Learning from Examples

Dr. Steven Bethard

Computer and Information Sciences
University of Alabama at Birmingham

31 Mar 2016

Outline

Outline

1. Supervised Learning
   - Hypothesis Functions
   - Features
   - Evaluating Hypotheses
   - Overfitting and Underfitting
2. Supervised Learning Algorithms
   - Decision Trees
   - Random Forests
   - k-Nearest Neighbors
   - Linear and Logistic Regression
   - Support Vector Machines
   - Neural Networks

Supervised Learning

Key Ideas

- Examine pairs of inputs and outputs
- Guess a possible function mapping input to output
- Predict outputs given new inputs

$$\begin{aligned} f(1) &= 1 \\ f(2) &= 4 \\ f(3) &= 9 \\ f(4) &= 16 \\ f(5) &= ? \end{aligned}$$

$$\begin{aligned} f(\text{Romeo and Juliet}) &= \text{Shakespeare} \\ f(\text{Tom Sawyer}) &= \text{Twain} \\ f(\text{Macbeth}) &= \text{Shakespeare} \\ f(\text{Huckleberry Finn}) &= \text{Twain} \\ f(\text{Othello}) &= ? \end{aligned}$$

Supervised Learning

Key Ideas

- Examine pairs of inputs and outputs
- Guess a possible function mapping input to output
- Predict outputs given new inputs

$$
\begin{aligned}
f(1) &= 1 \\
f(2) &= 4 \\
f(3) &= 9 \\
f(4) &= 16 \\
f(5) &= ?
\end{aligned}
\qquad\qquad
\begin{aligned}
f(\text{Romeo and Juliet}) &= \text{Shakespeare} \\
f(\text{Tom Sawyer}) &= \text{Twain} \\
f(\text{Macbeth}) &= \text{Shakespeare} \\
f(\text{Huckleberry Finn}) &= \text{Twain} \\
f(\text{Othello}) &= ?
\end{aligned}
$$

Supervised Learning

Key Ideas

- Examine pairs of inputs and outputs
- Guess a possible function mapping input to output
- Predict outputs given new inputs

$$
\begin{aligned}
f(1) &= 1 \\
f(2) &= 4 \\
f(3) &= 9 \\
f(4) &= 16 \\
f(5) &= 25
\end{aligned}
$$

$$
\begin{aligned}
f(\text{Romeo and Juliet}) &= \text{Shakespeare} \\
f(\text{Tom Sawyer}) &= \text{Twain} \\
f(\text{Macbeth}) &= \text{Shakespeare} \\
f(\text{Huckleberry Finn}) &= \text{Twain} \\
f(\text{Othello}) &= ?
\end{aligned}
$$

Supervised Learning

Key Ideas

- Examine pairs of inputs and outputs
- Guess a possible function mapping input to output
- Predict outputs given new inputs

$$\begin{aligned} f(1) &= 1 \\ f(2) &= 4 \\ f(3) &= 9 \\ f(4) &= 16 \\ f(5) &= \textcolor{red}{25} \end{aligned}$$

$$\begin{aligned} f(\text{Romeo and Juliet}) &= \text{Shakespeare} \\ f(\text{Tom Sawyer}) &= \text{Twain} \\ f(\text{Macbeth}) &= \text{Shakespeare} \\ f(\text{Huckleberry Finn}) &= \text{Twain} \\ f(\text{Othello}) &= \text{?} \end{aligned}$$

Supervised Learning

Key Ideas

- Examine pairs of inputs and outputs
- Guess a possible function mapping input to output
- Predict outputs given new inputs

$$
\begin{aligned}
f(1) &= 1 \\
f(2) &= 4 \\
f(3) &= 9 \\
f(4) &= 16 \\
f(5) &= \color{red}{25}
\end{aligned}
\qquad
\begin{aligned}
f(\text{Romeo and Juliet}) &= \text{Shakespeare} \\
f(\text{Tom Sawyer}) &= \text{Twain} \\
f(\text{Macbeth}) &= \text{Shakespeare} \\
f(\text{Huckleberry Finn}) &= \text{Twain} \\
f(\text{Othello}) &= \color{red}{\text{Shakespeare}}
\end{aligned}
$$

Supervised Learning Components

Original Function

An unknown function $f : D_f \rightarrow R_f$

Training Examples

Pairs of $(x, f(x))$ where $x \in D_f$ and $f(x) \in R_f$

Hypothesis Function

Some function $h : D_f \rightarrow R_f$

Learning Goal

Pick a hypothesis h as close to f as possible

Supervised Learning Components

Original Function

An unknown function $f : D_f \rightarrow R_f$

Training Examples

Pairs of $(x, f(x))$ where $x \in D_f$ and $f(x) \in R_f$

Hypothesis Function

Some function $h : D_f \rightarrow R_f$

Learning Goal

Pick a hypothesis h as close to f as possible

Supervised Learning Components

Original Function

An unknown function $f: D_f \rightarrow R_f$

Training Examples

Pairs of $(x, f(x))$ where $x \in D_f$ and $f(x) \in R_f$

Hypothesis Function

Some function $h: D_f \rightarrow R_f$

Learning Goal

Pick a hypothesis h as close to f as possible

Supervised Learning Components

Original Function

An unknown function $f: D_f \to R_f$

Training Examples

Pairs of $(x, f(x))$ where $x \in D_f$ and $f(x) \in R_f$

Hypothesis Function

Some function $h: D_f \to R_f$

Learning Goal

Pick a hypothesis h as close to f as possible

Problem Formulation

Defining a Machine Learning Problem

Describe the function $f : D_f \to R_f$

- What is D_f?
- What is R_f?

Ex: Income Prediction

D_f = people

$R_f = \mathbb{R}$

Ex: Face Recognition

D_f = image regions

$R_f = [0, 1]$

Most algorithms work best when $R_f \subseteq \mathbb{R}$ or $|R_f| \ll 100$

Problem Formulation

Defining a Machine Learning Problem

Describe the function $f : D_f \to R_f$

- What is D_f?
- What is R_f?

Ex: Income Prediction

D_f = people

R_f = $\mathbb{R}$

Ex: Face Recognition

D_f = image regions

R_f = $\{0, 1\}$

Most algorithms work best when $R_f \subseteq \mathbb{R}$ or $|R_f| \ll 100$

Problem Formulation

Defining a Machine Learning Problem

Describe the function $f : D_f \to R_f$

- What is D_f?
- What is R_f?

Ex: Income Prediction

D_f = people

R_f = $\mathbb{R}$

Ex: Face Recognition

D_f = image regions

R_f = $\{0, 1\}$

Most algorithms work best when $R_f \subseteq \mathbb{R}$ or $|R_f| \ll 100$

Problem Formulation

Defining a Machine Learning Problem

Describe the function $f : D_f \rightarrow R_f$

- What is D_f?
- What is R_f?

Ex: Income Prediction

D_f = people

$R_f = \mathbb{R}$

Ex: Face Recognition

D_f = image regions

$R_f = [0, 1]$

Most algorithms work best when $R_f \subseteq \mathbb{R}$ or $|R_f| \ll 100$

Problem Formulation

Defining a Machine Learning Problem

Describe the function $f : D_f \longrightarrow R_f$

- What is D_f?
- What is R_f?

