealing with the issue of overfitting the data.

Consider the decision trees shown in Figure . Assume they are generated from a data set that contains 16 binary attributes and 3 classes, C_1 , C_2 , and C_3 . Compute the total description length of each decision tree according to the minimum description length principle.

Which decision tree is better, according to the MDL principle?



(a) Decision tree with 7 errors



(b) Decision tree with 4 errors

Answer:

Because there are 16 attributes, the cost for each internal node in the decision tree is:

$$\log_2(m) = \log_2(16) = 4$$

Furthermore, because there are 3 classes, the cost for each leaf node is:

$$\lceil \log_2(k) \rceil = \lceil \log_2(3) \rceil = 2$$

The cost for each misclassification error is $\log_2(n)$.

The overall cost for the decision tree (a) is $2 \times 4 + 3 \times 2 + 7 \times \log_2 n = 14 + 7 \log_2 n$ and the overall cost for the decision tree (b) is $4 \times 4 + 5 \times 2 + 4 \times 5 = 26 + 4 \log_2 n$. According to the MDL principle, tree (a) is better than (b) if $n < 16$ and is worse than (b) if $n > 16$.

Decision Theory

- Suppose x is an input vector together with a corresponding vector t of target variables
- Goal: predict t given a new value for x .
- The joint probability distribution $p(x, t)$ provides a complete summary of the uncertainty associated with these variables.
- Determination of $p(x, t)$ from a set of training data is called ***inference*** and is a difficult problem.

Decision Theory

Inference step

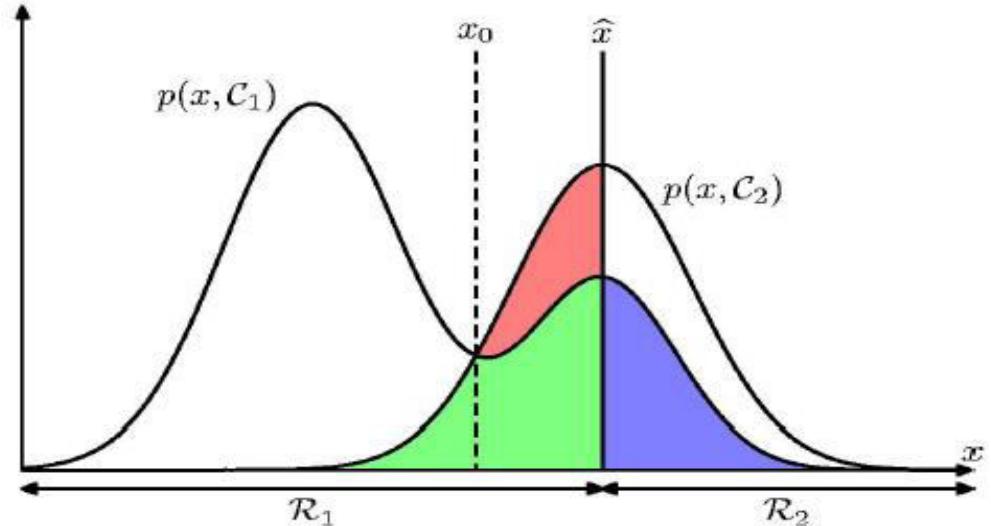
Determine either $p(t|\mathbf{x})$ or $p(\mathbf{x}, t)$.

Decision step

For given \mathbf{x} , determine optimal t .

Minimum Misclassification Rate

- Divide the input space into regions R_k called decision regions, one for each class, such that all points in R_k are assigned to class C_k
- Boundaries between decision regions are called decision boundaries or decision surfaces
- A mistake occurs when an input vector belonging to class C_1 is assigned to class C_2 or vice versa.



$$\begin{aligned}
 p(\text{mistake}) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \\
 &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x}.
 \end{aligned}$$

Minimum Misclassification Rate

$$\begin{aligned} p(\text{correct}) &= \sum_{k=1}^K p(\mathbf{x} \in \mathcal{R}_k, \mathcal{C}_k) \\ &= \sum_{k=1}^K \int_{\mathcal{R}_k} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x} \end{aligned}$$

Inference and Decision

- We have broken the **classification problem** down into two separate stages, the *inference stage* in which we use training data to learn a model for $p(C_k|x)$, and the subsequent *decision stage* in which we use these posterior probabilities to make optimal class assignments.
- An alternative possibility would be to solve both problems together and simply learn a function that maps inputs x directly into decisions. Such a function is called a *discriminant function*.
- **Three distinct approaches to solving decision problems**

Inference and Decision

1st approach

- Solve the inference problem of determining the posterior class probabilities $p(C_k|x)$, and then subsequently use decision theory to assign each new x to one of the classes.
- Approaches that model the posterior probabilities directly are called ***discriminative models***.
 - Logistic Regression

Inference and Decision

2nd approach

- Determine the class-conditional densities $p(\mathbf{x} | C_k)$ for each class C_k individually.
- Separately infer the prior class probabilities $p(C_k)$. Then use Bayes' theorem

$$p(C_k | \mathbf{x}) = \frac{p(\mathbf{x} | C_k)p(C_k)}{p(\mathbf{x})}$$

$$p(\mathbf{x}) = \sum_k p(\mathbf{x} | C_k)p(C_k)$$

Inference and Decision

3rd approach

- Find a function $f(x)$, called a ***discriminant function***, which maps each input x directly onto a class label.
- Example - For two-class problems, $f(\cdot)$ might be binary valued and such that $f = 0$ represents class C_1 , and $f = 1$ represents class C_2 . In this case, probabilities play no role.

Question 1

The sales of a company (in million dollars) for each year are shown in the table below.

- Find the least square regression line $y = a x + b$.
- Use the least squares regression line as a model to estimate the sales of the company in 2012.

x (year)	2005	2006	2007	2008	2009
y (sales)	12	19	29	37	45

$$\mathbf{y} = \mathbf{a} \cdot \mathbf{x} + \mathbf{b}$$

$$a = \frac{n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2}$$

$$b = \frac{1}{n} (\sum_{i=1}^n y_i - a \sum_{i=1}^n x_i)$$

Question 2

Suppose that the lifetime of *Badger* brand light bulbs is modeled by an exponential distribution with (unknown) parameter λ . We test 5 bulbs and find they have lifetimes of 2, 3, 1, 3, and 4 years, respectively. What is the estimate for λ ?

Question 3

Consider the hypothesis function $h(\mathbf{w}, \mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2$; with parameters

$$\mathbf{w} = \langle w_0, w_1, w_2, w_3, w_4 \rangle = \langle -20, -2, -4, 1, 1 \rangle.$$

Here x_1 and x_2 are two features.

- Derive the equation of the decision boundary $g(x_1, x_2)$ for logistic regression given by the equation:

$$y = \frac{1}{1+\exp\{-h(w, x)\}}$$

- Draw the decision boundary and predict the class labels [C_0 , C_1] for the examples given by A(-2, 2), B(6, 6) and C(-5, 5).

$$\omega = \langle w_0, w_1, w_2, w_3, w_4 \rangle = \langle -20, -2, -4, 1, 1 \rangle$$

$$h(\omega) = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2$$

$$\sum_{i=0}^4 w_i x_i \geq 0 \quad ; \quad w_0 = 0$$

for a decision boundary

Put $\sum_{i=0}^4 w_i x_i = 0$

$$\Rightarrow -20 - 2x_1 - 4x_2 + x_1^2 + x_2^2 = 0$$

$$x_1^2 + x_2^2 - 2x_1 - 4x_2 - 20 = 0$$

$$x_1^2 - 2x_1 + x_2^2 - 4x_2 = 20$$

$$x_1^2 - 2x_1 + 1 + x_2^2 - 4x_2 + 4 = 20 + 1 + 4$$

$$(x_1 - 1)^2 + (x_2 - 2)^2 = 25$$

~~Centre~~ $(1, 2)$, $\lambda = \sqrt{25} = 5$

[eqn of circle $\Rightarrow (x-h)^2 + (y-k)^2 = r^2$]

Centre (h, k) , radius r

$$x^2 + y^2 + 2gx + 2fy + c = 0$$

center = $(-g, -f)$, $r = \sqrt{g^2 + f^2 - c}$

	x_1	x_2
A	-2	2
B	6	6
C	-5	5

If points are inside the circle then
class ~~C1~~ C1

If points outside circle then
class CO

(1) A (-2, 2) and C (1, 2) find Euclidean
distance

$$d(A, C) = \sqrt{(2-1)^2 + (2-2)^2} = \sqrt{9+0} = 3$$

3 is < 5

$d(A, C) <$ radius so A is
inside boundary circle \therefore class = C1

B (6, 6) and C (1, 2).

$$d(B, C) = \sqrt{(6-1)^2 + (6-2)^2} = \sqrt{25+16} = \sqrt{41}$$

$d(B, C) >$ radius ($6 > 5$) so \therefore class \rightarrow CO

Question 4

Consider the loss function of linear regression given by: $J(\theta_0, \theta_1)$.

Given $(\theta_0, \theta_1) = 0, 0.5$, Estimate $\partial J / \partial \theta_1$ using the data points below:

x	2	4	7.0	8.0	10.0
y	1	2	2.5	3.5	5.5

Linear regression

$$(Q_0; Q_1) = (0, 0.5)$$

$$y = Q_0 + Q_1 x$$

$$y = 0.5x$$

x	1	2	4	7	8	10
y	1	2	2.5	3.5	5.5	

$$\frac{\partial J}{\partial Q_1} = \frac{1}{n} \sum_{i=1}^n [h_Q(x^{(i)}) - y^{(i)}] \cdot x_i^{(i)}$$

$$\frac{1}{5} \left[[(0.5 \times 2) - 1]^2 \cdot 2 + [(4 \times 0.5) - 2]^2 \cdot 4 + [(7 \times 0.5) - 2.5]^2 \cdot 7 + [(8 \times 0.5) - 3.5]^2 \cdot 8 + [(10 \times 0.5) - 5.5]^2 \cdot 10 \right]$$

$$= \cancel{\frac{1}{5}} \left[0 + 0 + 7 + 2 + 2.5 \right]$$

$$= \underline{\underline{2.3}}$$

Consider the following dataset with predictor x and response variable y. Linear regression is used to fit the model $y = a + b*x$ to this dataset. Distinct samples (of size 2 or 3) of instances are used to obtain the estimates of b. What is the expected value of b? Choose the closest answer.

Hint: Here subsets means all possible data samples i.e. obs (1,2),(2,3),(1,3), (1,2,3) . Each sample allow us to estimate coefficients and expectation of coefficient is desired

x	y
0	2
1	4
2	7

Consider the following dataset with predictor x and response variable y . Linear regression is used to fit the model $y = a + b \cdot x$ to this entire dataset. If L2 regularization with regularization constant $\alpha = 1$ is used what will be the value a .

x	y
0	2
1	4
2	7

Thank you

Good References

Decision Tree

- https://www.youtube.com/watch?v=eKD5gxPPeY0&list=PLBv09BD7ez_4temBw7vLA19p3tdQH6FYO&index=1

Overfitting

- https://www.youtube.com/watch?time_continue=1&v=t56Nid85Thg
- <https://www.youtube.com/watch?v=y6SpA2Wuyt8>

Random Forest

- <https://www.stat.berkeley.edu/~breiman/RandomForests/>



BITS Pilani
Pilani Campus

Support Vector Machines

Dr. Sugata Ghosal

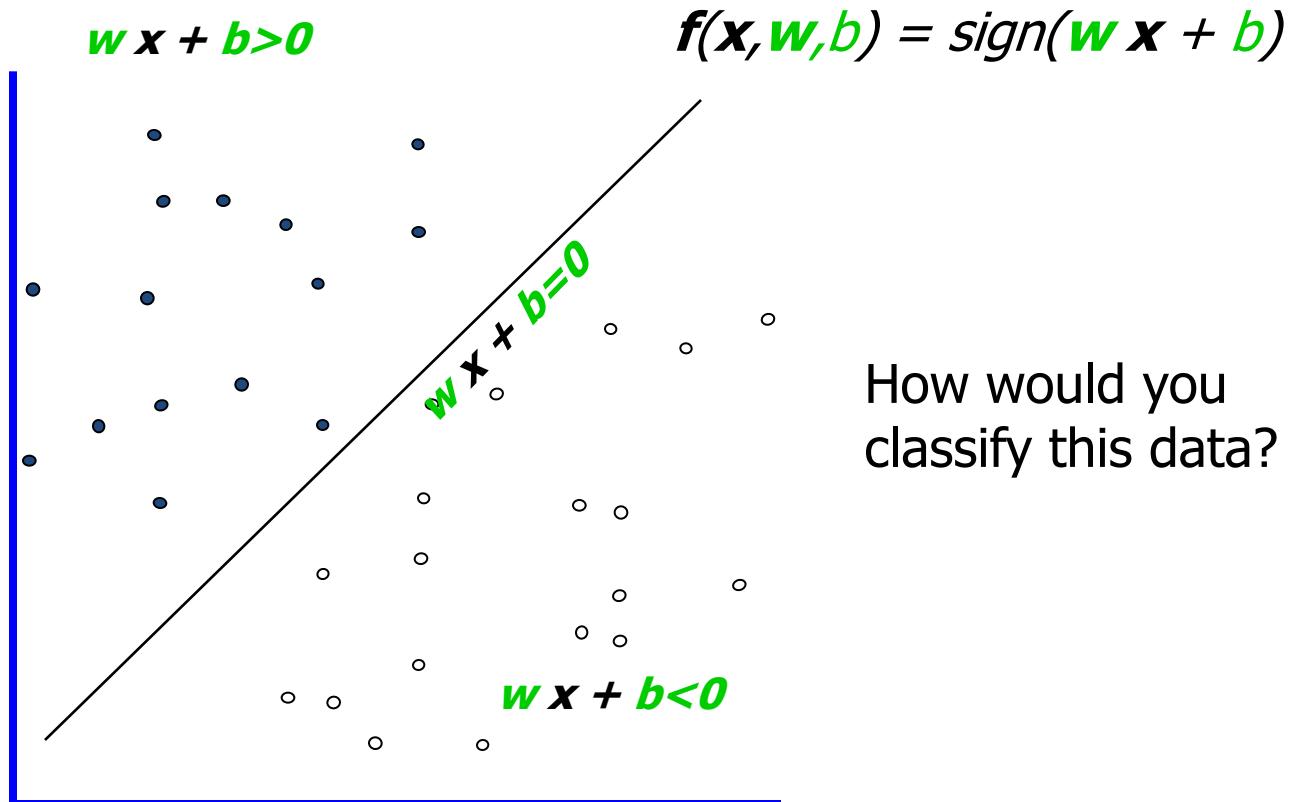
sugata.ghosal@pilani.bits-pilani.ac.in

SVM - I

- Linear Classifiers
- Maximum Margin Classification
- Linear SVM
- SVM optimization problem
- Soft Margin SVM

Linear Classifiers

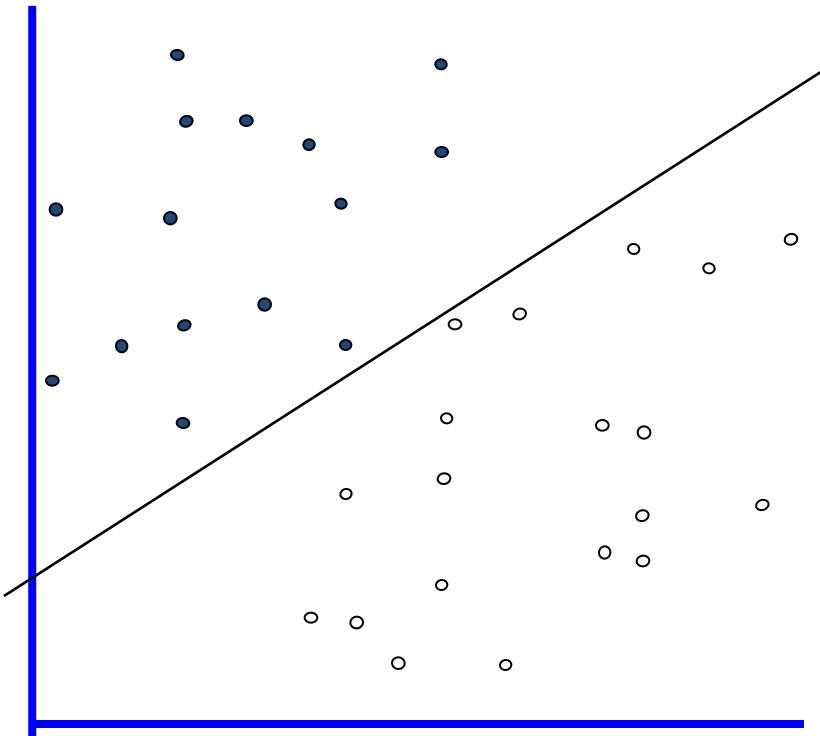
- denotes +1
- denotes -1



Linear Classifiers

$$\mathbf{f}(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

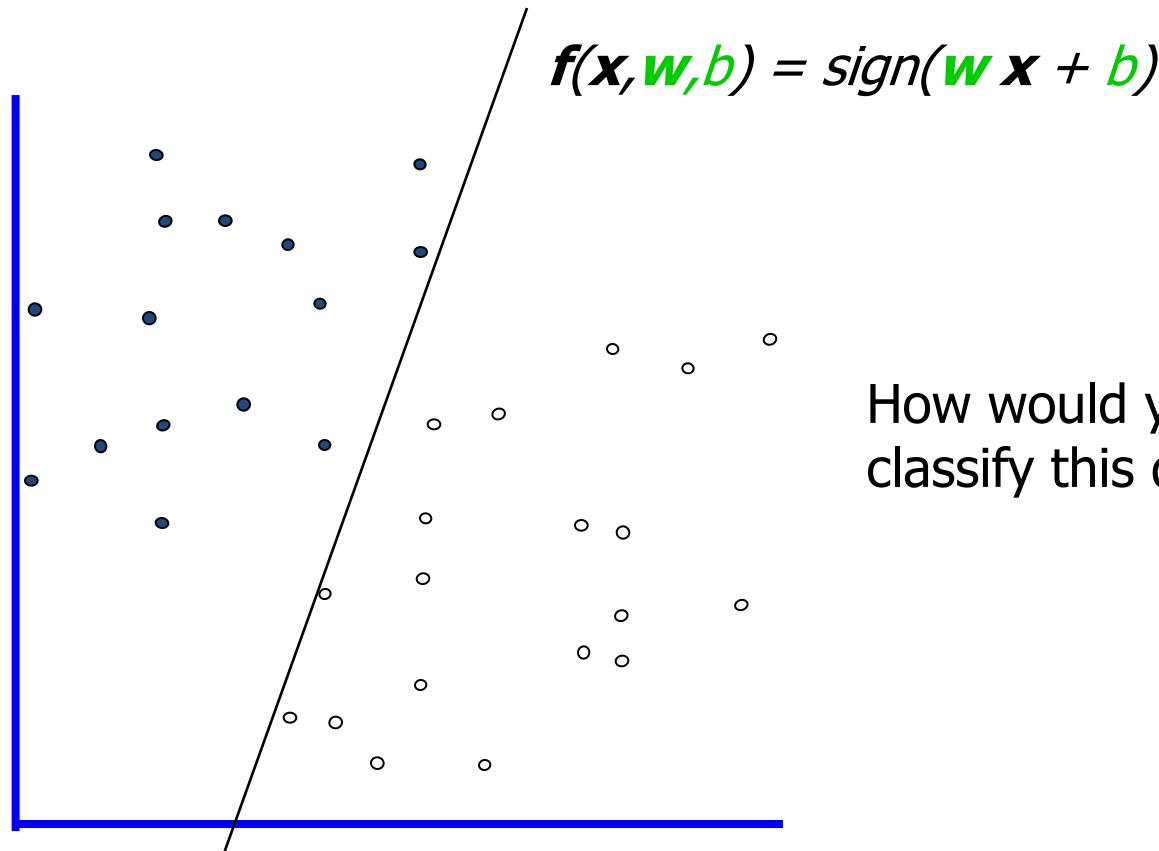
- denotes +1
- denotes -1



How would you
classify this data?

Linear Classifiers

- denotes +1
- denotes -1

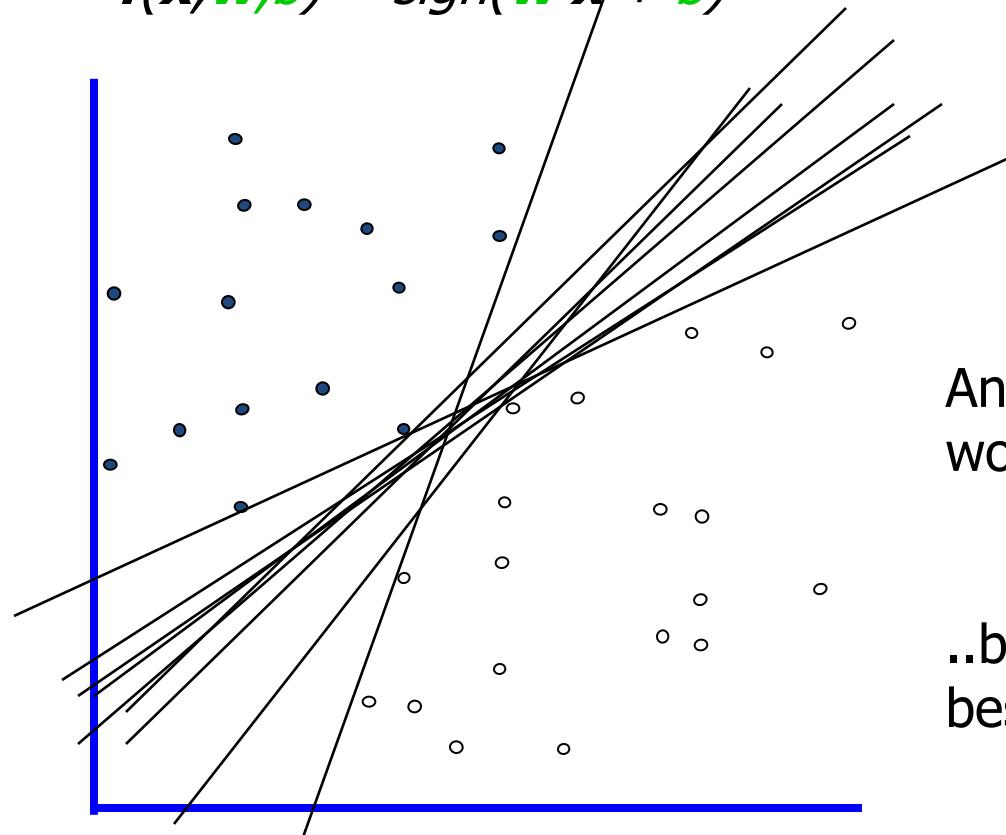


How would you
classify this data?

Linear Classifiers

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

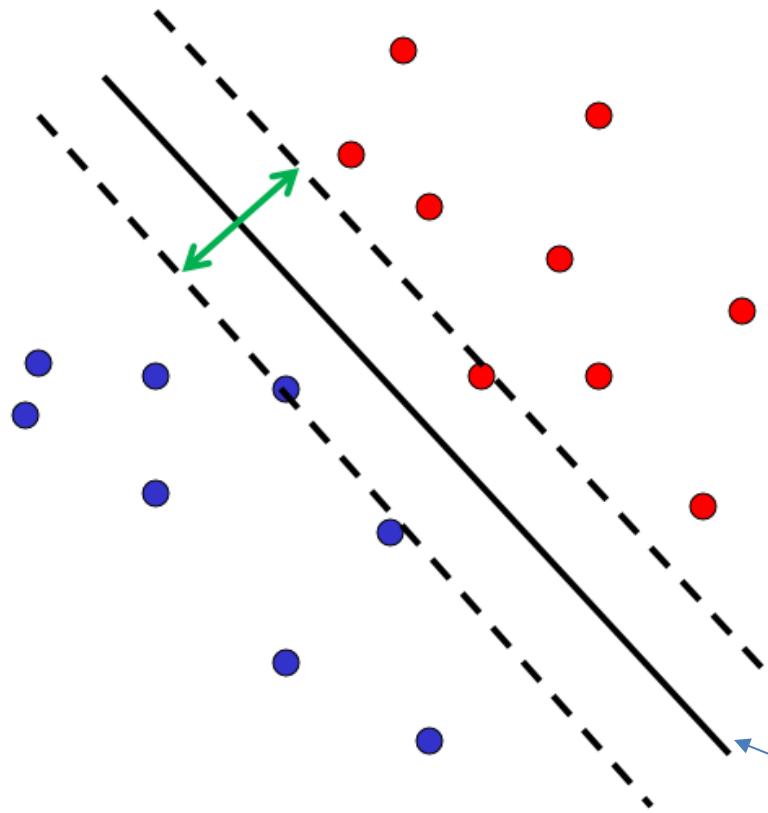
- denotes +1
- denotes -1



Any of these
would be fine..

..but which is
best?

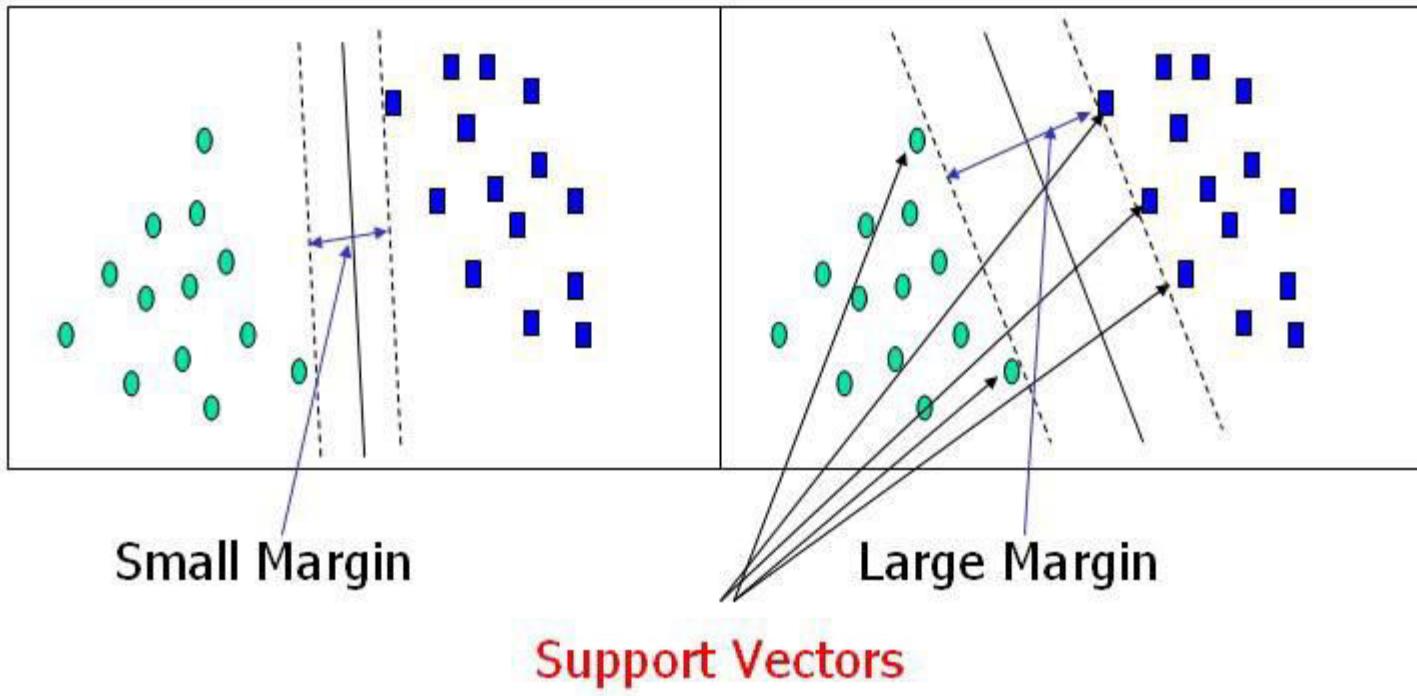
Linear Classifier



- Discriminative classifier based on *optimal separating line (for 2d case)*
- Maximize the *margin* between the positive and negative training examples

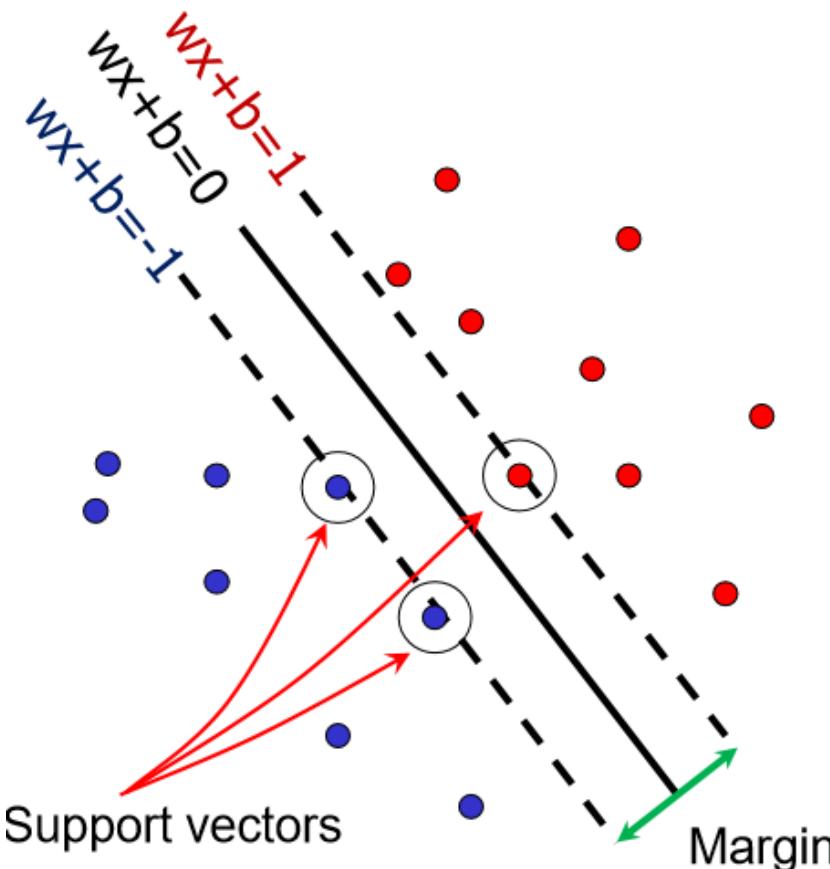
C. Burges, [A Tutorial on Support Vector Machines for Pattern Recognition](#), Data Mining and Knowledge Discovery, 1998

Large margin and support vectors



Support Vector Machines

- Want line that maximizes the margin.



\mathbf{x}_i positive ($y_i = 1$): $\mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

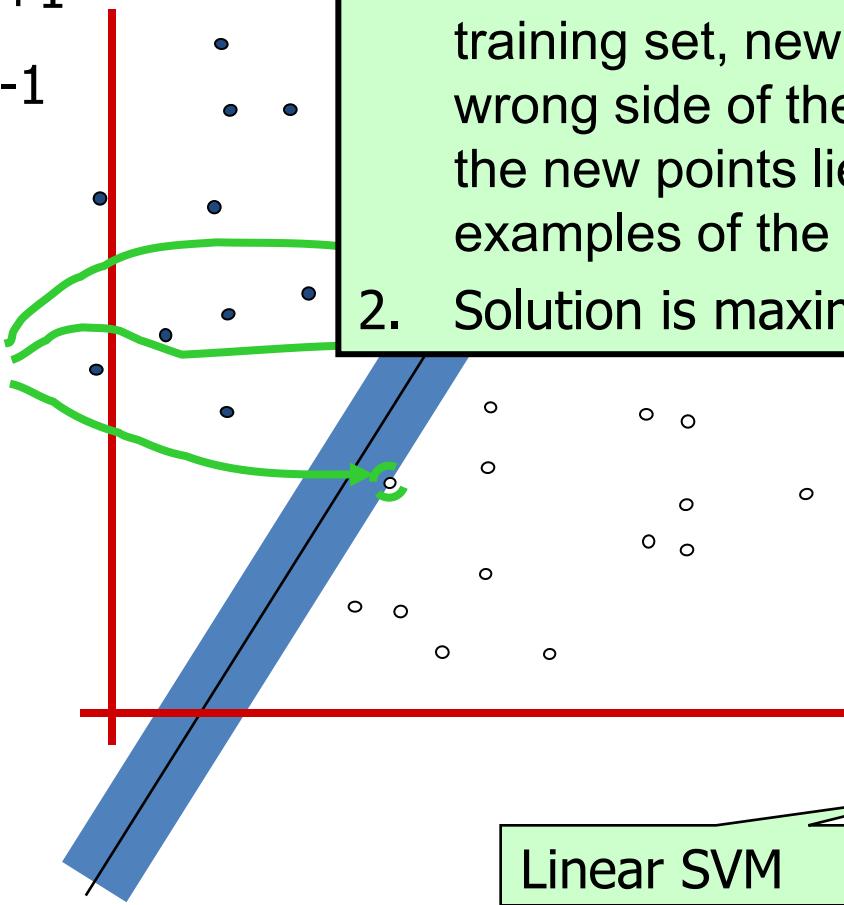
\mathbf{x}_i negative ($y_i = -1$): $\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support vectors, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Maximum Margin

- denotes +1
- denotes -1

Support Vectors
are those
datapoints that
the margin
pushes up
against



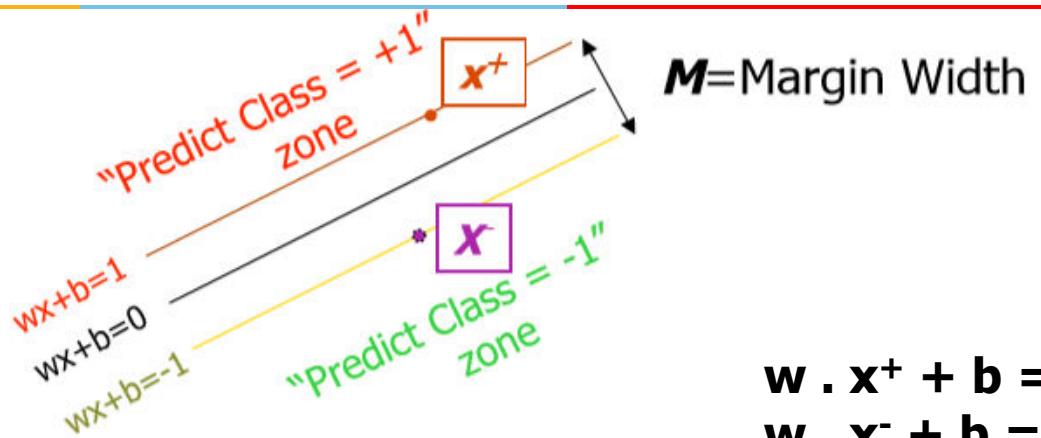
1. If hyperplane is oriented such that it is close to some of the points in your training set, new data may lie on the wrong side of the hyperplane, even if the new points lie close to training examples of the correct class.
2. Solution is maximizing the margin with the, maximum margin.

This is the
simplest kind of
SVM (Called an
LSVM)

Support Vectors

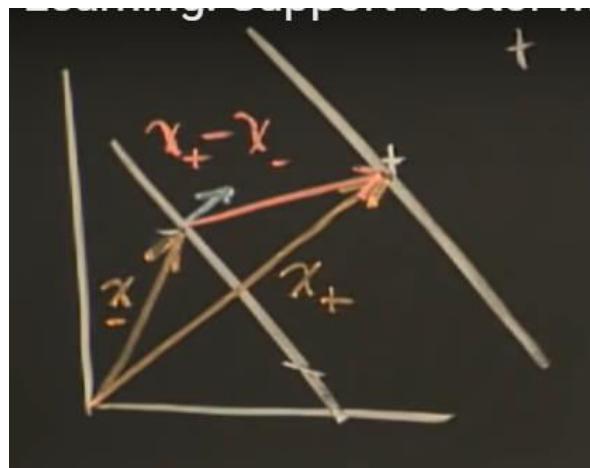
- Geometric description of SVM is that the max-margin hyperplane is completely determined by those points that lie nearest to it.
- Points that lie on this margin are the support vectors.
- The points of our data set which if removed, would alter the position of the dividing hyperplane

Linear SVM Mathematically



$$\mathbf{w} \cdot \mathbf{x}^+ + \mathbf{b} = +1$$

$$\mathbf{w} \cdot \mathbf{x}^- + \mathbf{b} = -1$$



Margin width

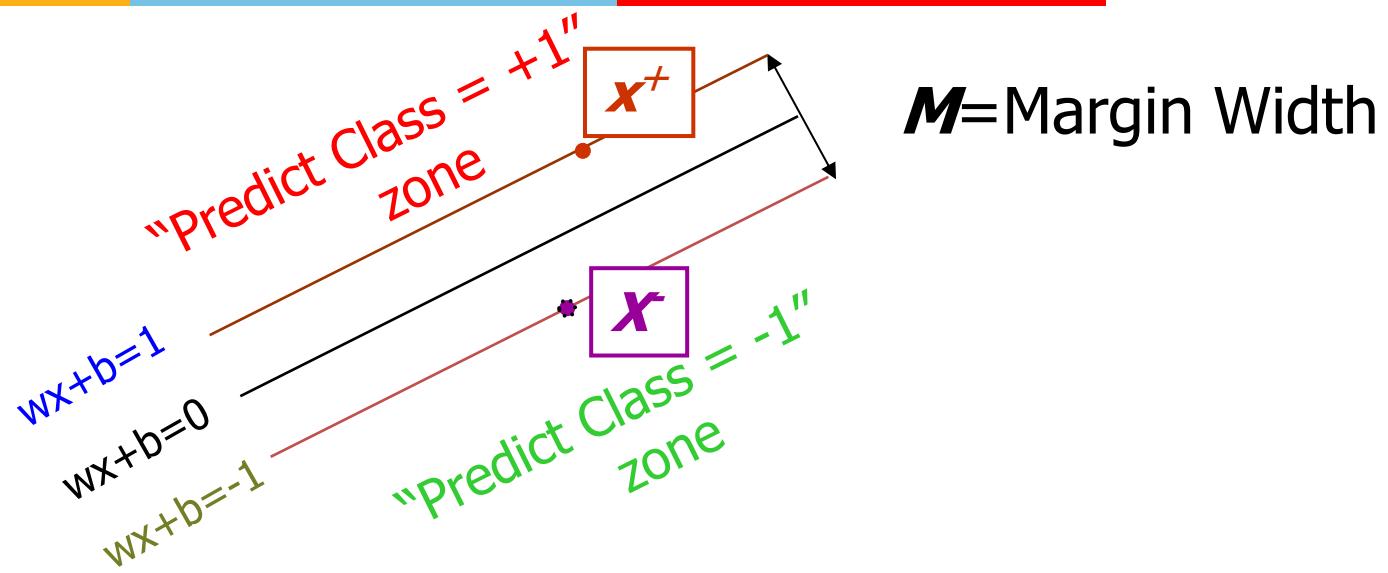
$$= \mathbf{x}^+ - \mathbf{x}^- \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

$$= \frac{\mathbf{w} \cdot \mathbf{x}^+ - \mathbf{w} \cdot \mathbf{x}^-}{\|\mathbf{w}\|}$$

$$= (1-\mathbf{b}) - (-1-\mathbf{b}) / \|\mathbf{w}\|$$

$$= \frac{2}{\|\mathbf{w}\|}$$

Linear SVM Mathematically



Distance between lines given by solving linear equation:

What we know:

- $\mathbf{w} \cdot \mathbf{x}^+ + b = +1$
- $\mathbf{w} \cdot \mathbf{x}^- + b = -1$

$$\text{Maximize margin: } M = \frac{2}{\|\mathbf{w}\|}$$

$$\text{Equivalent to minimize: } \frac{1}{2} \|\mathbf{w}\|^2$$

Solving the Optimization Problem

1. Maximize margin $2/\|\mathbf{w}\|$
2. Correctly classify all training data points:

$$\mathbf{x}_i \text{ positive } (y_i = 1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

Quadratic optimization problem:

Find \mathbf{w} and b such that

$\Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$ is minimized;

and for all $\{(\mathbf{x}_i, y_i)\}$: $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

$$+1 (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

$$-1 (\mathbf{w}^T \mathbf{x}_i + b) \leq 1$$

$$\text{same as } (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

Solving the Optimization Problem



Find w and b such that

$\Phi(w) = \frac{1}{2}\|w\|^2$ is minimized; Type equation here.

and for all $\{(x_i, y_i)\}$: $y_i (w^T x_i + b) \geq 1$

← Primal

- Need to optimize a *quadratic function subject to linear inequality constraints*.
- All constraints in SVM are linear
- Quadratic optimization problems are a well-known class of mathematical programming problems, and many (rather intricate) algorithms exist for solving them.
- The solution involves constructing a *unconstrained problem* where a *Lagrange multiplier α_i* is associated with every constraint in the primary problem:

Karush–Kuhn–Tucker (KKT) theorem

- KKT approach to nonlinear programming (quadratic) generalizes the method of Lagrange multipliers, which allows only equality constraints.
- KKT allows inequality constraints

Karush-Kuhn-Tucker (KKT) conditions

- Start with

$\max f(x)$ subject to

$g_i(x) = 0$ and $h_j(x) \geq 0$ for all i, j

- Make the Lagrangian function

$$\mathcal{L} = f(x) - \sum_i \lambda_i g_i(x) - \sum_j \mu_j h_j(x)$$

- Take gradient and set to 0 – but other conditions also.

KKT conditions

- Make the Lagrangian function

$$\mathcal{L} = f(x) - \sum_i \lambda_i g_i(x) - \sum_j \mu_j h_j(x)$$

- Necessary conditions to have a minimum are

$$\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) = 0$$

$$g_i(x^*) = 0 \text{ for all } i$$

$$h_j(x^*) \geq 0 \text{ for all } j$$

$$\mu_j \geq 0 \text{ for all } j$$

$$\mu_j^* h_j(x^*) = 0 \text{ for all } j$$

Solving the Optimization Problem

- The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:

$$L(w, b, \alpha_i) = \frac{1}{2} \|w\|^2 - \sum \alpha_i [y_i (w^T x_i + b) - 1]$$

- Taking partial derivative with respect to w , $\frac{\partial L}{\partial w} = 0$
 - $w - \sum \alpha_i y_i x_i = 0$
 - $w = \sum \alpha_i y_i x_i$
- Taking partial derivative with respect to b , $\frac{\partial L}{\partial b} = 0$
 - $-\sum \alpha_i y_i = 0$
 - $\sum \alpha_i y_i = 0$

Support Vectors

Using KKT conditions :

$$\alpha_i [y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1] = 0$$

For this condition to be satisfied
either $\alpha_i = 0$ and $y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 > 0$

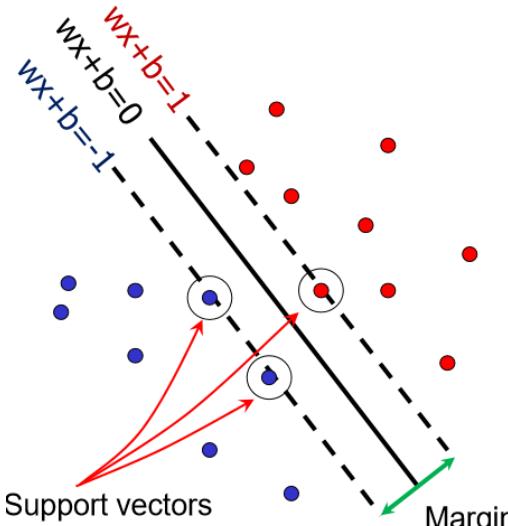
OR

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 = 0 \text{ and } \alpha_i > 0$$

For support vectors:
 $y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 = 0$

For all points other than
support vectors:
 $\alpha_i = 0$

- Want line that maximizes the margin.



$$\mathbf{x}_i \text{ positive } (y_i = 1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

$$\text{For support vectors, } \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$$

$$L(\mathbf{w}, b, \alpha_i) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum \alpha_i [y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1]$$

Solving the Optimization Problem

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

Learned
weight

Support
vector

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

Learned weight Support vector

Solving the Optimization Problem

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$
 $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ (for any support vector)

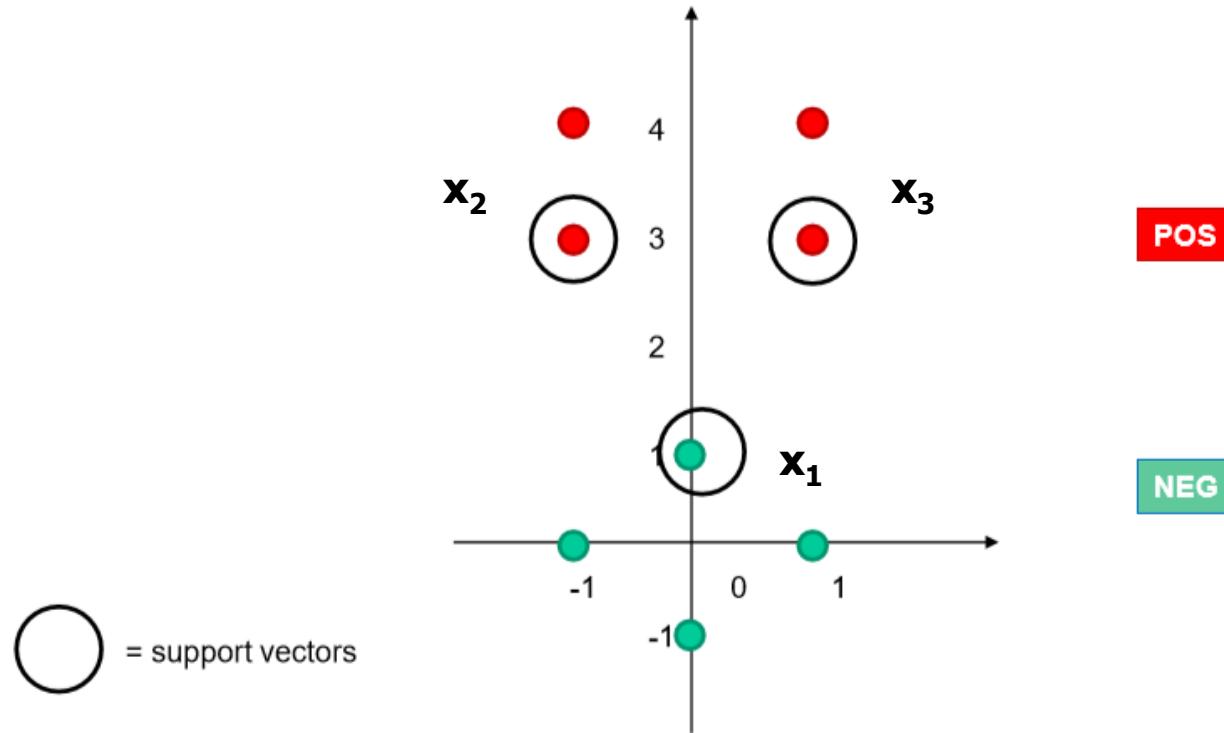
- Classification function:

$$\begin{aligned} f(\mathbf{x}) &= \text{sign} (\mathbf{w} \cdot \mathbf{x} + b) \\ &= \text{sign} \left(\sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b \right) \end{aligned}$$

If $f(\mathbf{x}) < 0$, classify as negative, otherwise classify as positive.