Ex: Income Prediction

D_f = people

$R_f = \mathbb{R}$

Ex: Face Recognition

D_f = image regions

$R_f = \{0, 1\}$

Most algorithms work best when $R_f \subseteq \mathbb{R}$ or $|R_f| \ll 100$

Problem Formulation

Defining a Machine Learning Problem

Describe the function $f : D_f \rightarrow R_f$

- What is D_f?
- What is R_f?

Ex: Income Prediction

D_f = people

R_f = $\mathbb{R}$

Ex: Face Recognition

D_f = image regions

R_f = $\{0, 1\}$

Most algorithms work best when $R_f \subseteq \mathbb{R}$ or $|R_f| \ll 100$

Problem Formulation

Defining a Machine Learning Problem

Describe the function $f : D_f \rightarrow R_f$

- What is D_f?
- What is R_f?

Ex: Income Prediction

D_f = people

$R_f = \mathbb{R}$

Ex: Face Recognition

D_f = image regions

$R_f = \{0, 1\}$

Most algorithms work best when $R_f \subseteq \mathbb{R}$ or $|R_f| \ll 100$

Problem Formulation

Defining a Machine Learning Problem

Describe the function $f : D_f \rightarrow R_f$

- What is D_f?
- What is R_f?

Ex: Income Prediction

D_f = people

R_f = $\mathbb{R}$

Ex: Face Recognition

D_f = image regions

R_f = $\{0, 1\}$

Most algorithms work best when $R_f \subseteq \mathbb{R}$ or $|R_f| \ll 100$

Decomposing Domain Objects

Smoker identification:

$$\begin{aligned} f(\text{John}) &= \textit{true} \\ f(\text{Mary}) &= \textit{false} \\ f(\text{Frank}) &= \textit{false} \\ f(\text{Sally}) &= ? \end{aligned}$$

Easier if we know, e.g.

- cigarette smell?
- teeth stains?
- deep cough?

Definition

Features (or attributes) are the components of a domain object believed to be important for learning the function f

Decomposing Domain Objects

Smoker identification:

$$\begin{aligned} f(\text{John}) &= \textit{true} \\ f(\text{Mary}) &= \textit{false} \\ f(\text{Frank}) &= \textit{false} \\ f(\text{Sally}) &= ? \end{aligned}$$

Easier if we know, e.g.

- cigarette smell?
- teeth stains?
- deep cough?

Definition

Features (or attributes) are the components of a domain object believed to be important for learning the function f

Decomposing Domain Objects

Smoker identification:

$$\begin{aligned} f(\text{John}) &= \textit{true} \\ f(\text{Mary}) &= \textit{false} \\ f(\text{Frank}) &= \textit{false} \\ f(\text{Sally}) &= ? \end{aligned}$$

Easier if we know, e.g.

- cigarette smell?
- teeth stains?
- deep cough?

Definition

Features (or attributes) are the components of a domain object believed to be important for learning the function f

Part of Speech Tagging Example

Input	*John*	*broke*	*the*	*red*	*lamp*
Output	NOUN	VERB	DET	ADJ	NOUN

Part of Speech Tagging Example

Input	*John*	*broke*	*the*	*red*	*lamp*
Output	NOUN	VERB	DET	ADJ	NOUN

Function Description

D_f =

R_f =

Part of Speech Tagging Example

Input	*John*	*broke*	*the*	*red*	*lamp*
Output	NOUN	VERB	DET	ADJ	NOUN

Function Description

$$D_f = \{a, aardvark, abacus, abalone, \ldots\}$$

$$R_f =$$

Part of Speech Tagging Example

Input	*John*	*broke*	*the*	*red*	*lamp*
Output	NOUN	VERB	DET	ADJ	NOUN

Function Description

D_f = {*a*, *aardvark*, *abacus*, *abalone*, . . .}

R_f = {NOUN, VERB, ADJ, ADV, DET, . . .}

Part of Speech Tagging Example

Input	*John*	*broke*	*the*	*red*	*lamp*
Output	NOUN	VERB	DET	ADJ	NOUN

Function Description

$$D_f = \{a, aardvark, abacus, abalone, \ldots\}$$

$$R_f = \{\textsc{Noun}, \textsc{Verb}, \textsc{Adj}, \textsc{Adv}, \textsc{Det}, \ldots\}$$

$f(bark)$ = NOUN? VERB?

. . . *the **bark** of the tree* . . .

. . . *heard the dog **bark*** . . .

Part of Speech Tagging Example

Input	*John*	*broke*	*the*	*red*	*lamp*
Output	Noun	Verb	Det	Adj	Noun

Function Description

D_f = {*a*, *aardvark*, *abacus*, *abalone*, . . .} in a sentence

R_f = {Noun, Verb, Adj, Adv, Det, . . .}

f(*bark*) = Noun? Verb?

. . . *the* ***bark*** *of the tree* . . .

. . . *heard the dog* ***bark*** . . .

Part of Speech Tagging Example

Input	*John*	*broke*	*the*	*red*	*lamp*
Output	NOUN	VERB	DET	ADJ	NOUN

Function Description

D_f = {*a*, *aardvark*, *abacus*, *abalone*, ...} in a sentence

R_f = {NOUN, VERB, ADJ, ADV, DET, ...}

$f(\textit{bark})$ = NOUN? VERB?

... *the* ***bark*** *of the tree* ...

... *heard the dog* ***bark*** ...

Feature Representation

$$
\begin{aligned}
f([w_0 = \textit{bark}, w_{-1} = \textit{the}]) &= \text{NOUN} \\
f([w_0 = \textit{bark}, w_{-1} = \textit{dog}]) &= \text{VERB}
\end{aligned}
$$

Named Entity Recognition Exercise

Definition

A named entity recognition program find spans of words that are people, locations, organizations, etc.

Input Bill works for Microsoft Corporation

Output [$_{\text{PER}}$ Bill] works for [$_{\text{ORG}}$ Microsoft Corporation]

Exercise

Named entity recognition as supervised learning:

- Describe the function domain
- Describe the function range (keep it small!)
- Describe the feature space

One Named Entity Recognition Approach

Function Description

$D_f = \{a, aardvark, abacus, abalone, \ldots\}$ in a sentence

$R_f = \{\text{B-Per}, \text{I-Per}, \text{B-Org}, \text{I-Org}, \ldots, \text{O}\}$

$$
\begin{aligned}
f(\mathit{Bill}) &= \text{B-Per} \\
f(\mathit{works}) &= \text{O} \\
f(\mathit{for}) &= \text{O} \\
f(\mathit{Microsoft}) &= \text{B-Org} \\
f(\mathit{Corporation}) &= \text{I-Org}
\end{aligned}
$$

Typical Features

Word itself

Preceding label

Capitalization

First in sentence?

Evaluating Learning Algorithms

1. Train learning algorithm on examples E_{train}
2. Learning algorithm produces a hypothesis h
3. Test hypothesis on new examples E_{test} - Don't peek!