- Notice that it relies on an *inner product* between the test point \mathbf{x} and the support vectors \mathbf{x}_i
- (Solving the optimization problem also involves computing the inner products $\mathbf{x}_i \cdot \mathbf{x}_j$ between all pairs of training points)

Example



Example adapted from Dan Ventura

Solving for α

- We know that for the support vectors, $f(x) = 1$ or -1 exactly
- Add a 1 in the feature representation for the bias
- The support vectors have coordinates and labels:
 - $x_1 = [0 \ 1 \ 1]$, $y_1 = -1$
 - $x_2 = [-1 \ 3 \ 1]$, $y_2 = +1$
 - $x_3 = [1 \ 3 \ 1]$, $y_3 = +1$
- Thus we can form the following system of linear equations:

Solving for α

- System of linear equations:

$$\alpha_1 y_1 \text{dot}(x_1, x_1) + \alpha_2 y_2 \text{dot}(x_1, x_2) + \alpha_3 y_3 \text{dot}(x_1, x_3) = y_1$$

$$\alpha_1 y_1 \text{dot}(x_2, x_1) + \alpha_2 y_2 \text{dot}(x_2, x_2) + \alpha_3 y_3 \text{dot}(x_2, x_3) = y_2$$

$$\alpha_1 y_1 \text{dot}(x_3, x_1) + \alpha_2 y_2 \text{dot}(x_3, x_2) + \alpha_3 y_3 \text{dot}(x_3, x_3) = y_3$$

$$-2 * \alpha_1 + 4 * \alpha_2 + 4 * \alpha_3 = -1$$

$$-4 * \alpha_1 + 11 * \alpha_2 + 9 * \alpha_3 = +1$$

$$-4 * \alpha_1 + 9 * \alpha_2 + 11 * \alpha_3 = +1$$

$$\alpha_i [-1 (\mathbf{w} \cdot \mathbf{x}_i + b)] = -1$$

$$\alpha_i [+1 (\mathbf{w} \cdot \mathbf{x}_i + b)] = 1$$

- Solution: $\alpha_1 = 3.5$, $\alpha_2 = 0.75$, $\alpha_3 = 0.75$
-

Solving for w and b

We know $w = \alpha_1 y_1 x_1 + \dots + \alpha_N y_N x_N$ where $N = \# \text{ SVs}$

$$\text{Thus } w = -3.5 * [0 \ 1 \ 1] + 0.75 [-1 \ 3 \ 1] + 0.75 [1 \ 3 \ 1] = [0 \ 1 \ -2]$$

Separating out weights and bias, we have: $w = [0 \ 1]$ and $b = -2$

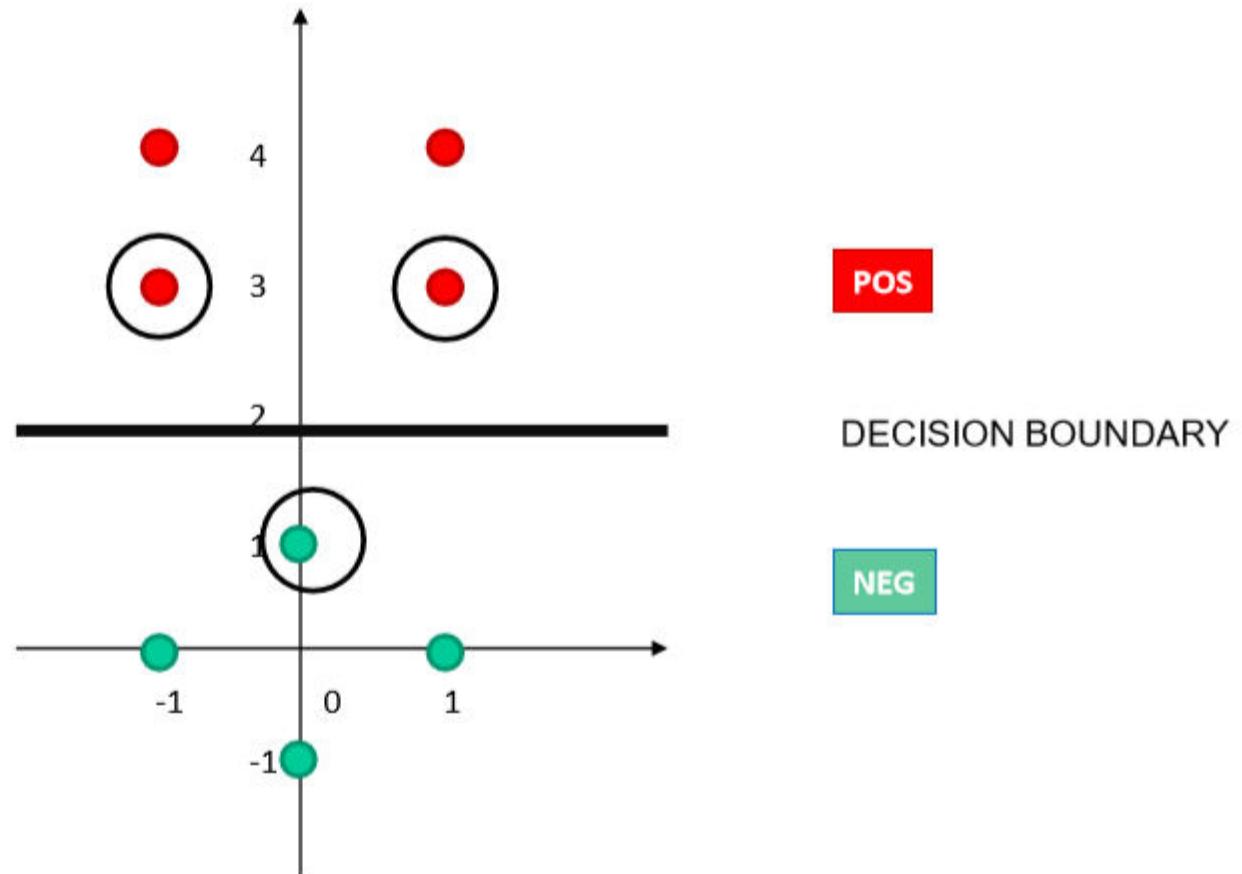
For SVMs, we used this eq for a line: $ax + cy + b = 0$
where $w = [a \ c]$

$$\text{Thus } ax + b = -cy \rightarrow y = (-a/c)x + (-b/c)$$

$$\text{Thus y-intercept is } -(-2)/1 = 2$$

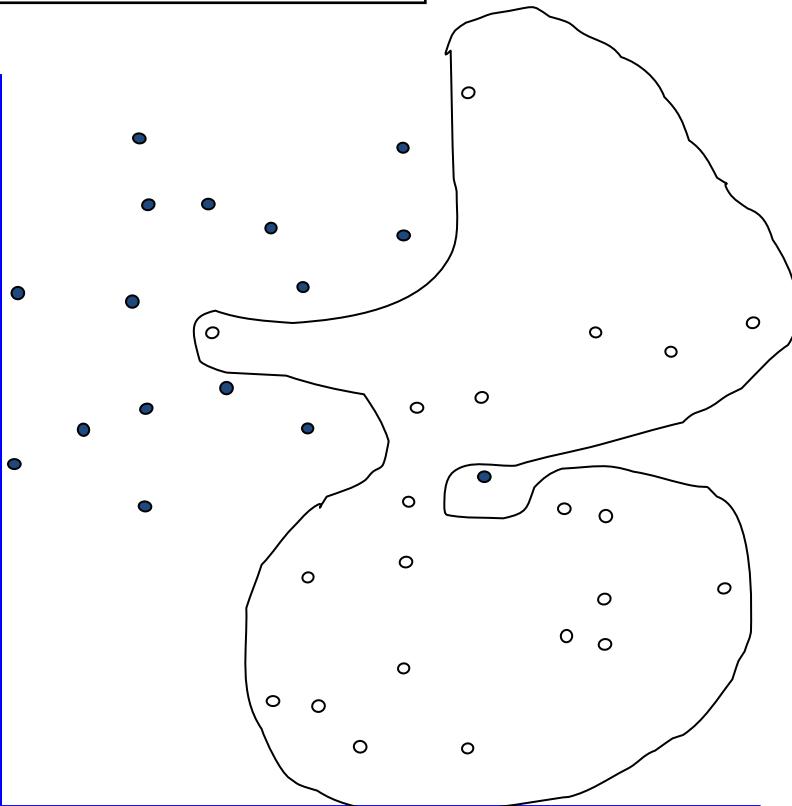
The decision boundary is perpendicular to w and it has slope $-0/1 = 0$

Decision boundary



Dataset with noise

- denotes +1
- denotes -1



- **Hard Margin:** So far we require all data points be classified correctly
 - No training error
- **What if the training set is noisy?**

Soft Margin Classification

Slack variables ξ_i can be added to allow misclassification of difficult or noisy examples.

What should our quadratic optimization criterion be?

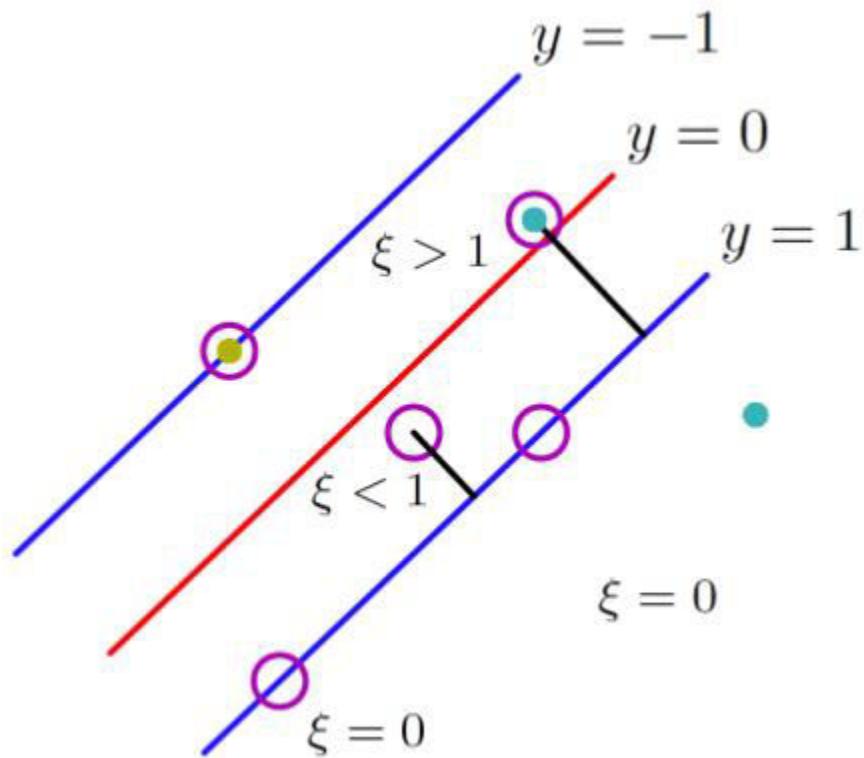
Minimize

$$\frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{k=1}^R \xi_k$$

Slack Variable

- **Slack variable** as giving the classifier some leniency when it comes to moving around points near the **margin**.
- When C is large, larger slacks penalize the objective function of SVM's more than when C is small.

Soft margin example



Soft Margin

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i$$

The \mathbf{w} that minimizes...
Maximize margin Minimize misclassification

Misclassification cost
data samples
Slack variable
 N
 ξ_i

subject to $y_i \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i,$
 $\xi_i \geq 0, \quad \forall i = 1, \dots, N$

Hard Margin versus Soft Margin

- **Hard Margin:**

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$
$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

- **Soft Margin incorporating slack variables:**

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_i \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$
$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0 \text{ for all } i$$

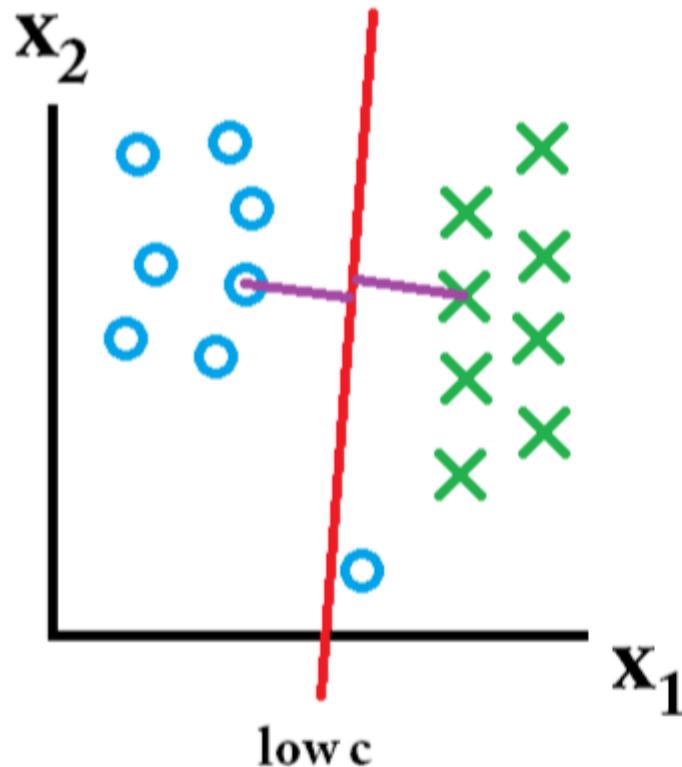
- **Parameter C can be viewed as a way to control overfitting.**

Value of C parameter

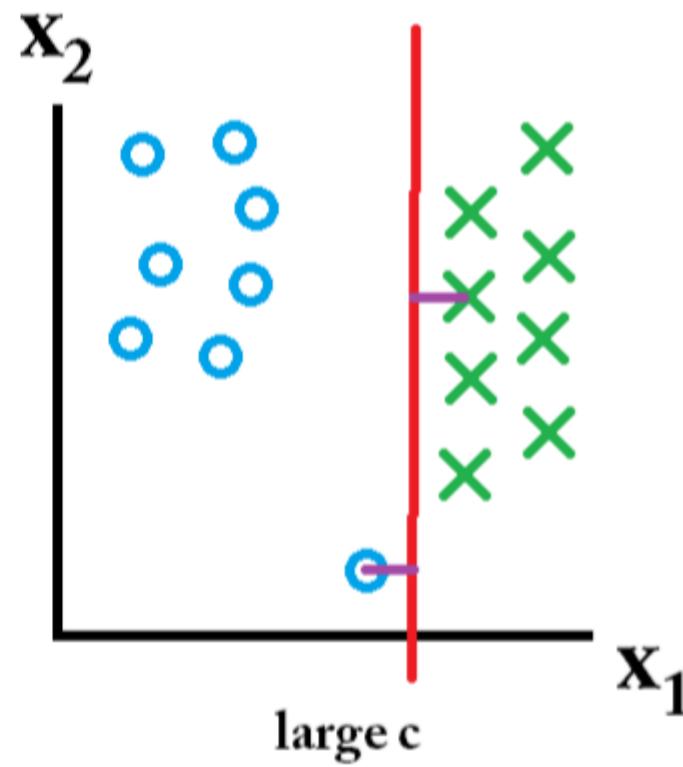
- C parameter tells the SVM optimization how much you want to avoid misclassifying each training example.
 - For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly.
 - Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.
-

Effect of Margin size v/s misclassification cost

Training set



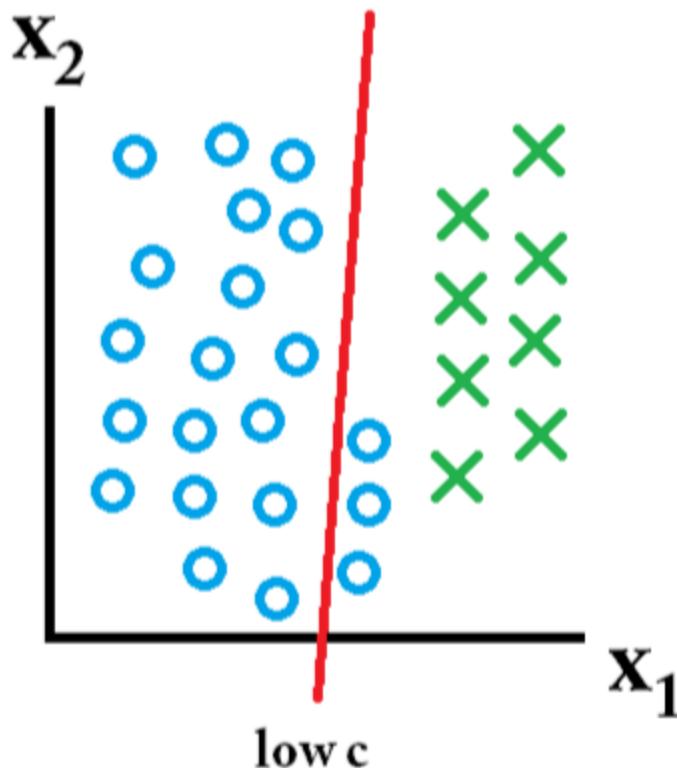
Misclassification ok, want large margin



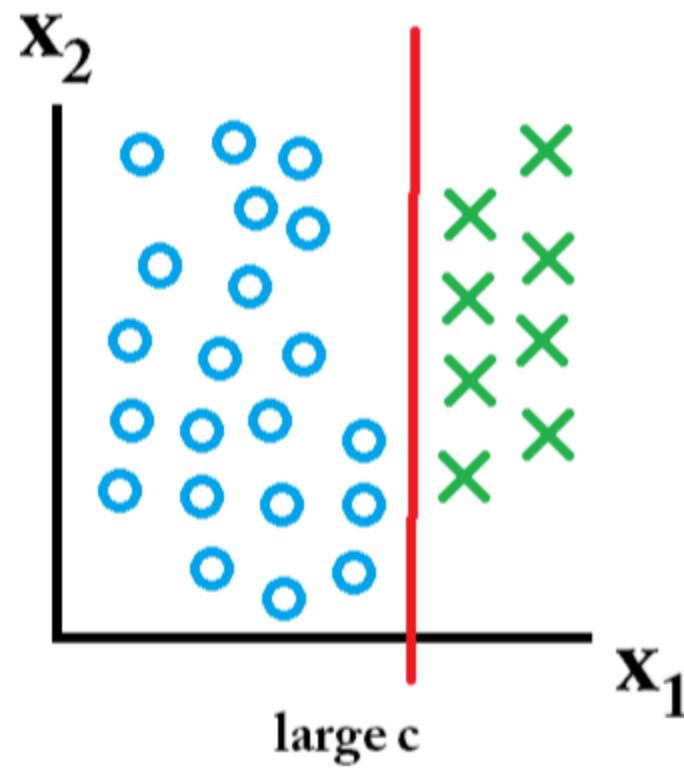
Misclassification not ok

Effect of Margin size v/s misclassification cost

Including test set A



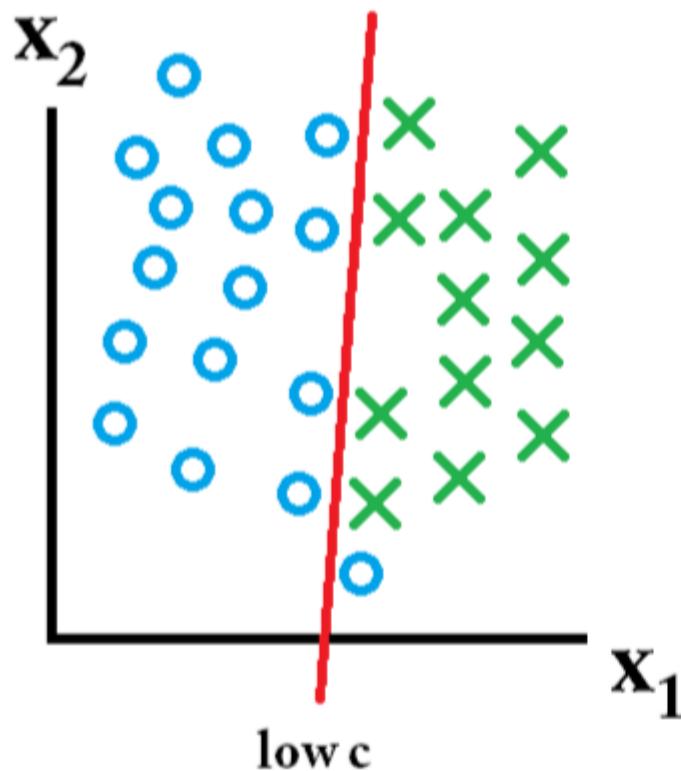
Misclassification ok, want large margin



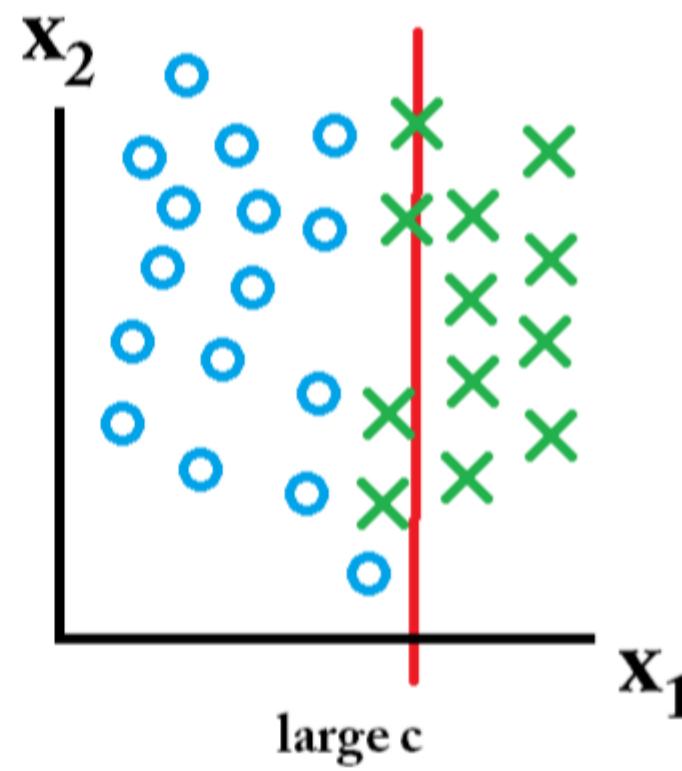
Misclassification not ok

Effect of Margin size v/s misclassification cost

Including test set B



Misclassification ok, want large margin



Misclassification not ok

Good Web References for SVM

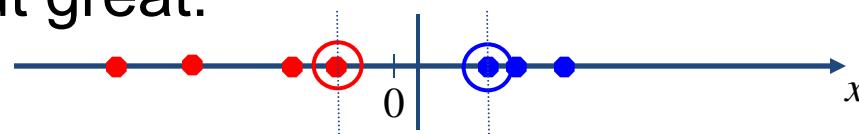
- **Text categorization with Support Vector Machines: learning with many relevant features** - T. Joachims, ECML
- **A Tutorial on Support Vector Machines for Pattern Recognition**, Kluwer Academic Publishers - Christopher J.C. Burges
- <http://www.cs.utexas.edu/users/mooney/cs391L/>
- <https://www.coursera.org/learn/machine-learning/home/week/7>
- <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>
- <https://data-flair.training/blogs/svm-kernel-functions/>
- [MIT 6.034 Artificial Intelligence, Fall 2010](https://mit-6.034-artificial-intelligence-fall-2010.readthedocs.io/en/latest/lectures/lec10.html)
- <https://stats.stackexchange.com/questions/30042/neural-networks-vs-support-vector-machines-are-the-second-definitely-superior>
- <https://www.sciencedirect.com/science/article/abs/pii/S0893608006002796>
- <https://medium.com/deep-math-machine-learning-ai/chapter-3-support-vector-machine-with-math-47d6193c82be>
- [Radial basis kernel](#)

SVM - II

-
- Nonlinear SVM
 - Kernel Trick
 - SVM Kernels
 - Multi-Class Problem
 - SVM vs Logistic Regression
 - SVM Applications
-

Non-linear SVMs

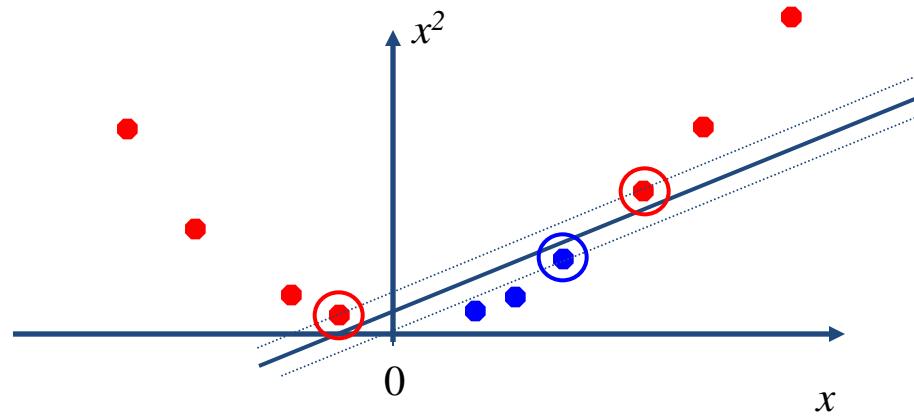
- Datasets that are linearly separable with some noise soft margin work out great:



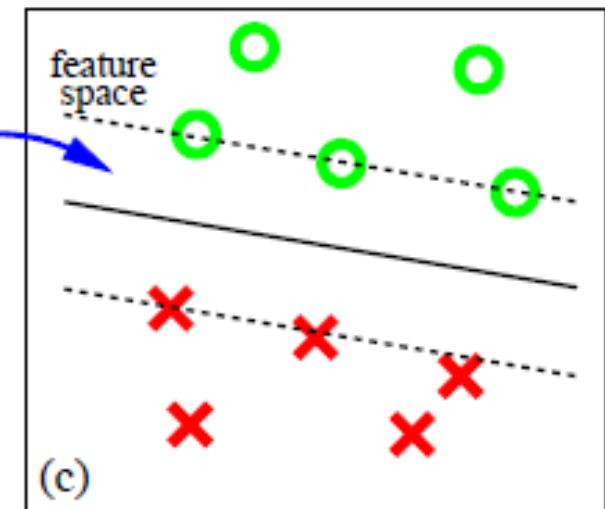
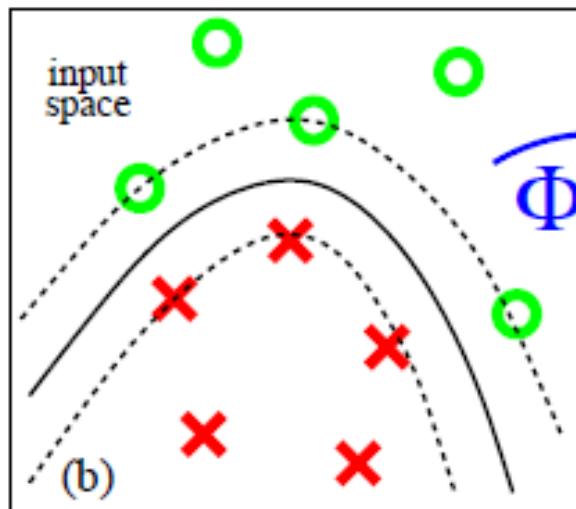
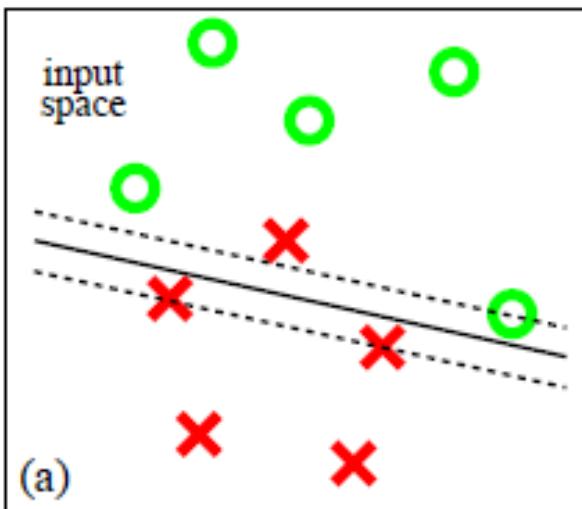
- But what are we going to do if the dataset is just too hard?



- How about... mapping data to a higher-dimensional space:

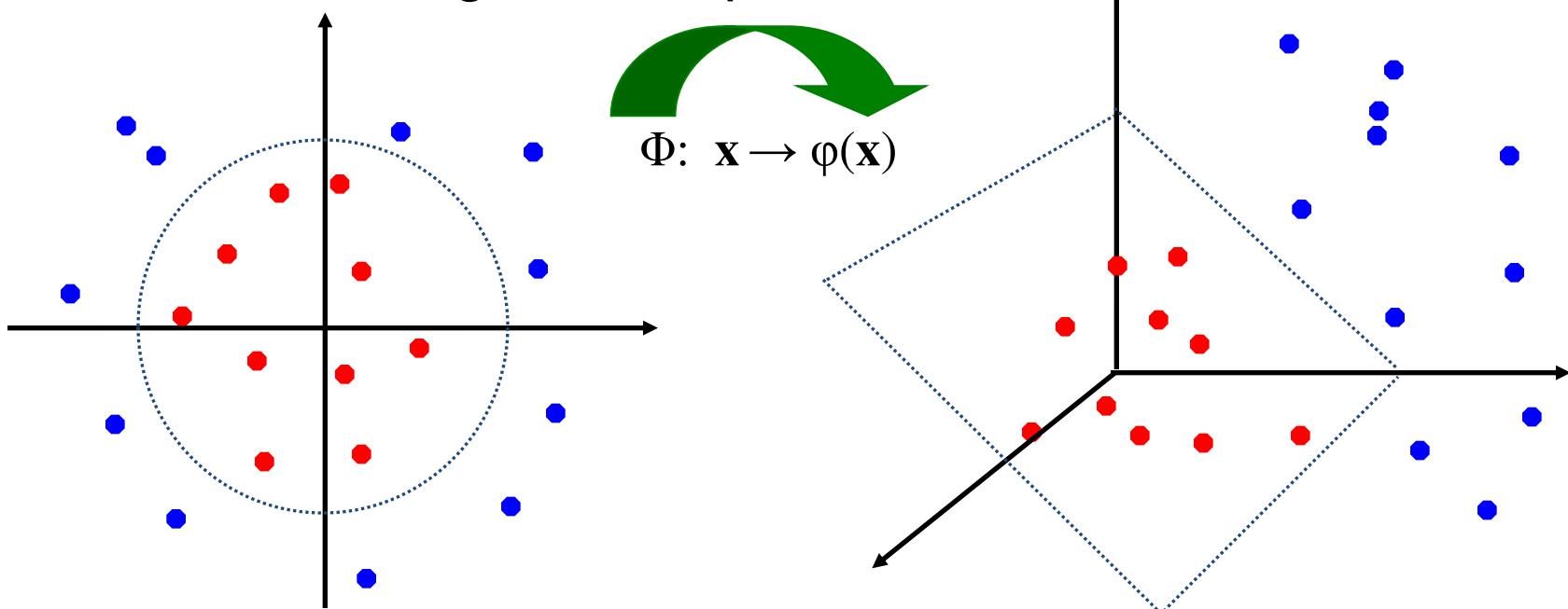


Find a feature space



Non-linear SVMs: Feature spaces

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



SVM – Overlapping Class Scenario

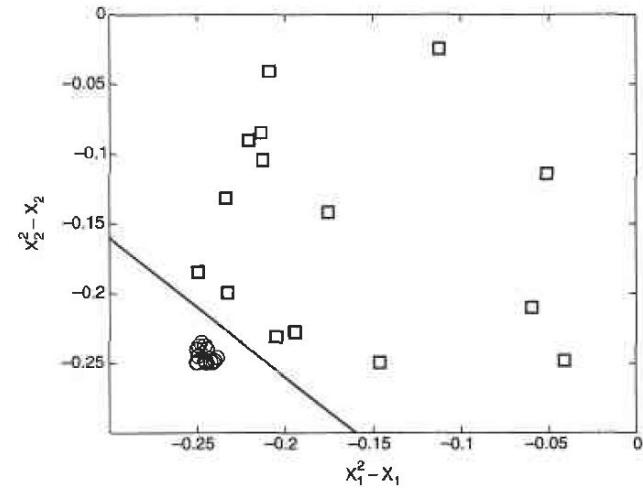
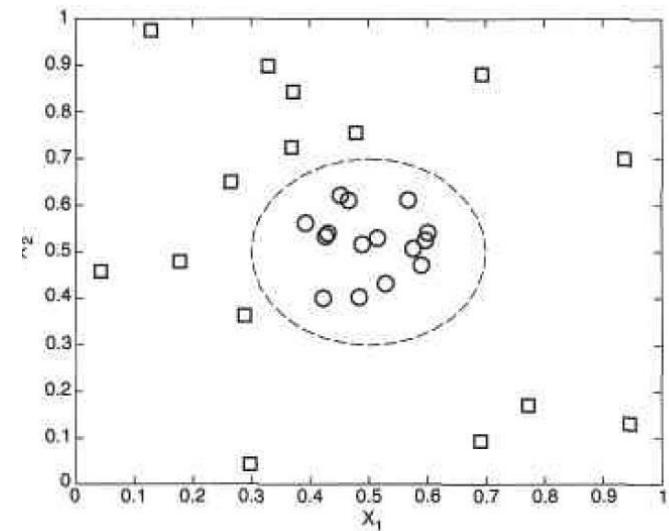
- Data is not separable linearly
- Margin will become inefficient
- Data needs to be transformed from original coordinate space \mathbf{x} to a new space $\Phi(\mathbf{x})$, so that linear decision boundary can be applied
- A non-linear transformation function is needed, like, ex:

$$\Phi : (x_1, x_2) \longrightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

- In the transformed space we can choose $w = (w_0, w_1, \dots, w_4)$ such that

$$w_4x_1^2 + w_3x_2^2 + w_2\sqrt{2}x_1 + w_1\sqrt{2}x_2 + w_0 = 0.$$

- The linear decision boundary in the transformed space has the following form: $w \cdot \Phi(\mathbf{x}) + b = 0$



The “Kernel Trick”

- The linear classifier relies on dot product between vectors
 - $x_i^T \cdot x_j$
- If every data point is mapped into high-dimensional space via some transformation $\Phi: x \rightarrow \phi(x)$, the dot product becomes:
$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$
- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.

SVM Kernels

- SVM algorithms use a set of mathematical functions that are defined as the kernel.
- Function of kernel is to take data as input and transform it into the required form.
- Different SVM algorithms use different types of kernel functions. Example *linear, nonlinear, polynomial, and sigmoid etc.*

Example: Polynomial Kernel

- Given two examples x and z we want to map them to a **high dimensional space** [for example, quadratic]

$$\phi(x_1, x_2, \dots, x_n) = [1, x_1, x_2, \dots, x_n, x_1^2, x_2^2, \dots, x_n^2, x_1x_2, \dots, x_{n-1}x_n]^T$$

Example: Polynomial Kernel

- Given two examples x and z we want to map them to a **high dimensional space** [for example, quadratic]

$$\phi(x_1, x_2, \dots, x_n) = [1, x_1, x_2, \dots, x_n, x_1^2, x_2^2, \dots, x_n^2, x_1x_2, \dots, x_{n-1}x_n]^T$$



All degree zero terms

Example: Polynomial Kernel

- Given two examples x and z we want to map them to a **high dimensional space** [for example, quadratic]

$$\phi(x_1, x_2, \dots, x_n) = [1, x_1, x_2, \dots, x_n, x_1^2, x_2^2, \dots, x_n^2, x_1x_2, \dots, x_{n-1}x_n]^T$$

All degree zero terms All degree one terms

Example: Polynomial Kernel

- Given two examples x and z we want to map them to a **high dimensional space** [for example, quadratic]

$$\phi(x_1, x_2, \dots, x_n) = [1, x_1, x_2, \dots, x_n, x_1^2, x_2^2, \dots, x_n^2, x_1x_2, \dots, x_{n-1}x_n]^T$$

All degree zero terms All degree one terms All degree two terms

Example: Polynomial Kernel

- Given two examples \mathbf{x} and \mathbf{z} we want to map them to a **high dimensional space** [for example, quadratic]

$$\phi(x_1, x_2, \dots, x_n) = [1, x_1, x_2, \dots, x_n, x_1^2, x_2^2, \dots, x_n^2, x_1x_2, \dots, x_{n-1}x_n]^T$$

All degree zero terms All degree one terms All degree two terms

and compute the dot product $A = \varphi(\mathbf{x})^T \varphi(\mathbf{z})$ [takes time]

Example: Polynomial Kernel

- Given two examples x and z we want to map them to a **high dimensional space** [for example, quadratic]

$$\phi(x_1, x_2, \dots, x_n) = [1, x_1, x_2, \dots, x_n, x_1^2, x_2^2, \dots, x_n^2, x_1x_2, \dots, x_{n-1}x_n]^T$$

and compute the dot product $A = \varphi(x)^T \varphi(z)$ [takes time]

Example: Polynomial Kernel

- Given two examples x and z we want to map them to a **high dimensional space** [for example, quadratic]

$$\phi(x_1, x_2, \dots, x_n) = [1, x_1, x_2, \dots, x_n, x_1^2, x_2^2, \dots, x_n^2, x_1x_2, \dots, x_{n-1}x_n]^T$$

and compute the dot product $A = \varphi(x)^T \varphi(z)$ [takes time]

- Instead, in the original space, compute

Example: Polynomial Kernel

- Given two examples \mathbf{x} and \mathbf{z} we want to map them to a **high dimensional space** [for example, quadratic]

$$\phi(x_1, x_2, \dots, x_n) = [1, x_1, x_2, \dots, x_n, x_1^2, x_2^2, \dots, x_n^2, x_1x_2, \dots, x_{n-1}x_n]^T$$

and compute the dot product $\mathbf{A} = \varphi(\mathbf{x})^\top \varphi(\mathbf{z})$ [takes time]

- Instead, in the original space, compute

$$B = K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^\top \mathbf{z})^2$$

Example: Polynomial Kernel

- Given two examples \mathbf{x} and \mathbf{z} we want to map them to a **high dimensional space** [for example, quadratic]

$$\phi(x_1, x_2, \dots, x_n) = [1, x_1, x_2, \dots, x_n, x_1^2, x_2^2, \dots, x_n^2, x_1x_2, \dots, x_{n-1}x_n]^T$$

and compute the dot product $\mathbf{A} = \varphi(\mathbf{x})^\top \varphi(\mathbf{z})$ [takes time]

- Instead, in the original space, compute

$$B = K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^\top \mathbf{z})^2$$

Claim: $\mathbf{A} = \mathbf{B}$ (Coefficients do not really matter)

Example: Two dimensions, quadratic kernel

$$A = \varphi(\mathbf{x})^\top \varphi(\mathbf{z}) \quad B = K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^T \mathbf{z})^2$$

$$\phi(x_1, x_2) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \\ x_1 x_2 \end{bmatrix}$$

The Kernel Trick

Suppose we wish to compute $K(x, z) = \varphi(x)^T \varphi(z)$

Here φ maps x and z to a high dimensional space

The Kernel Trick: Save time/space by computing the value of $K(x, z)$ by performing operations in the original space (without a feature transformation!)

Computing dot products efficiently

Kernel Trick: You want to work with degree 2 polynomial features, $\varphi(x)$. Then, your dot product will be operate using vectors in a space of dimensionality $n(n+1)/2$.

The kernel trick allows you to save time/space and compute dot products in an n dimensional space.

(Not just for degree 2 polynomials)

Which functions are kernels?

Kernel Trick: You want to work with degree 2 polynomial features, $\phi(x)$. Then, your dot product will be operate using vectors in a space of dimensionality $n(n+1)/2$.

The kernel trick allows you to save time/space and compute dot products in an n dimensional space.

(Not just for degree 2 polynomials)

- Can we use any function $K(.,.)$?

Which functions are kernels?

Kernel Trick: You want to work with degree 2 polynomial features, $\varphi(x)$. Then, your dot product will be operate using vectors in a space of dimensionality $n(n+1)/2$.

The kernel trick allows you to save time/space and compute dot products in an n dimensional space.

(Not just for degree 2 polynomials)

- Can we use any function $K(.,.)$?
 - No! A function $K(x,z)$ is a valid kernel if it corresponds to an inner product in some (perhaps infinite dimensional) feature space.

Which functions are kernels?

Kernel Trick: You want to work with degree 2 polynomial features, $\varphi(x)$. Then, your dot product will be operate using vectors in a space of dimensionality $n(n+1)/2$.

The kernel trick allows you to save time/space and compute dot products in an n dimensional space.

(Not just for degree 2 polynomials)

- Can we use any function $K(.,.)$?
 - No! A function $K(x,z)$ is a valid kernel if it corresponds to an inner product in some (perhaps infinite dimensional) feature space.
- General condition: construct the Gram matrix $\{K(x_i, z_j)\}$; check that it's positive semi definite

The Kernel Matrix

- The **Gram matrix** of a set of n vectors $S = \{\mathbf{x}_1 \dots \mathbf{x}_n\}$ is the $n \times n$ matrix \mathbf{G} with $G_{ij} = \mathbf{x}_i^T \mathbf{x}_j$
 - The kernel matrix is the Gram matrix of $\{\varphi(\mathbf{x}_1), \dots, \varphi(\mathbf{x}_n)\}$
 - (size depends on the # of examples, not dimensionality)

Mercer's condition

Let $K(\mathbf{x}, \mathbf{z})$ be a function that maps two n dimensional vectors to a real number

K is a valid kernel if for every finite set $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, for any choice of real valued c_1, c_2, \dots, c_n , we have

$$\sum_i \sum_j c_i c_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

Polynomial kernels

- Linear kernel: $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z}$
- Polynomial kernel of degree d : $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z})^d$
 - only d th-order interactions
- Polynomial kernel up to degree d : $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z} + c)^d$
 $(c > 0)$
 - all interactions of order d or lower

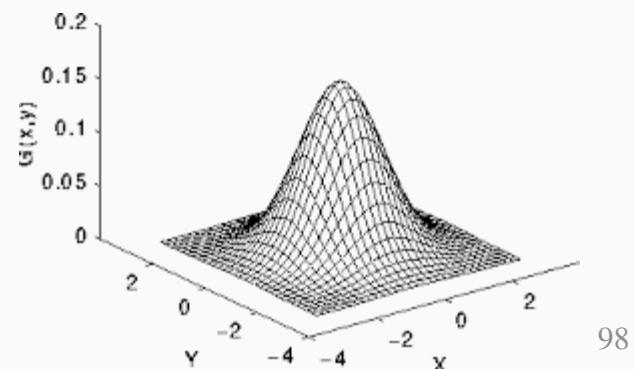
Gaussian Kernel

(or the radial basis function kernel)

$$K_{rbf}(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{c}\right)$$

- $(x - z)^2$: squared Euclidean distance between \mathbf{x} and \mathbf{z}
- $c = \sigma^2$: a free parameter
- very small c : $K \approx$ identity matrix (every item is different)
- very large c : $K \approx$ unit matrix (all items are the same)

- $k(\mathbf{x}, \mathbf{z}) \approx 1$ when \mathbf{x}, \mathbf{z} close
- $k(\mathbf{x}, \mathbf{z}) \approx 0$ when \mathbf{x}, \mathbf{z} dissimilar



Constructing New Kernels

You can construct new kernels $k'(\mathbf{x}, \mathbf{x}')$ from existing ones:

- Multiplying $k(\mathbf{x}, \mathbf{x}')$ by a constant c

$$ck(\mathbf{x}, \mathbf{x}')$$

- Multiplying $k(\mathbf{x}, \mathbf{x}')$ by a function f applied to \mathbf{x} and \mathbf{x}'

$$f(\mathbf{x})k(\mathbf{x}, \mathbf{x}')f(\mathbf{x}')$$

- Applying a polynomial (with non-negative coefficients) to $k(\mathbf{x}, \mathbf{x}')$

$$P(k(\mathbf{x}, \mathbf{x}')) \text{ with } P(z) = \sum_i a_i z^i \text{ and } a_i \geq 0$$

- Exponentiating $k(\mathbf{x}, \mathbf{x}')$

$$\exp(k(\mathbf{x}, \mathbf{x}'))$$

Constructing New Kernels (2)

- You can construct $k'(\mathbf{x}, \mathbf{x}')$ from $k_1(\mathbf{x}, \mathbf{x}')$, $k_2(\mathbf{x}, \mathbf{x}')$ by:
 - Adding $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$:
 $k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$
 - Multiplying $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$:
 $k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$

Name	Function	Type problem
Polynomial Kernel	$(x_i^T x_j + 1)^q$ q is degree of polynomial	Best for Image processing
Sigmoid Kernel	$\tanh(ax_i^T x_j + k)$ k is offset value	Very similar to neural network
Gaussian Kernel	$\exp(-\ x_i - x_j\ ^2 / 2\sigma^2)$	No prior knowledge on data
Linear Kernel	$(1 + x_i^T x_j) \min(x_i, x_j) - \frac{(x_i + x_j)}{2} \min(x_i, x_j)^2 + \frac{\min(x_i, x_j)^3}{3}$	Text Classification
Laplace Radial Basis Function (RBF)	$(e^{-\lambda \ x_i - x_j\ }), \lambda >= 0$	No prior knowledge on data

There are many more kernel functions.

Non-linear SVM using kernel

1. Select a kernel function.
2. Compute pairwise kernel values between labeled examples.
3. Use this “kernel matrix” to solve for SVM support vectors & alpha weights.
4. To classify a new example: compute kernel values between new input and support vectors, apply alpha weights, check sign of output.

Nonlinear SVM - Overview

- SVM locates a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the dot product in the feature space.

Multi-Class Problem

Instead of just two classes, we now have C classes

- E.g. predict which movie genre a viewer likes best
- Possible answers: action, drama, indie, thriller, etc.

Two approaches:

- One-vs-all
- One-vs-one

Multi-Class Problem

Instead of just two classes, we now have C classes

- E.g. predict which movie genre a viewer likes best
- Possible answers: action, drama, indie, thriller, etc.

Two approaches:

- One-vs-all
- One-vs-one

Multi-Class Problem

One-vs-all (a.k.a. one-vs-others)

- Train C classifiers
- In each, pos = data from class i , neg = data from classes other than i
- The class with the most confident prediction wins
- Example:
 - You have 4 classes, train 4 classifiers
 - 1 vs others: score 3.5
 - 2 vs others: score 6.2
 - 3 vs others: score 1.4
 - 4 vs other: score 5.5
 - Final prediction: class 2
- Issues?

Multi-Class Problem

One-vs-one (a.k.a. all-vs-all)

- Train $C(C-1)/2$ binary classifiers (all pairs of classes)
- They all vote for the label
- Example:
 - You have 4 classes, then train 6 classifiers
 - 1 vs 2, 1 vs 3, 1 vs 4, 2 vs 3, 2 vs 4, 3 vs 4
 - Votes: 1, 1, 4, 2, 4, 4
 - Final prediction is class 4

SVM versus Logistic Regression

- When viewed from the point of view of regularized empirical loss minimization, SVM and logistic regression appear quite similar:

$$\text{SVM: } \sum_{i=1}^n \left(1 - y_i [w_0 + \mathbf{x}_i^T \mathbf{w}_1]\right)^+ + \|\mathbf{w}_1\|^2/2$$

$$\text{Logistic: } \sum_{i=1}^n \underbrace{-\log P(y_i|\mathbf{x}, \mathbf{w})}_{-\log \sigma(y_i [w_0 + \mathbf{x}_i^T \mathbf{w}_1])} + \|\mathbf{w}_1\|^2/2$$

where $\sigma(z) = (1 + \exp(-z))^{-1}$ is the logistic function.

SVM versus Logistic Regression

- The difference comes from how we penalize “errors”:

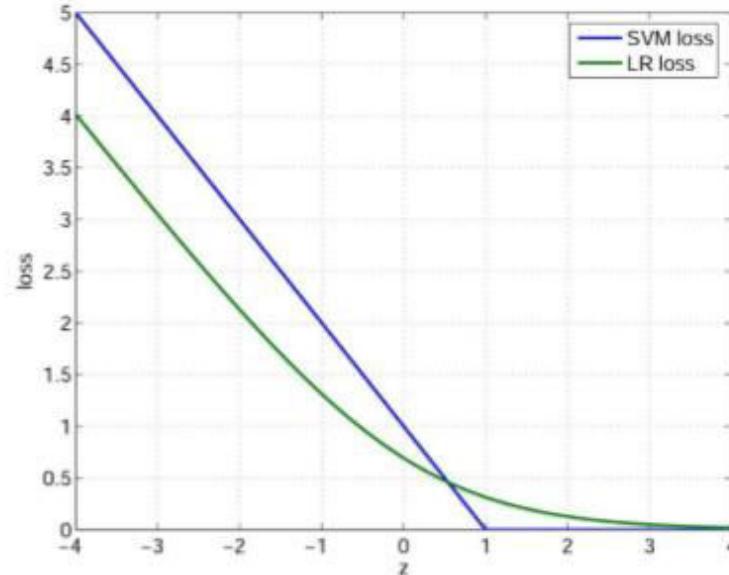
Both:
$$\sum_{i=1}^n \text{Loss}\left(\underbrace{y_i [w_0 + \mathbf{x}_i^T \mathbf{w}_1]}_z\right) + \|\mathbf{w}_1\|^2/2$$

- SVM:

$$\text{Loss}(z) = (1 - z)^+$$

- Regularized logistic reg:

$$\text{Loss}(z) = \log(1 + \exp(-z))$$

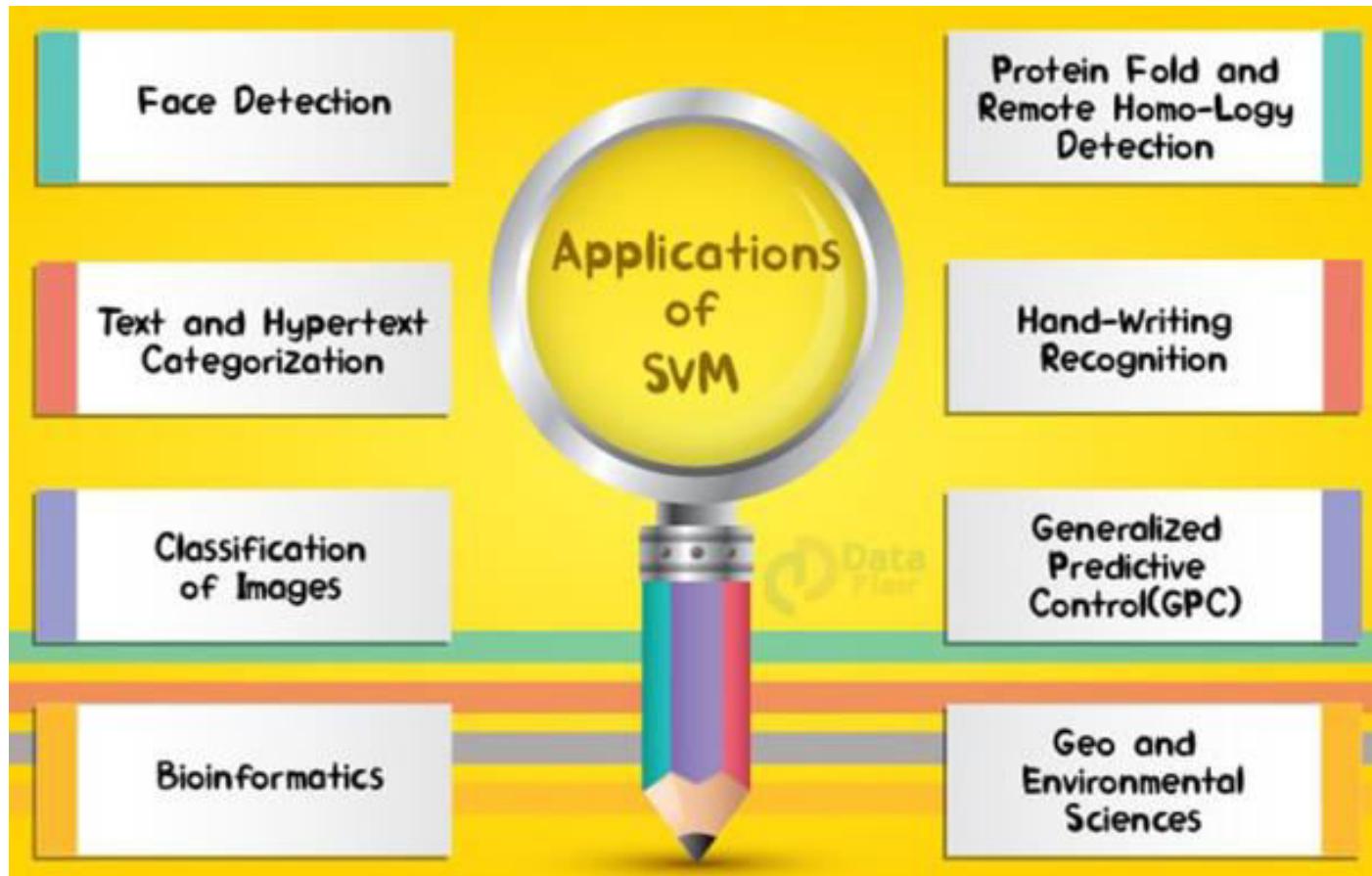


Properties of SVM

- **Flexibility in choosing a similarity function**
- **Sparseness of solution when dealing with large data sets**
 - Only support vectors are used to specify the separating hyperplane
 - Therefore SVM also called sparse kernel machine.
- **Ability to handle large feature spaces**
 - complexity does not depend on the dimensionality of the feature space
- **Overfitting can be controlled by soft margin approach**
- **Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution**
- **Feature Selection**

SVM Applications

SVM has been used successfully in many real-world problems



Application : Text Categorization

- Task: The classification of natural text (or hypertext) documents into a fixed number of predefined categories based on their content. A document can be assigned to more than one category, so this can be viewed as a series of binary classification problems, one for each category

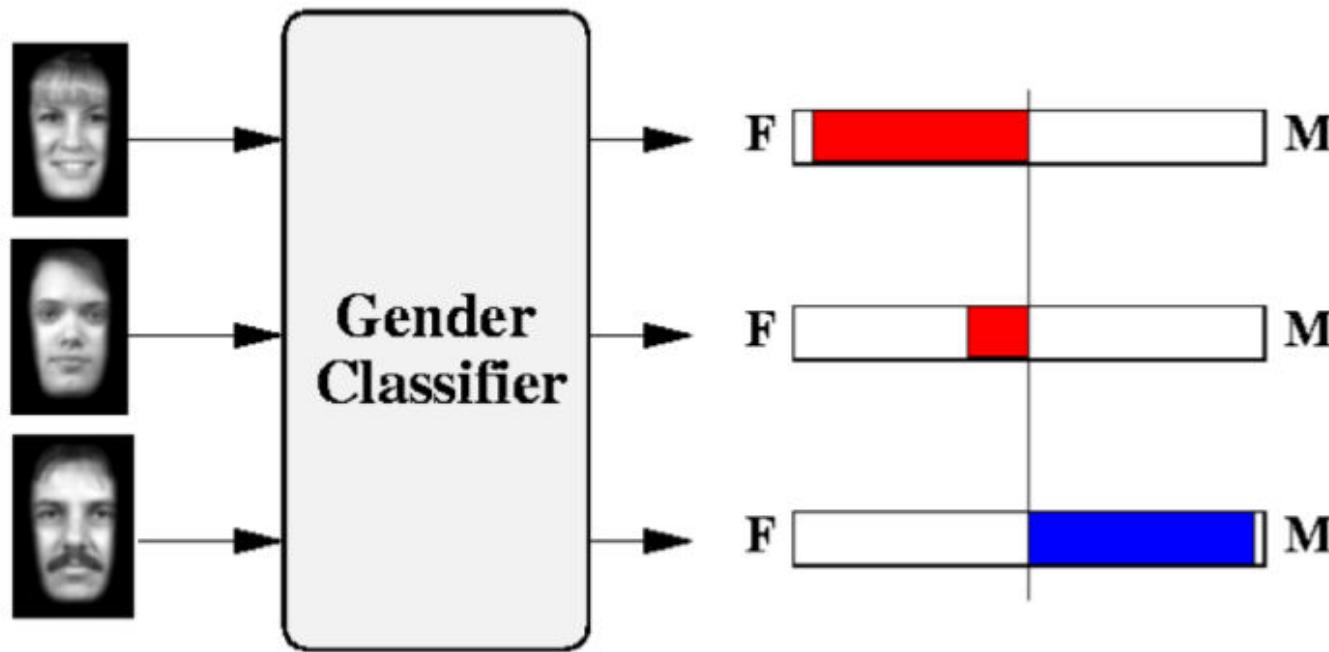
Text Categorization using SVM

- The distance between two documents is $\phi(x) \cdot \phi(z)$
- $K(x,z) = \phi(x) \cdot \phi(z)$ is a valid kernel, SVM can be used with $K(x,z)$ for discrimination.
- Why SVM?
 - High dimensional input space
 - Few irrelevant features (dense concept)
 - Sparse document vectors (sparse instances)
 - Text categorization problems are linearly separable

Using SVM

1. Select a kernel function.
 2. Compute pairwise kernel values between labeled examples.
 3. Use this “kernel matrix” to solve for SVM support vectors & alpha weights.
 4. To classify a new example: compute kernel values between new input and support vectors, apply alpha weights, check sign of output.
-

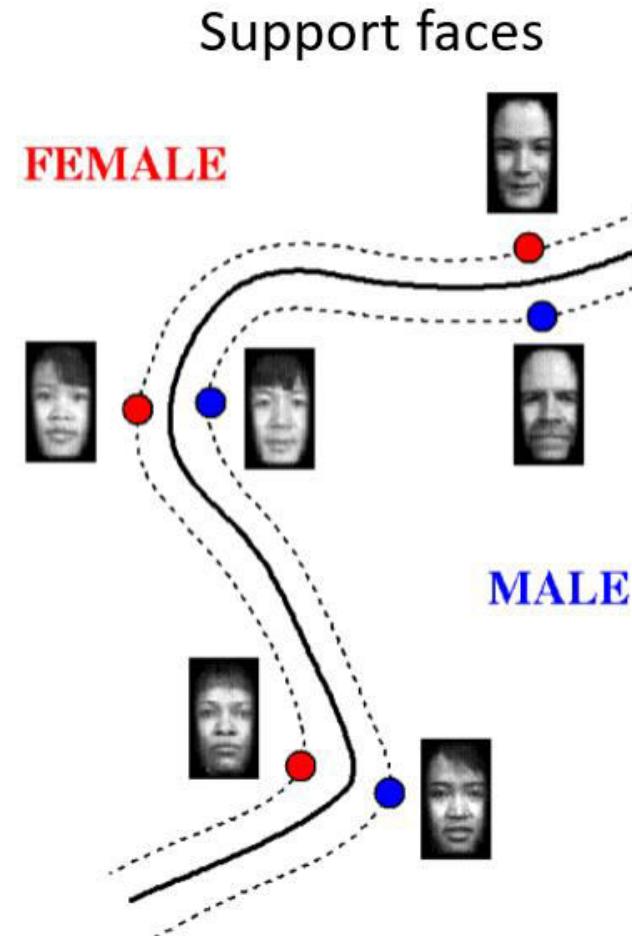
Learning Gender from image with SVM



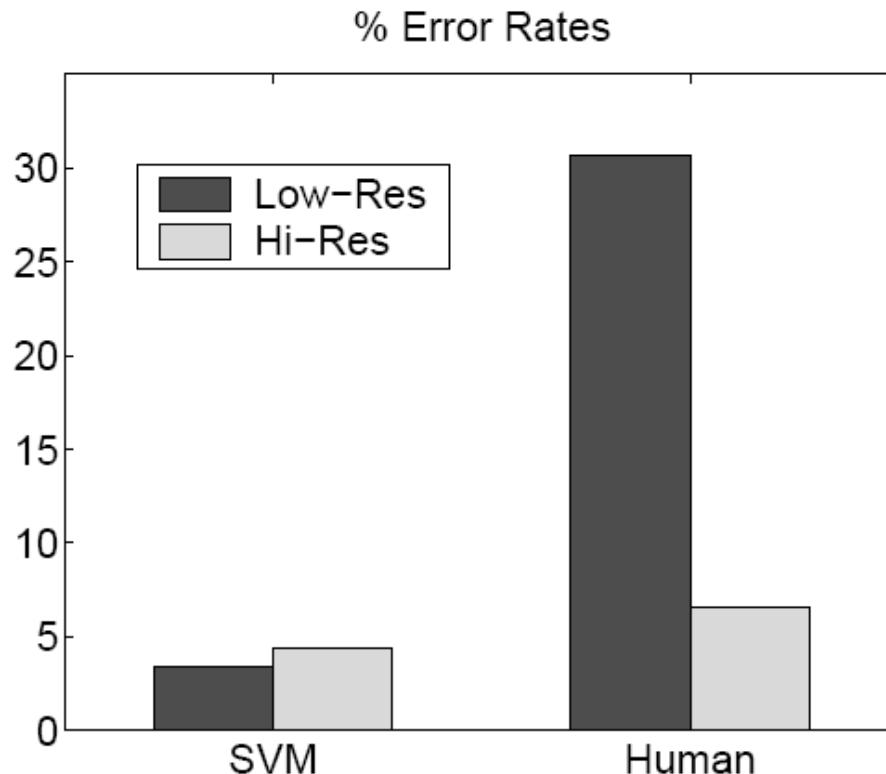
Moghaddam and Yang, Learning Gender with Support Faces, TPAMI 2002

Moghaddam and Yang, Face & Gesture 2000

Support faces



Accuracy of SVM Classifier



- SVMs performed better than humans, at either resolution

Figure 6. SVM vs. Human performance

Some Issues

- **Sensitive to noise**
 - A relatively small number of mislabeled examples can dramatically decrease the performance
 - **Choice of kernel**
 - Gaussian or polynomial kernel is default
 - if ineffective, more elaborate kernels are needed
 - domain experts can give assistance in formulating appropriate similarity measures
 - **Choice of kernel parameters**
 - e.g. σ in Gaussian kernel
 - σ is the distance between closest points with different classifications
 - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.
 - **Optimization criterion** – Hard margin v.s. Soft margin
 - a lengthy series of experiments in which various parameters are tested
-

Reference

- **Support Vector Machine Classification of Microarray Gene Expression Data**, Michael P. S. Brown William Noble Grundy, David Lin, Nello Cristianini, Charles Sugnet, Manuel Ares, Jr., David Haussler
- **Text categorization with Support Vector Machines: learning with many relevant features**

T. Joachims, ECML - 98

- Christopher Bishop: Pattern Recognition and Machine Learning, Springer International Edition
- **A Tutorial on Support Vector Machines for Pattern Recognition**, Kluwer Academic Publishers - Christopher J.C. Burges

Good Web References for SVM

- <http://www.cs.utexas.edu/users/mooney/cs391L/>
- <https://www.coursera.org/learn/machine-learning/home/week/7>
- <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>
- <https://data-flair.training/blogs/svm-kernel-functions/>
- [MIT 6.034 Artificial Intelligence, Fall 2010](#)
- <https://stats.stackexchange.com/questions/30042/neural-networks-vs-support-vector-machines-are-the-second-definitely-superior>
- <https://www.sciencedirect.com/science/article/abs/pii/S0893608006002796>
- <https://medium.com/deep-math-machine-learning-ai/chapter-3-support-vector-machine-with-math-47d6193c82be>
- [Radial basis kernel](#)
- <http://www.engr.mun.ca/~baxter/Publications/LagrangeForSVMs.pdf>



Thank You



Machine Learning



BITS Pilani
Pilani Campus

Dr. Sugata Ghosal

sugata.ghosal@pilani.bits-pilani.ac.in
9910267531



Lecture No. – 10 | Bayesian Learning

Date – 11/02/2023

Time – 4:15 PM – 6:15 PM

Grateful Acknowledgement : These slides were assembled leveraging the content created by the many instructors who made their course materials freely available online.

Agenda

- Maximum Likelihood Estimation
- Maximum A Posteriori Estimation
- Naïve Bayes Classifier
- Gaussian Naïve Bayes Classifier
- Image Classification Example
- Text Classification Example

Learning Function Approximation ?

- instead of $F: X \rightarrow Y$
learn $P(Y | X)$

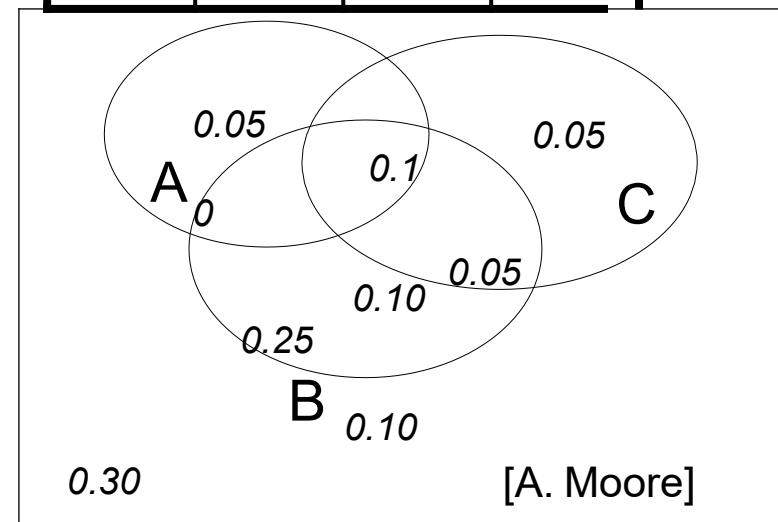
The Joint Distribution



Recipe for making a joint distribution of M variables:

Example: Boolean variables A, B, C

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10



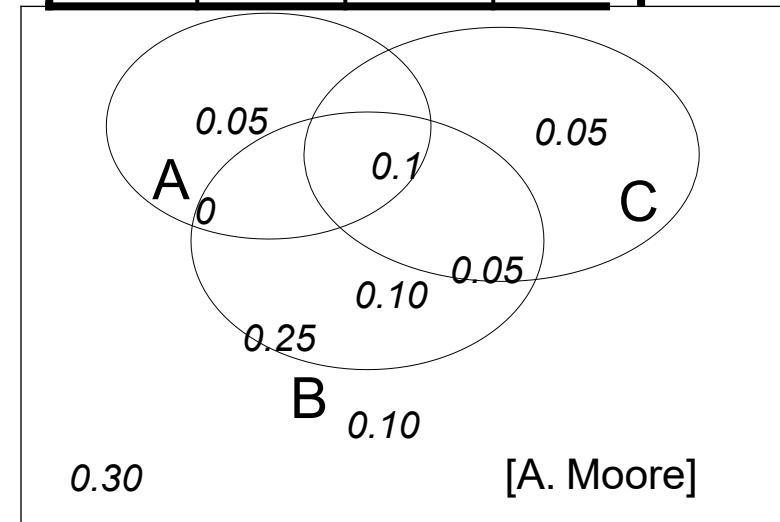
The Joint Distribution

Recipe for making a joint distribution of M variables:

1. Make a truth table listing all combinations of values (M Boolean variables $\rightarrow 2^M$ rows).

Example: Boolean variables A, B, C

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10



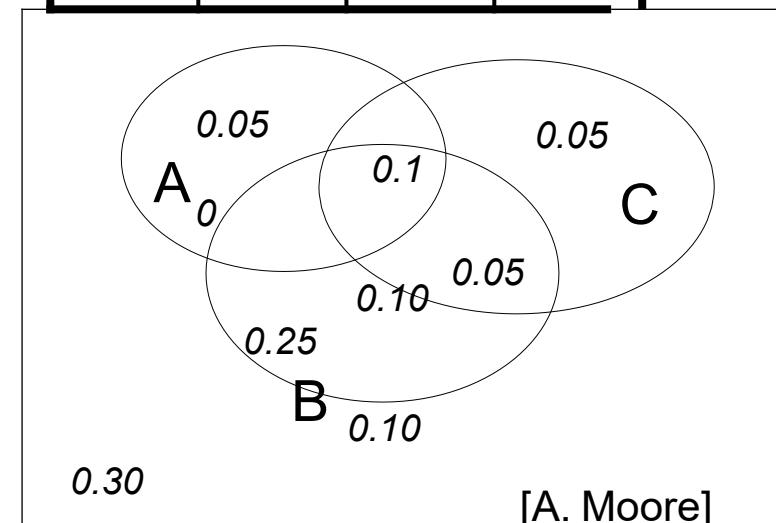
The Joint Distribution

Recipe for making a joint distribution of M variables:

1. Make a truth table listing all combinations of values (M Boolean variables $\rightarrow 2^M$ rows).
2. For each combination of values, say how probable it is.

Example: Boolean variables A, B, C

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10



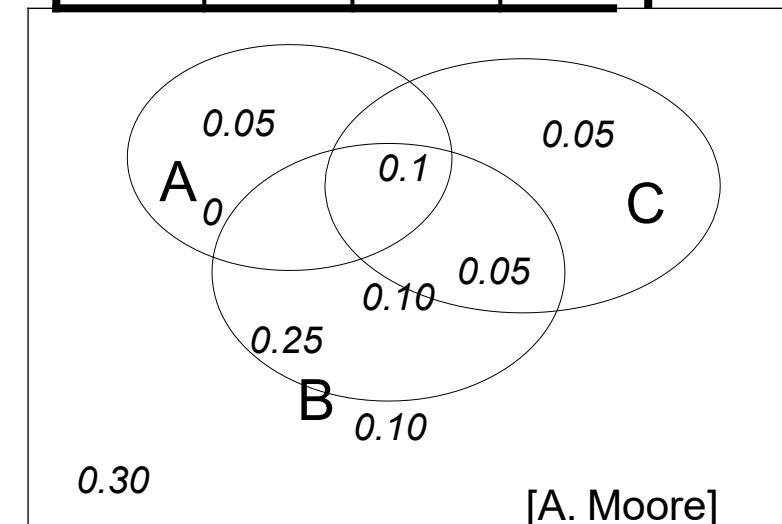
The Joint Distribution

Recipe for making a joint distribution of M variables:

1. Make a truth table listing all combinations of values (M Boolean variables $\rightarrow 2^M$ rows).
2. For each combination of values, say how probable it is.
3. If you subscribe to the axioms of probability, those probabilities must sum to 1.

Example: Boolean variables A, B, C

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10



[A. Moore]

Using the Joint Distribution

gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

Once you have the JD
you can ask for the
probability of **any** logical
expression involving
these variables

$$P(E) = \sum_{\text{rows matching } E} P(\text{row})$$

[A. Moore]

Using the Joint Distribution

gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

$$P(\text{Poor Male}) = 0.4654$$

$$P(E) = \sum_{\text{rows matching } E} P(\text{row})$$

[A. Moore]

Using the Joint Distribution

gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

$$P(\text{Poor}) = 0.7604$$

$$P(E) = \sum_{\text{rows matching } E} P(\text{row})$$

[A. Moore]

Inference with the Joint Distribution



gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

$$P(E_1 | E_2) = \frac{P(E_1 \wedge E_2)}{P(E_2)} = \frac{\sum_{\substack{\text{rows matching } E_1 \text{ and } E_2}} P(\text{row})}{\sum_{\substack{\text{rows matching } E_2}} P(\text{row})}$$

$$P(\text{Male} | \text{Poor}) = 0.4654 / 0.7604 = 0.612$$

[A. Moore]

Learning and the Joint Distribution



gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

Suppose we want to learn the function $f: \langle G, H \rangle \rightarrow W$
Equivalently, $P(W | G, H)$

Solution: learn joint distribution from data, calculate $P(W | G, H)$

e.g., $P(W=\text{rich} | G = \text{female}, H = 40.5-) =$

[A. Moore]

sounds like the solution to learning $F : X \rightarrow Y$ or $P(Y | X)$

Main problem: learning $P(Y|X)$
can require more data than we have

consider learning Joint Dist. with 100 attributes

of rows in this table? $2^{100} \sim 10^{30}$

of people on earth? 10^9

fraction of rows with 0 training examples? 0.9999

What to do?

1. Be smart about how we estimate probabilities from sparse data
 - maximum likelihood estimates
 - maximum a posteriori estimates

1. Be smart about how we estimate probabilities

Estimating Probability of Heads



- I show you the above coin X , and hire you to estimate the probability that it will turn up heads ($X = 1$) or tails ($X = 0$)
- You flip it repeatedly, observing
 - it turns up heads α_1 times
 - it turns up tails α_0 times
- Your estimate for $P(X = 1)$ is....?

Estimating $\theta = P(X=1)$



Case A:

- 100 flips: 51 Heads ($X=1$), 49 Tails ($X=0$)

Case B:

- 3 flips: 2 Heads ($X=1$), 1 Tails ($X=0$)

Case C: (online learning)

- keep flipping, want single learning algorithm that gives reasonable estimate after each flip

Principles for Estimating Probabilities

Principle 1 (maximum likelihood):

- choose parameters θ that maximize $P(\text{data} | \theta)$

Principle 2 (maximum a posteriori prob.):

- choose parameters θ that maximize $P(\theta | \text{data})$

Maximum Likelihood Estimation

$$P(X=1) = \theta \quad P(X=0) = (1-\theta)$$

Data D:

$$\{ 0, 1, 1, 1, 0, \dots \}$$



Flips produce data D with α_1 heads, α_0 tails

- flips are independent, identically distributed 1's and 0's (Bernoulli)
- α_1 and α_0 are counts that sum these outcomes (Binomial)

$$P(D|\theta) = P(\alpha_1, \alpha_0|\theta) = \theta^{\alpha_1} (1 - \theta)^{\alpha_0}$$

Maximum Likelihood Estimate for θ



$$\begin{aligned}\hat{\theta} &= \arg \max_{\theta} \ln P(\mathcal{D} | \theta) \\ &= \arg \max_{\theta} \ln \theta^{\alpha_H} (1 - \theta)^{\alpha_T}\end{aligned}$$

- Set derivative to zero:

$$\frac{d}{d\theta} \ln P(\mathcal{D} | \theta) = 0$$

$$\hat{\theta} = \arg \max_{\theta} \ln P(D|\theta)$$

- Set derivative to zero:

$$\frac{d}{d\theta} \ln P(\mathcal{D} | \theta) = 0$$

$$= \arg \max_{\theta} \ln [\theta^{\alpha_1} (1 - \theta)^{\alpha_0}]$$

$$\text{hint: } \frac{\partial \ln \theta}{\partial \theta} = \frac{1}{\theta}$$

[C. Guestrin]

Summary:

Maximum Likelihood Estimate



- Each flip yields boolean value for X

$$P(X=1) = \theta$$

$$P(X=0) = 1-\theta \\ (\text{Bernoulli})$$

- Data set D of independent, identically distributed (iid) flips produces α_1 ones, α_0 zeros (Binomial)

$$P(D|\theta) = P(\alpha_1, \alpha_0|\theta) = \theta^{\alpha_1}(1-\theta)^{\alpha_0}$$

$$\hat{\theta}^{MLE} = \operatorname{argmax}_{\theta} P(D|\theta) = \frac{\alpha_1}{\alpha_1 + \alpha_0}$$

Principles for Estimating Probabilities

Principle 1 (maximum likelihood):

- choose parameters θ that maximize $P(\text{data} | \theta)$

Principle 2 (maximum a posteriori prob.):

- choose parameters θ that maximize

$$P(\theta | \text{data}) = \frac{P(\text{data} | \theta) P(\theta)}{P(\text{data})}$$

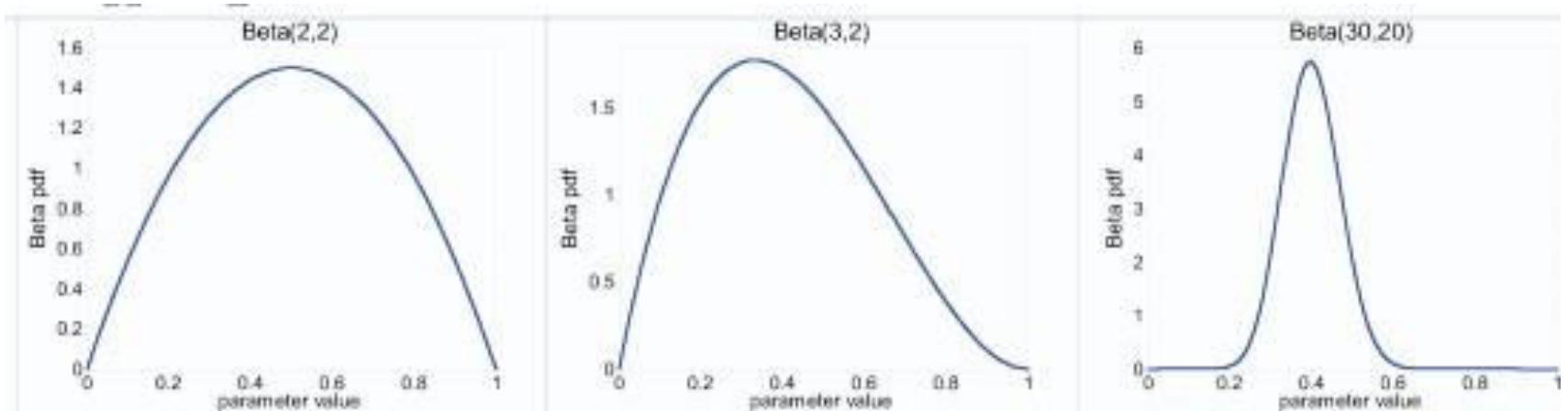
Example prior distribution – $P(\theta)$

$$P(\theta) = \frac{\theta^{\beta_H-1}(1-\theta)^{\beta_T-1}}{B(\beta_H, \beta_T)} \sim Beta(\beta_H, \beta_T)$$

- Likelihood function: $P(\mathcal{D} | \theta) = \theta^{\alpha_H}(1-\theta)^{\alpha_T}$
- Posterior: $P(\theta | \mathcal{D}) \propto P(\mathcal{D} | \theta)P(\theta)$

Beta prior distribution – $P(\theta)$

$$P(\theta) = \frac{\theta^{\beta_H-1}(1-\theta)^{\beta_T-1}}{B(\beta_H, \beta_T)} \sim Beta(\beta_H, \beta_T)$$



[C. Guestrin]

Eg. 1 Coin flip problem

Likelihood is \sim Binomial



$$P(\mathcal{D} | \theta) = \theta^{\alpha_H} (1 - \theta)^{\alpha_T}$$

If prior is Beta distribution,

$$P(\theta) = \frac{\theta^{\beta_H-1} (1 - \theta)^{\beta_T-1}}{B(\beta_H, \beta_T)} \sim Beta(\beta_H, \beta_T)$$

Then posterior is Beta distribution

$$P(\theta|D) \sim Beta(\alpha_H + \beta_H, \alpha_T + \beta_T)$$

and MAP estimate is therefore

$$\hat{\theta}^{MAP} = \frac{\alpha_H + \beta_H - 1}{(\alpha_H + \beta_H - 1) + (\alpha_T + \beta_T - 1)}$$

Eg. 2 Dice roll problem (6 outcomes instead of 2)



Likelihood is $\sim \text{Multinomial}(\theta = \{\theta_1, \theta_2, \dots, \theta_k\})$

$$P(\mathcal{D} | \theta) = \theta_1^{\alpha_1} \theta_2^{\alpha_2} \dots \theta_k^{\alpha_k}$$

If prior is Dirichlet distribution,

$$P(\theta) = \frac{\theta_1^{\beta_1-1} \theta_2^{\beta_2-1} \dots \theta_k^{\beta_k-1}}{B(\beta_1, \dots, \beta_k)} \sim \text{Dirichlet}(\beta_1, \dots, \beta_k)$$

Then posterior is Dirichlet distribution

$$P(\theta|D) \sim \text{Dirichlet}(\beta_1 + \alpha_1, \dots, \beta_k + \alpha_k)$$

and MAP estimate is therefore

$$\hat{\theta}_i^{MAP} = \frac{\alpha_i + \beta_i - 1}{\sum_{j=1}^k (\alpha_j + \beta_j - 1)}$$

Aside: Some terminology

- Likelihood function: $P(\text{data} | \theta)$
- Prior: $P(\theta)$
- Posterior: $P(\theta | \text{data})$
- **Conjugate prior:** $P(\theta)$ is the conjugate prior for likelihood function $P(\text{data} | \theta)$ if the forms of $P(\theta)$ and $P(\theta | \text{data})$ are the same.

Let's learn classifiers by learning $P(Y|X)$



Consider $Y = \text{Wealth}$, $X = \langle \text{Gender}, \text{HoursWorked} \rangle$

gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

Gender	HrsWorked	$P(\text{rich} G, \text{HW})$	$P(\text{poor} G, \text{HW})$
F	<40.5	.09	.91
F	>40.5	.21	.79
M	<40.5	.23	.77
M	>40.5	.38	.62

How many parameters must we estimate?

Suppose $X = \langle X_1, \dots, X_n \rangle$
 where X_i and Y are boolean RV's

Gender	HrsWorked	P(rich G,HW)	P(poor G,HW)
F	<40.5	.09	.91
F	>40.5	.21	.79
M	<40.5	.23	.77
M	>40.5	.38	.62

To estimate $P(Y | X_1, X_2, \dots, X_n)$
 how many parameters do we
 need to estimate?

If we have 30 boolean X_i 's: $P(Y | X_1, X_2, \dots, X_{30})$

Bayes Rule

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Which is shorthand for:

$$(\forall i, j) P(Y = y_i | X = x_j) = \frac{P(X = x_j | Y = y_i)P(Y = y_i)}{P(X = x_j)}$$

Equivalently:

$$(\forall i, j) P(Y = y_i | X = x_j) = \frac{P(X = x_j | Y = y_i)P(Y = y_i)}{\sum_k P(X = x_j | Y = y_k)P(Y = y_k)}$$

Naïve Bayes

Naïve Bayes assumes

$$P(X_1 \dots X_n | Y) = \prod_i P(X_i | Y)$$

i.e., that X_i and X_j are conditionally independent given Y , for all $i \neq j$

Conditional Independence

Definition: X is conditionally independent of Y given Z, if the probability distribution governing X is independent of the value of Y, given the value of Z

$$(\forall i, j, k) P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$$

Which we often write

$$P(X|Y, Z) = P(X|Z)$$

E.g.,

$$P(Thunder|Rain, Lightning) = P(Thunder|Lightning)$$

Naïve Bayes uses assumption that the X_i are conditionally independent, given Y . e.g., $P(X_1|X_2, Y) = P(X_1|Y)$

Given this assumption, then:

$$\begin{aligned}P(X_1, X_2|Y) &= P(X_1|X_2, Y)P(X_2|Y) \\&= P(X_1|Y)P(X_2|Y)\end{aligned}$$

in general: $P(X_1 \dots X_n|Y) = \prod_i P(X_i|Y)$

Naïve Bayes uses assumption that the X_i are conditionally independent, given Y . E.g., $P(X_1|X_2, Y) = P(X_1|Y)$

Given this assumption, then:

$$\begin{aligned}P(X_1, X_2|Y) &= P(X_1|X_2, Y)P(X_2|Y) \\&= P(X_1|Y)P(X_2|Y)\end{aligned}$$

in general: $P(X_1 \dots X_n|Y) = \prod_i P(X_i|Y)$

How many parameters to describe $P(X_1 \dots X_n|Y)$? $P(Y)$?

- Without conditional indep assumption?
- With conditional indep assumption?

Naïve Bayes in a Nutshell

Bayes rule:

$$P(Y = y_k | X_1 \dots X_n) = \frac{P(Y = y_k) P(X_1 \dots X_n | Y = y_k)}{\sum_j P(Y = y_j) P(X_1 \dots X_n | Y = y_j)}$$

Assuming conditional independence among X_i 's:

$$P(Y = y_k | X_1 \dots X_n) = \frac{P(Y = y_k) \prod_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j) \prod_i P(X_i | Y = y_j)}$$

So, to pick most probable Y for $X^{new} = < X_1, \dots, X_n >$

$$Y^{new} \leftarrow \arg \max_{y_k} P(Y = y_k) \prod_i P(X_i^{new} | Y = y_k)$$

- Train Naïve Bayes (examples)
for each* value y_k
estimate $\pi_k \equiv P(Y = y_k)$
for each* value x_{ij} of each attribute X_i
estimate $\theta_{ijk} \equiv P(X_i = x_{ij}|Y = y_k)$

- Classify (X^{new})

$$Y^{new} \leftarrow \arg \max_{y_k} P(Y = y_k) \prod_i P(X_i^{new}|Y = y_k)$$

$$Y^{new} \leftarrow \arg \max_{y_k} \pi_k \prod_i \theta_{ijk}$$

* probabilities must sum to 1, so need estimate only n-1 of these...

Estimating Parameters: Y , X_i

innovate achieve lead

discrete

Maximum likelihood estimates (MLE's):

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\}}{|D|}$$

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_{ij}|Y = y_k) = \frac{\#D\{X_i = x_{ij} \wedge Y = y_k\}}{\#D\{Y = y_k\}}$$

Number of items in
dataset D for which $Y=y_k$

Estimating Parameters: Y, X_i



discrete

MAP estimates (Beta, Dirichlet priors):

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\} + (\beta_k - 1)}{|D| + \sum_m (\beta_m - 1)}$$

Only difference:
“imaginary” examples

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_j | Y = y_k) = \frac{\#D\{X_i = x_j \wedge Y = y_k\} + (\beta_k - 1)}{\#D\{Y = y_k\} + \sum_m (\beta_m - 1)}$$

Naïve Bayes Classification Example 1

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(A|M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A|N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A|M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A|N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

$$\begin{aligned} P(A|M)P(M) &> \\ P(A|N)P(N) \end{aligned}$$

=> Mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

Naïve Bayes Classification Example 2



play →

The weather data, with counts and probabilities

outlook		temperature			humidity			windy		play	
	yes	no		yes	no		yes	no		yes	no
sunny	2	3	hot	2	2	high	3	4	false	6	2
overcast	4	0	mild	4	2	normal	6	1	true	3	3
rainy	3	2	cool	3	1						
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5
rainy	3/9	2/5	cool	3/9	1/5						

A new day

outlook		temperature			humidity			windy		play	
sunny		cool			high			true		?	

Naïve Bayes Classification Example 2 (contd.)



- Likelihood of yes

$$= \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.0053$$

- Likelihood of no

$$= \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.0206$$

- Therefore, the prediction is No

Estimating Parameters: X_i



Continuous

What if features are continuous?

- E.g., character recognition: X_i is intensity at i th pixel
- Gaussian Naïve Bayes (GNB):

$$P(X_i = x | Y = y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} e^{-\frac{(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$



distribution of feature X_i is Gaussian with a mean and variance that can depend on the label y_k and which feature X_i it is



What if features are continuous?

- E.g., character recognition: X_i is intensity at i th pixel
- Gaussian Naïve Bayes (GNB):

$$P(X_i = x | Y = y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} e^{-\frac{(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$



- Different mean and variance for each class k and each pixel i .
- Sometimes assume variance:
 - Is independent of Y (i.e., just have σ_i)
 - Or independent of X (i.e., just have σ_k)
 - Or both (i.e., just have σ)



Estimating parameters: Y discrete, X_i continuous

- Maximum likelihood estimates:

$$\hat{\mu}_{MLE} = \frac{1}{N} \sum_{j=1}^N x_j$$

$$\hat{\mu}_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j X_i^j \delta(Y^j = y_k)$$

ith pixel in jth training image
 jth training image
 kth class

$$\hat{\sigma}_{unbiased}^2 = \frac{1}{N-1} \sum_{j=1}^N (x_j - \hat{\mu})^2$$

$$\hat{\sigma}_{ik}^2 = \frac{1}{\sum_j \delta(Y^j = y_k) - 1} \sum_j (X_i^j - \hat{\mu}_{ik})^2 \delta(Y^j = y_k)$$

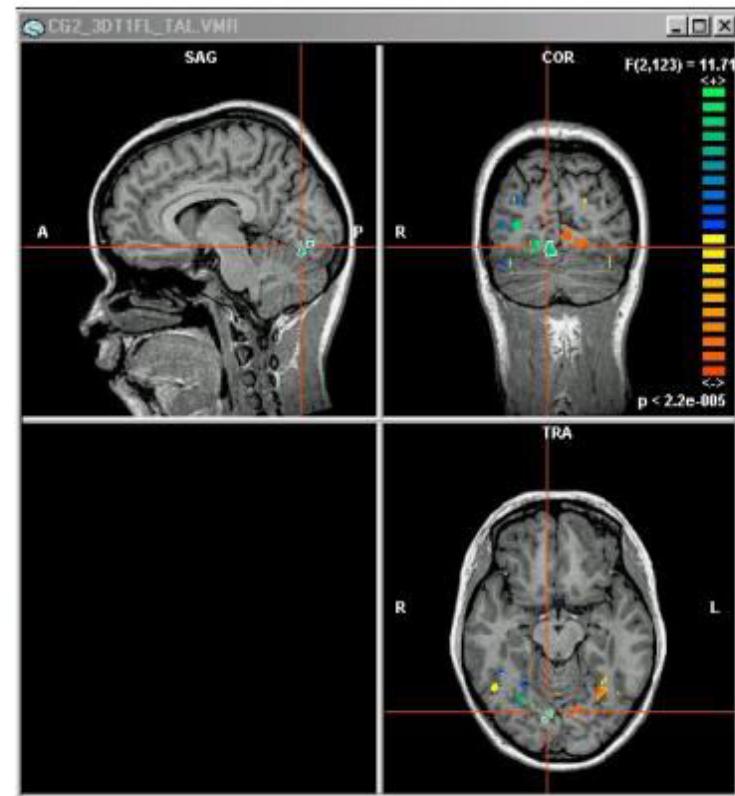


Example: GNB for classifying mental states

[Mitchell et al.]



- Classify a person's cognitive state, based on brain image
 - reading a sentence or viewing a picture?
 - reading the word describing a "Tool" or "Building"?
 - reading the word describing a "Person" or an "Animal"?
- Training: Patients were shown words of different categories and then a measurement was done to see what parts of the brain responded.