Train:

x_1	x_2	$f(x)$
1	a	*true*
0	b	*false*
0	c	*false*
0	b	*false*

Test:

x_1	x_2	$f(x)$
1	a	*true*
0	b	*false*
0	c	*false*
1	c	*true*

$$h_1(x) = \begin{cases} \mathit{true} & \text{if } x_1 = 1 \\ \mathit{false} & \text{otherwise} \end{cases}$$

$$h_2(x) = \begin{cases} \mathit{true} & \text{if } x_2 = a \\ \mathit{false} & \text{otherwise} \end{cases}$$

Performance: $\frac{4}{4} = 1.0$

Performance: $\frac{3}{4} = 0.75$

Evaluating Learning Algorithms

1. Train learning algorithm on examples E_{train}
2. Learning algorithm produces a hypothesis h
3. Test hypothesis on new examples E_{test} - Don't peek!

Train:

x_1	x_2	$f(x)$
1	a	*true*
0	b	*false*
0	c	*false*
0	b	*false*

Test:

x_1	x_2	$f(x)$
1	a	*true*
0	b	*false*
0	c	*false*
1	c	*true*

$$h_1(x) = \begin{cases} \textit{true} & \text{if } x_1 = 1 \\ \textit{false} & \text{otherwise} \end{cases}$$

$$h_2(x) = \begin{cases} \textit{true} & \text{if } x_2 = a \\ \textit{false} & \text{otherwise} \end{cases}$$

Evaluating Learning Algorithms

1. Train learning algorithm on examples E_{train}
2. Learning algorithm produces a hypothesis h
3. Test hypothesis on new examples E_{test} - Don't peek!

Train:

x_1	x_2	$f(x)$
1	a	*true*
0	b	*false*
0	c	*false*
0	b	*false*

Test:

x_1	x_2	$f(x)$
1	a	*true*
0	b	*false*
0	c	*false*
1	c	*true*

$$h_1(x) = \begin{cases} \mathit{true} & \text{if } x_1 = 1 \\ \mathit{false} & \text{otherwise} \end{cases}$$

$$h_2(x) = \begin{cases} \mathit{true} & \text{if } x_2 = a \\ \mathit{false} & \text{otherwise} \end{cases}$$

Performance: $\frac{4}{4} = 1.0$

Performance: $\frac{3}{4} = 0.75$

Evaluating Learning Algorithms

1. Train learning algorithm on examples E_{train}
2. Learning algorithm produces a hypothesis h
3. Test hypothesis on new examples E_{test} - Don't peek!

Train:

x_1	x_2	$f(x)$
1	a	*true*
0	b	*false*
0	c	*false*
0	b	*false*

Test:

x_1	x_2	$f(x)$
1	a	*true*
0	b	*false*
0	c	*false*
1	c	*true*

$$h_1(x) = \begin{cases} \mathit{true} & \text{if } x_1 = 1 \\ \mathit{false} & \text{otherwise} \end{cases}$$

Performance: $\frac{4}{4} = 1.0$

$$h_2(x) = \begin{cases} \mathit{true} & \text{if } x_2 = a \\ \mathit{false} & \text{otherwise} \end{cases}$$

Performance: $\frac{3}{4} = 0.75$

Evaluating Learning Algorithms

1. Train learning algorithm on examples E_{train}
2. Learning algorithm produces a hypothesis h
3. Test hypothesis on new examples E_{test} - Don't peek!

Train:

x_1	x_2	$f(x)$
1	a	*true*
0	b	*false*
0	c	*false*
0	b	*false*

Test:

x_1	x_2	$f(x)$
1	a	*true*
0	b	*false*
0	c	*false*
1	c	*true*

$$h_1(x) = \begin{cases} \mathit{true} & \text{if } x_1 = 1 \\ \mathit{false} & \text{otherwise} \end{cases}$$

$$h_2(x) = \begin{cases} \mathit{true} & \text{if } x_2 = a \\ \mathit{false} & \text{otherwise} \end{cases}$$

Performance: $\frac{4}{4} = 1.0$

Performance: $\frac{3}{4} = 0.75$

Evaluating Learning Algorithms

1. Train learning algorithm on examples E_{train}
2. Learning algorithm produces a hypothesis h
3. Test hypothesis on new examples E_{test} - Don't peek!

Train:

x_1	x_2	$f(x)$
1	a	*true*
0	b	*false*
0	c	*false*
0	b	*false*

Test:

x_1	x_2	$f(x)$
1	a	*true*
0	b	*false*
0	c	*false*
1	c	*true*

$$h_1(x) = \begin{cases} \mathit{true} & \text{if } x_1 = 1 \\ \mathit{false} & \text{otherwise} \end{cases}$$

Performance: $\frac{4}{4} = 1.0$

$$h_2(x) = \begin{cases} \mathit{true} & \text{if } x_2 = a \\ \mathit{false} & \text{otherwise} \end{cases}$$

Performance: $\frac{3}{4} = 0.75$

Evaluating Learning Algorithms

1. Train learning algorithm on examples E_{train}
2. Learning algorithm produces a hypothesis h
3. Test hypothesis on new examples E_{test} - Don't peek!

Train:

x_1	x_2	$f(x)$
1	a	*true*
0	b	*false*
0	c	*false*
0	b	*false*

Test:

x_1	x_2	$f(x)$
1	a	*true*
0	b	*false*
0	c	*false*
1	c	*true*

$$h_1(x) = \begin{cases} \mathit{true} & \text{if } x_1 = 1 \\ \mathit{false} & \text{otherwise} \end{cases}$$

Performance: $\frac{4}{4} = 1.0$

$$h_2(x) = \begin{cases} \mathit{true} & \text{if } x_2 = a \\ \mathit{false} & \text{otherwise} \end{cases}$$

Performance: $\frac{3}{4} = 0.75$

Evaluating Learning Algorithms

1. Train learning algorithm on examples E_{train}
2. Learning algorithm produces a hypothesis h
3. Test hypothesis on new examples E_{test} - Don't peek!

Train:

x_1	x_2	$f(x)$
1	a	*true*
0	b	*false*
0	c	*false*
0	b	*false*

Test:

x_1	x_2	$f(x)$
1	a	*true*
0	b	*false*
0	c	*false*
1	c	*true*

$$h_1(x) = \begin{cases} \textit{true} & \text{if } x_1 = 1 \\ \textit{false} & \text{otherwise} \end{cases}$$

Performance: $\frac{4}{4} = 1.0$

$$h_2(x) = \begin{cases} \textit{true} & \text{if } x_2 = a \\ \textit{false} & \text{otherwise} \end{cases}$$

Performance: $\frac{3}{4} = 0.75$

Train, Development and Test sets

Bad Idea

1. Evaluate on test set
2. Examine errors and adjust hypothesis
3. Goto 1

Tuning to your test set will overestimate model accuracy

Instead, split data into:

Train Used to train a hypothesis function

Dev Used to tune/adjust the hypothesis

Test Used to estimate hypothesis function's accuracy

Train, Development and Test sets

Bad Idea

1. Evaluate on test set
2. Examine errors and adjust hypothesis
3. Goto 1

Tuning to your test set will overestimate model accuracy

Instead, split data into:

- Train Used to train a hypothesis function
- Dev Used to tune/adjust the hypothesis
- Test Used to estimate hypothesis function's accuracy

Train, Development and Test sets

Bad Idea

1. Evaluate on test set
2. Examine errors and adjust hypothesis
3. Goto 1

Tuning to your test set will overestimate model accuracy

Instead, split data into:

Train Used to train a hypothesis function

Dev Used to tune/adjust the hypothesis

Test Used to estimate hypothesis function's accuracy

k-fold Cross Validation

Split data into k parts, then iteratively train and test:

Accuracy is average of the accuracies on each of the k folds

Warnings:

- Cross validation is not a substitution for a test set
- Cross validation can be used instead of train+dev

k-fold Cross Validation

Split data into k parts, then iteratively train and test:

Accuracy is average of the accuracies on each of the k folds

Warnings:

- Cross validation is not a substitution for a test set
- Cross validation can be used instead of train+dev

k-fold Cross Validation

Split data into k parts, then iteratively train and test:

Accuracy is average of the accuracies on each of the k folds

Warnings:

- Cross validation is not a substitution for a test set
- Cross validation can be used instead of train+dev

k-fold Cross Validation

Split data into k parts, then iteratively train and test:

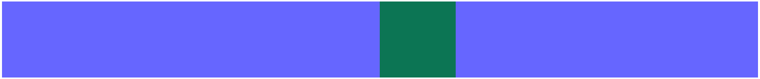
Accuracy is average of the accuracies on each of the k folds

Warnings:

- Cross validation is not a substitution for a test set
- Cross validation can be used instead of train+dev

k-fold Cross Validation

Split data into k parts, then iteratively train and test:



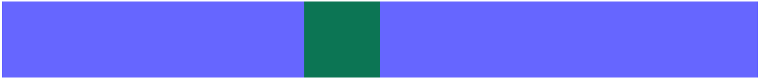
Accuracy is average of the accuracies on each of the k folds

Warnings:

- Cross validation is not a substitution for a test set
- Cross validation can be used instead of train+dev

k-fold Cross Validation

Split data into k parts, then iteratively train and test:



Accuracy is average of the accuracies on each of the k folds

Warnings:

- Cross validation is not a substitution for a test set
- Cross validation can be used instead of train+dev

k-fold Cross Validation

Split data into *k* parts, then iteratively train and test:

Accuracy is average of the accuracies on each of the *k* folds

Warnings:

- Cross validation is not a substitution for a test set
- Cross validation can be used instead of train+dev

k-fold Cross Validation

Split data into k parts, then iteratively train and test:

Accuracy is average of the accuracies on each of the k folds

Warnings:

- Cross validation is not a substitution for a test set
- Cross validation can be used instead of train+dev

k-fold Cross Validation

Split data into k parts, then iteratively train and test:

Accuracy is average of the accuracies on each of the k folds

Warnings:

- Cross validation is not a substitution for a test set
- Cross validation can be used instead of train+dev

k-fold Cross Validation

Split data into k parts, then iteratively train and test:

Accuracy is average of the accuracies on each of the k folds

Warnings:

- Cross validation is not a substitution for a test set
- Cross validation can be used instead of train+dev

k-fold Cross Validation

Split data into k parts, then iteratively train and test:

Accuracy is average of the accuracies on each of the k folds

Warnings:

- Cross validation is not a substitution for a test set
- Cross validation can be used instead of train+dev

k-fold Cross Validation

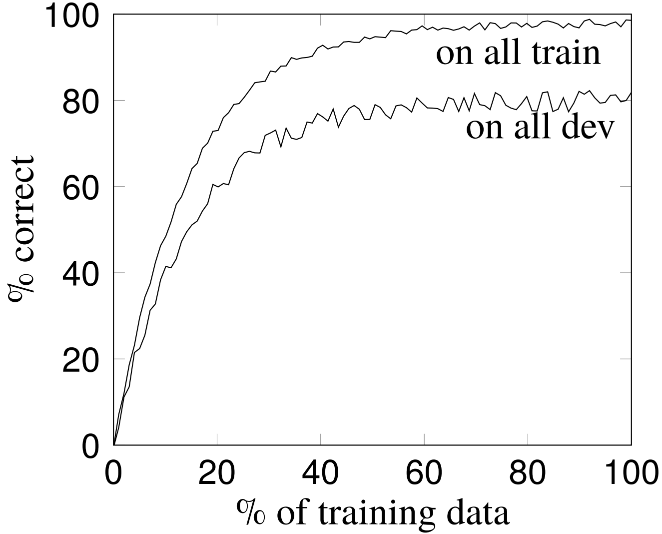
Split data into k parts, then iteratively train and test:

Accuracy is average of the accuracies on each of the k folds

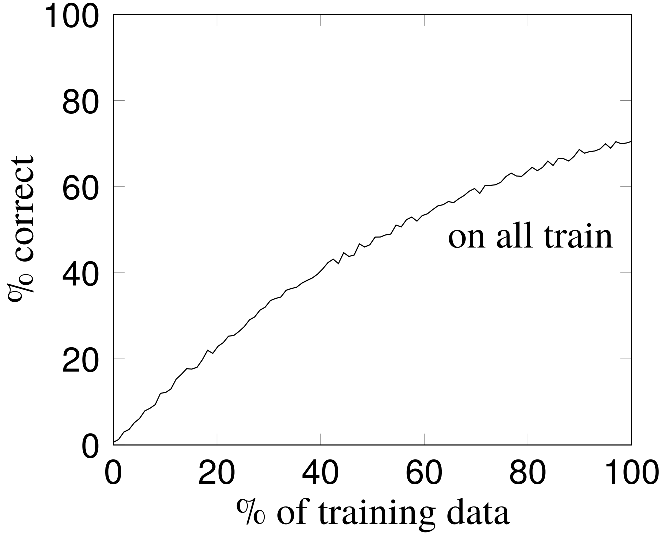
Warnings:

- Cross validation is not a substitution for a test set
- Cross validation can be used instead of train+dev