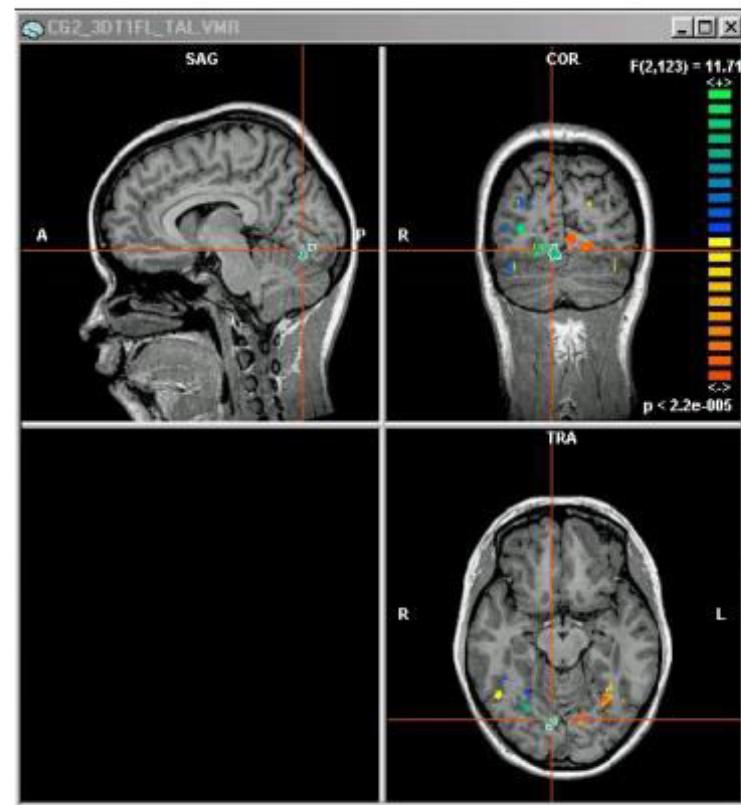


Example: GNB for classifying mental states

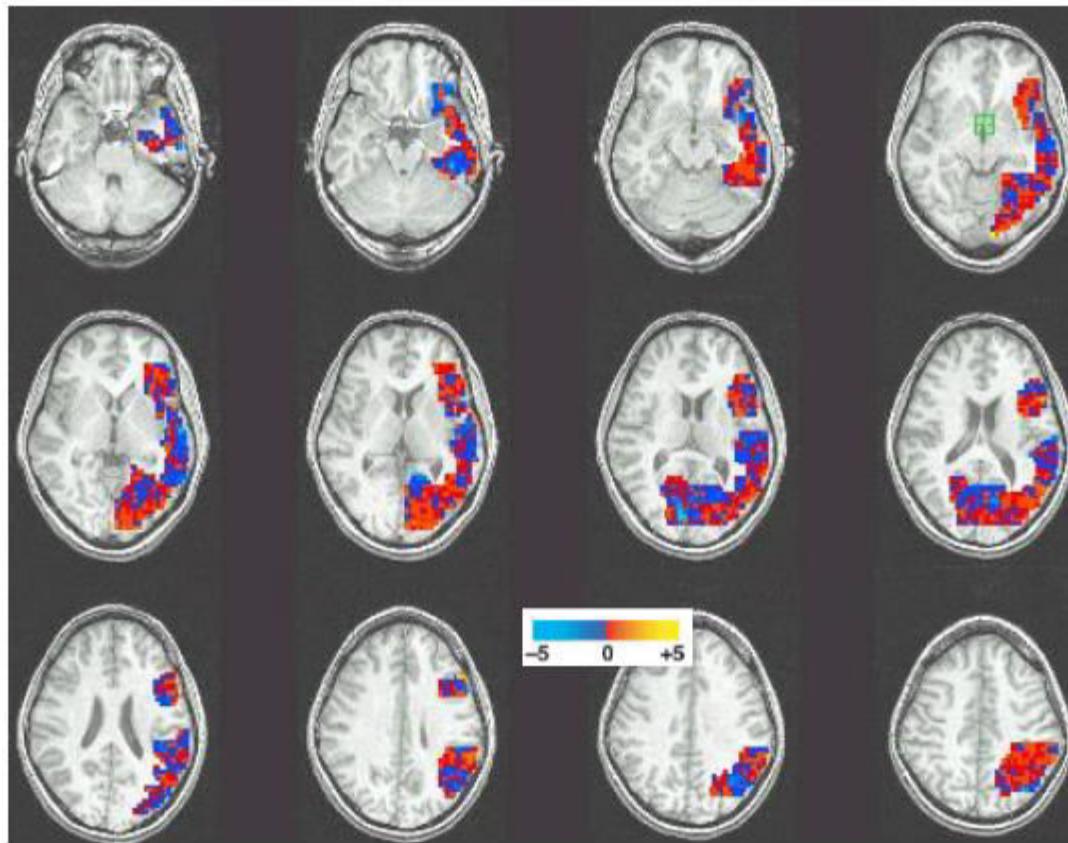
[Mitchell et al.]



- ~1mm resolution
- ~2 images per sec.
- 15,000 voxels/image
- Non-invasive, save
- Measures Blood Oxygen Level Dependent response (BOLD)



Gaussian Naïve Bayes: Learned $\mu_{\text{voxel}, \text{word}}$



[Mitchell et al.]

15,000 voxels
or features

10 training
examples or
subjects per
class

Learned Naïve Bayes Models – Means for $P(\text{BrainActivity} \mid \text{WordCategory})$

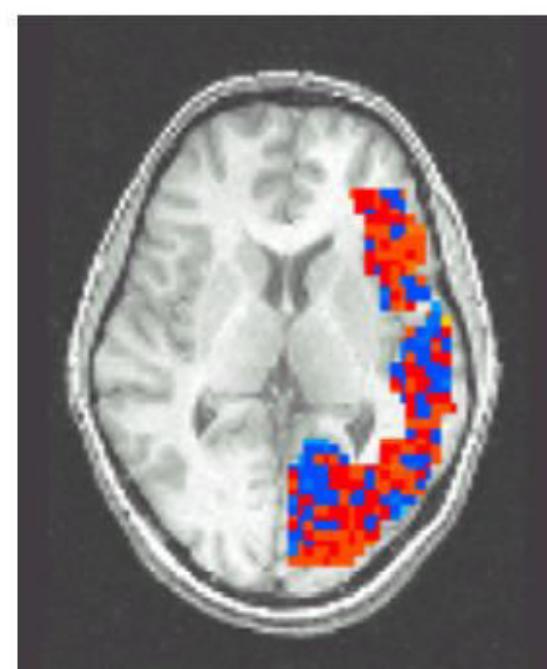
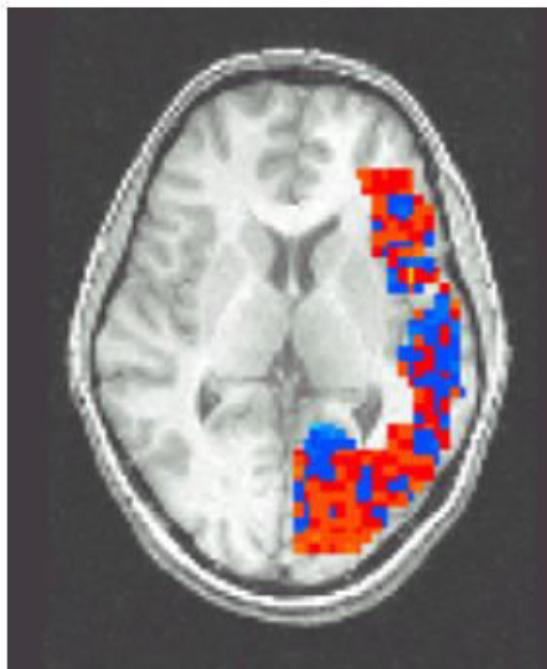
Pairwise classification accuracy: 85%

[Mitchell et al.]

People words



Animal words





Machine Learning

Dr. Sugata Ghosal

sugata.ghosal@pilani.bits-pilani.ac.in



BITS Pilani
Pilani Campus



Lecture No. – 11 | Bayesian Learning

Date – 18/02/2023

Time – 4:15 PM – 6:15 PM

Grateful Acknowledgement : These slides were assembled leveraging the content created by the many instructors who made their course materials freely available online.

Agenda

- Naïve Bayes Classifier
- Gaussian Naïve Bayes Classifier
- Image Classification Example
- Text Classification Example
- Optimal Bayes Classifier
- Regression from Bayesian Perspective

Bayes Rule

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Which is shorthand for:

$$(\forall i, j) P(Y = y_i | X = x_j) = \frac{P(X = x_j | Y = y_i)P(Y = y_i)}{P(X = x_j)}$$

Equivalently:

$$(\forall i, j) P(Y = y_i | X = x_j) = \frac{P(X = x_j | Y = y_i)P(Y = y_i)}{\sum_k P(X = x_j | Y = y_k)P(Y = y_k)}$$

Naïve Bayes

Naïve Bayes assumes

$$P(X_1 \dots X_n | Y) = \prod_i P(X_i | Y)$$

i.e., that X_i and X_j are conditionally independent given Y , for all $i \neq j$

Conditional Independence

Definition: X is conditionally independent of Y given Z, if the probability distribution governing X is independent of the value of Y, given the value of Z

$$(\forall i, j, k) P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$$

Which we often write

$$P(X|Y, Z) = P(X|Z)$$

E.g.,

$$P(Thunder|Rain, Lightning) = P(Thunder|Lightning)$$

Naïve Bayes uses assumption that the X_i are conditionally independent, given Y . E.g., $P(X_1|X_2, Y) = P(X_1|Y)$

Given this assumption, then:

$$\begin{aligned}P(X_1, X_2|Y) &= P(X_1|X_2, Y)P(X_2|Y) \\&= P(X_1|Y)P(X_2|Y)\end{aligned}$$

in general: $P(X_1 \dots X_n|Y) = \prod_i P(X_i|Y)$

How many parameters to describe $P(X_1 \dots X_n|Y)$? $P(Y)$?

- Without conditional indep assumption?
- With conditional indep assumption?

Naïve Bayes in a Nutshell

Bayes rule:

$$P(Y = y_k | X_1 \dots X_n) = \frac{P(Y = y_k) P(X_1 \dots X_n | Y = y_k)}{\sum_j P(Y = y_j) P(X_1 \dots X_n | Y = y_j)}$$

Assuming conditional independence among X_i 's:

$$P(Y = y_k | X_1 \dots X_n) = \frac{P(Y = y_k) \prod_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j) \prod_i P(X_i | Y = y_j)}$$

So, to pick most probable Y for $X^{new} = < X_1, \dots, X_n >$

$$Y^{new} \leftarrow \arg \max_{y_k} P(Y = y_k) \prod_i P(X_i^{new} | Y = y_k)$$

Naïve Bayes Algorithm – discrete X_i

- Train Naïve Bayes (examples)
for each^{*} value y_k
estimate $\pi_k \equiv P(Y = y_k)$
for each^{*} value x_{ij} of each attribute X_i
estimate $\theta_{ijk} \equiv P(X_i = x_{ij}|Y = y_k)$

- Classify (X^{new})

$$Y^{new} \leftarrow \arg \max_{y_k} P(Y = y_k) \prod_i P(X_i^{new}|Y = y_k)$$

$$Y^{new} \leftarrow \arg \max_{y_k} \pi_k \prod_i \theta_{ijk}$$

* probabilities must sum to 1, so need estimate only n-1 of these...

Estimating Parameters: Y , X_i discrete

Maximum likelihood estimates (MLE's):

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\}}{|D|}$$

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_{ij}|Y = y_k) = \frac{\#D\{X_i = x_{ij} \wedge Y = y_k\}}{\#D\{Y = y_k\}}$$

Number of items in
dataset D for which $Y=y_k$

Number of items in

dataset D for which $Y=y_k$

Estimating Parameters: Y, X_i discrete

MAP estimates (Beta, Dirichlet priors):

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\} + (\beta_k - 1)}{|D| + \sum_m (\beta_m - 1)}$$

Only difference:
“imaginary” examples

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_j | Y = y_k) = \frac{\#D\{X_i = x_j \wedge Y = y_k\} + (\beta_k - 1)}{\#D\{Y = y_k\} + \sum_m (\beta_m - 1)}$$

Naïve Bayes Classification Example 1

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(A|M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A|N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A|M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A|N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

$$\begin{aligned} P(A|M)P(M) &> \\ P(A|N)P(N) \end{aligned}$$

=> Mammals

Issues with Naïve Bayes Classifier

Naïve Bayes Classifier:

$$P(\text{Refund} = \text{Yes} | \text{No}) = 3/7$$

$$P(\text{Refund} = \text{No} | \text{No}) = 4/7$$

$$P(\text{Refund} = \text{Yes} | \text{Yes}) = 0$$

$$P(\text{Refund} = \text{No} | \text{Yes}) = 1$$

$$P(\text{Marital Status} = \text{Single} | \text{No}) = 2/7$$

$$P(\text{Marital Status} = \text{Divorced} | \text{No}) = 1/7$$

$$P(\text{Marital Status} = \text{Married} | \text{No}) = 4/7$$

$$P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 1/3$$

$$P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0$$

For Taxable Income:

If class = No: sample mean = 110

sample variance = 2975

If class = Yes: sample mean = 90

sample variance = 25

- | $P(\text{Yes}) = 3/10$
- | $P(\text{No}) = 7/10$
- | $P(\text{Yes} | \text{Married}) = 0 \times 3/10 / P(\text{Married})$
- | $P(\text{No} | \text{Married}) = 4/7 \times 7/10 / P(\text{Married})$

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Issues with Naïve Bayes Classifier

Consider the table with Tid = 7 deleted

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Naïve Bayes Classifier:

$P(\text{Refund} = \text{Yes} | \text{No}) = 2/6$
 $P(\text{Refund} = \text{No} | \text{No}) = 4/6$
 $P(\text{Refund} = \text{Yes} | \text{Yes}) = 0$
 $P(\text{Refund} = \text{No} | \text{Yes}) = 1$
 $P(\text{Marital Status} = \text{Single} | \text{No}) = 2/6$
 $P(\text{Marital Status} = \text{Divorced} | \text{No}) = 0$
 $P(\text{Marital Status} = \text{Married} | \text{No}) = 4/6$
 $P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$
 $P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 1/3$
 $P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0/3$

For Taxable Income:

If class = No: sample mean = 91
sample variance = 685

If class = Yes: sample mean = 90
sample variance = 25

Given $X = (\text{Refund} = \text{Yes}, \text{Divorced}, 120\text{K})$

$$P(X | \text{No}) = 2/6 \times 0 \times 0.0083 = 0$$

$$P(X | \text{Yes}) = 0 \times 1/3 \times 1.2 \times 10^{-9} = 0$$

Naïve Bayes will not be able to classify X as Yes or No!

Issues with Naïve Bayes Classifier

- | If one of the conditional probabilities is zero, then the entire expression becomes zero
- | Need to use other estimates of conditional probabilities than simple fractions
- | Probability estimation:

$$\text{Original} : P(A_i | C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace} : P(A_i | C) = \frac{N_{ic} + 1}{N_c + c}$$

$$\text{m - estimate} : P(A_i | C) = \frac{N_{ic} + mp}{N_c + m}$$

c: number of classes

p: prior probability of the class

m: parameter

N_c : number of instances in the class

N_{ic} : number of instances having attribute value A_i in class c

A Simple Example

Text	Tag	Which tag does the sentence <i>A very close game</i> belong to? i.e. $P(\text{sports} \text{A very close game})$
“A great game”	Sports	Feature Engineering: Bag of words i.e use word frequencies without considering order
“The election was over”	Not sports	
“Very clean match”	Sports	Using Bayes Theorem: $P(\text{sports} \text{A very close game})$ $= \frac{P(\text{A very close game} \text{sports}) P(\text{sports})}{P(\text{A very close game})}$ -----
“It was a close election”	Not sports	

We assume that every word in a sentence is **independent** of the other ones

“close” doesn’t appear in sentences of sports tag, So $P(\text{close} | \text{sports}) = 0$, which makes product 0

Laplace smoothing

- Laplace smoothing: we add 1 or in general constant k to every count so it's never zero.
- To balance this, we add the number of possible words to the divisor, so the division will never be greater than 1
- In our case, the possible words are ['a', 'great', 'very', 'over', 'it', 'but', 'game', 'election', 'clean', 'close', 'the', 'was', 'forgettable', 'match'].

Apply Laplace Smoothing

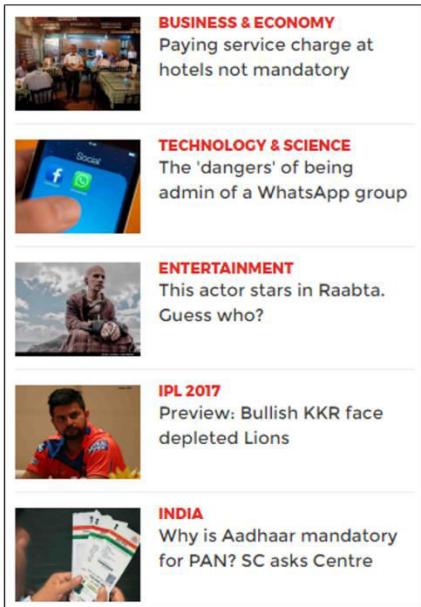
Word	P(word Sports)	P(word Not Sports)
a	2+1 / 11+14	1+1 / 9+14
very	1+1 / 11+14	0+1 / 9+14
close	0+1 / 11+14	1+1 / 9+14
game	2+1 / 11+14	0+1 / 9+14

$$\begin{aligned}
 & P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports) \times \\
 & P(Sports) \\
 & = 2.76 \times 10^{-5} \\
 & = 0.0000276
 \end{aligned}$$

$$\begin{aligned}
 & P(a|Not\ Sports) \times P(very|Not\ Sports) \times P(close|Not\ Sports) \times \\
 & P(game|Not\ Sports) \times P(Not\ Sports) \\
 & = 0.572 \times 10^{-5} \\
 & = 0.00000572
 \end{aligned}$$

Naïve Bayes Classifier Applications

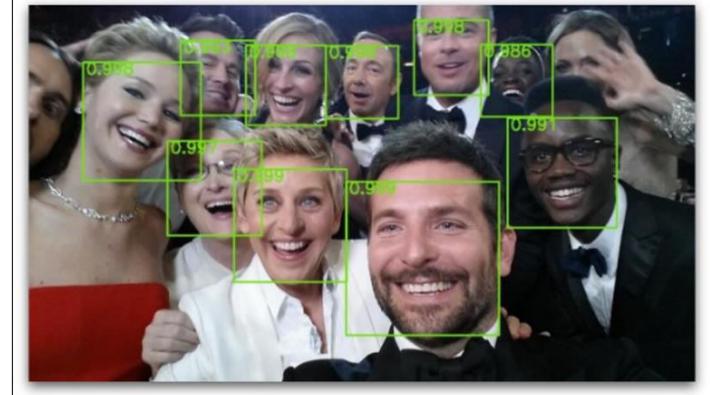
Categorizing News



Email Spam Detection



Face Recognition



Sentiment Analysis



Estimating Parameters: X_i



Continuous

What if features are continuous?

- E.g., character recognition: X_i is intensity at i th pixel
- Gaussian Naïve Bayes (GNB):

$$P(X_i = x | Y = y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} e^{-\frac{(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$



distribution of feature X_i is Gaussian with a mean and variance that can depend on the label y_k and which feature X_i it is



What if features are continuous?

- E.g., character recognition: X_i is intensity at i th pixel
- Gaussian Naïve Bayes (GNB):

$$P(X_i = x | Y = y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} e^{-\frac{(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$



- Different mean and variance for each class k and each pixel i .
- Sometimes assume variance:
 - Is independent of Y (i.e., just have σ_i)
 - Or independent of X (i.e., just have σ_k)
 - Or both (i.e., just have σ)



Estimating parameters: Y discrete, X_i continuous

- Maximum likelihood estimates:

$$\hat{\mu}_{MLE} = \frac{1}{N} \sum_{j=1}^N x_j$$

$$\hat{\mu}_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j X_i^j \delta(Y^j = y_k)$$

ith pixel in jth training image
 jth training image
 kth class

$$\hat{\sigma}_{unbiased}^2 = \frac{1}{N-1} \sum_{j=1}^N (x_j - \hat{\mu})^2$$

$$\hat{\sigma}_{ik}^2 = \frac{1}{\sum_j \delta(Y^j = y_k) - 1} \sum_j (X_i^j - \hat{\mu}_{ik})^2 \delta(Y^j = y_k)$$



Naive Bayes Classifier for Text

- Along with decision trees, neural networks, one of the most practical learning methods.
 - When to use
 - Moderate or large training set available
 - Attributes that describe instances are conditionally independent given classification
 - Successful applications:
 - Diagnosis
 - Classifying text documents
-

Learning to Classify Text

- Why?
 - Learn which news articles are of interest
 - Learn to classify web pages by topic
- Naive Bayes is among most effective algorithms
- What attributes shall we use to represent text documents??

Baseline: Bag of Words Approach

the world of

TOTAL



all about the company

Our energy exploration, production, and distribution operations span the globe, with activities in more than 100 countries.

At TOTAL, we draw our greatest strength from our fast-growing oil and gas reserves. Our strategic emphasis on natural gas provides a strong position in a rapidly expanding market.

Our expanding refining and marketing operations in Asia and the Mediterranean Rim complement already solid positions in Europe, Africa, and the U.S.

Our growing specialty chemicals sector adds balance and profit to the core energy business.

All About The Company

- ▶ All About The Company
- Global Activities
- Corporate Structure
- TOTAL's Story
- Upstream Strategy
- Downstream Strategy
- Chemicals Strategy
- TOTAL Foundation
- Homepage

aardvark	0
about	2
all	2
Africa	1
apple	0
anxious	0
...	
gas	1
...	
oil	1
...	
Zaire	0

Case Study: Text Classification

- Classify e-mails
 - $Y = \{\text{Spam}, \text{NotSpam}\}$
- Classify news articles
 - $Y = \text{what is the topic of the article?}$
- Classify webpages
 - $Y = \{\text{student}, \text{professor}, \text{project}, \dots\}$
- What about the features X ?
 - The text!

Features X are entire document - X_i for i th word in article

Article from rec.sport.hockey

Path: cantaloupe.srv.cs.cmu.edu!das-news.harvard.e
From: xxx@yyy.zzz.edu (John Doe)
Subject: Re: This year's biggest and worst (opinic
Date: 5 Apr 93 09:53:39 GMT

I can only comment on the Kings, but the most obvious candidate for pleasant surprise is Alex Zhitnik. He came highly touted as a defensive defenseman, but he's clearly much more than that. Great skater and hard shot (though wish he were more accurate). In fact, he pretty much allowed the Kings to trade away that huge defensive liability Paul Coffey. Kelly Hrudey is only the biggest disappointment if you thought he was any good to begin with. But, at best, he's only a mediocre goaltender. A better choice would be Tomas Sandstrom, though not through any fault of his own, but because some thugs in Toronto decided

Naïve Bayes for Text Classification

- **Naïve Bayes assumption helps a lot!**
 - $P(X_i = x_i | Y = y)$ is just the probability of observing word x_i at the i th position in a document on topic y .
 - Assume X_i is independent of all other words in document given the label y :
$$P(X_i = x_i | Y = y, X_{-i}) = P(X_i = x_i | Y = y).$$

$$h_{NB}(x) = \arg \max_y P(y) \prod_{i=1}^{\text{lengthDoc}} P(X_i = x_i | y)$$

- For each label y , have 1000 distributions of size 10000 to estimate.
- This is 10000×1000 items, which is big but much less than 10000^{1000} ...

Bag of Words Model

- Typical additional assumption – **Position in document doesn't matter:**

$$P(X_i = x_i | Y = y) = P(X_k = x_i | Y = y)$$

the probability distributions of words are the same at each position: $P_i = P_j$ for all i, j .

- “**Bag of Words**” model – order of words in the document is ignored
- Now, only 10000 quantities $P(x_i|y)$ to estimate for each label y (the 10000 possible values that x_i can be) plus the prior.

$$h_{NB}(x) = \arg \max_y P(y) \prod_{i=1}^{1000} P(x_i|y)$$



Bag of Words model

- Typical additional assumption – **Position in document doesn't matter:**
 $P(X_i = x_i | Y = y) = P(X_k = x_i | Y = y)$
- “**Bag of Words**” model – order of words on the page ignored

Can simplify further:

$$\prod_{i=1}^{\text{lengthDoc}} P(x_i|y) = \prod_{w=1}^W P(w|y)^{\text{count}(w)}$$



Bag of Words representation

- Since we are assuming the order of words doesn't matter, an alternative representation of document is as vector of counts:
 - $x^{(j)}$ = number of occurrences of word j in document x .
 - Typical document: [0 0 1 0 0 3 0 0 0 1 0 0 0 1 0 0 2 0 0 ...]
 - Called “bag of words” or “term vector” or “vector space model” representation



Naïve Bayes with Bag of Words for text classification

- Learning phase
 - Class Prior $P(Y)$
 - $P(X_i|Y)$
- Test phase:
 - For each document
 - Use naïve Bayes decision rule

$$h_{NB}(x) = \arg \max_y P(y) \prod_{i=1}^{1000} P(x_i|y)$$



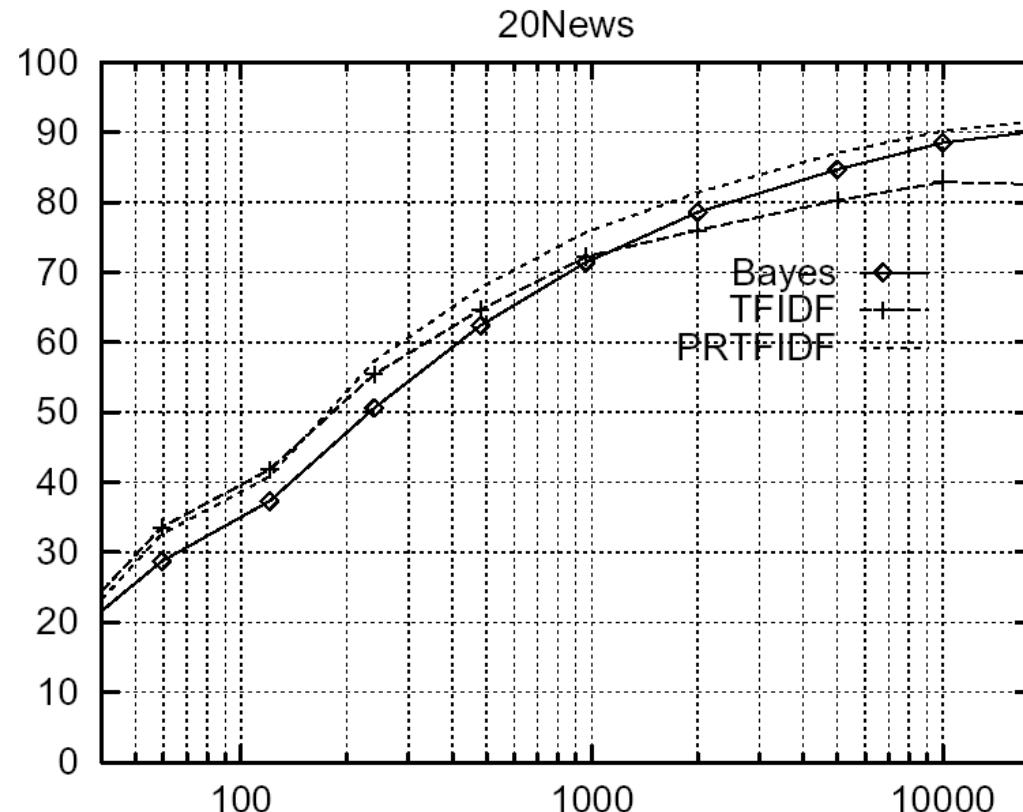
Twenty NewsGroups

- Given 1000 training documents from each group Learn to classify new documents according to which newsgroup it came from

comp.graphics	misc.forsale	alt.atheism	sci.space
comp.os.ms-windows.misc	rec.autos	soc.religion.christian	sci.crypt
comp.sys.ibm.pc.hardware	rec.motorcycles	talk.religion.misc	sci.electronics
comp.sys.mac.hardware	rec.sport.baseball	talk.politics.mideast	sci.med
comp.windows.x	rec.sport.hockey	talk.politics.misc	
		talk.politics.guns	

- Naive Bayes: 89% classification accuracy

Learning Curve for 20 Newsgroups



- Accuracy vs. Training set size (1/3 withheld for test)

Summary: Learning to Classify Text

Target concept Interesting? : *Document* $\rightarrow \{+, -\}$

1. Represent each document by vector of words
 - one attribute per word position in document
2. Learning: Use training examples to estimate
 - $P(+)$
 - $P(-)$
 - $P(doc|+)$
 - $P(doc|-)$

Naive Bayes conditional independence assumption

$$P(doc|v_j) = \prod_{i=1}^{length(doc)} P(a_i = w_k | v_j)$$

where $P(a_i = w_k | v_j)$ is probability that word in position i is w_k , given v_j

one more assumption: $P(a_i = w_k | v_j) = P(a_m = w_k | v_j), \forall i, m$

Summary: Learning to Classify Text

`LEARN_NAIVE_BAYES_TEXT (Examples, V)`

1. *collect all words and other tokens that occur in Examples*

- *Vocabulary* \leftarrow all distinct words and other tokens in *Examples*

2. *calculate the required $P(v_j)$ and $P(w_k \mid v_j)$ probability terms*

- For each target value v_j in V do
 - $docs_j \leftarrow$ subset of *Examples* for which the target value is v_j
 - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
 - $Text_j \leftarrow$ a single document created by concatenating all members of $docs_j$

Summary: Learning to Classify Text

- $n \leftarrow$ total number of words in $Text_j$ (counting duplicate words multiple times)
- for each word w_k in $Vocabulary$
 - * $n_k \leftarrow$ number of times word w_k occurs in $Text_j$
 - * $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

CLASSIFY_NAIVE_BAYES_TEXT (Doc)

- $positions \leftarrow$ all word positions in Doc that contain tokens found in $Vocabulary$
- Return v_{NB} where $v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_{i \in positions} P(a_i|v_j)$

Probabilistic Generative Model versus Probabilistic Discriminative Model

Generative	Discriminative
Ex: Naïve Bayes	Ex: Logistic Regression
Estimate $P(Y)$ and $P(X Y)$	Finds class label directly $P(Y X)$
Prediction $\hat{y} = \operatorname{argmax}_y P(Y = y)P(X = x_{new} Y = y)$	Prediction $\hat{y} = P(Y = y X = x_{new})$

Most Probable Classification of New Instances

- So far we've sought the most probable *hypothesis* given the data D (i.e., h_{MAP})
- Given new instance x , what is its most probable *classification*?
 - $h_{MAP}(x)$ is not the most probable classification!
- Consider:
 - Three possible hypotheses:
$$P(h_1|D) = .4, P(h_2|D) = .3, P(h_3|D) = .3$$
 - Given new instance x ,
$$h_1(x) = +, h_2(x) = -, h_3(x) = -$$
 - What's most probable classification of x ?

Bayes Optimal Classifier

- **Bayes optimal classification:**

$$\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

- Example:

$$P(h_1|D) = .4, P(-|h_1) = 0, P(+|h_1) = 1$$

$$P(h_2|D) = .3, P(-|h_2) = 1, P(+|h_2) = 0$$

$$P(h_3|D) = .3, P(-|h_3) = 1, P(+|h_3) = 0$$

therefore

$$\sum_{h_i \in H} P(+|h_i)P(h_i|D) = .4$$

$$\sum_{h_i \in H} P(-|h_i)P(h_i|D) = .6$$

and

$$\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) = -$$

Gibbs Classifier

- Bayes optimal classifier provides best result, but can be expensive if many hypotheses.
- Gibbs algorithm:
 1. Choose one hypothesis at random, according to $P(h | D)$
 2. Use this to classify new instance
- Surprising fact: Assume target concepts are drawn at random from H according to priors on H . Then:

$$E[\text{error}_{\text{Gibbs}}] \leq 2E[\text{error}_{\text{BayesOptional}}]$$

Features of Bayesian learning

- Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct.
 - Flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example.
 - Bayesian methods can accommodate hypotheses that make probabilistic predictions (e.g., hypotheses such as "this pneumonia patient has a 93% chance of complete recovery").
-

Features of Bayesian learning

- Prior knowledge can be combined with observed data to determine the final probability of a hypothesis.
 - Prior knowledge is provided by asserting
 - prior probability for each candidate hypothesis, and
 - probability distribution over observed data for each possible hypothesis.
 - New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.
-

Practical Issues of Bayesian learning

- Require initial knowledge of many probabilities
 - Often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions.
- Significant computational cost required to determine the Bayes optimal hypothesis in the general case (linear in the number of candidate hypotheses)

Logistic Regression from Bayesian Perspective

- Consider learning $f: X \rightarrow Y$, where
 - X is a vector of real-valued features, $\langle X_1 \dots X_n \rangle$
 - Y is boolean
 - assume all X_i are conditionally independent given Y
 - model $P(X_i | Y = y_k)$ as Gaussian $N(\mu_{ik}, \sigma_i)$
 - model $P(Y)$ as Bernoulli ($P(Y=1) = \pi$)
- What does that imply about the form of $P(Y|X)$?

Derive form for $P(Y|X)$ for Gaussian $P(X_i|Y=y_k)$ assuming $\sigma_{ik} = \sigma_i$

$$\begin{aligned}
 P(Y=1|X) &= \frac{P(Y=1)P(X|Y=1)}{P(Y=1)P(X|Y=1) + P(Y=0)P(X|Y=0)} \\
 &= \frac{1}{1 + \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}} & P(x | y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} e^{\frac{-(x-\mu_{ik})^2}{2\sigma_{ik}^2}} \\
 &= \frac{1}{1 + \exp(\ln \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)})} & P(Y=1) = \pi \\
 &= \frac{1}{1 + \exp(-\ln \frac{1-\pi}{\pi} + \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)})} \\
 && \text{The term } \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)} \text{ is highlighted with a red box and has an arrow pointing to it from the expression below.} \\
 && \boxed{\sum_i \left(\frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} X_i + \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \right)} \\
 P(Y=1|X) &= \frac{1}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)}
 \end{aligned}$$

Very convenient!

$$P(Y = 1|X = \langle X_1, \dots, X_n \rangle) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

implies

$$P(Y = 0|X = \langle X_1, \dots, X_n \rangle) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

implies

$$\frac{P(Y = 0|X)}{P(Y = 1|X)} = \exp(w_0 + \sum_i w_i X_i)$$

implies

$$\ln \frac{P(Y = 0|X)}{P(Y = 1|X)} = w_0 + \sum_i w_i X_i$$

linear
classification
rule!

Training Logistic Regression: MCLE

- Choose parameters $W = \langle w_0, \dots, w_n \rangle$ to maximize conditional likelihood of training data

where

$$P(Y = 0|X, W) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

$$P(Y = 1|X, W) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

- Training data $D = \{\langle X^1, Y^1 \rangle, \dots, \langle X^L, Y^L \rangle\}$
- Data likelihood = $\prod_l P(X^l, Y^l | W)$
- Data conditional likelihood = $\prod_l P(Y^l | X^l, W)$

$$W_{MCLE} = \arg \max_W \prod_l P(Y^l | W, X^l)$$

Expressing Conditional Log Likelihood

$$l(W) \equiv \ln \prod_l P(Y^l | X^l, W) = \sum_l \ln P(Y^l | X^l, W)$$

$$P(Y = 0 | X, W) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

$$P(Y = 1 | X, W) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

$$\begin{aligned} l(W) &= \sum_l Y^l \ln P(Y^l = 1 | X^l, W) + (1 - Y^l) \ln P(Y^l = 0 | X^l, W) \\ &= \sum_l Y^l \ln \frac{P(Y^l = 1 | X^l, W)}{P(Y^l = 0 | X^l, W)} + \ln P(Y^l = 0 | X^l, W) \\ &= \sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1 + \exp(w_0 + \sum_i^n w_i X_i^l)) \end{aligned}$$

MLE vs MAP

- Maximum conditional likelihood estimate

$$W \leftarrow \arg \max_W \ln \prod_l P(Y^l | X^l, W)$$

$$w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))$$

- Maximum a posteriori estimate with prior $W \sim N(0, \sigma I)$

$$W \leftarrow \arg \max_W \ln [P(W) \prod_l P(Y^l | X^l, W)]$$

$$w_i \leftarrow w_i - \eta \lambda w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))$$

MAP estimates and Regularization

- Maximum a posteriori estimate with prior $W \sim N(0, \sigma^2 I)$

$$W \leftarrow \arg \max_W \ln[P(W) \prod_l P(Y^l | X^l, W)]$$

$$w_i \leftarrow w_i - \eta \lambda w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))$$

λ is called a “regularization” term

- helps reduce overfitting
- keep weights nearer to zero (if $P(W)$ is zero mean Gaussian prior), or whatever the prior suggests
- used very frequently in Logistic Regression

Naïve Bayes versus Logistic Regression

- Naïve Bayes are Generative Models which Logistic Regression are Discriminative Models
 - Naïve Bayes easy to construct
 - Naïve Bayes better on smaller datasets
 - Naive Bayes also assumes that the features are conditionally independent. Real data sets are never perfectly independent
 - When the training size reaches infinity, logistic regression performs better than the generative model Naive Bayes.
 - Optional reading by Ng and Jordan has proofs and experiments
 - Logistic regression allows arbitrary features
-

Naïve Bayes vs Logistic Regression

Consider Y boolean, X_i continuous, $X = \langle X_1 \dots X_n \rangle$

Number of parameters:

- NB: $4n + 1$
- LR: $n+1$

Estimation method:

- NB parameter estimates are uncoupled
- LR parameter estimates are coupled

G. Naïve Bayes vs. Logistic Regression

[Ng & Jordan, 2002]

Recall two assumptions deriving form of LR from GNBayes:

1. X_i conditionally independent of X_k given Y
2. $P(X_i | Y = y_k) = N(\mu_{ik}, \sigma_i)$, \leftarrow not $N(\mu_{ik}, \sigma_{ik})$

Consider three learning methods:

- GNB (assumption 1 only) -- decision surface can be non-linear
- GNB2 (assumption 1 and 2) – decision surface linear
- LR -- decision surface linear, trained without assumption 1.

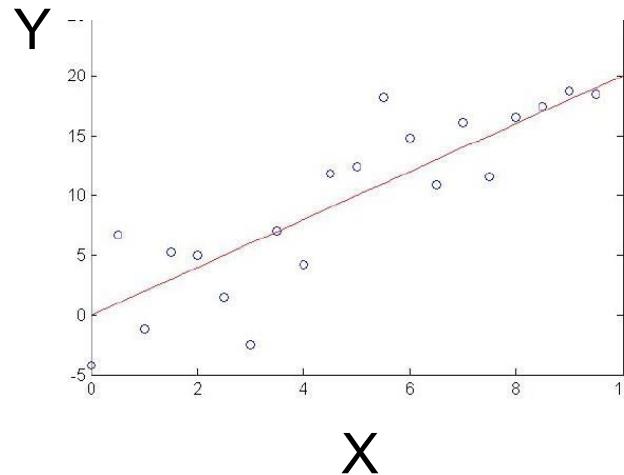
Which method works better if we have infinite training data, and...

- Both (1) and (2) are satisfied: $LR = GNB2 = GNB$
- (1) is satisfied, but not (2) : $GNB > GNB2, GNB > LR, LR > GNB2$
- Neither (1) nor (2) is satisfied: $GNB > GNB2, LR > GNB2, LR > <GNB$

Maximum likelihood and least-squared error hypotheses

- A set of m training examples is provided, where the target value of each example is corrupted by random noise drawn according to a Normal probability distribution.
- Each training example is a pair of the form (x_i, d_i) where $d_i = f(x_i) + e_i$. Here $f(x_i)$ is the noise-free value of the target function and e_i is a random variable representing the noise.
 - values of the e_i are drawn independently and that they are distributed according to a Normal distribution with zero mean

Choose parameterized form for $P(Y|X; \theta)$



Assume Y is some deterministic $f(X)$, plus random noise

$$y = f(x) + \epsilon \quad \text{where } \epsilon \sim N(0, \sigma)$$

Therefore Y is a random variable that follows the distribution

$$p(y|x) = N(f(x), \sigma)$$

and the expected value of y for any given x is $f(x)$

Training Linear Regression : Maximum Conditional Likelihood Estimate (MCLE)

$$p(y|x; W) = N(w_0 + w_1x, \sigma^2)$$

How can we learn W from the training data?

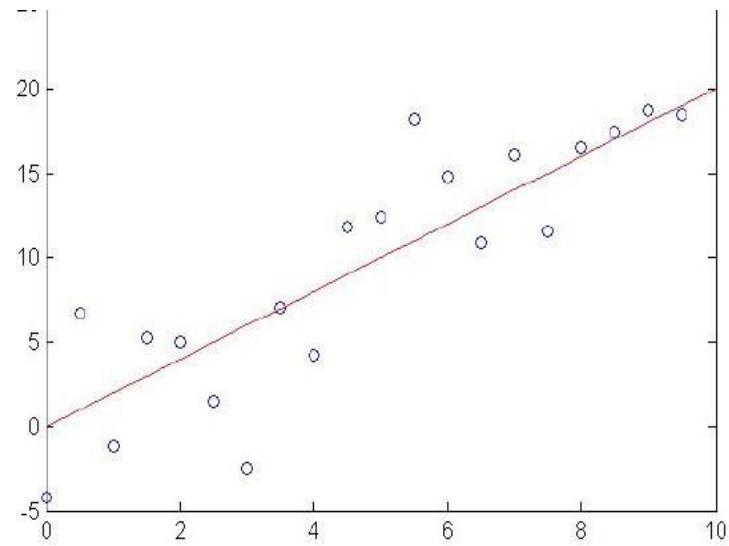
Learn Maximum Conditional Likelihood Estimate!

$$W_{MCLE} = \arg \max_W \prod_l p(y^l|x^l, W)$$

$$W_{MCLE} = \arg \max_W \sum_l \ln p(y^l|x^l, W)$$

where

$$p(y|x; W) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{y-f(x;W)}{\sigma})^2}$$



Training Linear Regression: MCLE

Learn Maximum Conditional Likelihood Estimate

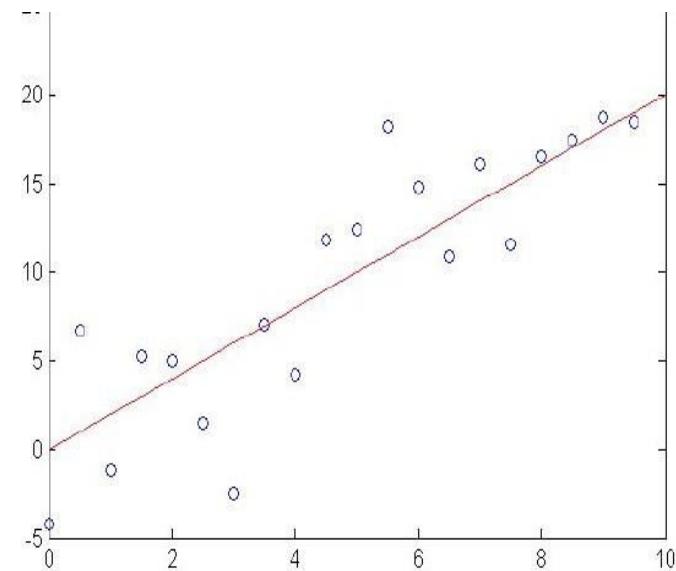
$$W_{MCLE} = \arg \max_W \sum_l \ln p(y^l | x^l, W)$$

where

$$p(y|x; W) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{y-f(x;W)}{\sigma})^2}$$

so:

$$W_{MCLE} = \arg \min_W \sum_l (y - f(x; W))^2$$



Decision Theory

- Suppose \mathbf{x} is an input vector together with a corresponding vector \mathbf{t} of target variables
- Goal: predict \mathbf{t} given a new value for \mathbf{x} .
- The joint probability distribution $p(\mathbf{x}, \mathbf{t})$ provides a complete summary of the uncertainty associated with these variables.
- Determination of $p(\mathbf{x}, \mathbf{t})$ from a set of training data is called *inference*

Decision Theory

Inference step

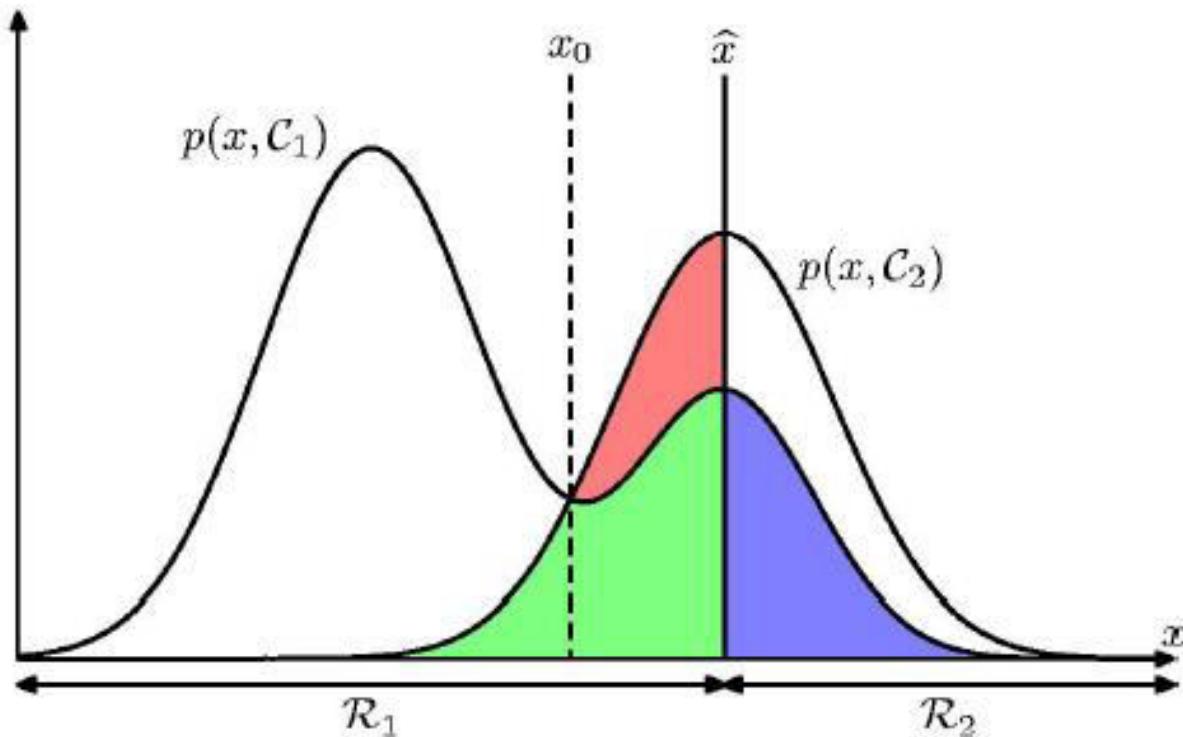
Determine either $p(t|x)$ or $p(x,t)$.

Decision step

For given x , determine optimal t .

$$p(\mathcal{C}_k|x) = \frac{p(x|\mathcal{C}_k)p(\mathcal{C}_k)}{p(x)}$$

Minimum Misclassification Rate



$$\begin{aligned}
 p(\text{mistake}) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \\
 &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x}.
 \end{aligned}$$

Minimum Misclassification Rate

$$\begin{aligned} p(\text{correct}) &= \sum_{k=1}^K p(\mathbf{x} \in \mathcal{R}_k, \mathcal{C}_k) \\ &= \sum_{k=1}^K \int_{\mathcal{R}_k} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x} \end{aligned}$$



BITS Pilani
Pilani Campus

Ensemble Learning

Dr. Sugata Ghosal

sugata.ghosal@pilani.bits-pilani.ac.in

Contents

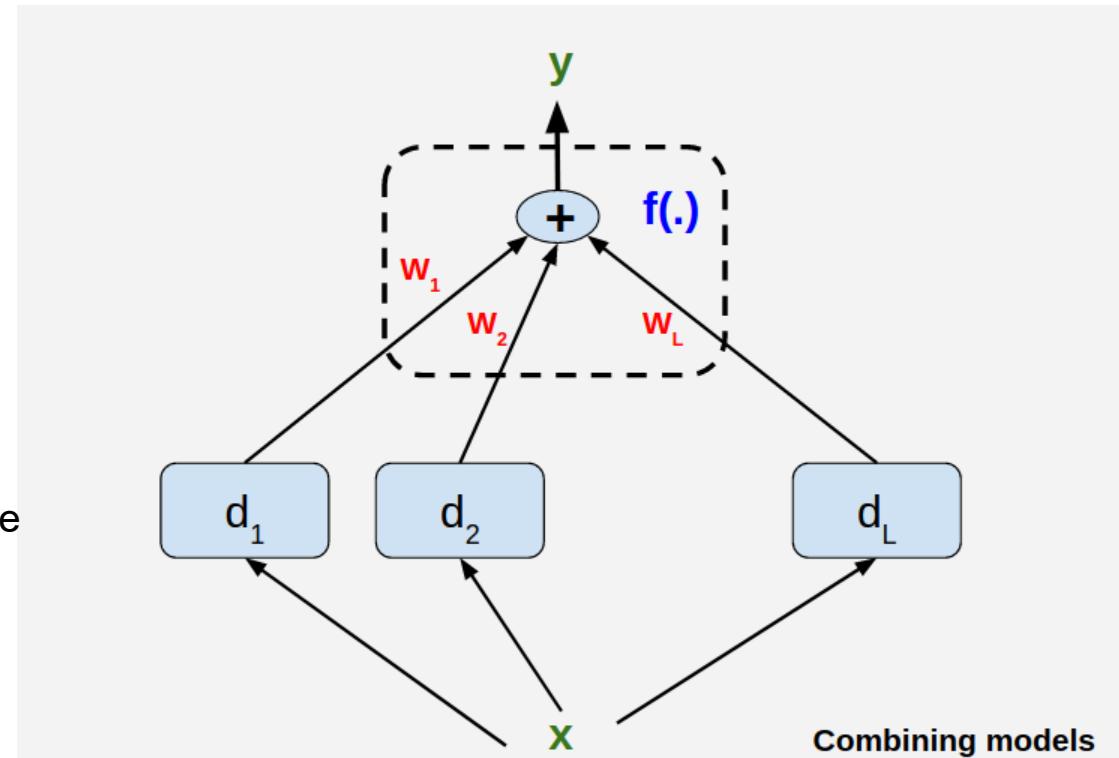
- Combining classifiers
- Bagging
- Boosting
- Random Forest Algorithm
- AdaBoost Algorithm
- Gradient Boosting

Getting Started

- No Free Lunch Theorem: There is no algorithm that is always the most accurate
- Each learning algorithm dictates a certain model that comes with a set of assumptions
 - Each algorithm converges to a different solution and fails under different circumstances
 - The best tuned learners could miss some examples and there could be other learners which works better on (may be only) those !
 - In the absence of a single expert (*a superior model*) , a committee (*combinations of models*) can do better !
 - A committee can work in many ways ...

Committee of Models

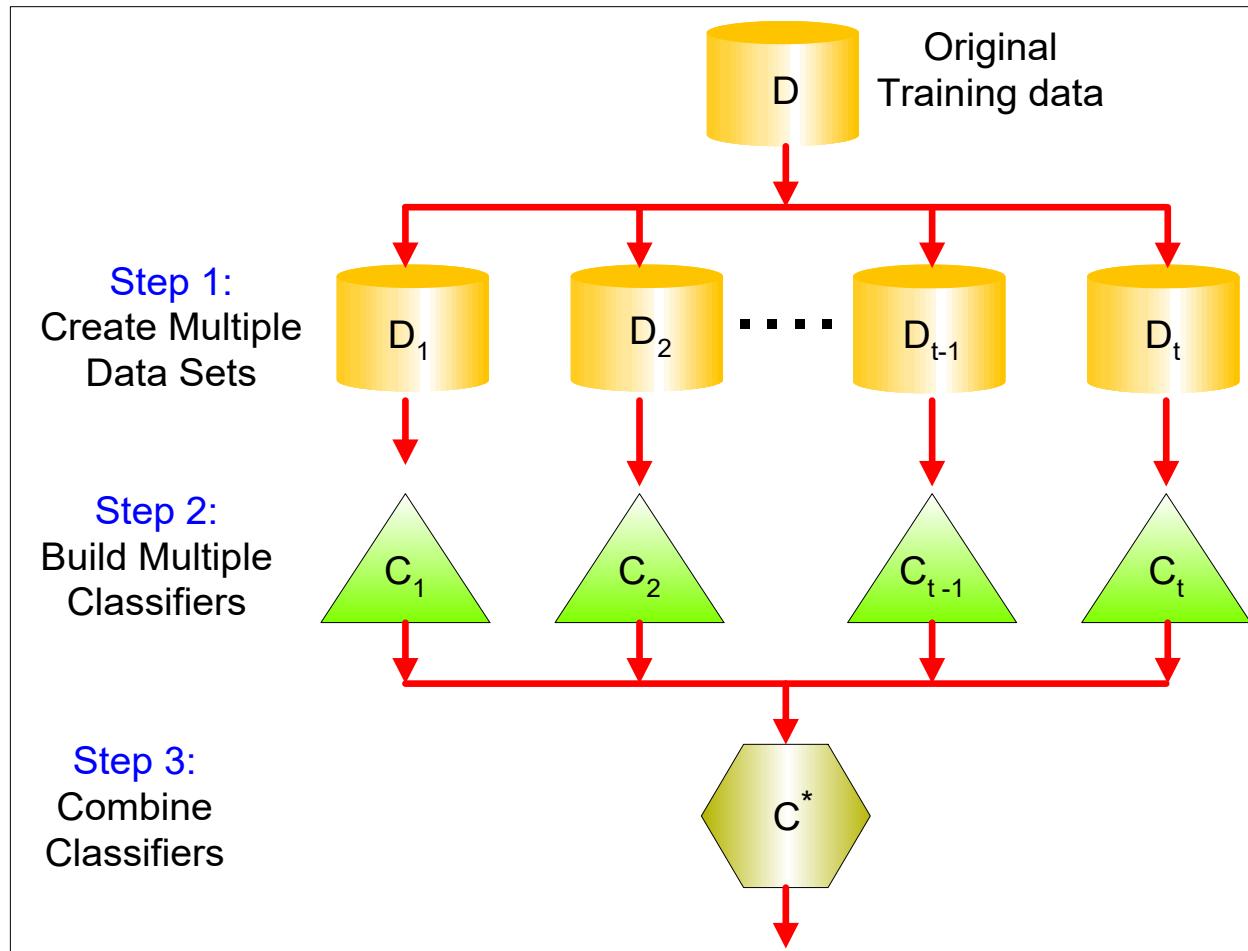
- Committee Members are base learners !
- Major challenges dealing with this committee
 - Expertise of each of the members (Does it help / not?)
 - Combining the results from the members for better performance



Ensemble Methods

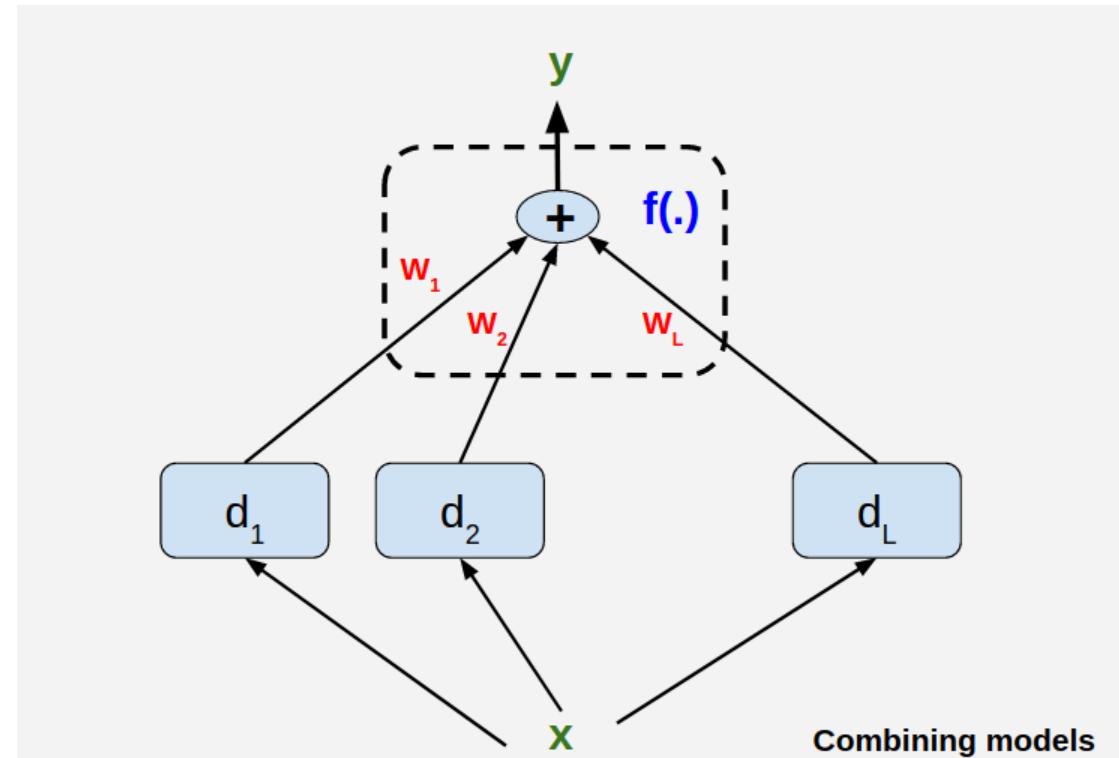
- **Ensemble methods** use multiple learning algorithms to obtain better [predictive performance](#) than could be obtained from any of the constituent learning algorithms alone
- Construct a set of classifiers from the training data
- Predict class label of test records by combining the predictions made by multiple classifiers
- Tend to reduce problems related to over-fitting of the training data.

General Approach



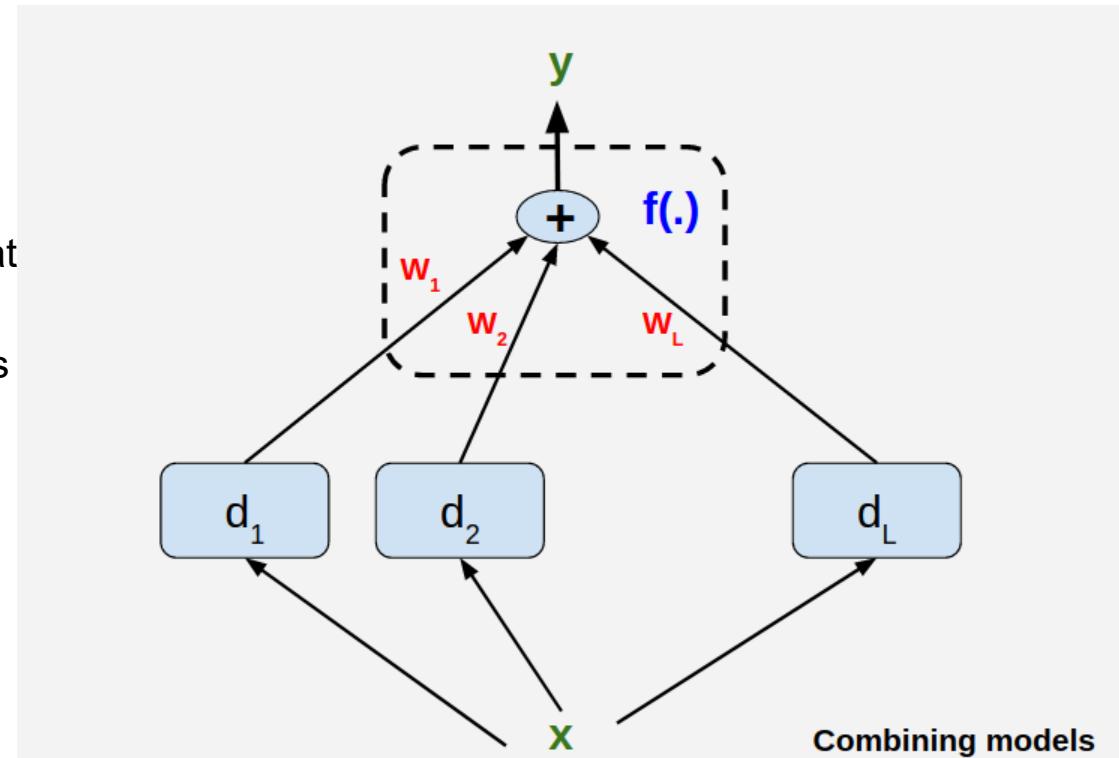
Issue 1 : On the members (Base Learners)

- It does not help if all learners are good/bad at roughly same thing
 - Need Diverse Learners



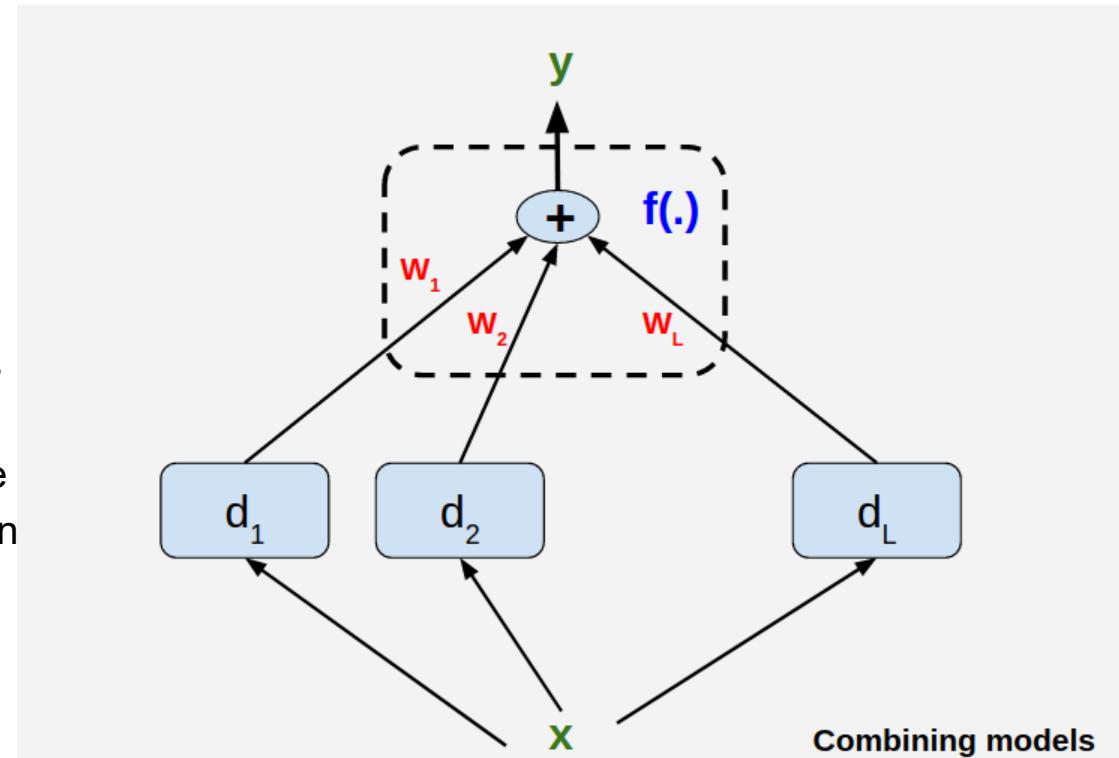
Issue 1 : On the members (Base Learners)

- Use Different Algorithms
 - Different algorithms make different assumptions
- Use Different Hyperparameters, that is ,
 - vary the structure of neural nets



Issue 1 : On the members (Base Learners)

- Different input representations
 - Uttered words + video information of speakers clips
 - image + text annotations
- Different training sets
 - Draw different random samples of data
 - Partition data in the input space and have learners specialized in those spaces (mixture of experts)



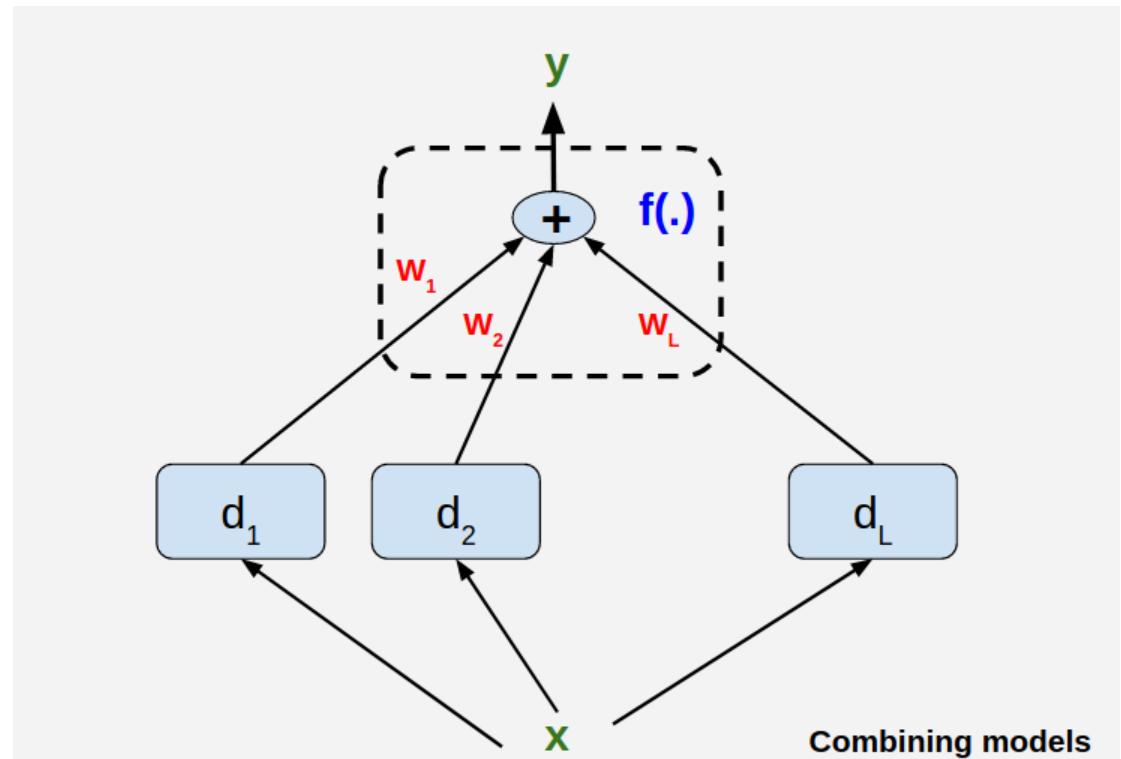
Issue -2 : Combining Results

$$y = f(d_1, d_2, \dots, d_L | \Phi)$$

A Simple Combination Scheme:

$$y = \sum_{j=1}^L w_j d_j$$

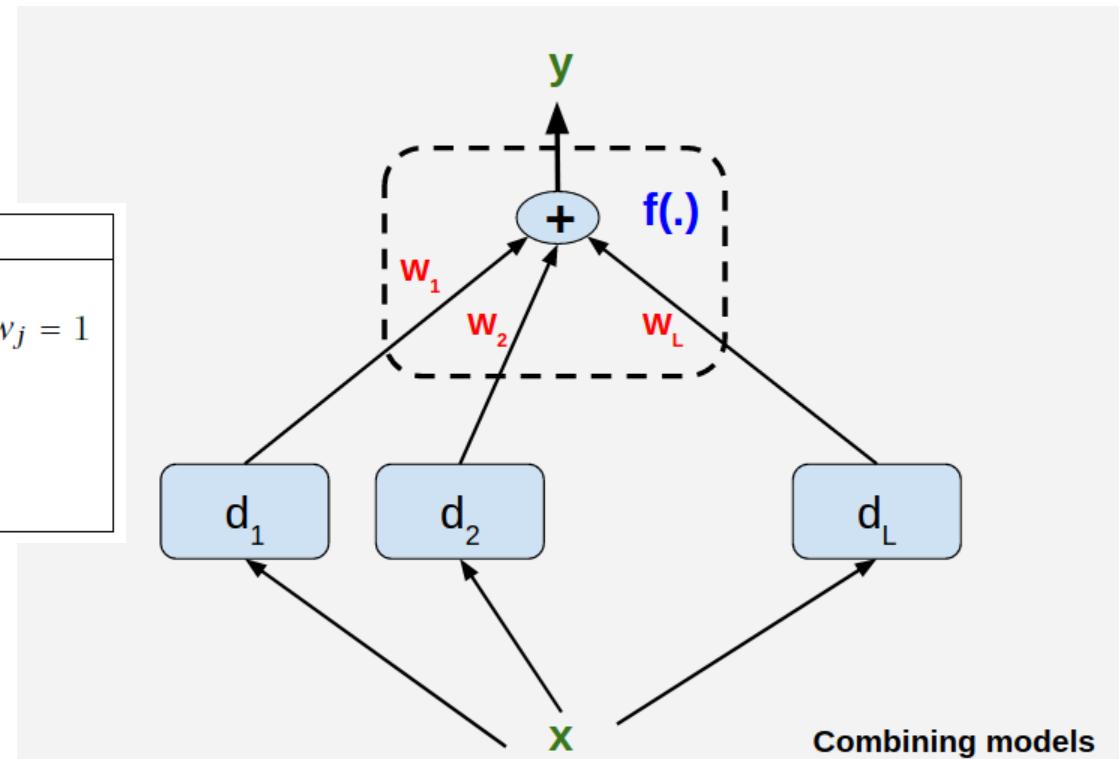
$$w_j \geq 0 \text{ and } \sum_{j=1}^L w_j = 1$$



Issue -2 : Combining Results

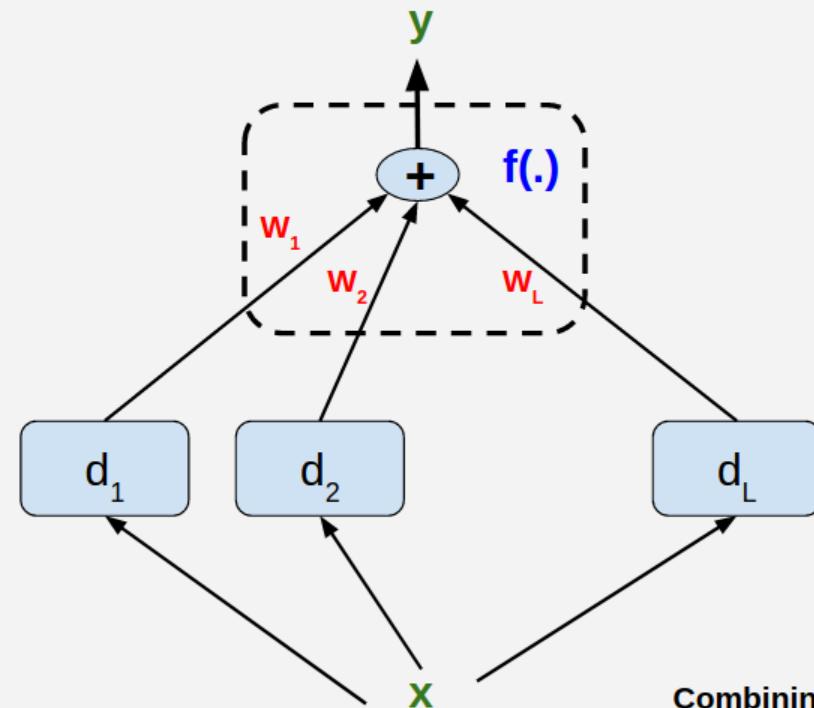
$$y = f(d_1, d_2, \dots, d_L | \Phi)$$

Rule	Fusion function $f(\cdot)$
Sum	$y_i = \frac{1}{L} \sum_{j=1}^L d_{ji}$
Weighted sum	$y_i = \sum_j w_j d_{ji}, w_j \geq 0, \sum_j w_j = 1$
Median	$y_i = \text{median}_j d_{ji}$
Minimum	$y_i = \min_j d_{ji}$
Maximum	$y_i = \max_j d_{ji}$
Product	$y_i = \prod_j d_{ji}$



Issue -2 : Combining Results

	C_1	C_2	C_3
d_1	0.2	0.5	0.3
d_2	0.0	0.6	0.4
d_3	0.4	0.4	0.2
Sum	0.2	0.5	0.3
Median	0.2	0.5	0.4
Minimum	0.0	0.4	0.2
Maximum	0.4	0.6	0.4
Product	0.0	0.12	0.032



When does Ensemble work?

- Ensemble classifier performs better than the base classifiers when error rate is smaller than 0.5
- Necessary conditions for an ensemble classifier to perform better than a single classifier:
 - Base classifiers should be independent of each other
 - Base classifiers should do better than a classifier that performs random guessing

Why Majority Vote?

- assume n independent classifiers with a base error rate ϵ
- here, independent means that the errors are uncorrelated
- assume a binary classification task
- assume the error rate is better than random guessing (i.e., lower than 0.5 for binary classification)

$$\forall \epsilon_i \in \{\epsilon_1, \epsilon_2, \dots, \epsilon_n\}, \epsilon_i < 0.5$$

Why Majority Vote?

- assume n independent classifiers with a base error rate ϵ
- here, independent means that the errors are uncorrelated
- assume a binary classification task
- assume the error rate is better than random guessing (i.e., lower than 0.5 for binary classification)

$$\forall \epsilon_i \in \{\epsilon_1, \epsilon_2, \dots, \epsilon_n\}, \epsilon_i < 0.5$$

The probability that we make a wrong prediction via the ensemble if k classifiers predict the same class label

$$P(k) = \binom{n}{k} \epsilon^k (1 - \epsilon)^{n-k} \quad k > [n/2]$$

(Probability mass func. of a binomial distr.)

Why Majority Vote?

The probability that we make a wrong prediction via the ensemble if k classifiers predict the same class label

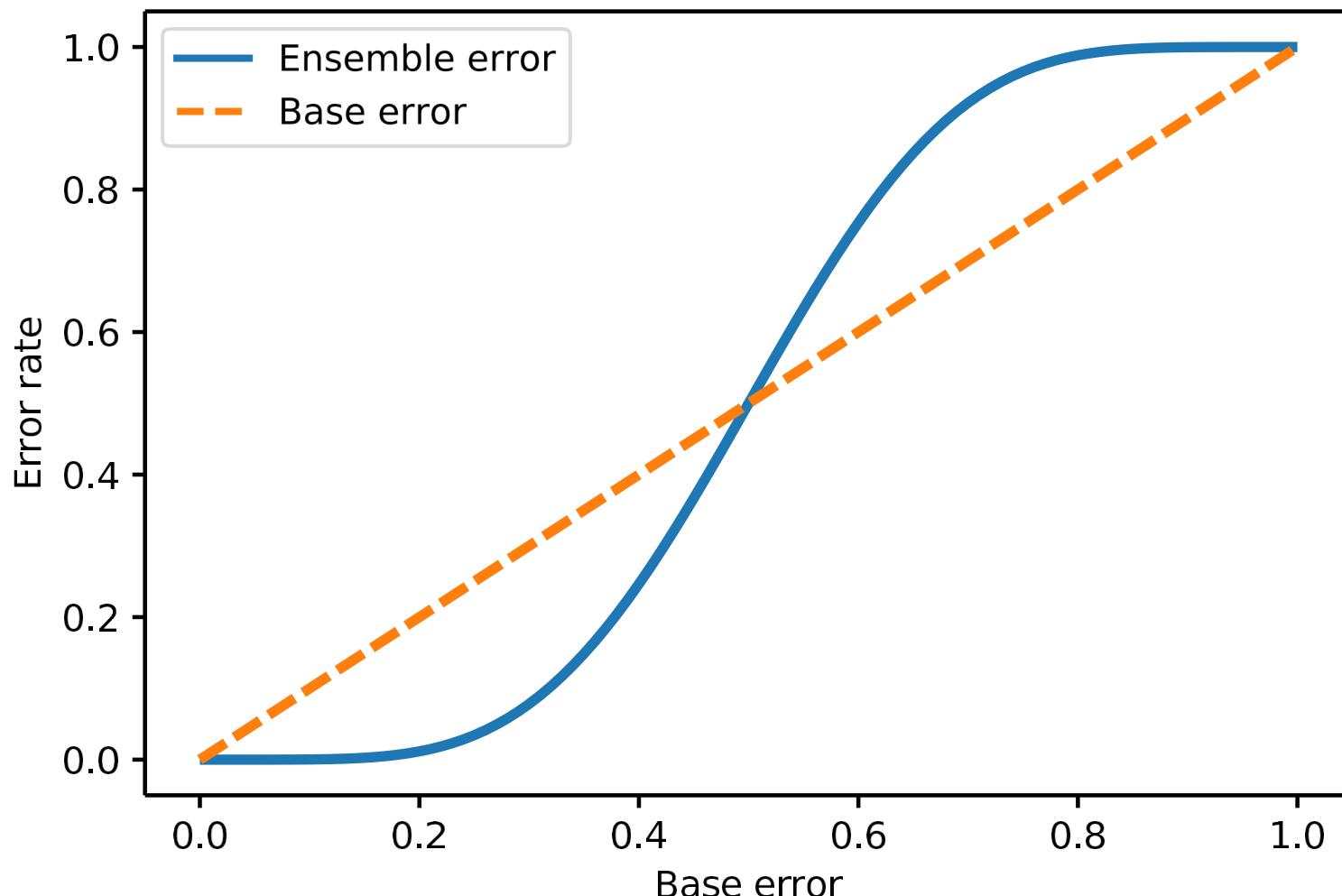
$$P(k) = \binom{n}{k} \epsilon^k (1 - \epsilon)^{n-k} \quad k > [n/2]$$

Ensemble error:

$$\epsilon_{\text{ens}} = \sum_k^n \binom{n}{k} \epsilon^k (1 - \epsilon)^{n-k}$$

$$\epsilon_{\text{ens}} = \sum_{k=6}^{11} \binom{11}{k} 0.25^k (1 - 0.25)^{11-k} = 0.034$$

$$\epsilon_{\text{ens}} = \sum_k^n \binom{n}{k} \epsilon^k (1 - \epsilon)^{n-k}$$



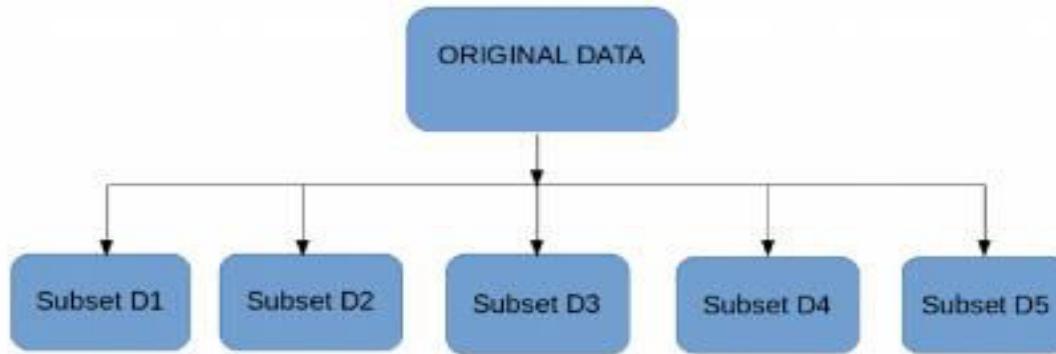
Types of Ensemble Methods

- Manipulate data distribution
 - Example: bagging, boosting
- Manipulate input features
 - Example: random forests

Bagging (Bootstrap Aggregating)

- Technique uses these subsets (bags) to get a fair idea of the distribution (complete set).
- The size of subsets created for bagging may be less than the original set.
- Bootstrapping is a sampling technique in which we create subsets of observations from the original dataset, **with replacement**.
- When you sample with replacement, items are independent. One item does not affect the outcome of the other. You have $1/7$ chance of choosing the first item and a $1/7$ chance of choosing the second item.
- If the two items are **dependent**, or linked to each other. When you choose the first item, you have a $1/7$ probability of picking a item. Assuming you don't replace the item, you only have six items to pick from. That gives you a $1/6$ chance of choosing a second item.

Bagging



- Multiple subsets are created from the original dataset, selecting observations with replacement.
- A base model (weak model) is created on each of these subsets.
- The models run in parallel and are independent of each other.
- The final predictions are determined by combining the predictions from all the models.

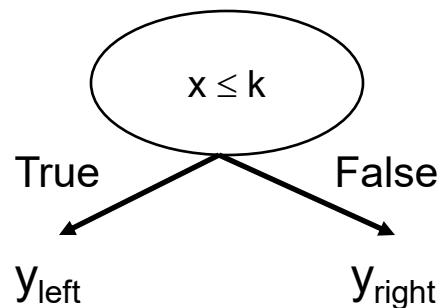
Bagging Example

- Consider 1-dimensional data set:

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Classifier is a decision stump
 - Decision rule: $x \leq k$ versus $x > k$
 - Split point k is chosen based on entropy



Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

$x \leq 0.35 \rightarrow y = 1$
 $x > 0.35 \rightarrow y = -1$

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

$x \leq 0.01 \rightarrow y = -1$
 $x > 0.01 \rightarrow y = 1$

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$x \leq 0.35 \rightarrow y = 1$
 $x > 0.35 \rightarrow y = -1$

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$x \leq 0.3 \rightarrow y = 1$
 $x > 0.3 \rightarrow y = -1$

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

$x \leq 0.35 \rightarrow y = 1$
 $x > 0.35 \rightarrow y = -1$

Bagging Example

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

$$x \leq 0.75 \rightarrow y = -1$$
$$x > 0.75 \rightarrow y = 1$$

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

$$x \leq 0.75 \rightarrow y = -1$$
$$x > 0.75 \rightarrow y = 1$$

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$$x \leq 0.75 \rightarrow y = -1$$
$$x > 0.75 \rightarrow y = 1$$

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$$x \leq 0.75 \rightarrow y = -1$$
$$x > 0.75 \rightarrow y = 1$$

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

$$x \leq 0.05 \rightarrow y = 1$$
$$x > 0.05 \rightarrow y = 1$$

Bagging Example

- Summary of Training sets:

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

Bagging Example

- Assume test set is the same as the original data
- Use majority vote to determine class of ensemble classifier

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1

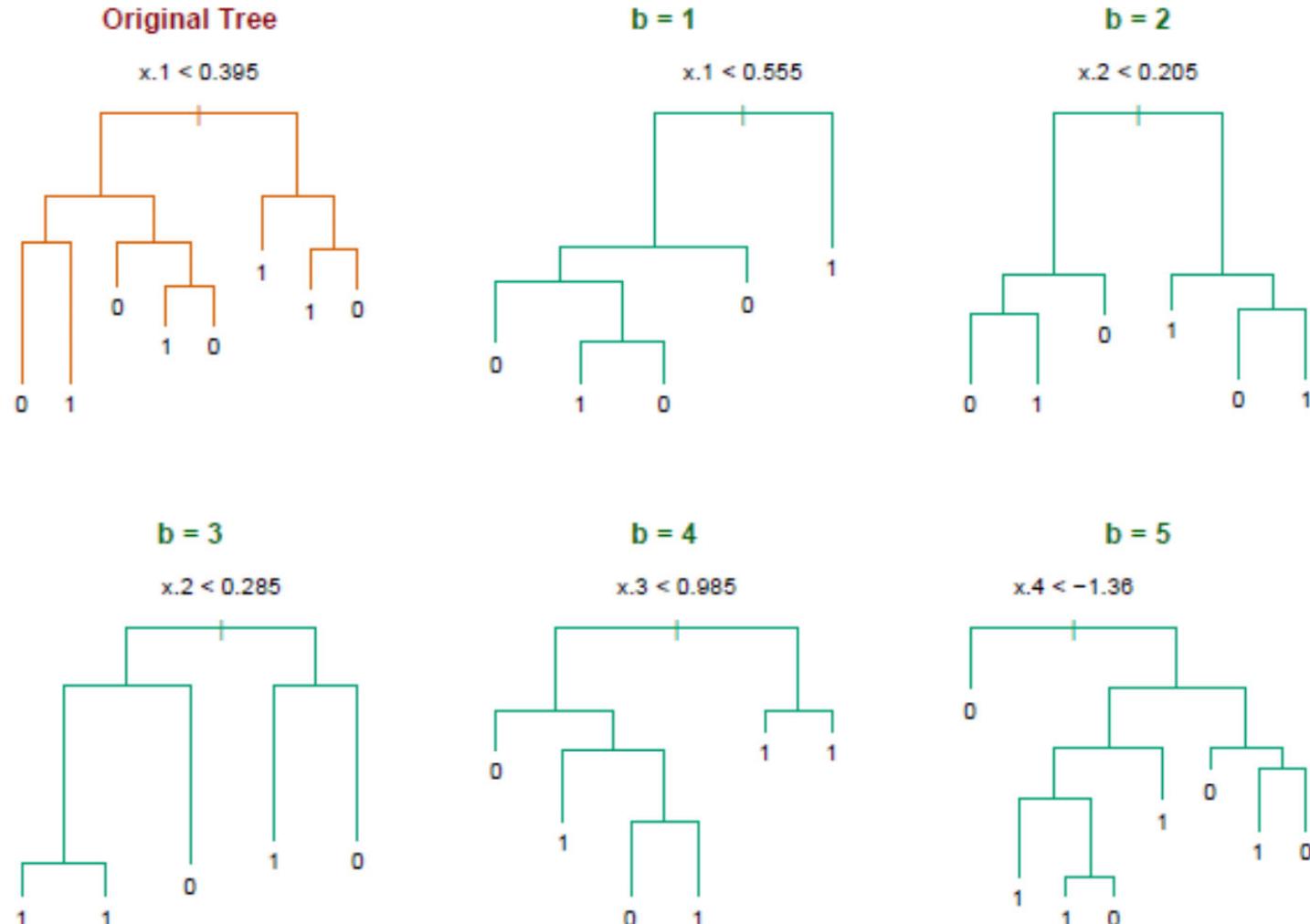
Predicted
Class

Bagging Algorithm

Algorithm 5.6 Bagging Algorithm

- 1: Let k be the number of bootstrap samples.
 - 2: for $i = 1$ to k do
 - 3: Create a bootstrap sample of size n , D_i .
 - 4: Train a base classifier C_i on the bootstrap sample D_i .
 - 5: end for
 - 6: $C^*(x) = \arg \max_y \sum_i \delta(C_i(x) = y)$, $\{\delta(\cdot) = 1 \text{ if its argument is true, and } 0 \text{ otherwise.}\}$
-

Bagging decision trees



Out-of-Bag Error Estimation

- No cross validation?
- Remember, in bootstrapping we sample with replacement, and therefore **not all observations are used for each bootstrap sample**. On average 1/3 of them are not used!
- We call them out-of-bag samples (OOB)
- We can predict the response for the i -th observation using each of the trees in which that observation was OOB and do this for n observations
- Calculate overall OOB MSE or classification error

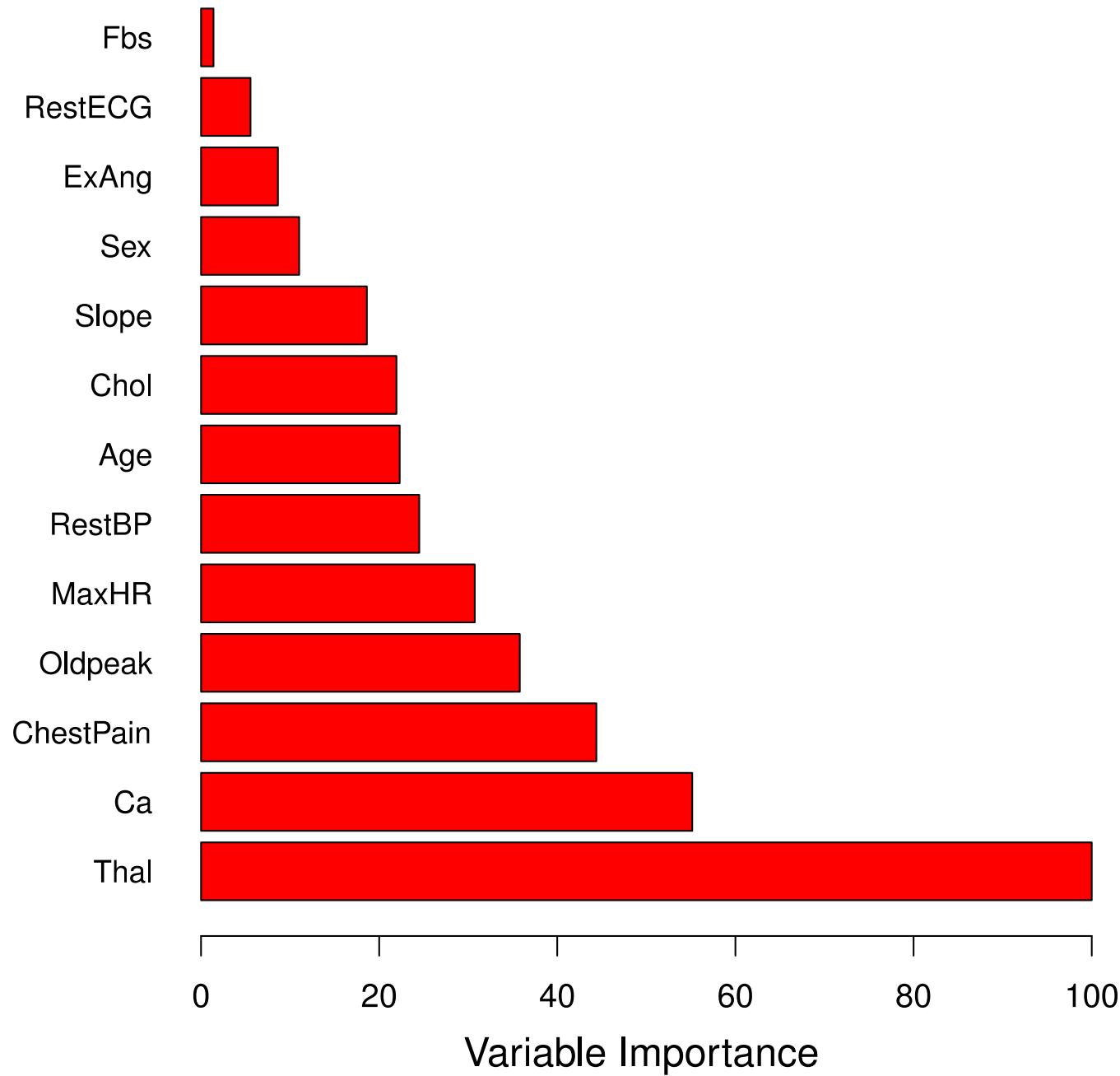
Bagging

- Reduces overfitting (variance)
- Normally uses one type of classifier
- Decision trees are popular
- Easy to parallelize

Variable Importance Measures

- Bagging results in improved accuracy over prediction using a single tree
- Unfortunately, difficult to interpret the resulting model. Bagging improves prediction accuracy at the expense of interpretability.

Calculate the total amount that the RSS or entropy is decreased due to splits over a given predictor, averaged over all B trees.



Bagging - issues

Each tree is identically distributed (i.d.)

→ the expectation of the average of B such trees is the same as the expectation of any one of them

→ the bias of bagged trees is the same as that of the individual trees

i.d. and not i.i.d

Bagging - issues

An average of B i.i.d. random variables, each with variance σ^2 , has variance: σ^2/B

If i.d. (identical but not independent) and pair correlation ρ is present, then the variance is:

$$\rho \sigma^2 + \frac{1 - \rho}{B} \sigma^2$$

As B increases the second term disappears but the first term remains

Why does bagging generate correlated trees?

Bagging - issues

Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors.

Then all bagged trees will select the strong predictor at the top of the tree and therefore all trees will look similar.

How do we avoid this?

Bagging - issues

Remember we want i.i.d such as the bias to be the same and variance to be less?

Other ideas?

What if we consider only a subset of the predictors at each split?

We will still get correlated trees unless
we **randomly** select the subset !

Boosting

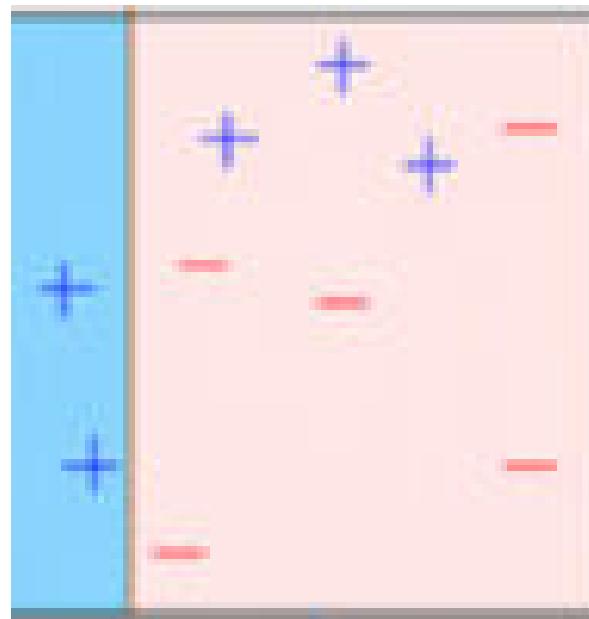
- What if a data point is incorrectly predicted by the first model, and then the next (probably all models), will combining the predictions provide better results? Such situations are taken care of by boosting.
- Boosting is a sequential process, where each subsequent model attempts to correct the errors of the previous model.
- The succeeding models are dependent on the previous model.

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
 - Initially, all N records are assigned equal weights
 - Unlike bagging, weights may change at the end of each boosting round

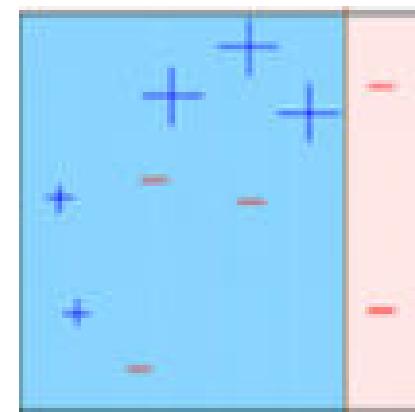
Boosting

- A subset is created from the original dataset.
- Initially, all data points are given equal weights.
- A base model is created on this subset.
- This model is used to make predictions on the whole dataset.



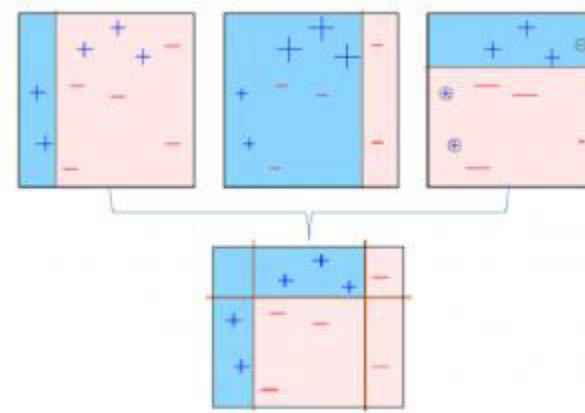
Boosting

- Errors are calculated using the actual values and predicted values.
- The observations which are incorrectly predicted, are given higher weights. (Here, the three misclassified blue-plus points will be given higher weights)
- Another model is created and predictions are made on the dataset. (This model tries to correct the errors from the previous model)



Boosting

- Similarly, multiple models are created, each correcting the errors of the previous model.
- The final model (strong learner) is the weighted mean of all the models (weak learners).