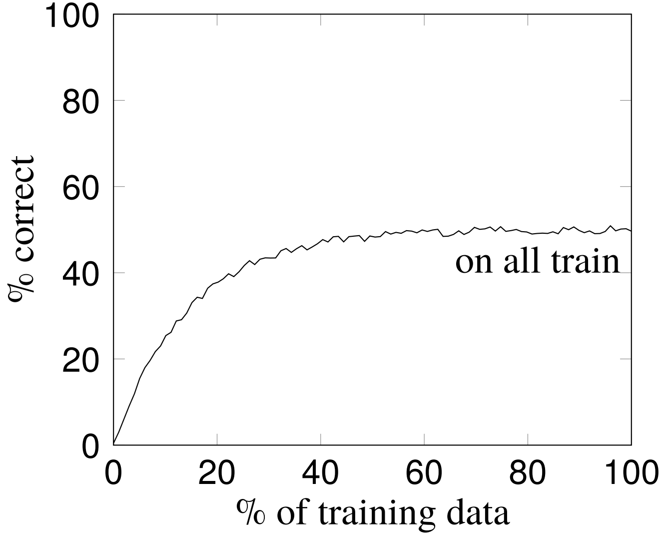
Learning Curves



Insufficient Training Data



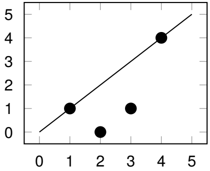
Underfitting



Addressing Underfitting

x_1	$f(x)$	$h(x)$
1	1	1
2	0	2
3	1	3
4	4	4

$h(x) = x_1$



More complex model

x_1	$f(x)$	$h(x)$
1	1	1
2	0	0
3	1	1
4	4	4

$h(x) = x_1^2 - 4x_1 + 4$

More features

x_1	x_2	$f(x)$	$h(x)$
1	1	1	1
2	4	0	0
3	9	1	1
4	16	4	4

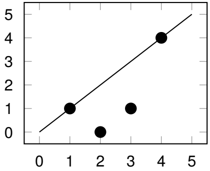
$h(x) = -4x_1 + x_2 + 4$

Addressing Underfitting

More complex model

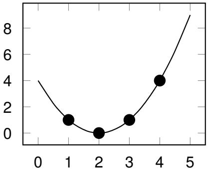
x_1	$f(x)$	$h(x)$
1	1	1
2	0	2
3	1	3
4	4	4

$h(x) = x_1$



x_1	$f(x)$	$h(x)$
1	1	1
2	0	0
3	1	1
4	4	4

$h(x) = x_1^2 - 4x_1 + 4$



More features

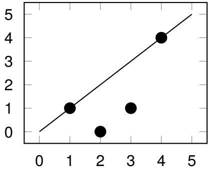
x_1	x_2	$f(x)$	$h(x)$
1	1	1	1
2	4	0	0
3	9	1	1
4	16	4	4

$h(x) = -4x_1 + x_2 + 4$

Addressing Underfitting

x_1	$f(x)$	$h(x)$
1	1	1
2	0	2
3	1	3
4	4	4

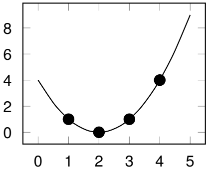
$h(x) = x_1$



More complex model

x_1	$f(x)$	$h(x)$
1	1	1
2	0	0
3	1	1
4	4	4

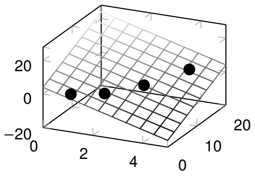
$h(x) = x_1^2 - 4x_1 + 4$



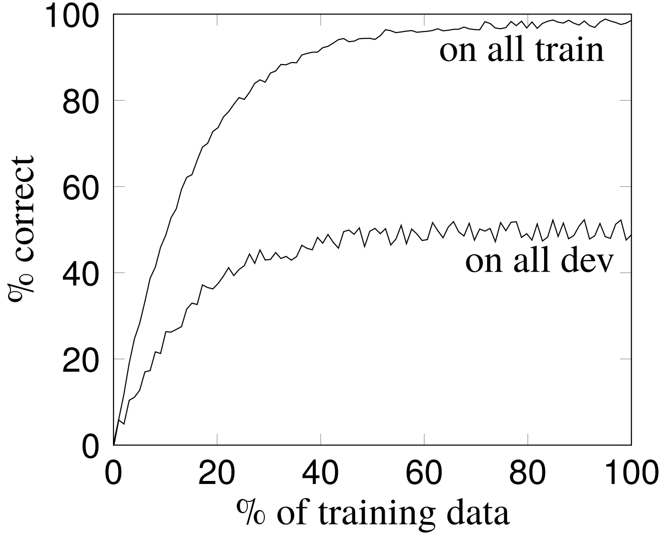
More features

x_1	x_2	$f(x)$	$h(x)$
1	1	1	1
2	4	0	0
3	9	1	1
4	16	4	4

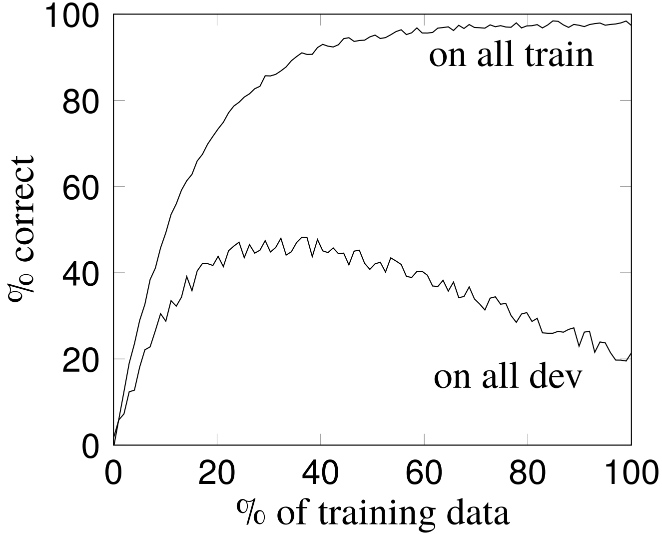
$h(x) = -4x_1 + x_2 + 4$



Overfitting



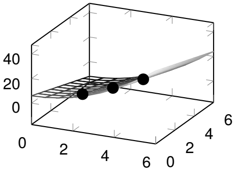
Overfitting



Addressing Overfitting

x_1	x_2	$f(x)$	$h(x)$
2	1	7	7
3	2	9	9
4	3	13	13

$h(x) = x_1^2 - 3x_2 + 6$



Simpler model

x_1	x_2	$f(x)$	$h(x)$
2	1	7	5
3	2	9	9
4	3	13	13
1	2	3	3
5	1	15	14

$h(x) = 3x_1 + x_2 - 2$

Fewer features

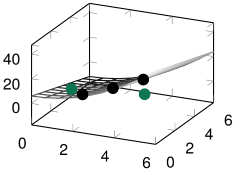
x_1	$f(x)$	$h(x)$
2	7	6
3	9	9
4	13	12
1	3	3
5	15	15

$h(x) = 3x_1$

Addressing Overfitting

x_1	x_2	$f(x)$	$h(x)$
2	1	7	7
3	2	9	9
4	3	13	13
1	2	3	1
5	1	15	28

$h(x) = x_1^2 - 3x_2 + 6$



Simpler model

x_1	x_2	$f(x)$	$h(x)$
2	1	7	5
3	2	9	9
4	3	13	13
1	2	3	3
5	1	15	14

$h(x) = 3x_1 + x_2 - 2$

Fewer features

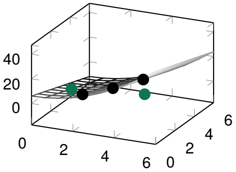
x_1	$f(x)$	$h(x)$
2	7	6
3	9	9
4	13	12
1	3	3
5	15	15