- Individual models would not perform well on the entire dataset, but they work well for some part of the dataset. Thus, each model actually boosts the performance of the ensemble.

Algorithms based on Bagging and Boosting

Bagging algorithms:

- Random forest

Boosting algorithms:

- AdaBoost
- Gradient Boosting

Random Forest

- Random Forest is ensemble machine learning algorithm that follows the bagging technique.
- The base estimators in random forest are decision trees.
- Random forest randomly selects a set of features which are used to decide the best split at each node of the decision tree.

Random Forest

- Random subsets are created from the original dataset (bootstrapping).
- At each node in the decision tree, only a random set of features are considered to decide the best split.
- A decision tree model is fitted on each of the subsets.
- The final prediction is calculated by averaging the predictions from all decision trees.

Random Forests Algorithm

- For $b = 1$ to B :
 - (a) Draw a bootstrap sample Z^* of size N from the training data.
 - (b) Grow a random-forest tree to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
 - i. Select m variables at random from the p variables.
 - ii. Pick the best variable/split-point among the m .
 - iii. Split the node into two daughter nodes.
- Output the ensemble of trees.
- To make a prediction at a new point x we do:
 - For regression: average the results
 - For classification: majority vote

Random Forests Tuning

The inventors make the following recommendations:

- For classification, the default value for m is \sqrt{p} and the minimum node size is one.
- For regression, the default value for m is $p/3$ and the minimum node size is five.

In practice the best values for these parameters will depend on the problem, and they should be treated as tuning parameters.

Like with Bagging, we can use OOB and therefore RF can be fit in one sequence, with cross-validation being performed along the way. Once the OOB error stabilizes, the training can be terminated.

Advantages of Random Forest

- Algorithm can solve both type of problems i.e. classification and regression
- Power to handle large data set with higher dimensionality.
- It can handle thousands of input variables and identify most significant variables so it is considered as one of the dimensionality reduction methods.
- Model outputs **Importance of variable**, which can be a very handy feature (on some random data set).

RF: Variable Importance Measures

Record the prediction accuracy on the oob samples for each tree

Randomly permute the data for column j in the oob samples the record the accuracy again.

The decrease in accuracy as a result of this permuting is averaged over all trees, and is used as a measure of the importance of variable j in the random forest.

Disadvantages of Random Forest

- May over-fit data sets that are particularly noisy.
- Random Forest can feel like a black box approach for statistical modelers – you have very little control on what the model does. You can at best – try different parameters and random seeds!

Random Forests Issues

When the number of variables is large, but the fraction of relevant variables is small, random forests are likely to perform poorly when m is small

Why?

Because:

At each split the chance can be small that the relevant variables will be selected

For example, with 3 relevant and 100 not so relevant variables the probability of any of the relevant variables being selected at any split is ~0.25

Can RF overfit?

Random forests “cannot overfit” the data wrt to number of trees.

Why?

The number of trees, B does not mean increase in the flexibility of the model

AdaBoost

- Adaptive boosting or AdaBoost is one of the simplest boosting algorithms. Usually, decision trees are used for modelling. Multiple sequential models are created, each correcting the errors from the last model.
- AdaBoost assigns weights to the observations which are incorrectly predicted and the subsequent model works to predict these values correctly.

AdaBoost Algorithm

- Initially, all observations (n) in the dataset are given equal weights ($1/n$).
- A model is built on a subset of data.
- Using this model, predictions are made on the whole dataset.
- Errors are calculated by comparing the predictions and actual values.
- While creating the next model, higher weights are given to the data points which were predicted incorrectly.

Adaboost Algorithm

- Weights can be determined using the error value. For instance, higher the error more is the weight assigned to the observation.
- This process is repeated until the error function does not change, or the maximum limit of the number of estimators is reached.

- Base classifiers C_i : C_1, C_2, \dots, C_T
- Error rate:
 - N input samples

$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(x_j) \neq y_j)$$

- Importance of a classifier:

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$

https://en.wikipedia.org/wiki/AdaBoost#Choosing_at

AdaBoost: Weight Update

Weight Update:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases} \quad \text{Eqn:5.88}$$

where Z_j is the normalization factor

$$C^*(x) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(x) = y)$$

- Reduce weight if correctly classified else increase
- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to $1/n$ and the resampling procedure is repeated

AdaBoost Algorithm

Algorithm 5.7 AdaBoost Algorithm

```

1:  $w = \{w_j = 1/n \mid j = 1, 2, \dots, n\}$ . {Initialize the weights for all  $n$  instances.}
2: Let  $k$  be the number of boosting rounds.
3: for  $i = 1$  to  $k$  do
4:   Create training set  $D_i$  by sampling (with replacement) from  $D$  according to  $w$ .
5:   Train a base classifier  $C_i$  on  $D_i$ .
6:   Apply  $C_i$  to all instances in the original training set,  $D$ .
7:    $\epsilon_i = \frac{1}{n} [\sum_j w_j \delta(C_i(x_j) \neq y_j)]$  {Calculate the weighted error}
8:   if  $\epsilon_i > 0.5$  then
9:      $w = \{w_j = 1/n \mid j = 1, 2, \dots, n\}$ . {Reset the weights for all  $n$  instances.}
10:    Go back to Step 4.
11:   end if
12:    $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$ .
13:   Update the weight of each instance according to equation (5.88).
14: end for
15:  $C^*(x) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(x) = y)$ .

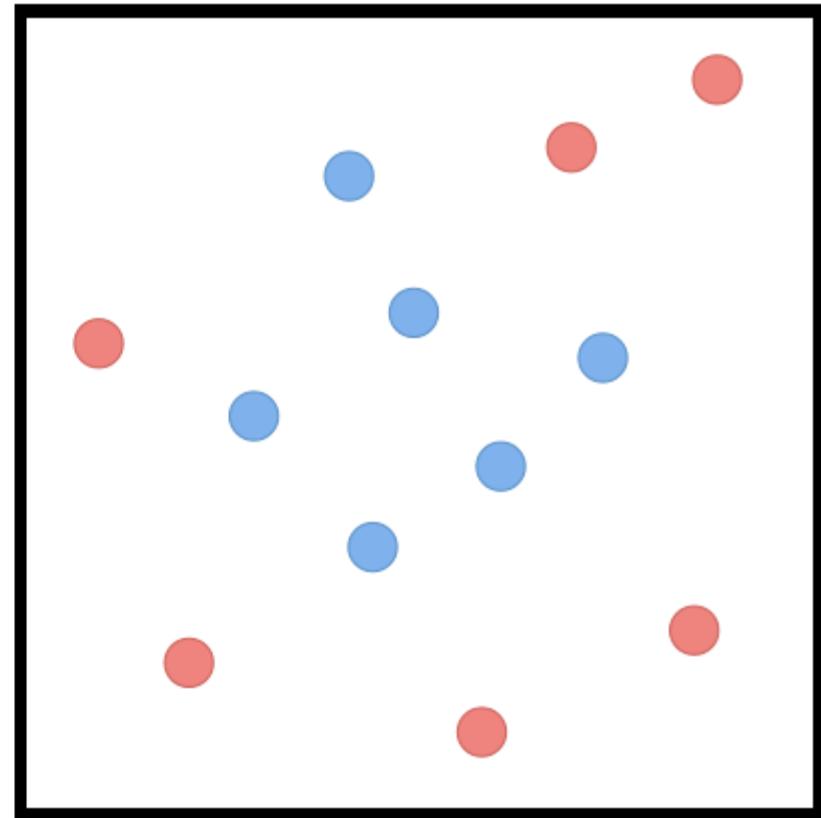
```

AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
  
$$H(\mathbf{x}) = \text{sign} \left( \sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$


```



- Size of point represents the instance's weight

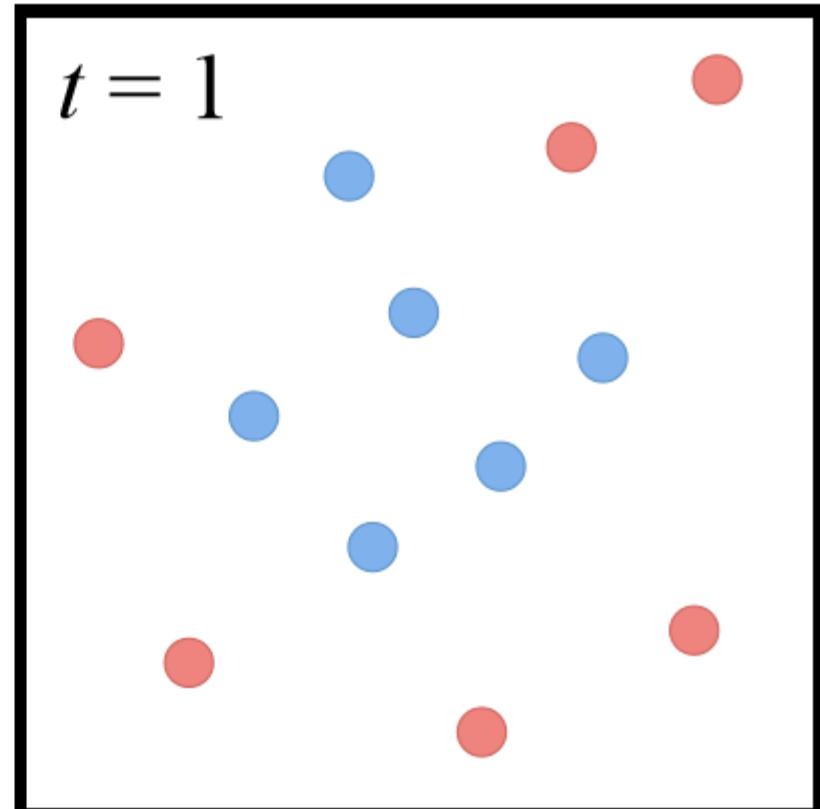
α in earlier slide same as β = weight of class

AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
  
```

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

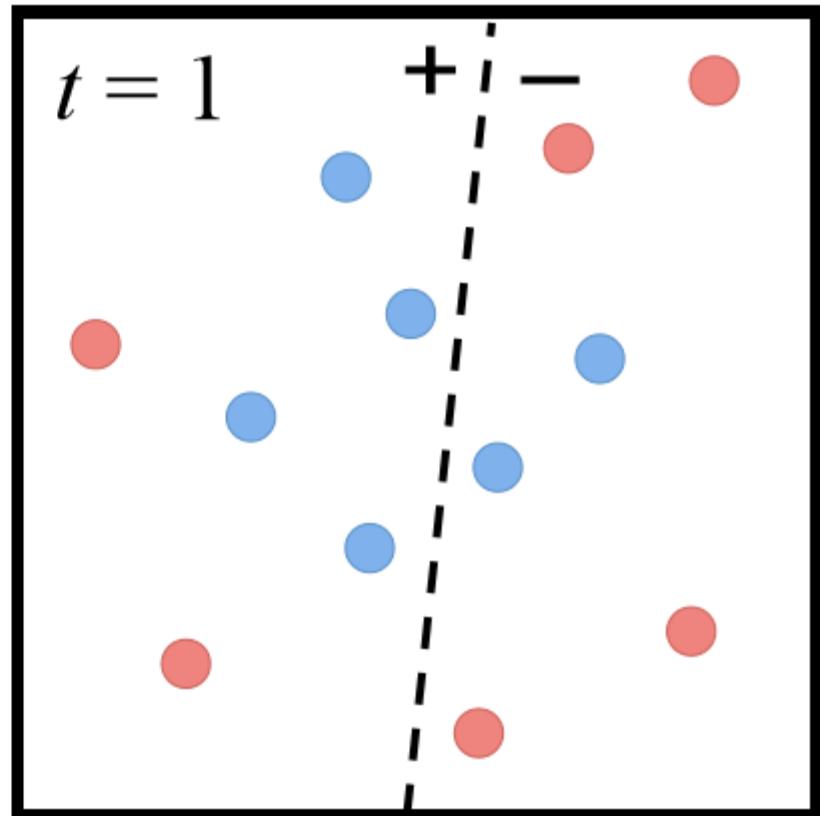


AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
  
```

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

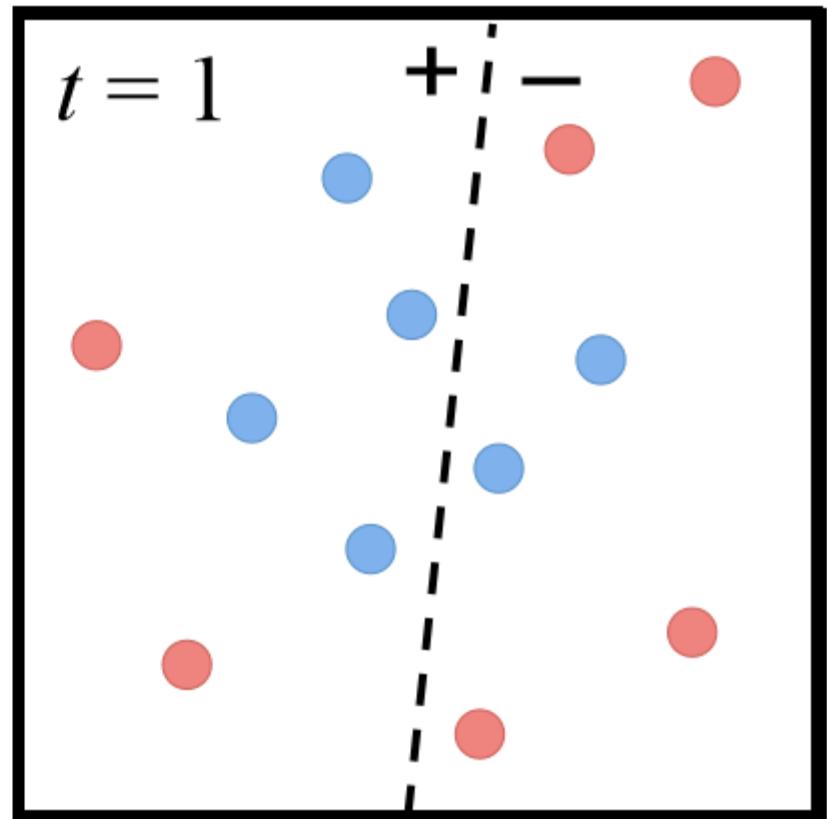


AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
  
```

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



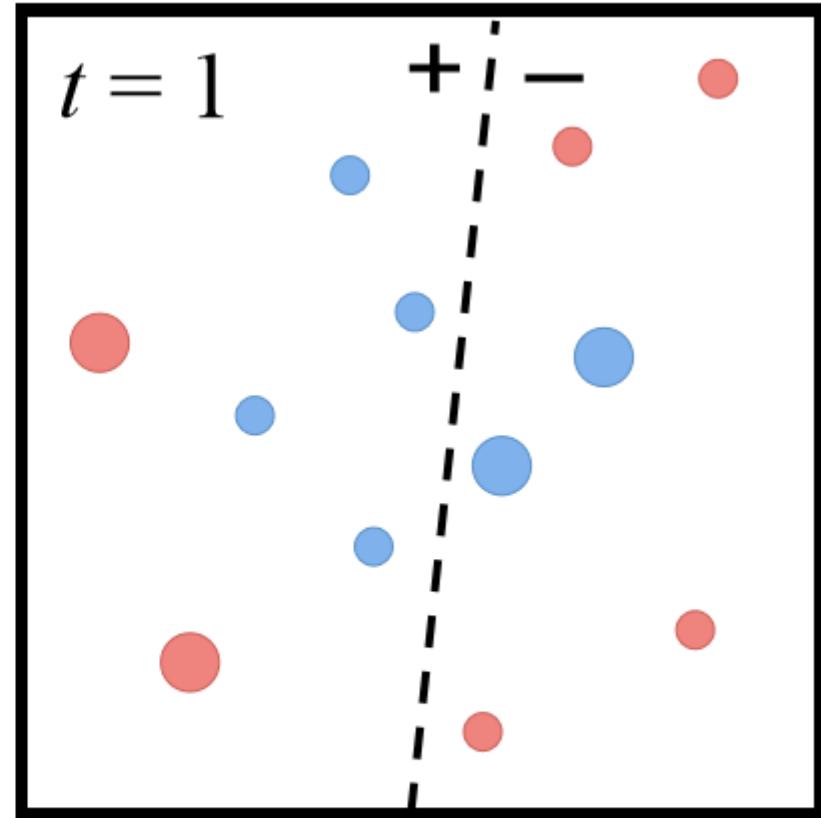
- β_t measures the importance of h_t
- If $\epsilon_t \leq 0.5$, then $\beta_t \geq 0$ (β_t grows as ϵ_t gets smaller)

AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
  
```

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

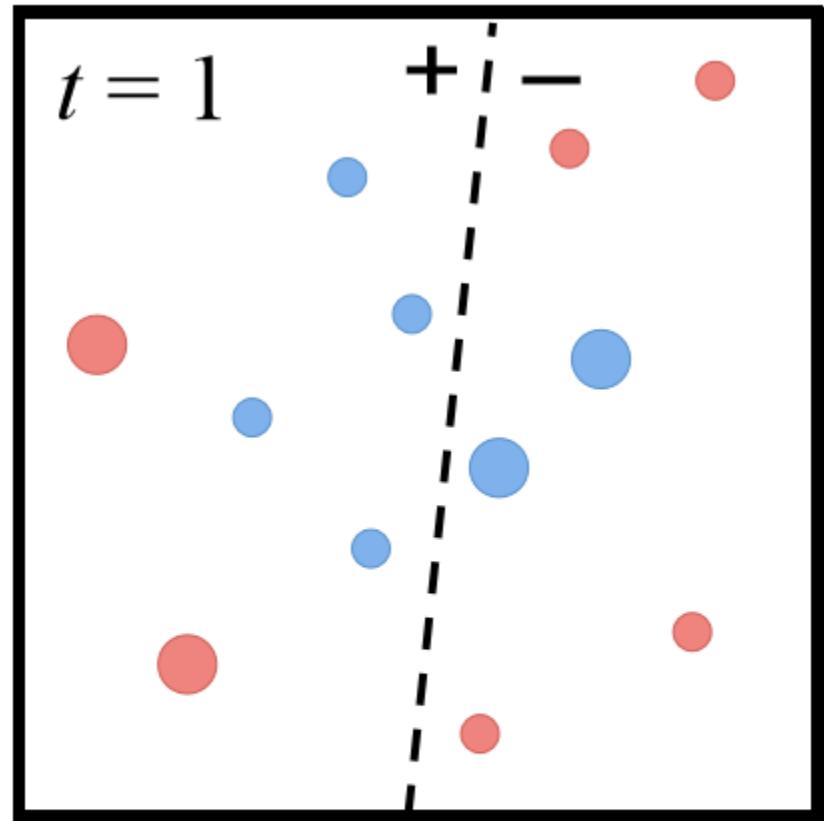


- Weights of correct predictions are multiplied by $e^{-\beta_t} \leq 1$
- Weights of incorrect predictions are multiplied by $e^{\beta_t} \geq 1$

AdaBoost Algorithm

```
1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
```

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



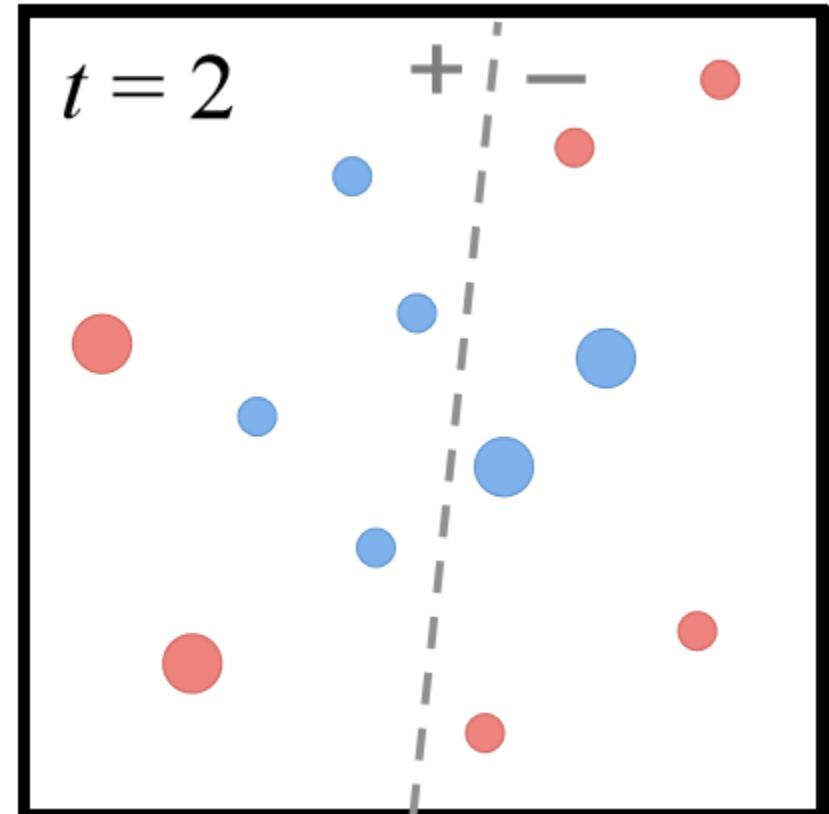
Disclaimer: Note that resized points in the illustration above are not necessarily to scale with β_t

AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
  
```

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

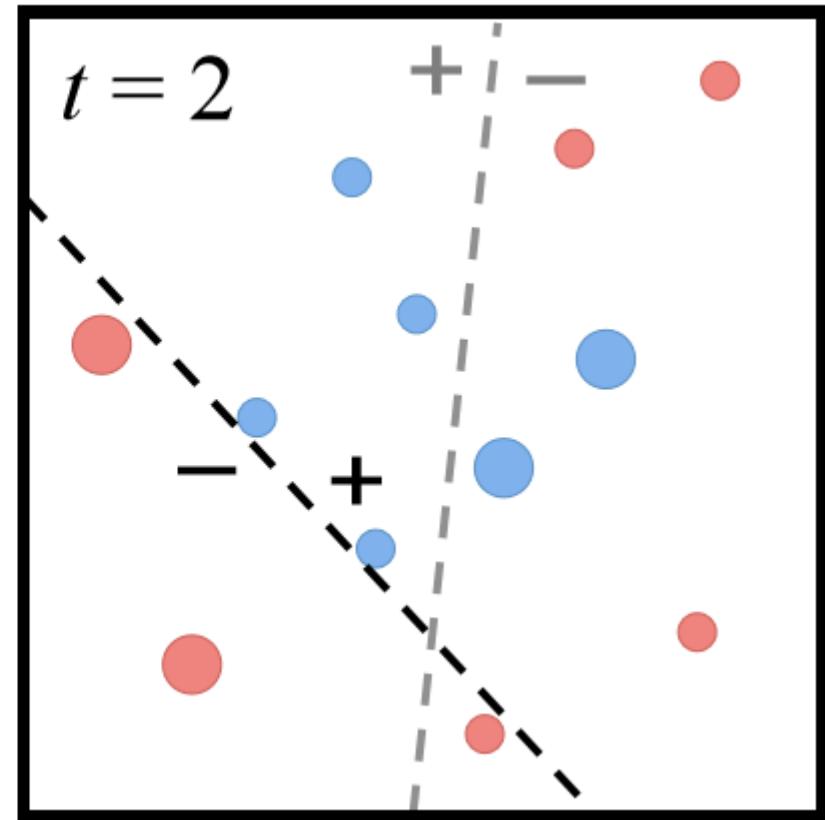


AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
   $H(\mathbf{x}) = \text{sign} \left( \sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$ 

```

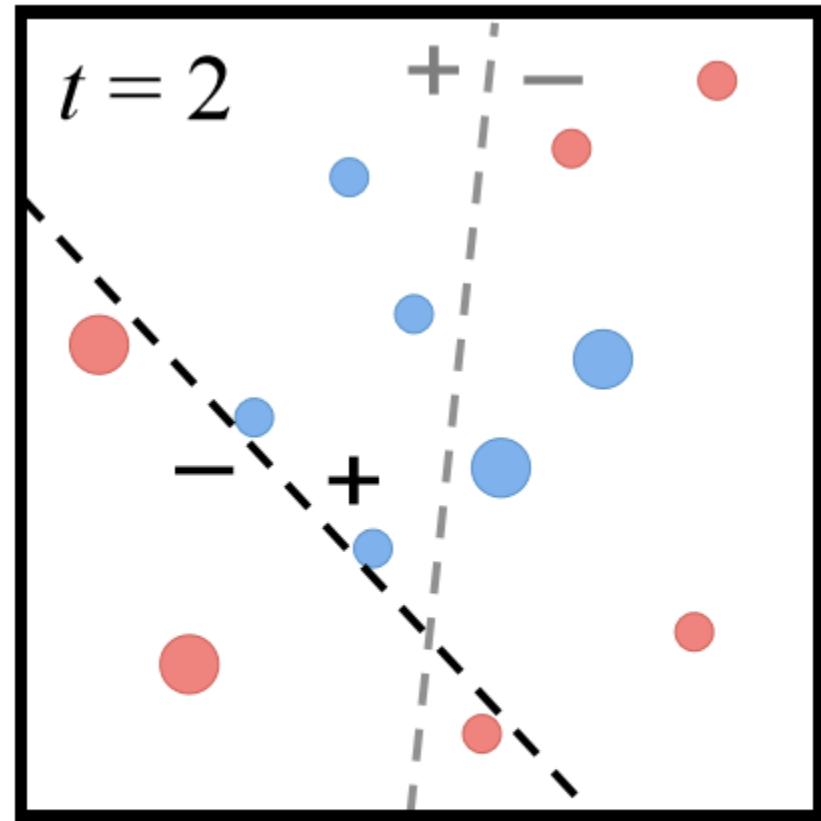


AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
  
```

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



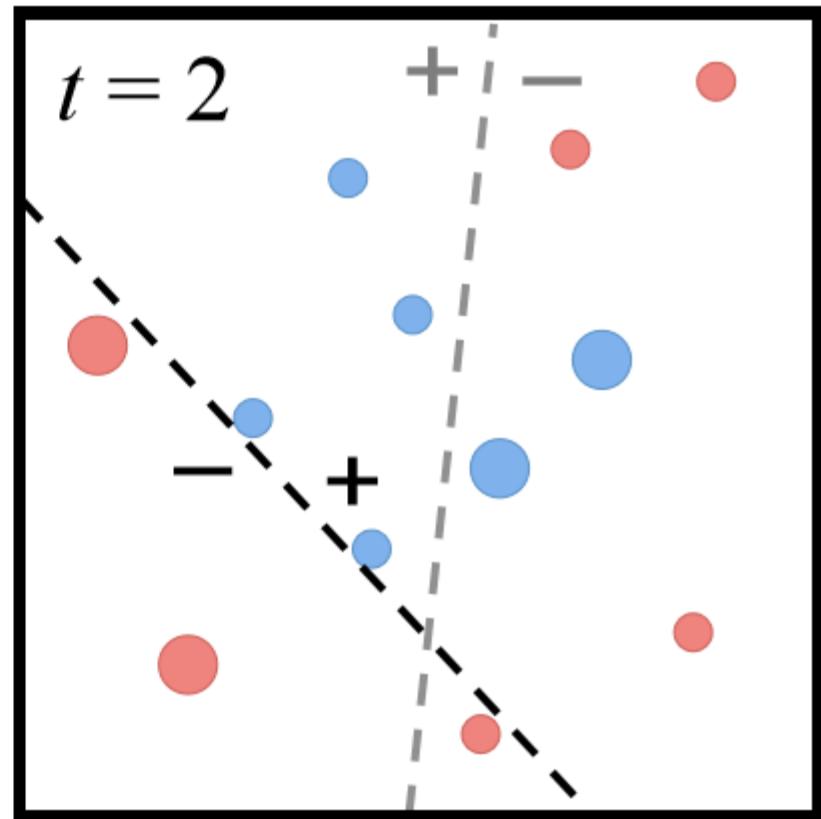
- β_t measures the importance of h_t
- If $\epsilon_t \leq 0.5$, then $\beta_t \geq 0$ (β_t grows as ϵ_t gets smaller)

AdaBoost Algorithm

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

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



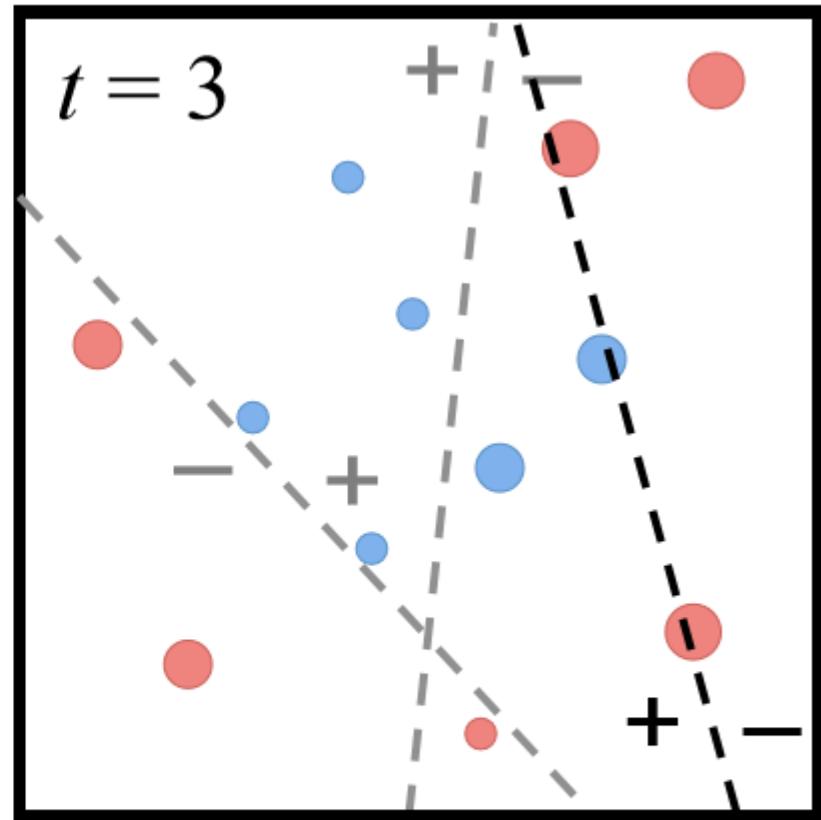
- Weights of correct predictions are multiplied by $e^{-\beta_t} \leq 1$
- Weights of incorrect predictions are multiplied by $e^{\beta_t} \geq 1$

AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
   $H(\mathbf{x}) = \text{sign} \left( \sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$ 

```



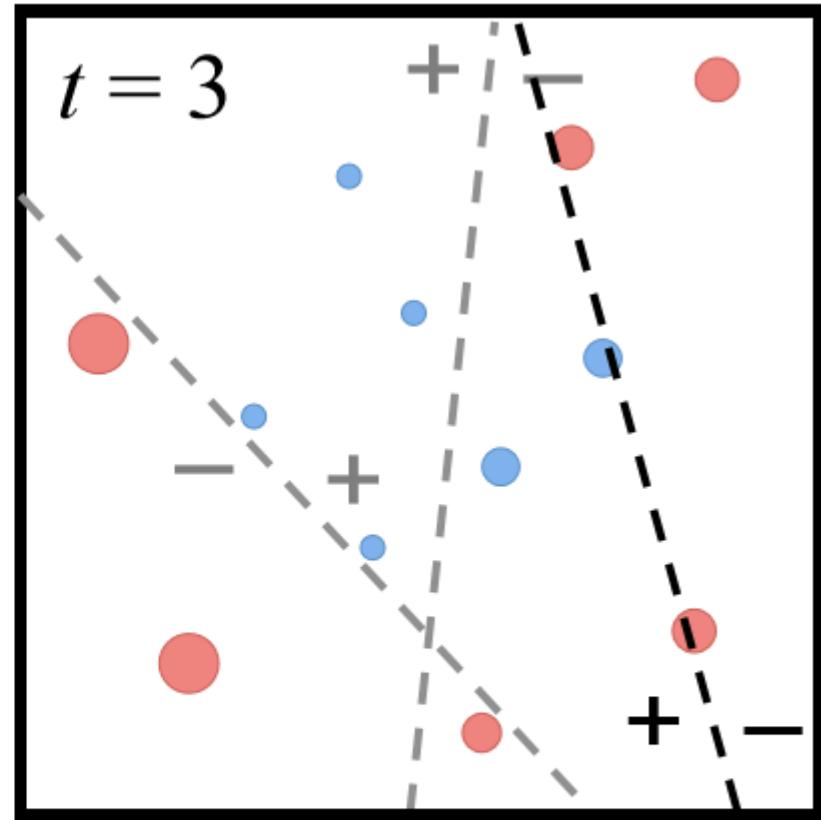
- β_t measures the importance of h_t
- If $\epsilon_t \leq 0.5$, then $\beta_t \geq 0$ (β_t grows as ϵ_t gets smaller)

AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
  
```

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



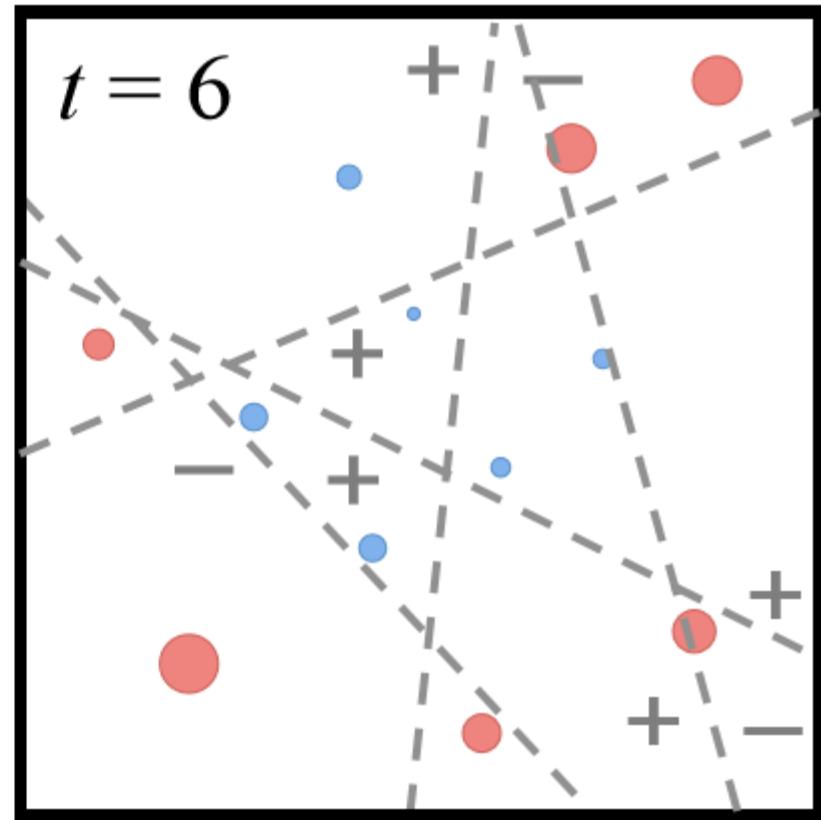
- Weights of correct predictions are multiplied by $e^{-\beta_t} \leq 1$
- Weights of incorrect predictions are multiplied by $e^{\beta_t} \geq 1$

AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
   $H(\mathbf{x}) = \text{sign} \left( \sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$ 

```



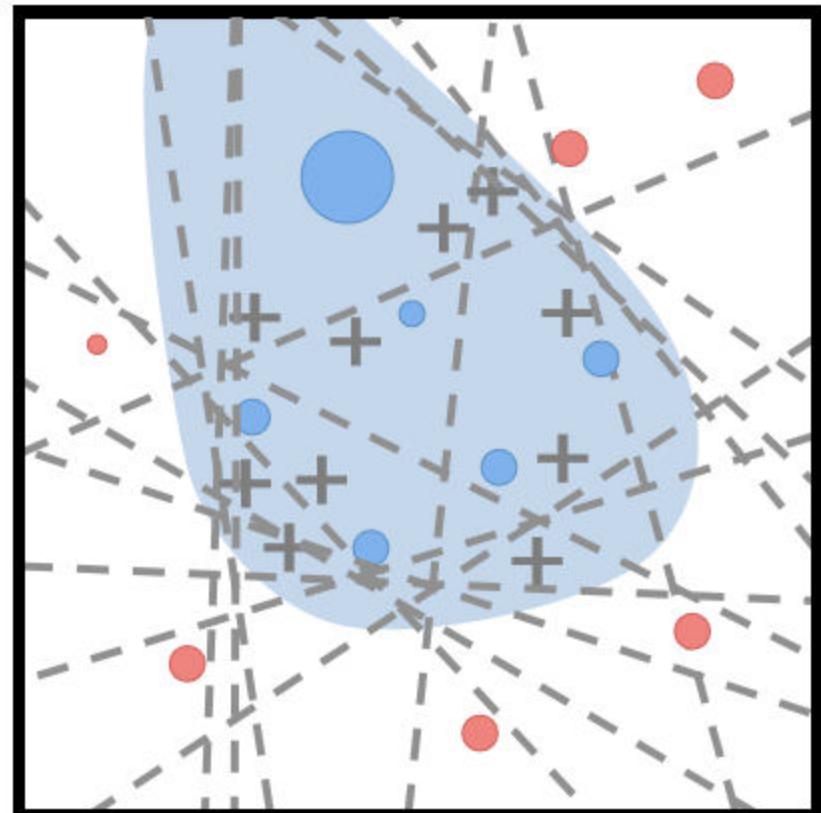
AdaBoost Algorithm

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

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

$t = T$



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

AdaBoost Algorithm

INPUT: training data $X, y = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$,
the number of iterations T

1: Initialize a vector of n uniform weights $\mathbf{w}_1 = [\frac{1}{n}, \dots, \frac{1}{n}]$

2: **for** $t = 1, \dots, T$

3: Train model h_t on X, y with instance weights \mathbf{w}_t

4: Compute the weighted training error rate of h_t :

$$\epsilon_t = \sum_{i:y_i \neq h_t(\mathbf{x}_i)} w_{t,i}$$

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

6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i)) \quad \forall i = 1, \dots, n$$

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

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^n w_{t+1,j}} \quad \forall i = 1, \dots, n$$

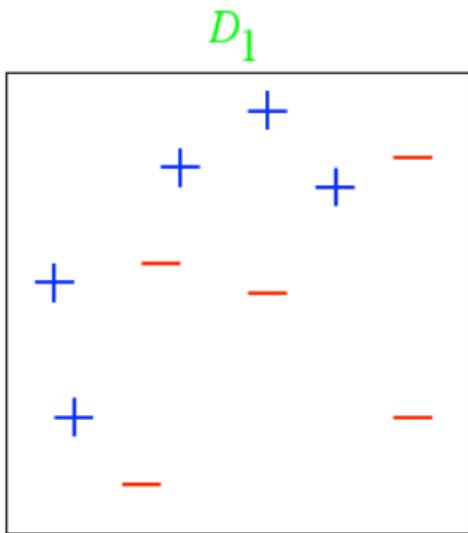
8: **end for**

9: **Return** the hypothesis

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

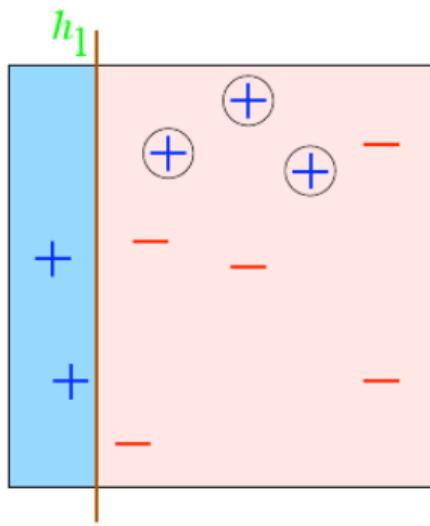
Member classifier with less error are given more weight in final ensemble hypothesis. Final prediction is a weighted combination of each members prediction

Example



From, Léon Bottou

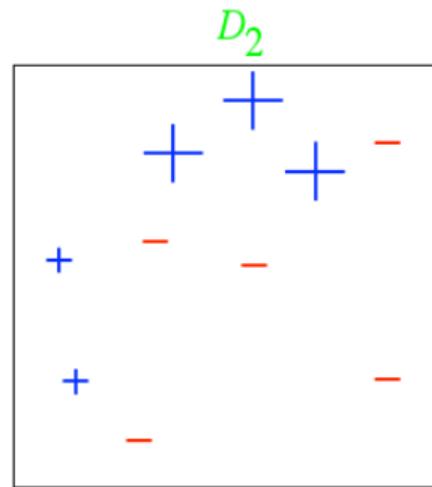
Example



$$\epsilon_1 = 0.30$$

$$\alpha_1 = 0.42$$

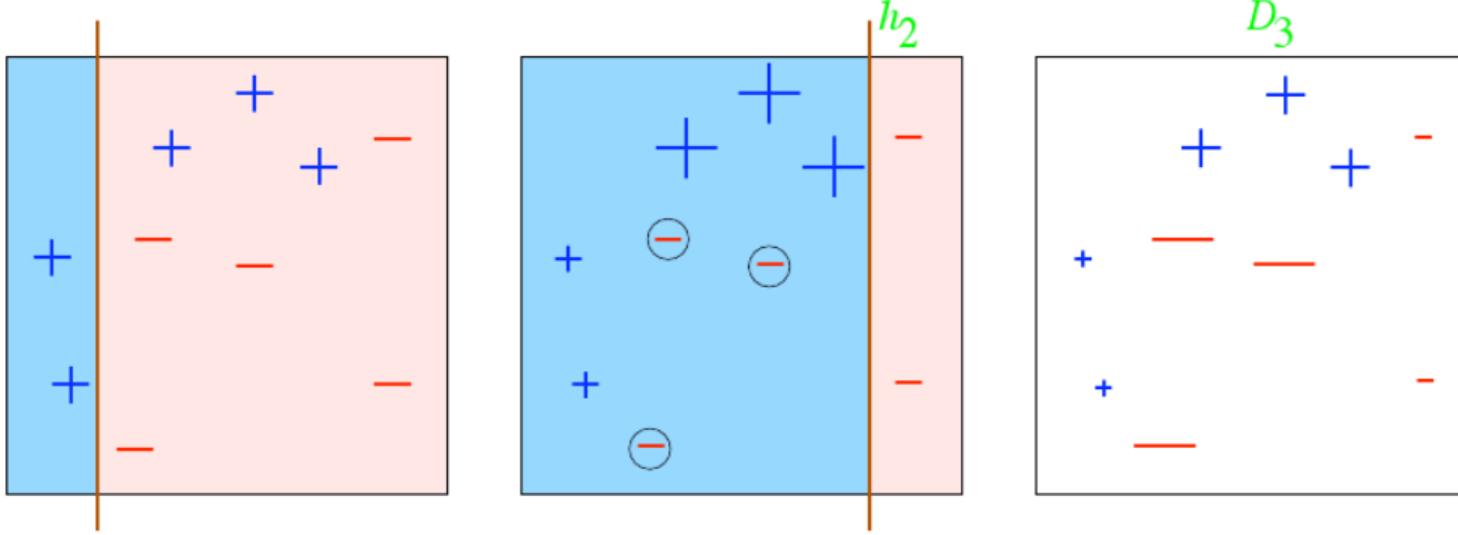
From, Léon Bottou



$$\epsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(x_j) \neq y_j)$$

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \epsilon_i}{\epsilon_i} \right)$$

Example



$$\varepsilon_2 = 0.21$$

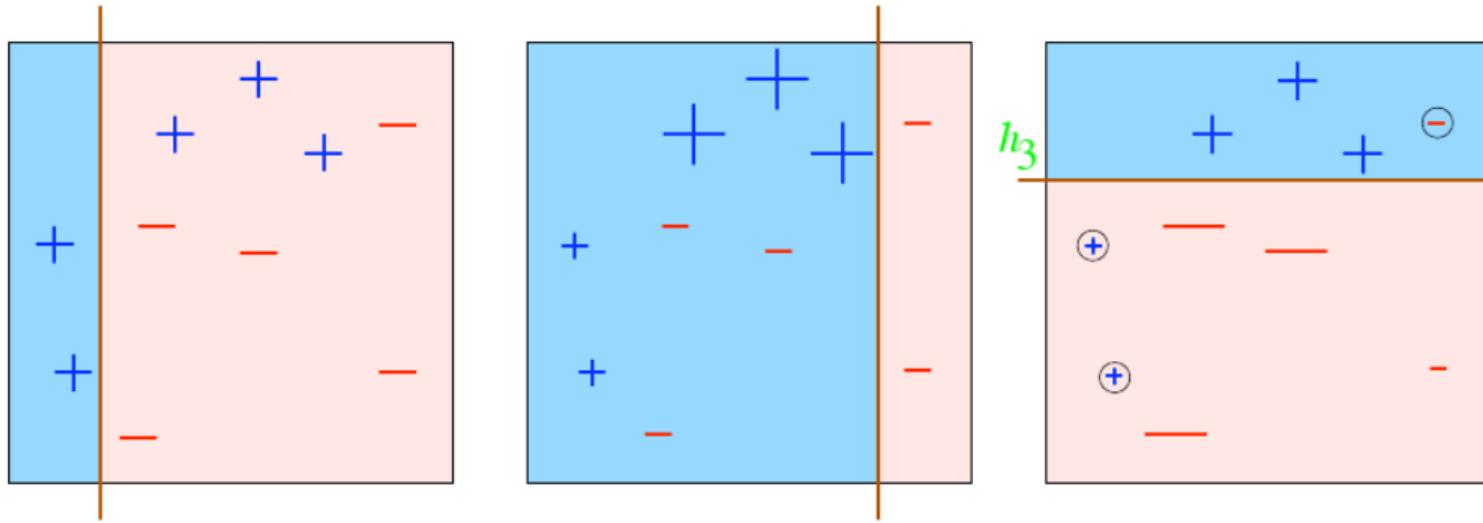
$$\alpha_2 = 0.65$$

$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(x_j) \neq y_j)$$

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$

From, Léon Bottou

Example



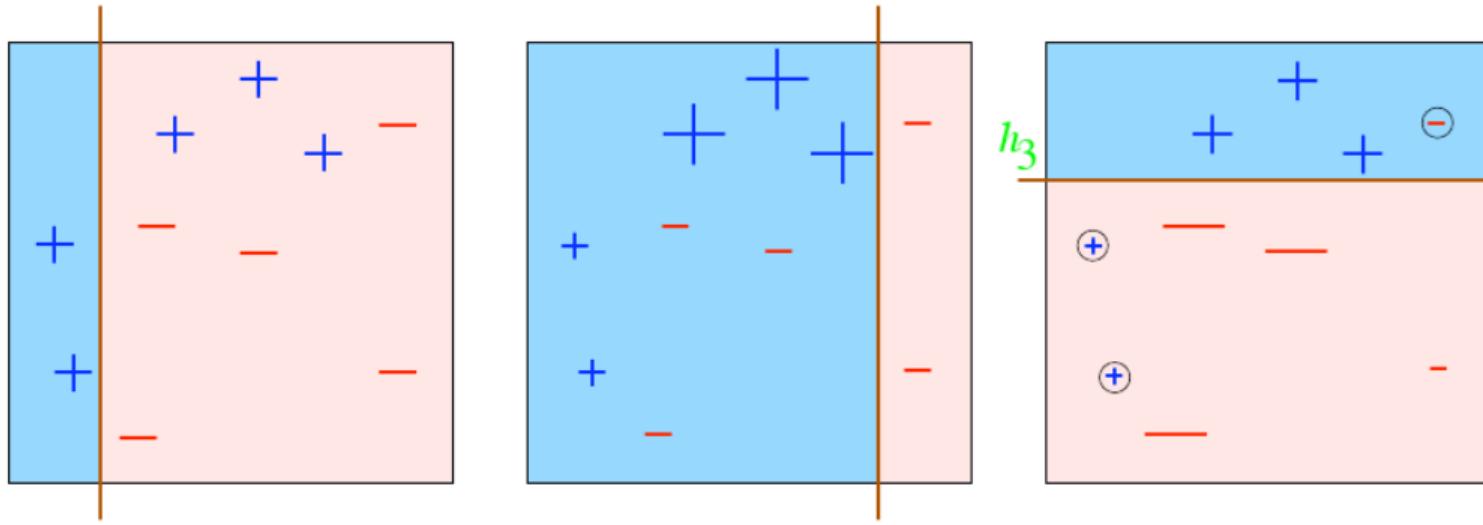
$$\varepsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

From, Léon Bottou

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$

Example



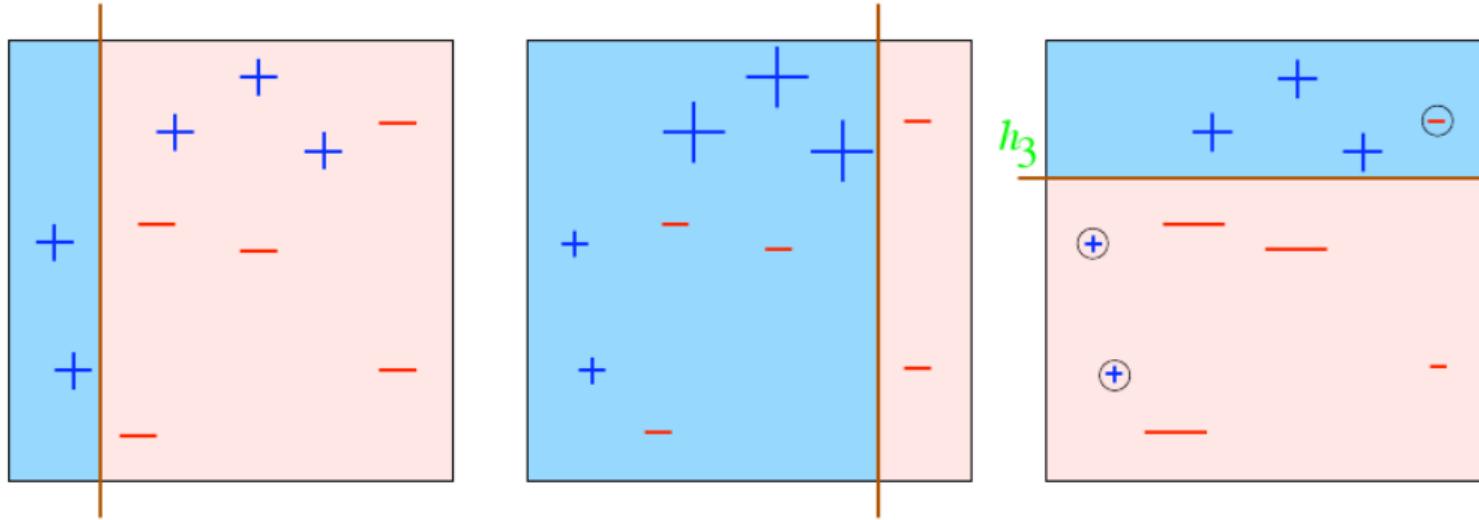
$$\varepsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

From, Léon Bottou

Example

How do we combine the results now?



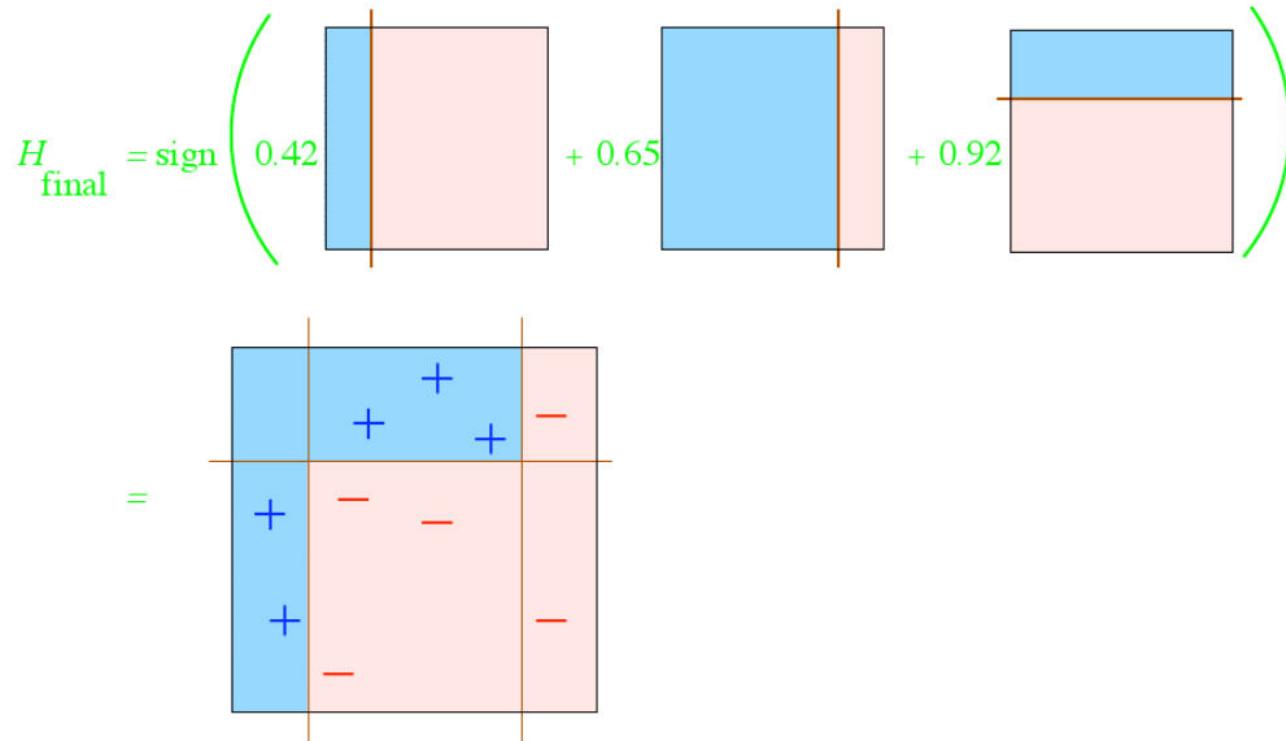
$$\varepsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

From, Léon Bottou

Example

How do we combine the results now?



From, Léon Bottou

AdaBoost Example

- Training sets for the first 3 boosting rounds:

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

- Summary:

Round	Split Point	Left Class	Right Class	alpha
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

AdaBoost Example

- Weights

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

- Classification

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Predicted Class	Sign	1	1	1	-1	-1	-1	1	1	1

AdaBoost error function takes into account the fact that only the sign of the final result is used, thus sum can be far larger than 1 without increasing error

AdaBoost base learners

- AdaBoost works best with “weak” learners
 - Should not be complex
 - Typically high bias classifiers
 - Works even when weak learner has an error rate just slightly under 0.5 (i.e., just slightly better than random)
 - Can prove training error goes to 0 in $O(\log n)$ iterations
- Examples:
 - Decision stumps (1 level decision trees)
 - Depth-limited decision trees
 - Linear classifiers

AdaBoost in practice

Strengths:

- Fast and simple to program
- No parameters to tune (besides T)
- No assumptions on weak learner

When boosting can fail:

- Given insufficient data
- Overly complex weak hypotheses
- Can be susceptible to noise
- When there are a large number of outliers

Fine Tuning Ensembles

- Model combination does not always guaranteed to decrease error, unless
 - base-learners are diverse and accurate
- Ignore poor base learners
 - Use accuracy as a cut-off
 - Introduce some pruning with which at each iteration remove poor learners / learners whose absence lead to improvement (if any)
 - Modify iterations to allow both additions / deletions of learners
 - Discarding appropriately leads to better performance

Gradient Boosting

- In Gradient Boosting, "shortcomings" are identified by gradients.
- Recall that, in Adaboost, “shortcomings” are identified by high-weight data points.
- Both high-weight data points and gradients tell us how to improve our model.

Gradient Boosting

- Gradient Boosting for Different Problems
Difficulty: regression ==> classification
==> ranking

Gradient Boosting

- You are given $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, and the task is to fit a model $F(x)$ to minimize square loss
- There are some mistakes: $F(x_1) = 0.8$, while $y_1 = 0.9$, and $F(x_2) = 1.4$ while $y_2 = 1.3\dots$ How can you improve this model?
- Rules:
 - You are not allowed to remove anything from F or change any parameter in F .
 - You can add an additional model (regression tree)
 h to F , so the new prediction will be $F(x) + h(x)$.
100

Gradient Boosting

- You wish to improve the model such that
 - $F(x_1) + h(x_1) = y_1$
 - $F(x_2) + h(x_2) = y_2 \dots$
 - $F(x_n) + h(x_n) = y_n$

Or, equivalently, you wish

$$h(x_1) = y_1 - F(x_1)$$

$$h(x_2) = y_2 - F(x_2) \dots$$

$$h(x_n) = y_n - F(x_n)$$

Fit a regression tree h to data

$$(x_1, y_1 - F(x_1)), (x_2, y_2 - F(x_2)), \dots, (x_n, y_n - F(x_n))$$

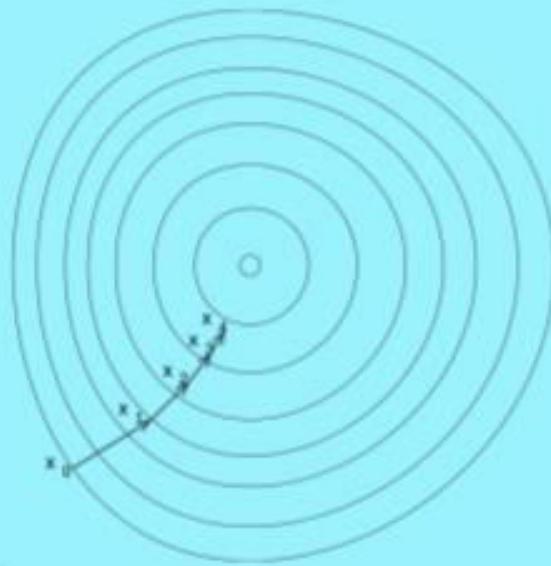
Gradient Boosting

- Simple solution: $y_i - F(x_i)$ are called residuals. These are the parts that existing model F cannot do well.
- The role of h is to compensate the shortcoming of existing model F .
- If the new model $F + h$ is still not satisfactory, we can add another regression tree...
- We are improving the predictions of training data, is the procedure also useful for test data?

Gradient Descent

Minimize a function by moving in the opposite direction of the gradient.

$$\theta_i := \theta_i - \rho \frac{\partial J}{\partial \theta_i}$$



Gradient Boosting for regression

Loss function $L(y, F(x)) = (y - F(x))^2/2$

We want to minimize $J = \sum_i L(y_i, F(x_i))$ by adjusting $F(x_1), F(x_2), \dots, F(x_n)$.

Notice that $F(x_1), F(x_2), \dots, F(x_n)$ are just some numbers. We can treat $F(x_i)$ as parameters and take derivatives

$$\frac{\partial J}{\partial F(x_i)} = \frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} = \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = F(x_i) - y_i$$

So we can interpret residuals as negative gradients.

$$y_i - F(x_i) = -\frac{\partial J}{\partial F(x_i)}$$

$$F(x_i) := F(x_i) + h(x_i)$$

$$F(x_i) := F(x_i) + y_i - F(x_i)$$

$$F(x_i) := F(x_i) - 1 \frac{\partial J}{\partial F(x_i)}$$

$$\theta_i := \theta_i - \rho \frac{\partial J}{\partial \theta_i}$$

For regression with **square loss**,

residual \Leftrightarrow *negative gradient*

fit h to residual \Leftrightarrow *fit h to negative gradient*

update F based on residual \Leftrightarrow *update F based on negative gradient*

So we are actually updating our model using **gradient descent!**

Gradient Boosting Algorithm

- It involves three elements
 - A loss function to be optimized (minimizes expected value)

$$\hat{F} = \arg \min_F \mathbb{E}_{x,y} [L(y, F(x))]$$

- Approximation of $F(x)$ in terms of weighted sum of base(weak) learners $h_i(x)$ to make

$$\hat{F}(x) = \sum_{i=1}^M \gamma_i h_i(x) + \text{const}$$

- An additive model to minimize the loss function, starting with $F_0(x)$ and incrementally expanding in greedy fashion

$$F_0(x) = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma),$$

$$F_m(x) = F_{m-1}(x) + \arg \min_{h_m \in \mathcal{H}} \left[\sum_{i=1}^n L(y_i, F_{m-1}(x_i) + h_m(x_i)) \right]$$

Why XGBoost so popular?

- **Speed** : faster than other ensemble classifiers.
- **Core algorithm is parallelizable**: harness the power of multi-core computers and networks of computers enabling to train on very large datasets **Consistently outperforms other algorithm methods** : It has shown better performance on a variety of machine learning benchmark datasets.
- **Wide variety of tuning parameters** : cross-validation, regularization, missing values, tree parameters, etc
- XGBoost (Extreme Gradient Boosting) uses the gradient boosting (GBM) framework at its core.

References

The-Morgan-Kaufmann-Series-in-Data-Management-
Systems-Jiawei-Han-Micheline-Kamber-Jian-Pei-Data-
Mining.-Concepts-and-Techniques-3rd-Edition-Morgan-
Kaufmann-2011

Bishop - Pattern Recognition And Machine Learning -
Springer 2006

A Gentle Introduction to Gradient Boosting
Cheng Li chengli@ccs.neu.edu College of Computer and
Information Science Northeastern University

https://www.youtube.com/watch?time_continue=647&v=LsK-xG1cLYA&feature=emb_logo



Thank You!



BITS Pilani
Pilani Campus

Ensemble Learning

Dr. Sugata Ghosal

sugata.ghosal@pilani.bits-pilani.ac.in

Contents

- Bagging - Issues
- Boosting
- Random Forest Algorithm
- AdaBoost Algorithm
- Gradient Boosting

Bagging - issues

Each tree is identically distributed (i.d.)

→ the expectation of the average of B such trees is the same as the expectation of any one of them

→ the bias of bagged trees is the same as that of the individual trees

i.d. and not i.i.d

Bagging - issues

An average of B i.i.d. random variables, each with variance σ^2 , has variance: σ^2/B

If i.d. (identical but not independent) and pair correlation ρ is present, then the variance is:

$$\rho \sigma^2 + \frac{1 - \rho}{B} \sigma^2$$

As B increases the second term disappears but the first term remains

Why does bagging generate correlated trees?

Bagging - issues

Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors.

Then all bagged trees will select the strong predictor at the top of the tree and therefore all trees will look similar.

How do we avoid this?

Bagging - issues

Remember we want i.i.d such as the bias to be the same and variance to be less?

Other ideas?

What if we consider only a subset of the predictors at each split?

We will still get correlated trees unless
we **randomly** select the subset !

Boosting

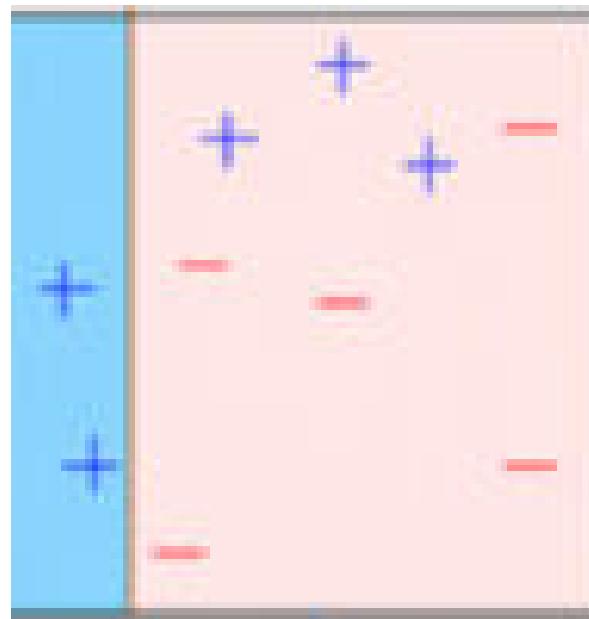
- What if a data point is incorrectly predicted by the first model, and then the next (probably all models), will combining the predictions provide better results? Such situations are taken care of by boosting.
- Boosting is a sequential process, where each subsequent model attempts to correct the errors of the previous model.
- The succeeding models are dependent on the previous model.

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
 - Initially, all N records are assigned equal weights
 - Unlike bagging, weights may change at the end of each boosting round

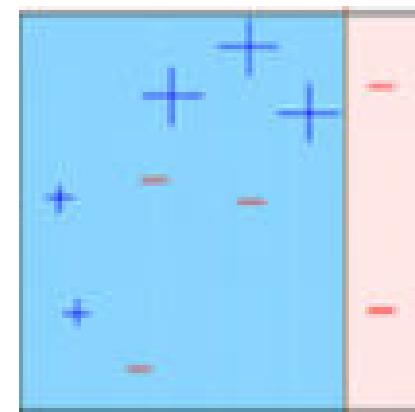
Boosting

- A subset is created from the original dataset.
- Initially, all data points are given equal weights.
- A base model is created on this subset.
- This model is used to make predictions on the whole dataset.



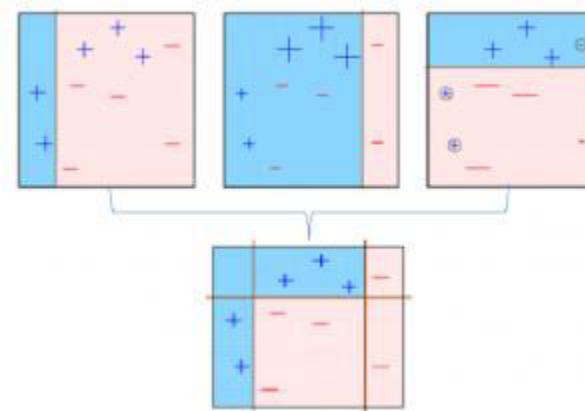
Boosting

- Errors are calculated using the actual values and predicted values.
- The observations which are incorrectly predicted, are given higher weights. (Here, the three misclassified blue-plus points will be given higher weights)
- Another model is created and predictions are made on the dataset. (This model tries to correct the errors from the previous model)



Boosting

- Similarly, multiple models are created, each correcting the errors of the previous model.
- The final model (strong learner) is the weighted mean of all the models (weak learners).



- Individual models would not perform well on the entire dataset, but they work well for some part of the dataset. Thus, each model actually boosts the performance of the ensemble.

Algorithms based on Bagging and Boosting

Bagging algorithms:

- Random forest

Boosting algorithms:

- AdaBoost
- Gradient Boosting

Random Forest

- Random Forest is ensemble machine learning algorithm that follows the bagging technique.
- The base estimators in random forest are decision trees.
- Random forest randomly selects a set of features which are used to decide the best split at each node of the decision tree.

Random Forest

- Random subsets are created from the original dataset (bootstrapping).
- At each node in the decision tree, only a random set of features are considered to decide the best split.
- A decision tree model is fitted on each of the subsets.
- The final prediction is calculated by averaging the predictions from all decision trees.

Random Forests Algorithm

- For $b = 1$ to B :
 - (a) Draw a bootstrap sample Z^* of size N from the training data.
 - (b) Grow a random-forest tree to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
 - i. Select m variables at random from the p variables.
 - ii. Pick the best variable/split-point among the m .
 - iii. Split the node into two daughter nodes.
- Output the ensemble of trees.
- To make a prediction at a new point x we do:
 - For regression: average the results
 - For classification: majority vote

Random Forests Tuning

The inventors make the following recommendations:

- For classification, the default value for m is \sqrt{p} and the minimum node size is one.
- For regression, the default value for m is $p/3$ and the minimum node size is five.

In practice the best values for these parameters will depend on the problem, and they should be treated as tuning parameters.

Like with Bagging, we can use OOB and therefore RF can be fit in one sequence, with cross-validation being performed along the way. Once the OOB error stabilizes, the training can be terminated.

Advantages of Random Forest

- Algorithm can solve both type of problems i.e. classification and regression
- Power to handle large data set with higher dimensionality.
- It can handle thousands of input variables and identify most significant variables so it is considered as one of the dimensionality reduction methods.
- Model outputs **Importance of variable**, which can be a very handy feature (on some random data set).

RF: Variable Importance Measures

Record the prediction accuracy on the oob samples for each tree

Randomly permute the data for column j in the oob samples the record the accuracy again.

The decrease in accuracy as a result of this permuting is averaged over all trees, and is used as a measure of the importance of variable j in the random forest.

Disadvantages of Random Forest

- May over-fit data sets that are particularly noisy.
- Random Forest can feel like a black box approach for statistical modelers – you have very little control on what the model does. You can at best – try different parameters and random seeds!

Random Forests Issues

When the number of variables is large, but the fraction of relevant variables is small, random forests are likely to perform poorly when m is small

Why?

Because:

At each split the chance can be small that the relevant variables will be selected

For example, with 3 relevant and 100 not so relevant variables the probability of any of the relevant variables being selected at any split is ~0.25

Can RF overfit?

Random forests “cannot overfit” the data wrt to number of trees.

Why?

The number of trees, B does not mean increase in the flexibility of the model

AdaBoost

- Adaptive boosting or AdaBoost is one of the simplest boosting algorithms. Usually, decision trees are used for modelling. Multiple sequential models are created, each correcting the errors from the last model.
- AdaBoost assigns weights to the observations which are incorrectly predicted and the subsequent model works to predict these values correctly.

AdaBoost Algorithm

- Initially, all observations (n) in the dataset are given equal weights ($1/n$).
- A model is built on a subset of data.
- Using this model, predictions are made on the whole dataset.
- Errors are calculated by comparing the predictions and actual values.
- While creating the next model, higher weights are given to the data points which were predicted incorrectly.

Adaboost Algorithm

- Weights can be determined using the error value. For instance, higher the error more is the weight assigned to the observation.
- This process is repeated until the error function does not change, or the maximum limit of the number of estimators is reached.

- Base classifiers C_i : C_1, C_2, \dots, C_T
- Error rate:
 - N input samples

$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(x_j) \neq y_j)$$

- Importance of a classifier:

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$

https://en.wikipedia.org/wiki/AdaBoost#Choosing_at

AdaBoost: Weight Update

Weight Update:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases} \quad \text{Eqn:5.88}$$

where Z_j is the normalization factor

$$C^*(x) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(x) = y)$$

- Reduce weight if correctly classified else increase
- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to $1/n$ and the resampling procedure is repeated

AdaBoost Algorithm

Algorithm 5.7 AdaBoost Algorithm

```

1:  $w = \{w_j = 1/n \mid j = 1, 2, \dots, n\}$ . {Initialize the weights for all  $n$  instances.}
2: Let  $k$  be the number of boosting rounds.
3: for  $i = 1$  to  $k$  do
4:   Create training set  $D_i$  by sampling (with replacement) from  $D$  according to  $w$ .
5:   Train a base classifier  $C_i$  on  $D_i$ .
6:   Apply  $C_i$  to all instances in the original training set,  $D$ .
7:    $\epsilon_i = \frac{1}{n} [\sum_j w_j \delta(C_i(x_j) \neq y_j)]$  {Calculate the weighted error}
8:   if  $\epsilon_i > 0.5$  then
9:      $w = \{w_j = 1/n \mid j = 1, 2, \dots, n\}$ . {Reset the weights for all  $n$  instances.}
10:    Go back to Step 4.
11:   end if
12:    $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$ .
13:   Update the weight of each instance according to equation (5.88).
14: end for
15:  $C^*(x) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(x) = y)$ .

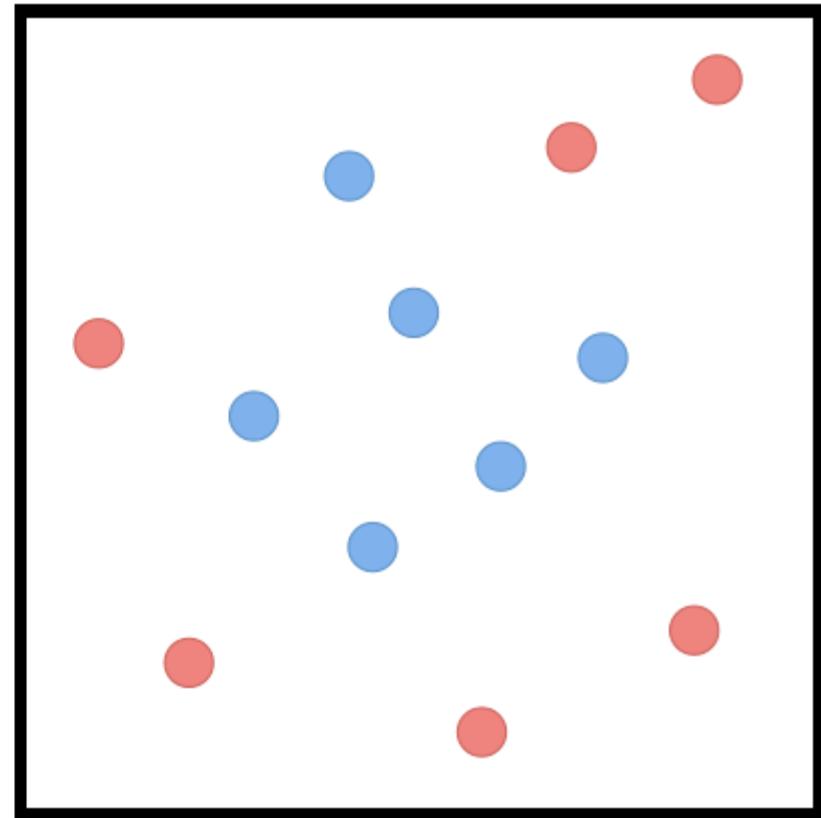
```

AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
  
$$H(\mathbf{x}) = \text{sign} \left( \sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$


```



- Size of point represents the instance's weight

α in earlier slide same as β = weight of class

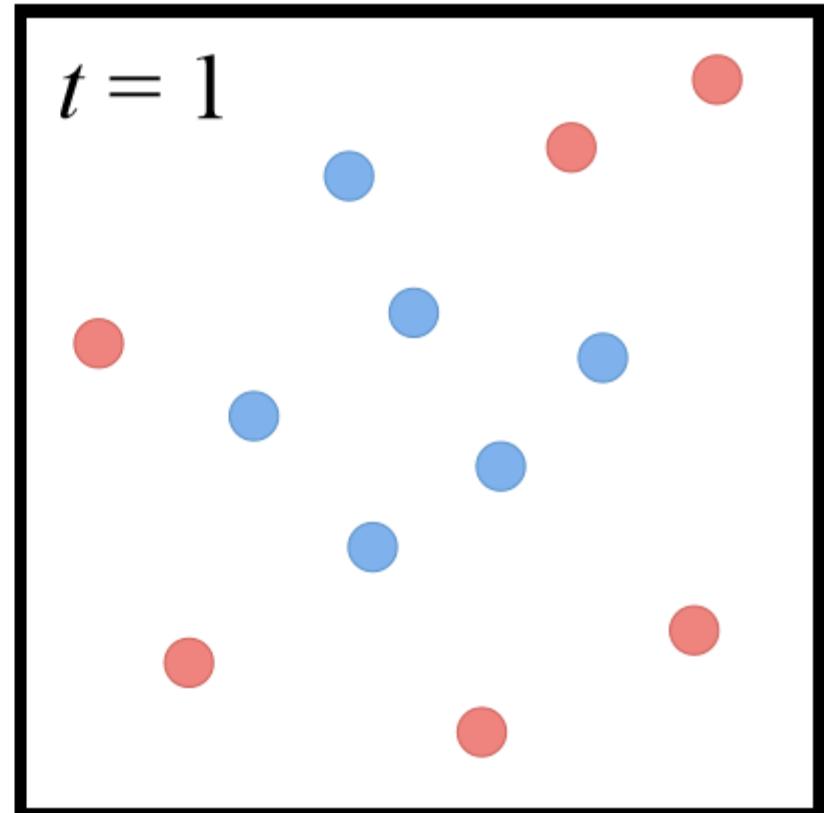
AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis

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


```

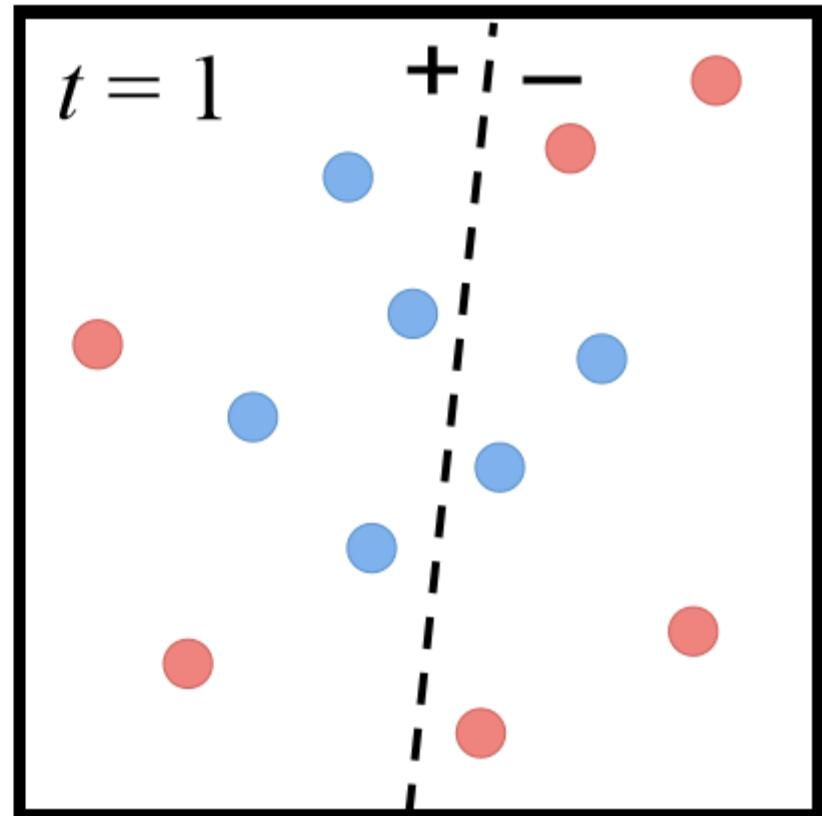


AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
  
```

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

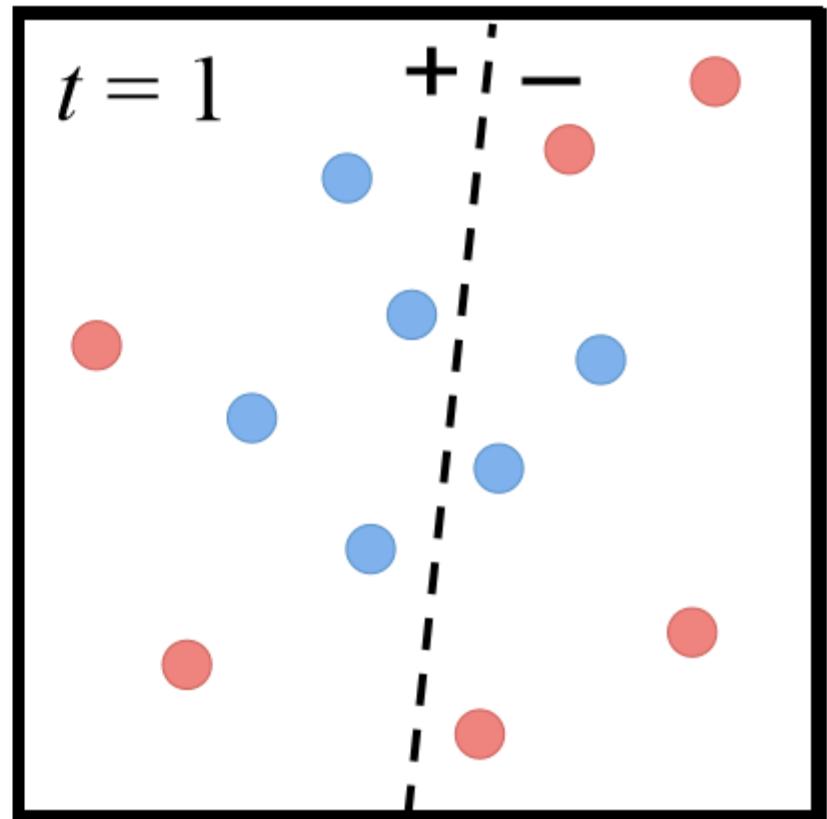


AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
  
```

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



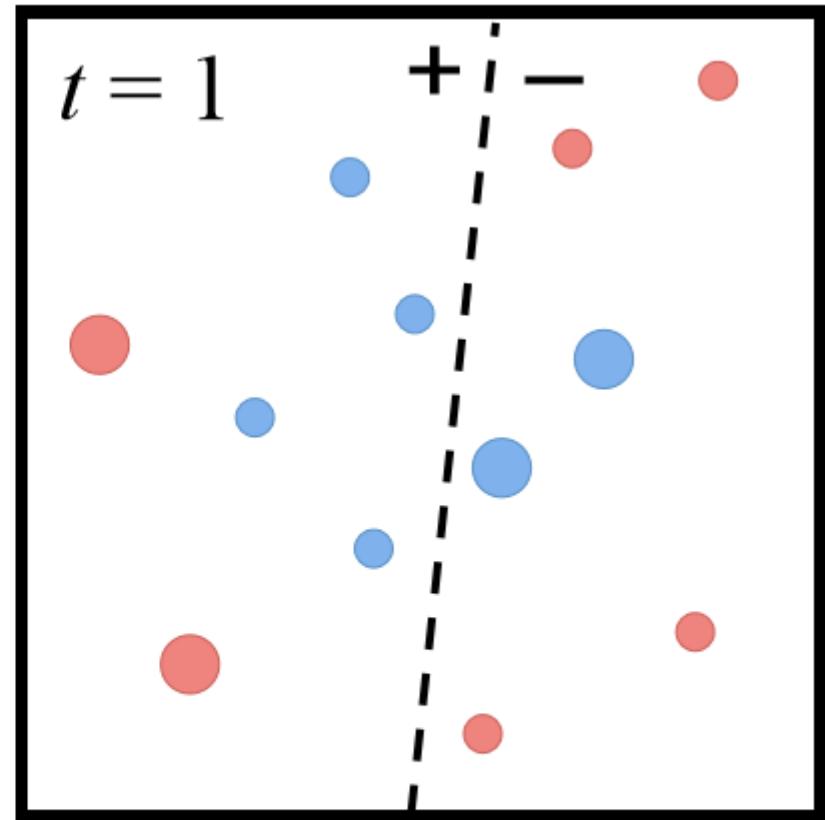
- β_t measures the importance of h_t
- If $\epsilon_t \leq 0.5$, then $\beta_t \geq 0$ (β_t grows as ϵ_t gets smaller)

AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
  
```

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

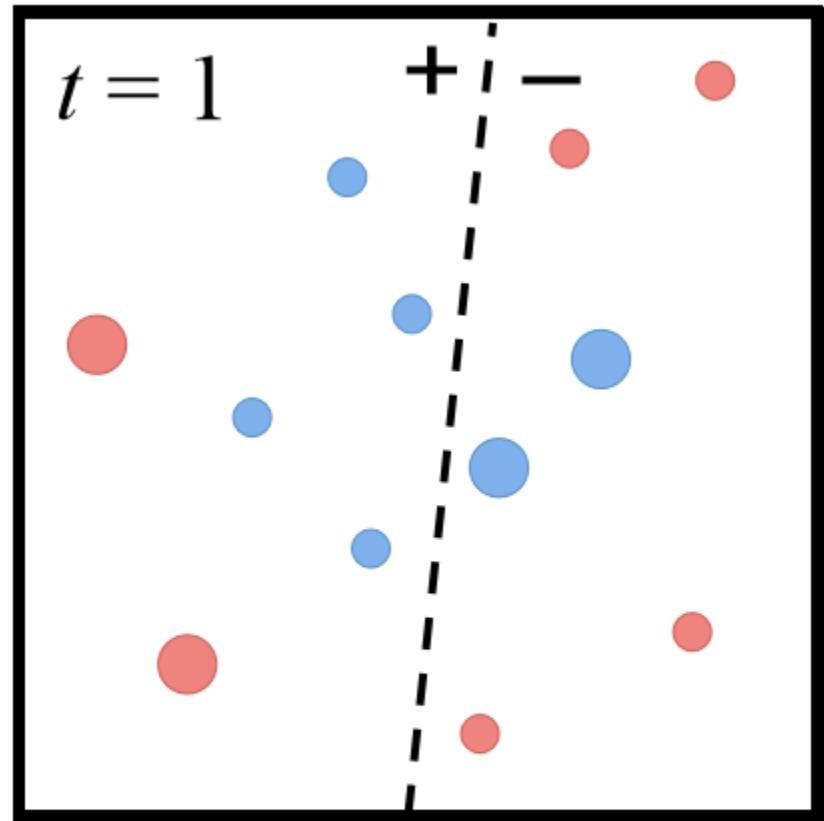


- Weights of correct predictions are multiplied by $e^{-\beta_t} \leq 1$
- Weights of incorrect predictions are multiplied by $e^{\beta_t} \geq 1$

AdaBoost Algorithm

```
1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
```

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



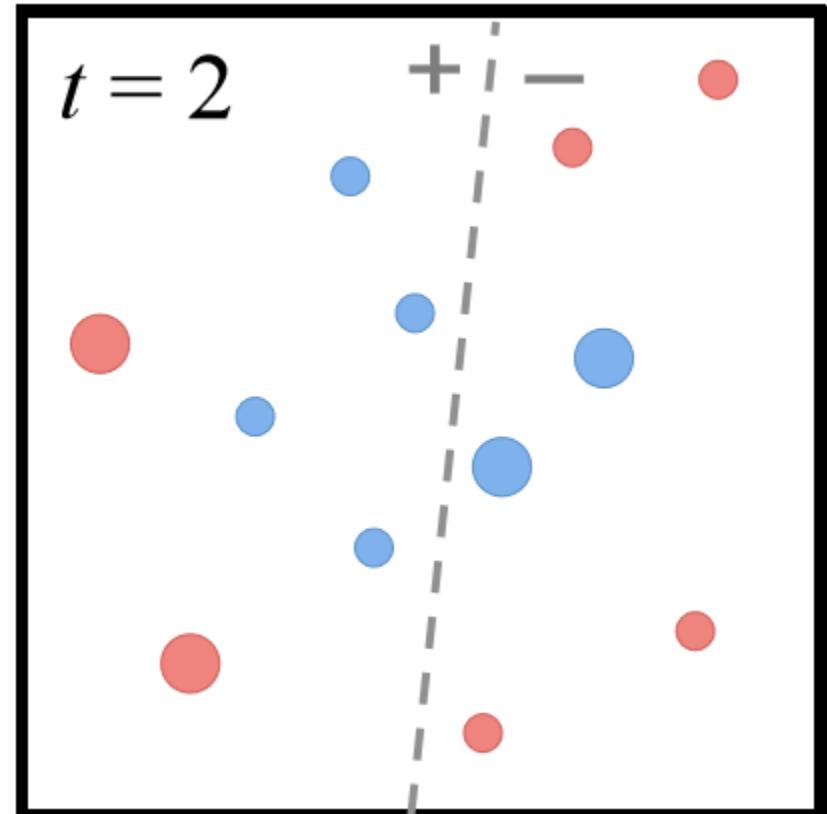
Disclaimer: Note that resized points in the illustration above
are not necessarily to scale with β_t

AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
  
```

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

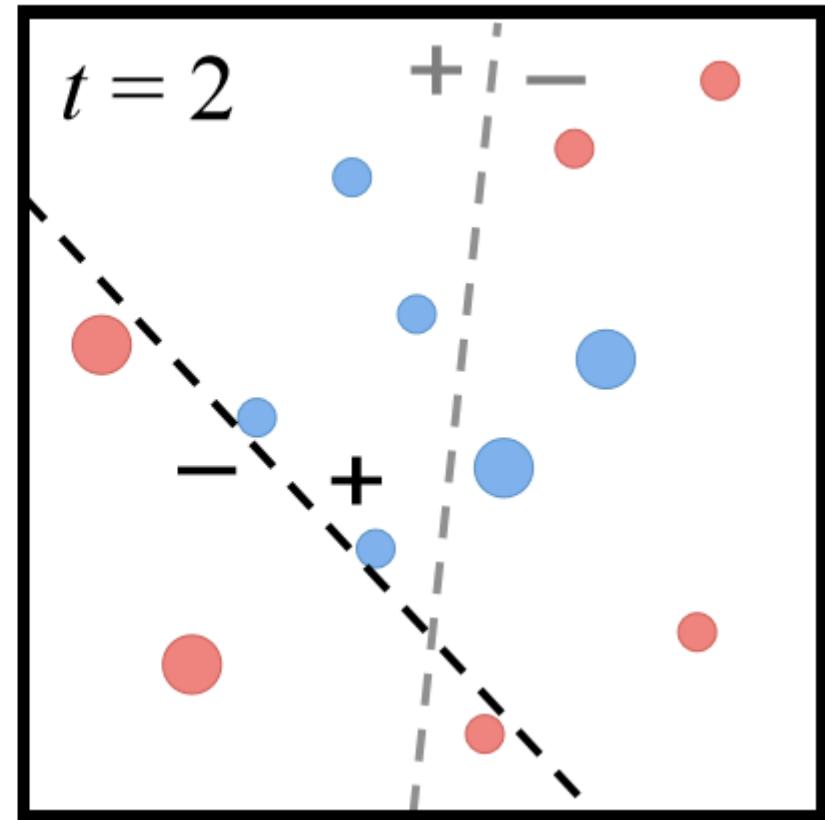


AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
   $H(\mathbf{x}) = \text{sign} \left( \sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$ 