$h(x) = 3x_1$

Addressing Overfitting

x_1	x_2	$f(x)$	$h(x)$
2	1	7	7
3	2	9	9
4	3	13	13
1	2	3	1
5	1	15	28

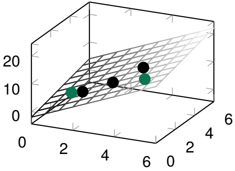
$h(x) = x_1^2 - 3x_2 + 6$



Simpler model

x_1	x_2	$f(x)$	$h(x)$
2	1	7	5
3	2	9	9
4	3	13	13
1	2	3	3
5	1	15	14

$h(x) = 3x_1 + x_2 - 2$



Fewer features

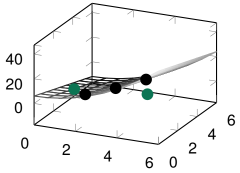
x_1	$f(x)$	$h(x)$
2	7	6
3	9	9
4	13	12
1	3	3
5	15	15

$h(x) = 3x_1$

Addressing Overfitting

x_1	x_2	$f(x)$	$h(x)$
2	1	7	7
3	2	9	9
4	3	13	13
1	2	3	1
5	1	15	28

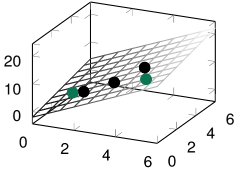
$h(x) = x_1^2 - 3x_2 + 6$



Simpler model

x_1	x_2	$f(x)$	$h(x)$
2	1	7	5
3	2	9	9
4	3	13	13
1	2	3	3
5	1	15	14

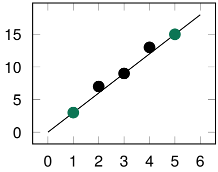
$h(x) = 3x_1 + x_2 - 2$



Fewer features

x_1	$f(x)$	$h(x)$
2	7	6
3	9	9
4	13	12
1	3	3
5	15	15

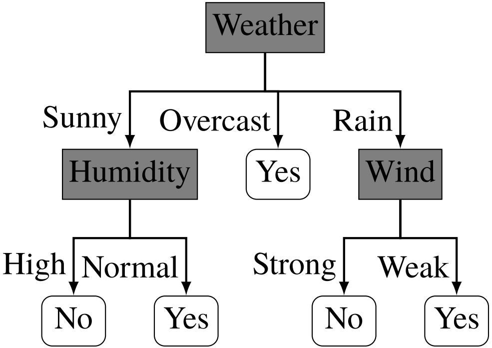
$h(x) = 3x_1$



Outline

Decision Trees

Should I play golf today?



Function

D_f = days

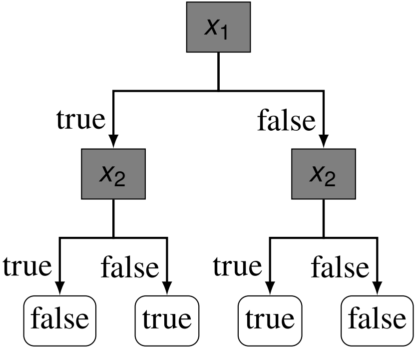
$R_f = \{Yes, No\}$

Features

- Weather
- Humidity
- Wind

Decision Trees as Functions

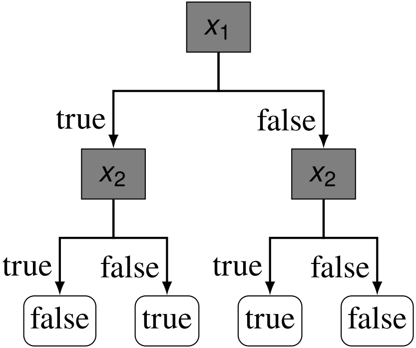
$f(x) = x_1 \; xor \; x_2$



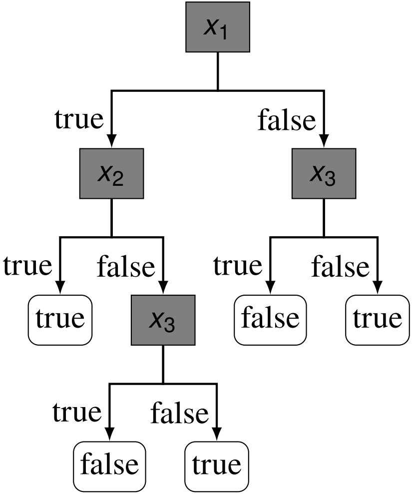
$f(x) = (x_1 \wedge x_2) \vee \neg x_3$

Decision Trees as Functions

$f(x) = x_1 \; xor \; x_2$



$f(x) = (x_1 \wedge x_2) \vee \neg x_3$



Learning Decision Trees

1. Select a feature x_i for the node
2. For each value of x_i create a child node
3. Sort training examples into child nodes
4. If examples are sorted perfectly, terminate
5. Else, repeat the process for each child node

x_1	x_2	$f(x)$
true	*true*	*true*
true	*false*	*false*
false	*true*	*true*
false	*false*	*true*

Learning Decision Trees

1. Select a feature x_i for the node
2. For each value of x_i create a child node
3. Sort training examples into child nodes
4. If examples are sorted perfectly, terminate
5. Else, repeat the process for each child node

x_1	x_2	$f(x)$
true	*true*	*true*
true	*false*	*false*
false	*true*	*true*
false	*false*	*true*

Learning Decision Trees

1. Select a feature x_i for the node
2. For each value of x_i create a child node
3. Sort training examples into child nodes
4. If examples are sorted perfectly, terminate
5. Else, repeat the process for each child node

3 true, 1 false

x_1	x_2	$f(x)$
true	*true*	*true*
true	*false*	*false*
false	*true*	*true*
false	*false*	*true*

Learning Decision Trees

1. Select a feature x_i for the node
2. For each value of x_i create a child node
3. Sort training examples into child nodes
4. If examples are sorted perfectly, terminate
5. Else, repeat the process for each child node

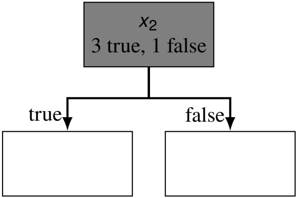
x_1	x_2	$f(x)$
true	*true*	*true*
true	*false*	*false*
false	*true*	*true*
false	*false*	*true*

x_2
3 true, 1 false

Learning Decision Trees

1. Select a feature x_i for the node
2. For each value of x_i create a child node
3. Sort training examples into child nodes
4. If examples are sorted perfectly, terminate
5. Else, repeat the process for each child node

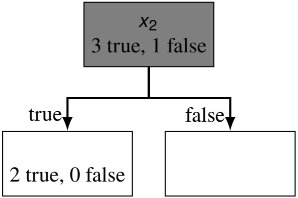
x_1	x_2	$f(x)$
true	*true*	*true*
true	*false*	*false*
false	*true*	*true*
false	*false*	*true*



Learning Decision Trees

1. Select a feature x_i for the node
2. For each value of x_i create a child node
3. Sort training examples into child nodes
4. If examples are sorted perfectly, terminate
5. Else, repeat the process for each child node

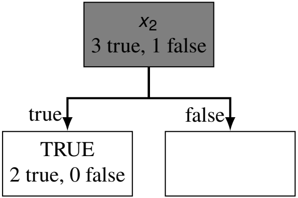
x_1	x_2	$f(x)$
true	*true*	*true*
true	*false*	*false*
false	*true*	*true*
false	*false*	*true*



Learning Decision Trees

1. Select a feature x_i for the node
2. For each value of x_i create a child node
3. Sort training examples into child nodes
4. If examples are sorted perfectly, terminate
5. Else, repeat the process for each child node

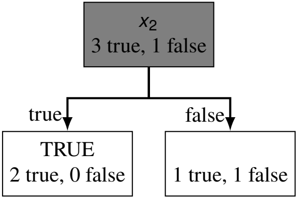
x_1	x_2	$f(x)$
true	*true*	*true*
true	*false*	*false*
false	*true*	*true*
false	*false*	*true*



Learning Decision Trees

1. Select a feature x_i for the node
2. For each value of x_i create a child node
3. Sort training examples into child nodes
4. If examples are sorted perfectly, terminate
5. Else, repeat the process for each child node

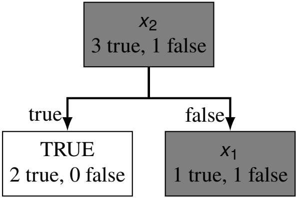
x_1	x_2	$f(x)$
true	*true*	*true*
true	*false*	*false*
false	*true*	*true*
false	*false*	*true*



Learning Decision Trees

1. Select a feature x_i for the node
2. For each value of x_i create a child node
3. Sort training examples into child nodes
4. If examples are sorted perfectly, terminate
5. Else, repeat the process for each child node

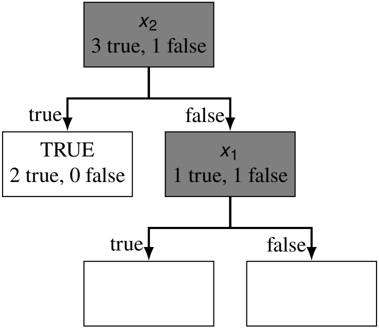
x_1	x_2	$f(x)$
true	*true*	*true*
true	*false*	*false*
false	*true*	*true*
false	*false*	*true*



Learning Decision Trees

1. Select a feature x_i for the node
2. For each value of x_i create a child node
3. Sort training examples into child nodes
4. If examples are sorted perfectly, terminate
5. Else, repeat the process for each child node

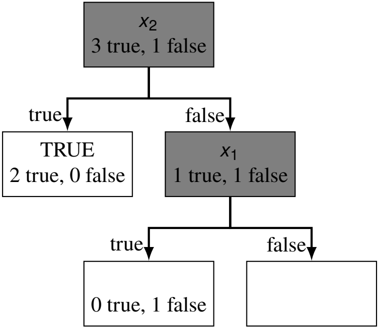
x_1	x_2	$f(x)$
true	*true*	*true*
true	*false*	*false*
false	*true*	*true*
false	*false*	*true*



Learning Decision Trees

1. Select a feature x_i for the node
2. For each value of x_i create a child node
3. Sort training examples into child nodes
4. If examples are sorted perfectly, terminate
5. Else, repeat the process for each child node

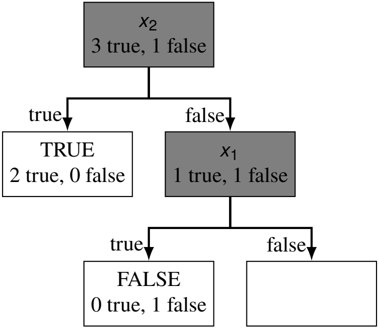
x_1	x_2	$f(x)$
true	*true*	*true*
true	*false*	*false*
false	*true*	*true*
false	*false*	*true*



Learning Decision Trees

1. Select a feature x_i for the node
2. For each value of x_i create a child node
3. Sort training examples into child nodes
4. If examples are sorted perfectly, terminate
5. Else, repeat the process for each child node

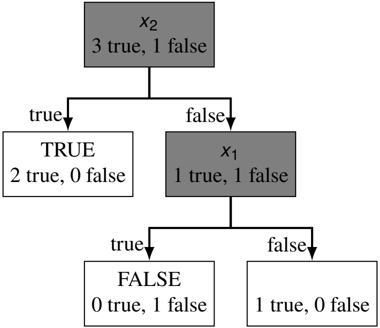
x_1	x_2	$f(x)$
true	*true*	*true*
true	*false*	*false*
false	*true*	*true*
false	*false*	*true*



Learning Decision Trees

1. Select a feature x_i for the node
2. For each value of x_i create a child node
3. Sort training examples into child nodes
4. If examples are sorted perfectly, terminate
5. Else, repeat the process for each child node

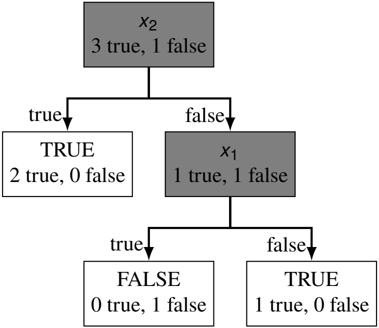
x_1	x_2	$f(x)$
true	*true*	*true*
true	*false*	*false*
false	*true*	*true*
false	*false*	*true*



Learning Decision Trees

1. Select a feature x_i for the node
2. For each value of x_i create a child node
3. Sort training examples into child nodes
4. If examples are sorted perfectly, terminate
5. Else, repeat the process for each child node

x_1	x_2	$f(x)$
true	*true*	*true*
true	*false*	*false*
false	*true*	*true*
false	*false*	*true*



Decision Tree Exercise

Build a decision tree for:

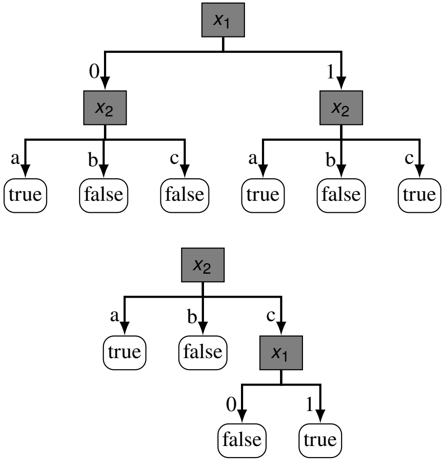
x_1	x_2	$f(x)$
0	a	*true*
0	b	*false*
1	c	*true*
0	c	*false*
1	b	*false*
1	a	*true*

Decision Tree Exercise

Build a decision tree for:

x_1	x_2	$f(x)$
0	a	*true*
0	b	*false*
1	c	*true*
0	c	*false*
1	b	*false*
1	a	*true*

Two possible solutions:



Selecting Features for Decision Trees

Feature Selection Order

- Different orders result in different trees
- "Good" features should be used before "poor" ones

What is a "good" feature?

One whose values predict the class labels

x_1	x_2	$f(x)$
a	0	*true*
a	1	*true*
b	0	*false*
b	1	*false*

x_1 is a good feature

x_2 is a poor feature

Selecting Features for Decision Trees

Feature Selection Order

- Different orders result in different trees
- “Good” features should be used before “poor” ones

What is a “good” feature?

One whose values predict the class labels

x_1	x_2	$f(x)$
a	0	*true*
a	1	*true*
b	0	*false*
b	1	*false*

x_1 is a good feature

x_2 is a poor feature

Selecting Features for Decision Trees

Feature Selection Order

- Different orders result in different trees
- "Good" features should be used before "poor" ones

What is a "good" feature?