```

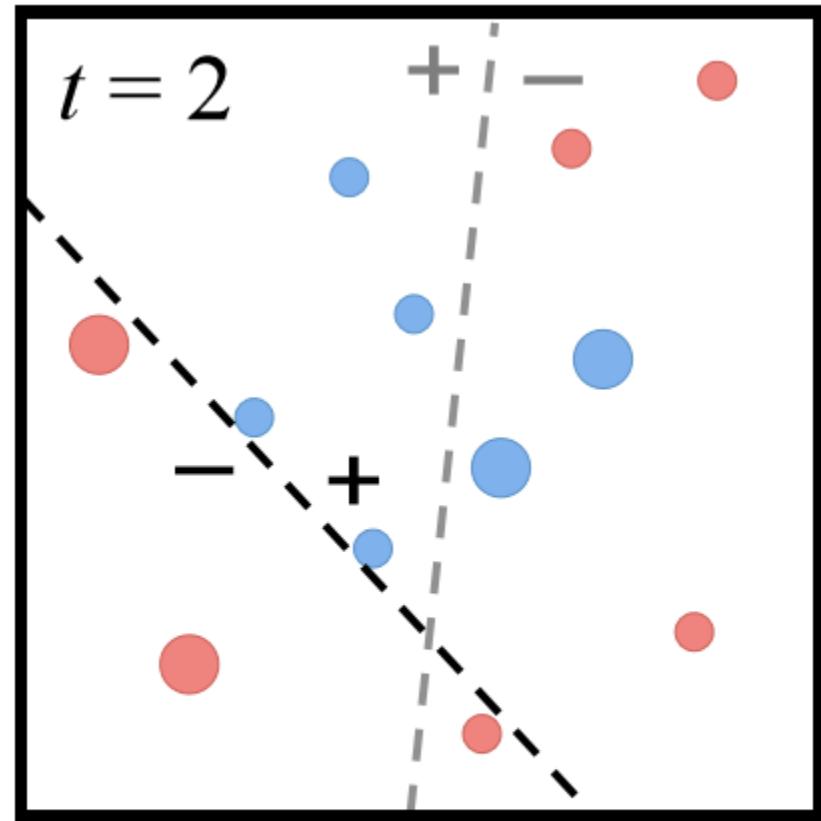


AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
  
```

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



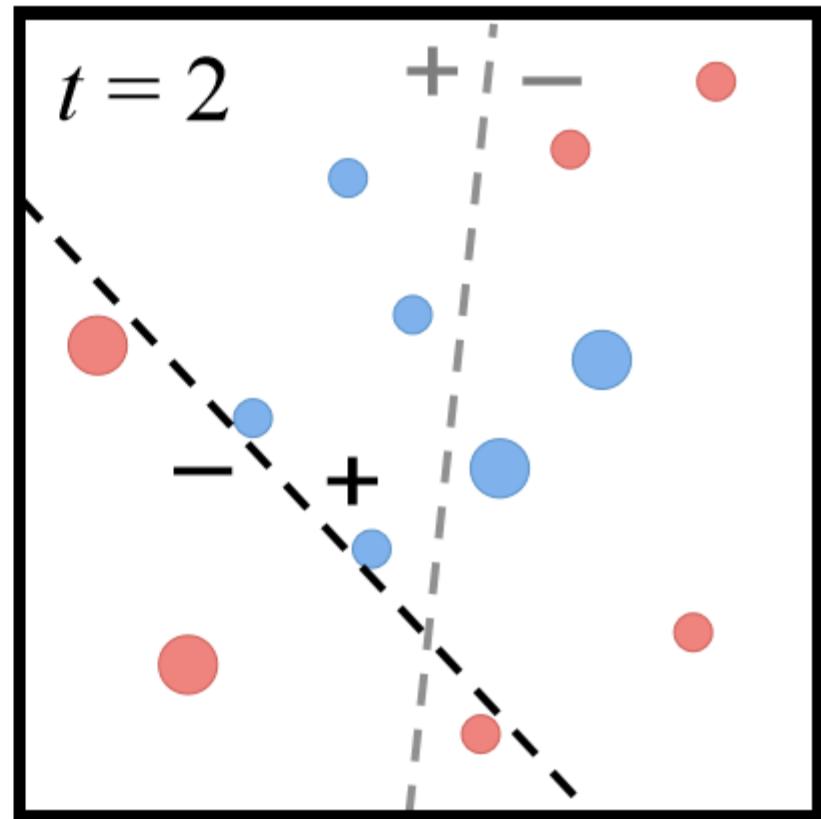
- β_t measures the importance of h_t
- If $\epsilon_t \leq 0.5$, then $\beta_t \geq 0$ (β_t grows as ϵ_t gets smaller)

AdaBoost Algorithm

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

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



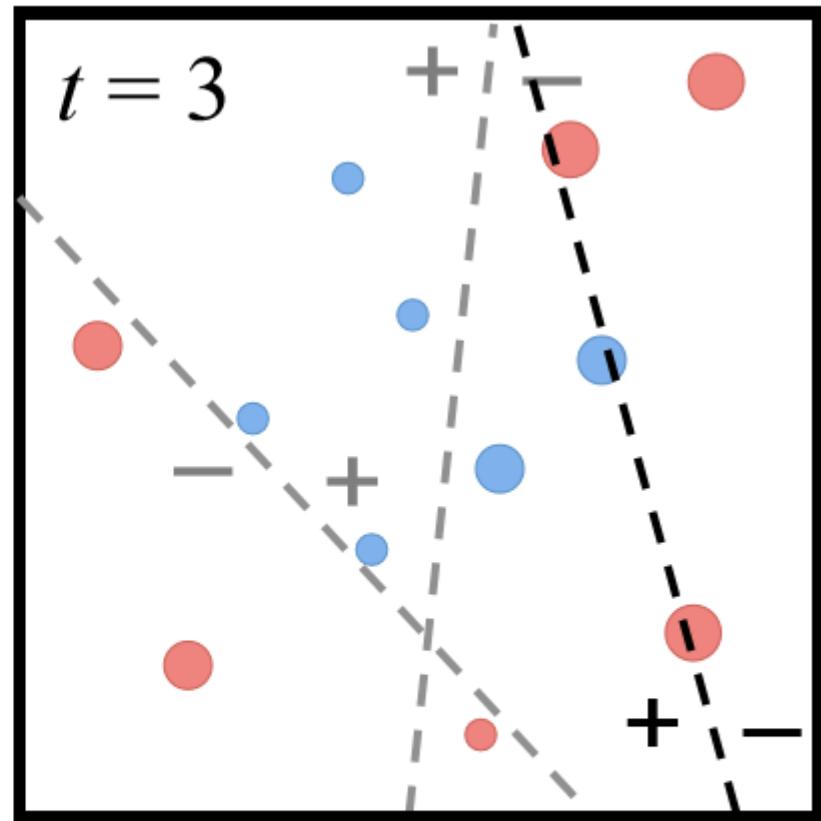
- Weights of correct predictions are multiplied by $e^{-\beta_t} \leq 1$
- Weights of incorrect predictions are multiplied by $e^{\beta_t} \geq 1$

AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
   $H(\mathbf{x}) = \text{sign} \left( \sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$ 

```



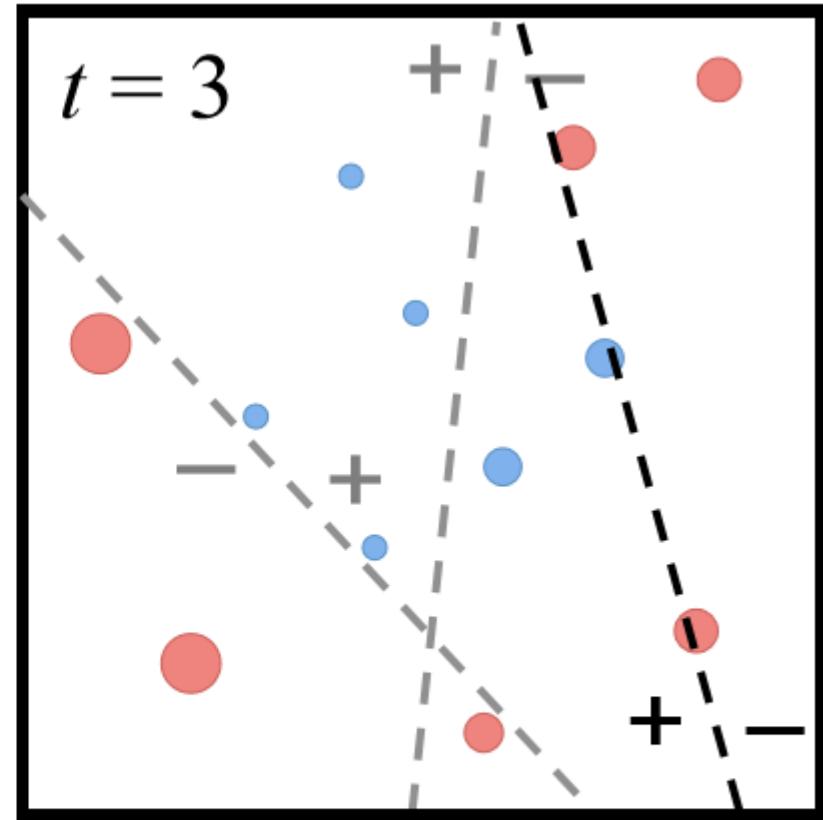
- β_t measures the importance of h_t
- If $\epsilon_t \leq 0.5$, then $\beta_t \geq 0$ (β_t grows as ϵ_t gets smaller)

AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
  
```

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



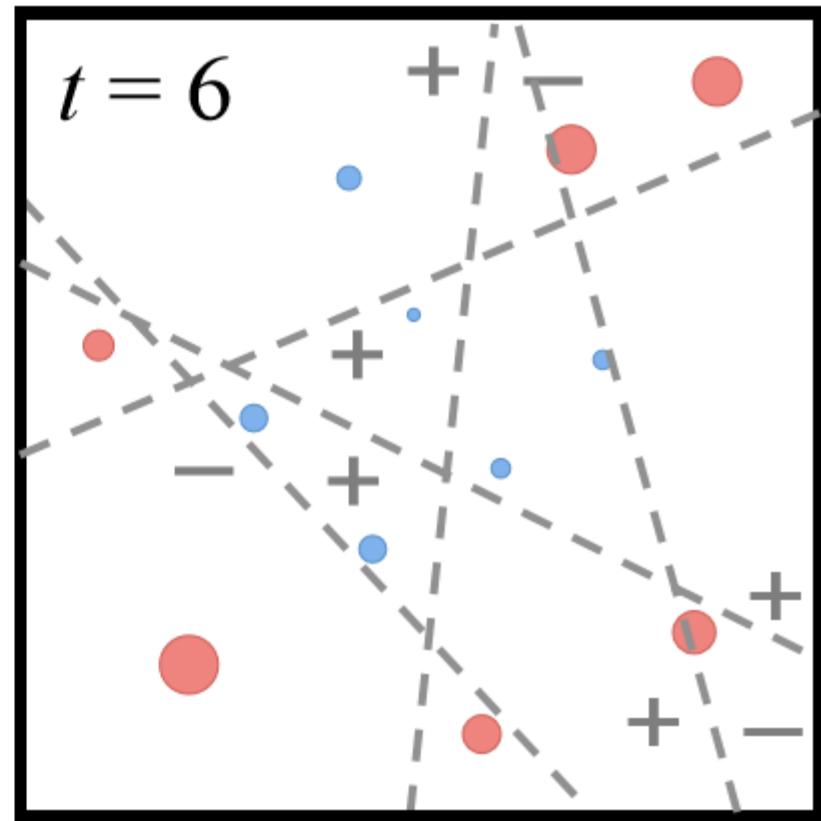
- Weights of correct predictions are multiplied by $e^{-\beta_t} \leq 1$
- Weights of incorrect predictions are multiplied by $e^{\beta_t} \geq 1$

AdaBoost Algorithm

```

1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
       $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$ 
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
9: Return the hypothesis
   $H(\mathbf{x}) = \text{sign} \left( \sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$ 

```



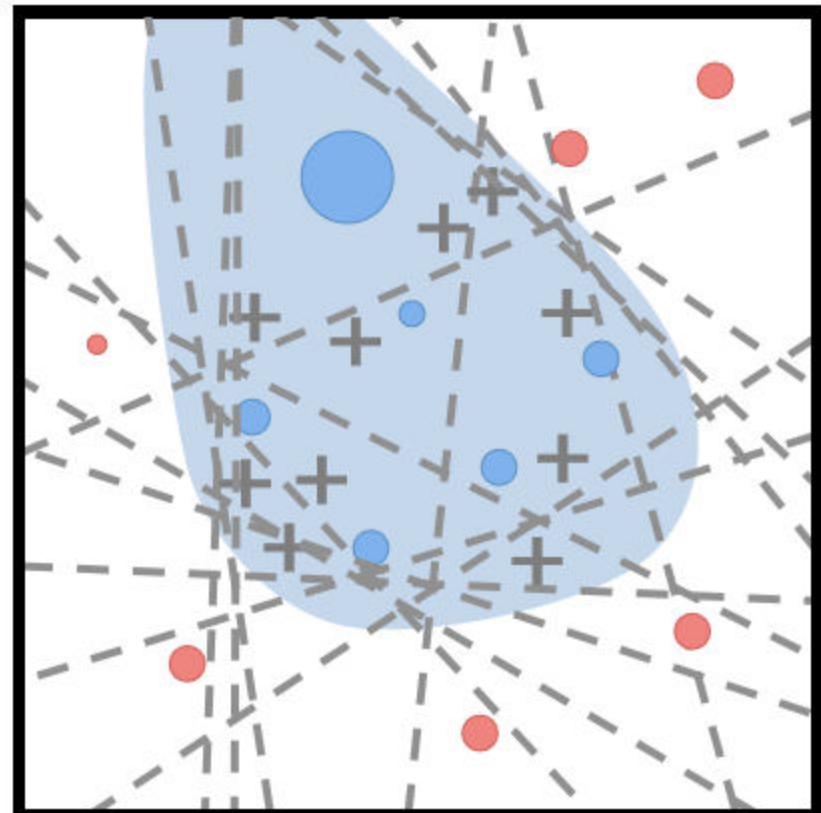
AdaBoost Algorithm

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

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

$t = T$



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

AdaBoost Algorithm

INPUT: training data $X, y = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$,
the number of iterations T

1: Initialize a vector of n uniform weights $\mathbf{w}_1 = [\frac{1}{n}, \dots, \frac{1}{n}]$

2: **for** $t = 1, \dots, T$

3: Train model h_t on X, y with instance weights \mathbf{w}_t

4: Compute the weighted training error rate of h_t :

$$\epsilon_t = \sum_{i:y_i \neq h_t(\mathbf{x}_i)} w_{t,i}$$

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

6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i)) \quad \forall i = 1, \dots, n$$

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

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^n w_{t+1,j}} \quad \forall i = 1, \dots, n$$

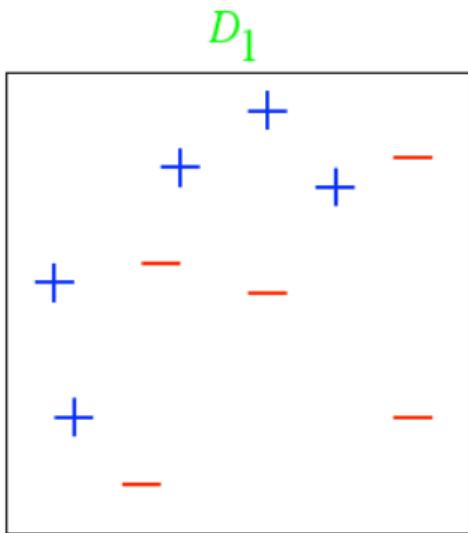
8: **end for**

9: **Return** the hypothesis

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

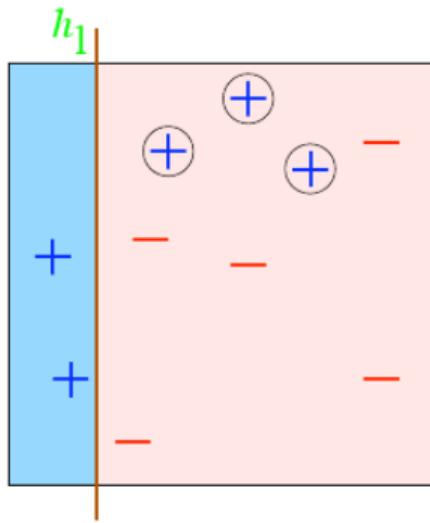
Member classifier with less error are given more weight in final ensemble hypothesis.
Final prediction is a weighted combination of each members prediction

Example



From, Léon Bottou

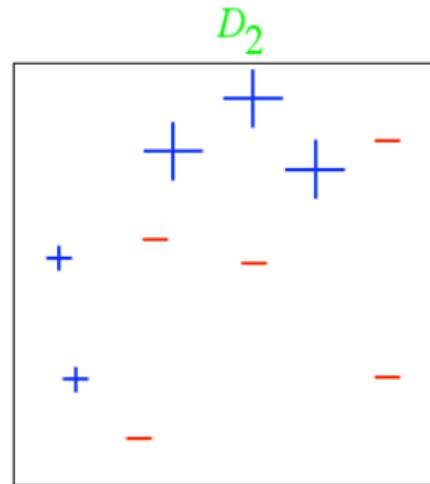
Example



$$\epsilon_1 = 0.30$$

$$\alpha_1 = 0.42$$

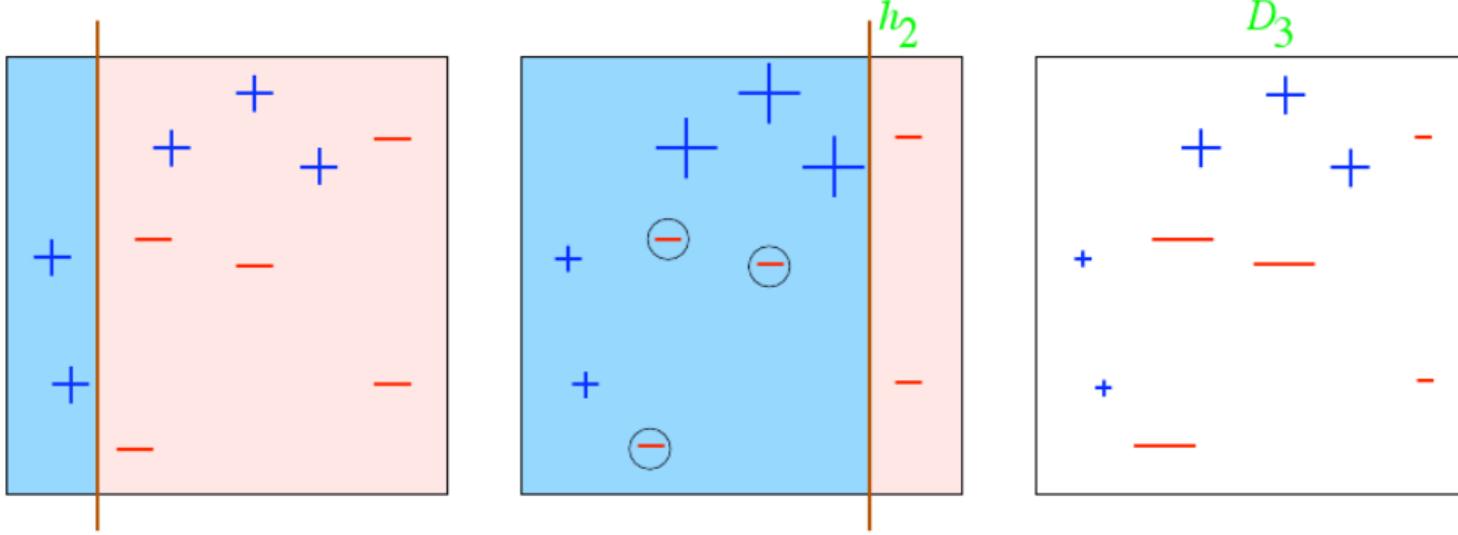
From, Léon Bottou



$$\epsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(x_j) \neq y_j)$$

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \epsilon_i}{\epsilon_i} \right)$$

Example



$$\varepsilon_2 = 0.21$$

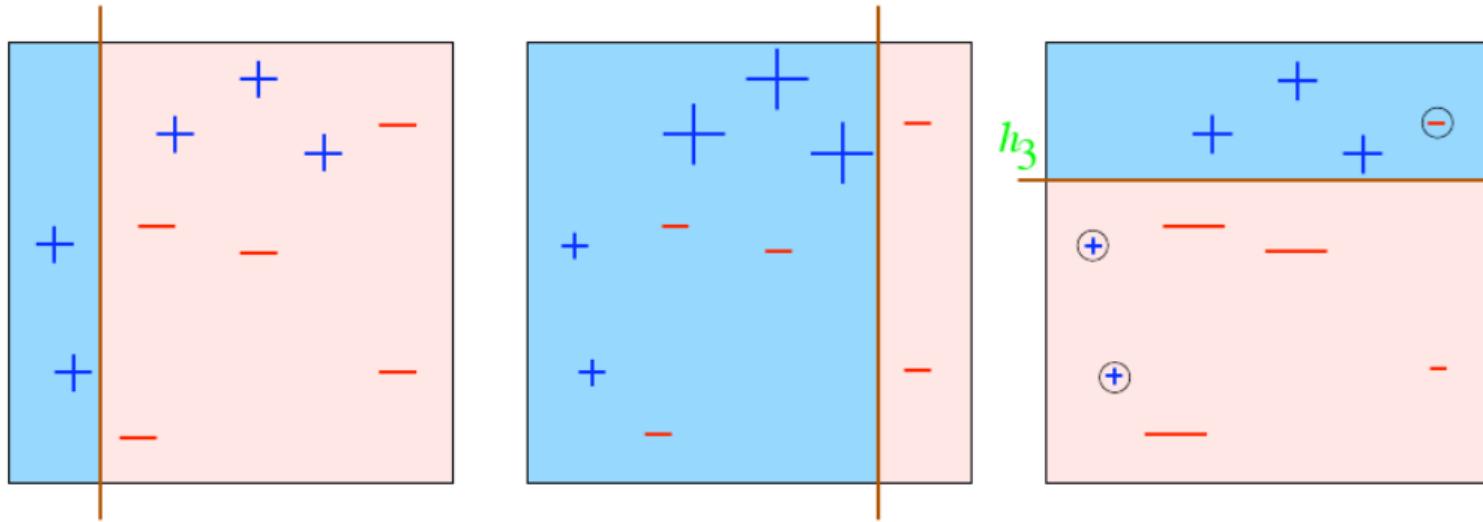
$$\alpha_2 = 0.65$$

$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(x_j) \neq y_j)$$

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$

From, Léon Bottou

Example



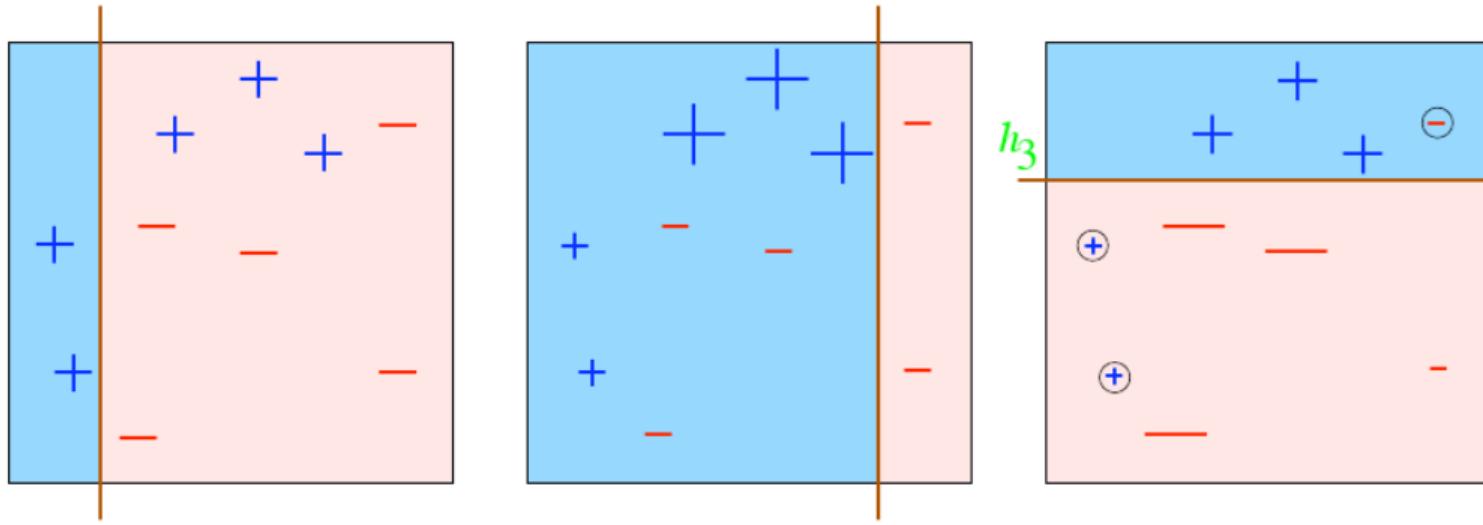
$$\varepsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

From, Léon Bottou

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$

Example



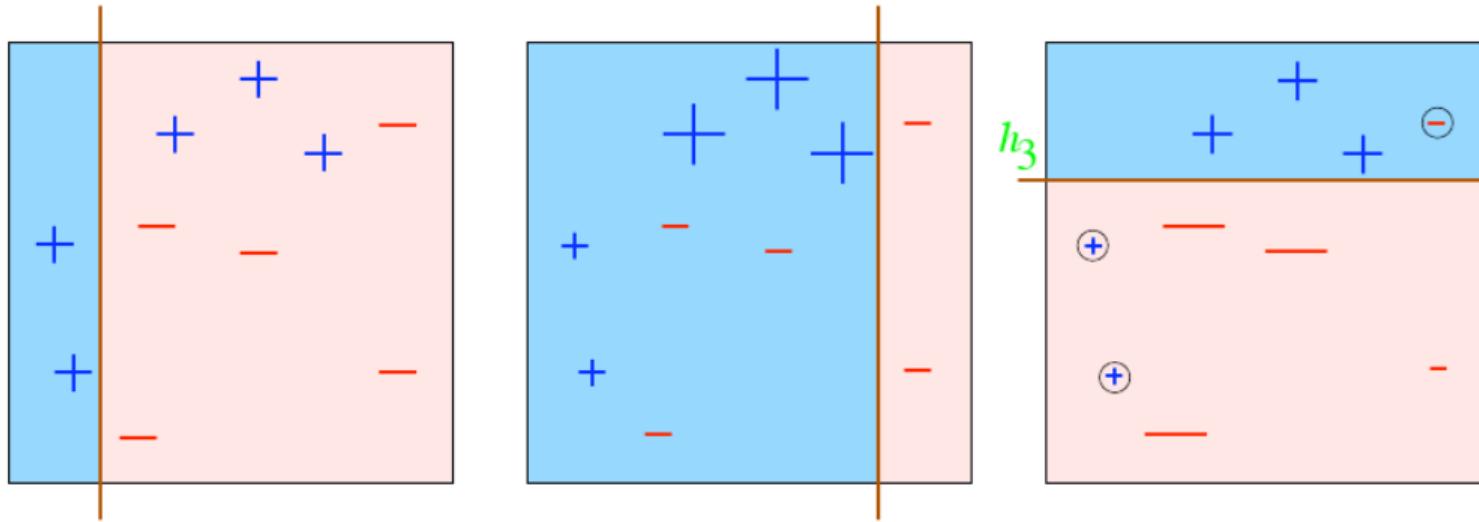
$$\varepsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

From, Léon Bottou

Example

How do we combine the results now?



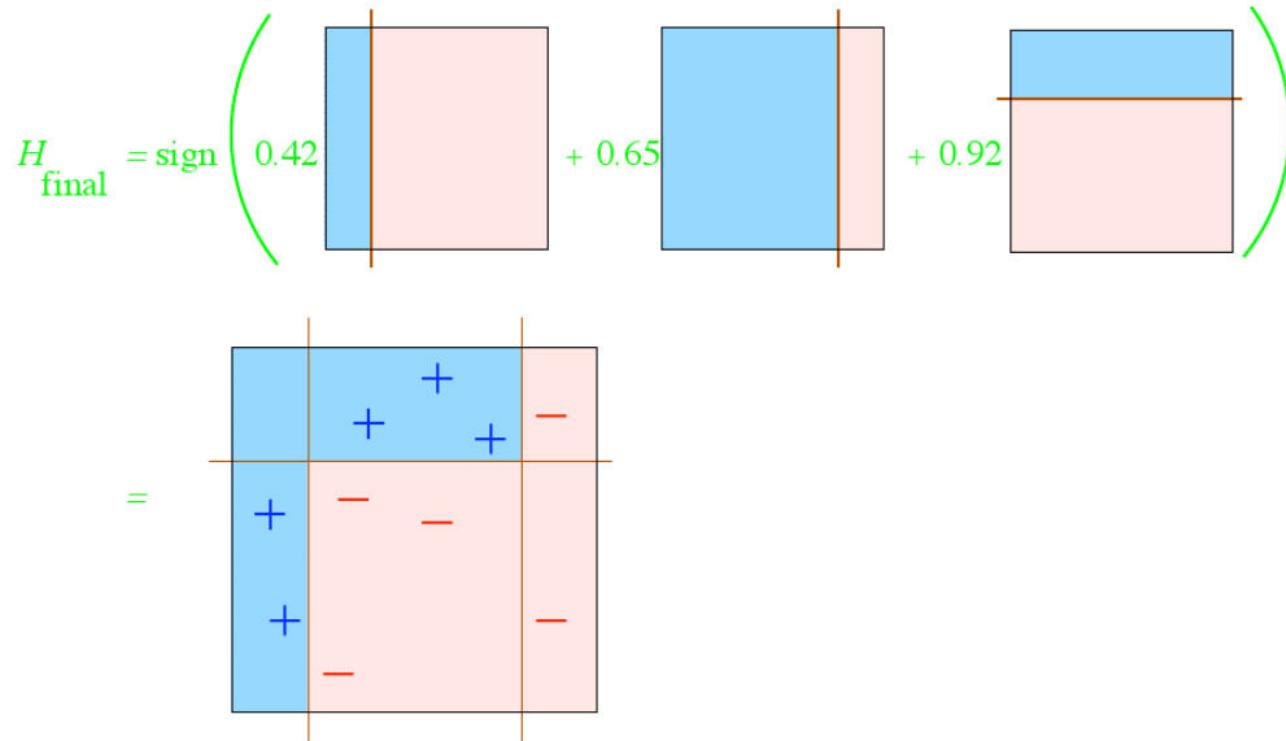
$$\varepsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

From, Léon Bottou

Example

How do we combine the results now?



From, Léon Bottou

AdaBoost Example

- Training sets for the first 3 boosting rounds:

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

- Summary:

Round	Split Point	Left Class	Right Class	alpha
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

AdaBoost Example

- Weights

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

- Classification

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Predicted Class

AdaBoost error function takes into account the fact that only the sign of the final result is used, thus sum can be far larger than 1 without increasing error

AdaBoost base learners

- AdaBoost works best with “weak” learners
 - Should not be complex
 - Typically high bias classifiers
 - Works even when weak learner has an error rate just slightly under 0.5 (i.e., just slightly better than random)
 - Can prove training error goes to 0 in $O(\log n)$ iterations
- Examples:
 - Decision stumps (1 level decision trees)
 - Depth-limited decision trees
 - Linear classifiers

AdaBoost in practice

Strengths:

- Fast and simple to program
- No parameters to tune (besides T)
- No assumptions on weak learner

When boosting can fail:

- Given insufficient data
- Overly complex weak hypotheses
- Can be susceptible to noise
- When there are a large number of outliers

Fine Tuning Ensembles

- Model combination does not always guaranteed to decrease error, unless
 - base-learners are diverse and accurate
- Ignore poor base learners
 - Use accuracy as a cut-off
 - Introduce some pruning with which at each iteration remove poor learners / learners whose absence lead to improvement (if any)
 - Modify iterations to allow both additions / deletions of learners
 - Discarding appropriately leads to better performance

Gradient Boosting

Gradient Boosting

Gradient boosting is somewhat similar to AdaBoost:

- trees are fit sequentially to improve error of previous trees
- boost weak learners to a strong learner

The way how the trees are fit sequentially differs in AdaBoost and Gradient Boosting, though ...

Gradient Boosting -- Conceptual Overview

- **Step 1:** Construct a base tree (just the root node)
- **Step 2:** Build next tree based on errors of the previous tree
- **Step 3:** Combine tree from step 1 with trees from step 2. Go back to step 2.

Gradient Boosting -- Conceptual Overview

--> A Regression-based Example

x1# Rooms	x2=City	x3=Age	y=Price
5	Boston	30	1.5
10	Madison	20	0.5
6	Lansing	20	0.25
5	Waunakee	10	0.1

In million US Dollars

- **Step 1:** Construct a base tree (just the root node)

$$\hat{y}_1 = \frac{1}{n} \sum_{i=1}^n y^{(i)} = 0.5875$$

Gradient Boosting -- Conceptual Overview

--> A Regression-based Example

- **Step 2:** Build next tree based on errors of the previous tree

First, compute (pseudo) residuals: $r_1 = y_1 - \hat{y}_1$

In million US Dollars

x1#	x2=City	x3=Age	y=Price	r1=Res
5	Boston	30	1.5	1.5 - 0.5875 = 0.9125
10	Madison	20	0.5	0.5 - 0.5875 = -0.0875
6	Lansing	20	0.25	0.25 - 0.5875 = -0.3375
5	Waunake	10	0.1	0.1 - 0.5875 = -0.4875

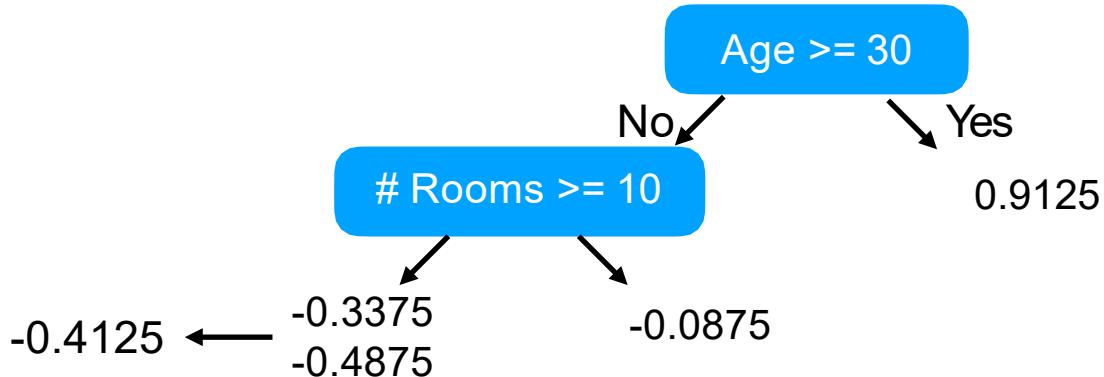
Gradient Boosting -- Conceptual Overview

--> A Regression-based Example

- **Step 2:** Build next tree based on errors of the previous tree

Then, create a tree based on X_1, \dots, X_m to fit the residuals

x1#	x2=City	x3=Age	y=Price	r1=Residual
5	Boston	30	1.5	$1.5 - 0.5875 = 0.9125$
10	Madison	20	0.5	$0.5 - 0.5875 = -0.0875$
6	Lansing	20	0.25	$0.25 - 0.5875 = -0.3375$
5	Waunake	10	0.1	$0.1 - 0.5875 = -0.4875$



Gradient Boosting -- Conceptual Overview

--> A Regression-based Example

- Step 3: Combine tree from step 1 with trees from step 2

x1#	x2=City	x3=Age	y=Price	r=Res
5	Boston	30	1.5	$1.5 - 0.5875 = 0.9125$
10	Madison	20	0.5	$0.5 - 0.5875 = -0.0875$
6	Lansing	20	0.25	$0.25 - 0.5875 = -0.3375$
5	Waunake	10	0.1	$0.1 - 0.5875 = -0.4875$

$$\hat{y}_1 = \frac{1}{n} \sum_{i=1}^n y^{(i)} = 0.5875 +$$

Age ≥ 30

Rooms ≥ 10

0.9125

-0.3375
-0.4875

-0.0875

-0.4125

Gradient Boosting -- Conceptual Overview

--> A Regression-based Example

- **Step 3:** Combine tree from step 1 with trees from step 2

x1#	x2=City	x3=Age	y=Price	r=Res
5	Boston	30	1.5	1.5 - 0.5875 = 0.9125
10	Madison	20	0.5	0.5 - 0.5875 = -0.0875
6	Lansing	20	0.25	0.25 - 0.5875 = -0.3375
5	Waunakee	10	0.1	0.1 - 0.5875 = -0.4875

E.g.,
predict
Lansing →

$$\hat{y}_1 = \frac{1}{n} \sum_{i=1}^n y^{(i)} = 0.5875 + \begin{array}{c} \text{Age } \geq 30 \\ \text{# Rooms } \geq 10 \end{array}$$

0.9125 -0.3375 -0.0875
No Yes

-0.4125 ← -0.4875

E.g.,
predict
Lansing

0.5875 + $\alpha \times (-0.4125)$

Where learning rate between 0 and 1 (if $\alpha=1$, low bias but high variance)

Gradient Boosting -- Algorithm Overview

Step 0: Input data $\{\langle \mathbf{x}^{(i)}, y^{(i)} \rangle\}_{i=1}^n$

Differentiable Loss function $L(y^{(i)}, h(\mathbf{x}^{(i)}))$

Step 1: Initialize model $h_0(\mathbf{x}) = \operatorname{argmin}_{\hat{y}} \sum_{i=1}^n L(y^{(i)}, \hat{y})$

Step 2: for $t = 1$ to T

A. Compute pseudo residual $r_{i,t} = - \left[\frac{\partial L(y^{(i)}, h(\mathbf{x}^{(i)}))}{\partial h(\mathbf{x}^{(i)})} \right]$

$h(\mathbf{x}) = h_{t-1}(\mathbf{x})$
for $i=1$ to n

B. Fit tree to $r_{i,t}$ values, and create terminal nodes $R_{j,t}$ for $j = 1, \dots, J_t$

• • •

Gradient Boosting -- Algorithm Overview

Step 2: for $t = 1$ to T

A. Compute pseudo residual $r_{i,t} = - \left[\frac{\partial L(y^{(i)}, h(\mathbf{x}^{(i)}))}{\partial h(\mathbf{x}^{(i)})} \right]$

$$h(\mathbf{x}) = h_{t-1}(\mathbf{x}) \\ \text{for } i=1 \text{ to } n$$

B. Fit tree to $r_{i,t}$ values, and create terminal nodes

$$R_{j,t} \text{ for } j = 1, \dots, J_t$$

C. for $j = 1, \dots, J_t$, compute

$$\hat{y}_{j,t} = \underset{\hat{y}}{\operatorname{argmin}} \sum_{\mathbf{x}^{(i)} \in R_{j,t}} L(y^{(i)}, h_{t-1}(\mathbf{x}^{(i)}) + \hat{y})$$

D. Update $h_t(\mathbf{x}) = h_{t-1}(\mathbf{x}) + \alpha \sum_{j=1}^{J_t} \hat{y}_{j,t} \mathbb{I}(\mathbf{x} \in R_{j,t})$

Step 3: Return $h_t(\mathbf{x})$

Gradient Boosting -- Algorithm Overview Discussion

Step 0: Input data $\{\langle \mathbf{x}^{(i)}, y^{(i)} \rangle\}_{i=1}^n$

Differentiable Loss function $L(y^{(i)}, h(\mathbf{x}^{(i)}))$

E.g., Sum-squared error in regression

$$SSE' = \frac{1}{2} (y^{(i)} - h(\mathbf{x}^{(i)}))^2$$

$$\frac{\partial}{\partial h(\mathbf{x}^{(i)})} \frac{1}{2} (y^{(i)} - h(\mathbf{x}^{(i)}))^2 \quad [\text{chain rule}]$$

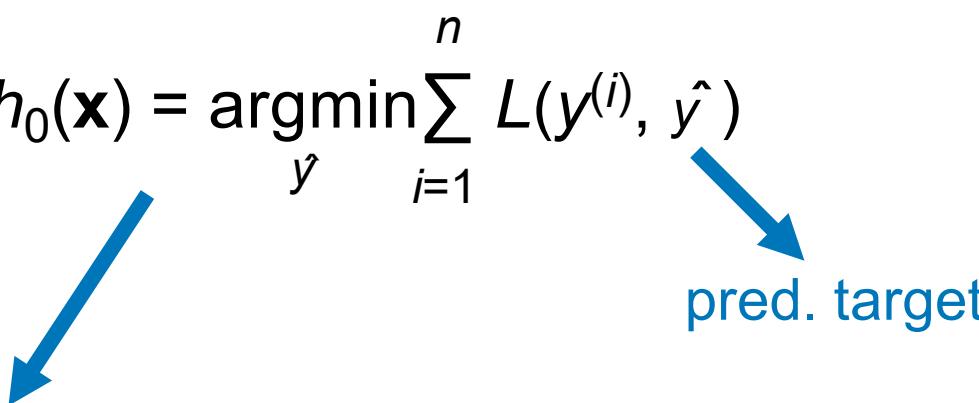
$$= 2 \times \frac{1}{2} (y^{(i)} - h(\mathbf{x}^{(i)})) \times (0 - 1) = - (y^{(i)} - h(\mathbf{x}^{(i)}))$$

[neg. residual]

Gradient Boosting -- Algorithm Overview Discussion

Step 1:

Initialize model $h_0(\mathbf{x}) = \operatorname{argmin}_{\hat{y}} \sum_{i=1}^n L(y^{(i)}, \hat{y})$



pred. target

turns out to be the average (in regression)

$$\frac{1}{n} \sum_{i=1}^n y^{(i)}$$

Gradient Boosting -- Algorithm Overview Discussion

Step 2: for $t = 1$ to T

A. Compute pseudo residual $r_{i,t} = - \left[\frac{\partial L(y^{(i)}, h(\mathbf{x}^{(i)}))}{\partial h(\mathbf{x}^{(i)})} \right]$

pseudo residual of the t -th tree
and i -th example

Loop to make T trees (e.g., $T=100$)

$h(\mathbf{x}) = h_{t-1}(\mathbf{x})$
for $i = 1$ to n

Derivative of the loss function

Gradient Boosting -- Algorithm Overview Discussion

Step 2: for $t = 1$ to T

A. Compute pseudo residual $r_{i,t} = - \left[\frac{\partial L(y^{(i)}, h(\mathbf{x}^{(i)}))}{\partial h(\mathbf{x}^{(i)})} \right]$

pseudo residual of the t -th tree
and i -th example

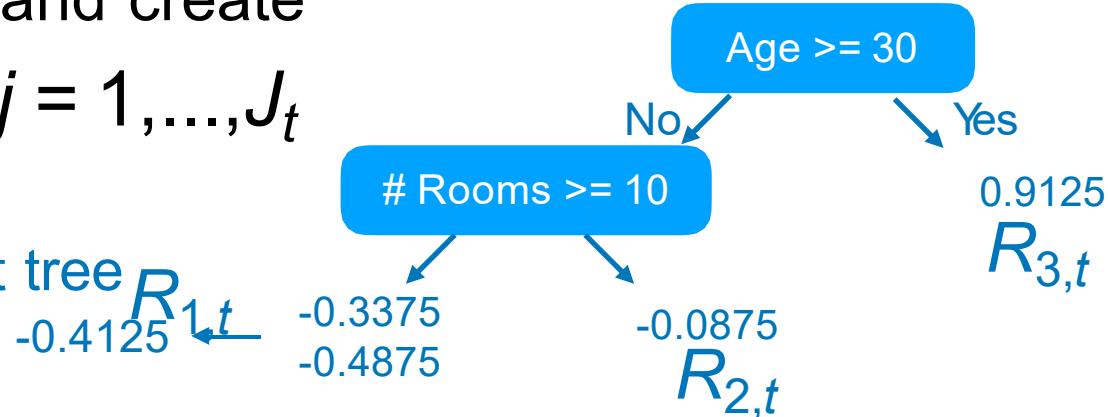
Derivative of the loss function

Loop to make T trees (e.g., $T=100$)

$h(\mathbf{x}) = h_{t-1}(\mathbf{x})$
for $i = 1$ to n

B. Fit tree to $r_{i,t}$ values, and create terminal nodes $R_{j,t}$ for $j = 1, \dots, J_t$

Use features in dataset to fit tree $R_{1,t}$



Gradient Boosting -- Algorithm Overview Discussion

Step 2: for $t = 1$ to T

A. Compute pseudo residual $r_{i,t} = - \left[\frac{\partial L(y^{(i)}, h(\mathbf{x}^{(i)}))}{\partial h(\mathbf{x}^{(i)})} \right]$

$h(\mathbf{x}) = h_{t-1}(\mathbf{x})$
for $i = 1$ to n

B. Fit tree to $r_{i,t}$ values, and create terminal nodes $R_{j,t}$ for $j = 1, \dots, J_t$

C. for $j = 1, \dots, J_t$, compute

$$\hat{y}_{j,t} = \underset{\hat{y}}{\operatorname{argmin}} \sum_{\mathbf{x}^{(i)} \in R_{j,t}} L(y^{(i)}, h_{t-1}(\mathbf{x}^{(i)}) + \hat{y})$$

Compute the residual for each leaf node

Only consider examples at that leaf node

Like step 1 but add previous prediction

Gradient Boosting -- Algorithm Overview Discussion

Step 2: for $t = 1$ to T

A. Compute pseudo residual $r_{i,t} = - \left[\frac{\partial L(y^{(i)}, h(\mathbf{x}^{(i)}))}{\partial h(\mathbf{x}^{(i)})} \right]$

$h(\mathbf{x}) = h_{t-1}(\mathbf{x})$
for $i = 1$ to n

B. Fit tree to $r_{i,t}$ values, and create terminal nodes $R_{j,t}$ for $j = 1, \dots, J_t$

C. for $j = 1, \dots, J_t$, compute

$$\hat{y}_{j,t} = \underset{\hat{y}}{\operatorname{argmin}} \sum_{\mathbf{x}^{(i)} \in R_{j,t}} L(y^{(i)}, h_{t-1}(\mathbf{x}^{(i)}) + \hat{y})$$

D. Update $h_t(\mathbf{x}) = h_{t-1}(\mathbf{x}) + \alpha \sum_{j=1}^{J_t} \hat{y}_{j,t} \mathbb{I}(\mathbf{x} \in R_{j,t})$

learning rate
between 0 and 1
(usually 0.1)

Summation just in case examples end up in multiple nodes

Gradient Boosting -- Algorithm Overview Discussion

For prediction, combine all T trees, e.g.,

$$h_0(\mathbf{x}) = \underset{\hat{y}}{\operatorname{argmin}} \sum_{i=1}^n L(y^{(i)}, \hat{y})$$

$$+\alpha \hat{y}_{j,t=1} = \underset{\hat{y}}{\operatorname{argmin}} \sum_{\mathbf{x}^{(i)} \in R_{i,j}} L(y^{(i)}, h_{(t=1)-1}(\mathbf{x}^{(i)}) + \hat{y})$$

...

$$+\alpha \hat{y}_{j,T} = \underset{\hat{y}}{\operatorname{argmin}} \sum_{\mathbf{x}^{(i)} \in R_{i,j}} L(y^{(i)}, h_{T-1}(\mathbf{x}^{(i)}) + \hat{y})$$

Gradient Boosting -- Algorithm Overview Discussion

For prediction, combine all T trees, e.g.,

$$h_0(\mathbf{x}) = \operatorname{argmin}_{\hat{y}} \sum_{i=1}^n L(y^{(i)}, \hat{y})$$

$$+ \alpha \hat{y}_{j,t=1}$$

...

$$+ \alpha \hat{y}_{j,T}$$

The idea is that we decrease the pseudo residuals by a small amount at each step

Good Reading

<https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/>

XGBoost

<https://arxiv.org/abs/1603.02754>

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 785-794). ACM.

Summary and Main Points:

- scalable implementation of gradient boosting
- Improvements include: regularized loss, sparsity-aware algorithm, weighted quantile sketch for approximate tree learning, caching of access patterns, data compression, sharding
- Decision trees based on CART
- Regularization term for penalizing model (tree) complexity
- Uses second order approximation for optimizing the objective
- Options for column-based and row-based subsampling
- Single-machine version of XGBoost supports the exact greedy algorithm

Why XGBoost so popular?

- **Speed** : faster than other ensemble classifiers.
- **Core algorithm is parallelizable**: harness the power of multi-core computers and networks of computers enabling to train on very large datasets **Consistently outperforms other algorithm methods** : It has shown better performance on a variety of machine learning benchmark datasets.
- **Wide variety of tuning parameters** : cross-validation, regularization, missing values, tree parameters, etc
- XGBoost (Extreme Gradient Boosting) uses the gradient boosting (GBM) framework at its core.

Gradient Boosting

- In Gradient Boosting, "shortcomings" are identified by gradients.
- Recall that, in Adaboost, “shortcomings” are identified by high-weight data points.
- Both high-weight data points and gradients tell us how to improve our model.

Gradient Boosting

- Gradient Boosting for Different Problems
Difficulty: regression ==> classification
==> ranking

References

The-Morgan-Kaufmann-Series-in-Data-Management-
Systems-Jiawei-Han-Micheline-Kamber-Jian-Pei-Data-
Mining.-Concepts-and-Techniques-3rd-Edition-Morgan-
Kaufmann-2011

Bishop - Pattern Recognition And Machine Learning -
Springer 2006

A Gentle Introduction to Gradient Boosting
Cheng Li chengli@ccs.neu.edu College of Computer and
Information Science Northeastern University

https://www.youtube.com/watch?time_continue=647&v=LsK-xG1cLYA&feature=emb_logo



Thank You!