One whose values predict the class labels

x_1	x_2	$f(x)$
a	0	*true*
a	1	*true*
b	0	*false*
b	1	*false*

x_1 is a good feature

x_2 is a poor feature

Selecting Features for Decision Trees

Feature Selection Order

- Different orders result in different trees
- “Good” features should be used before “poor” ones

What is a “good” feature?

One whose values predict the class labels

x_1	x_2	$f(x)$
a	0	*true*
a	1	*true*
b	0	*false*
b	1	*false*

x_1 is a good feature

x_2 is a poor feature

Information

Good features provide more information

Information can be quantified in terms of bits

Task: Encode abacabad using as few bits as possible

Simple Encoding:

a	00
b	01
c	10
d	11

abacabad → 16 bits

`0001001000010011`

Using Probability:

a	0
b	10
c	110
d	111

abacabad → 14 bits

`01001100100111`

More likely values can be encoded in fewer bits!

Information

Good features provide more information

Information can be quantified in terms of bits

Task: Encode `abacabad` using as few bits as possible

Simple Encoding:

a	00
b	01
c	10
d	11

abacabad → 16 bits

`0001001000010011`

Using Probability:

a	0
b	10
c	110
d	111

abacabad → 14 bits

`01001100100111`

More likely values can be encoded in fewer bits!

Information

Good features provide more information

Information can be quantified in terms of bits

Task: Encode `abacabad` using as few bits as possible

Simple Encoding:

a	00
b	01
c	10
d	11

`abacabad` $\rightarrow$ 16 bits

`0001001000010011`

Using Probability:

a	0
b	10
c	110
d	111

`abacabad` $\rightarrow$ 14 bits

`01001100100111`

More likely values can be encoded in fewer bits!

Information

Good features provide more information

Information can be quantified in terms of bits

Task: Encode `abacabad` using as few bits as possible

Simple Encoding:

a	00
b	01
c	10
d	11

`abacabad` → 16 bits

`0001001000010011`

Using Probability:

a	0
b	10
c	110
d	111

`abacabad` → 14 bits

`01001100100111`

More likely values can be encoded in fewer bits!

Information

Good features provide more information

Information can be quantified in terms of bits

Task: Encode `abacabad` using as few bits as possible

Simple Encoding:

a	00
b	01
c	10
d	11

`abacabad` → 16 bits

`0001001000010011`

Using Probability:

a	0
b	10
c	110
d	111

`abacabad` → 14 bits

`01001100100111`

More likely values can be encoded in fewer bits!

Information

Good features provide more information

Information can be quantified in terms of bits

Task: Encode abacabad using as few bits as possible

Simple Encoding:

a	00
b	01
c	10
d	11

abacabad → 16 bits

0001001000010011

Using Probability:

a	0
b	10
c	110
d	111

abacabad → 14 bits

01001100100111

More likely values can be encoded in fewer bits!

Good features provide more information

Information can be quantified in terms of bits

Task: Encode `abacabad` using as few bits as possible

Simple Encoding:

a	00
b	01
c	10
d	11

`abacabad` $\rightarrow$ 16 bits

`0001001000010011`

Using Probability:

a	0
b	10
c	110
d	111

`abacabad` $\rightarrow$ 14 bits

`01001100100111`

More likely values can be encoded in fewer bits!

Entropy

Definition

The entropy of a random variable X is:

$$H(X) = -\sum_{x \in X} P(x) \log_2 P(x)$$

Bit-based Interpretation

Smallest number of bits that can encode a stream of values from X's distribution

Intuitions

- High entropy $\rightarrow$ unpredictable distribution
- Low entropy $\rightarrow$ predictable distribution

Entropy

Definition

The entropy of a random variable X is:

$$H(X) = -\sum_{x \in X} P(x) \log_2 P(x)$$

Bit-based Interpretation

Smallest number of bits that can encode a stream of values from X's distribution

Intuitions

- High entropy $\rightarrow$ unpredictable distribution
- Low entropy $\rightarrow$ predictable distribution

Entropy

Definition

The entropy of a random variable X is:

$$H(X) = -\sum_{x \in X} P(x) \log_2 P(x)$$

Bit-based Interpretation

Smallest number of bits that can encode a stream of values from X's distribution

Intuitions

- High entropy $\rightarrow$ unpredictable distribution
- Low entropy $\rightarrow$ predictable distribution

Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

$$
\begin{aligned}
H(X) &= -\sum_{x \in X} P(x) \log_2 P(x) \\
&= -P(math) \log_2 P(math) \\
&\quad -P(history) \log_2 P(history) \\
&\quad -P(cs) \log_2 P(cs) \\
&= -\tfrac{1}{2} \log_2 \tfrac{1}{2} - \tfrac{1}{4} \log_2 \tfrac{1}{4} - \tfrac{1}{4} \log_2 \tfrac{1}{4} \\
&= -\tfrac{1}{2}(-1) - \tfrac{1}{4}(-2) - \tfrac{1}{4}(-2) \\
&= \tfrac{1}{2} + \tfrac{1}{2} + \tfrac{1}{2} \\
&= 1.5
\end{aligned}
$$

$$
H(Y) = 1.0
$$

Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

$$
\begin{aligned}
H(X) &= -\sum_{x \in X} P(x)\log_2 P(x) \\
&= -P(math)\log_2 P(math) \\
&\quad -P(history)\log_2 P(history) \\
&\quad -P(cs)\log_2 P(cs) \\
&= -\tfrac{1}{2}\log_2 \tfrac{1}{2} - \tfrac{1}{4}\log_2 \tfrac{1}{4} - \tfrac{1}{4}\log_2 \tfrac{1}{4} \\
&= -\tfrac{1}{2}(-1) - \tfrac{1}{4}(-2) - \tfrac{1}{4}(-2) \\
&= \tfrac{1}{2} + \tfrac{1}{2} + \tfrac{1}{2} \\
&= 1.5
\end{aligned}
$$

$$H(Y) = 1.0$$

Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

$$
\begin{aligned}
H(X) &= -\sum_{x \in X} P(x) \log_2 P(x) \\
&= -P(math) \log_2 P(math) \\
&\quad -P(history) \log_2 P(history) \\
&\quad -P(cs) \log_2 P(cs) \\
&= -\tfrac{1}{2} \log_2 \tfrac{1}{2} - \tfrac{1}{4} \log_2 \tfrac{1}{4} - \tfrac{1}{4} \log_2 \tfrac{1}{4} \\
&= -\tfrac{1}{2}(-1) - \tfrac{1}{4}(-2) - \tfrac{1}{4}(-2) \\
&= \tfrac{1}{2} + \tfrac{1}{2} + \tfrac{1}{2} \\
&= 1.5
\end{aligned}
$$

$$H(Y) = 1.0$$

Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

$$
\begin{aligned}
H(X) &= -\sum_{x \in X} P(x)\log_2 P(x) \\
&= -P(math)\log_2 P(math) \\
&\quad -P(history)\log_2 P(history) \\
&\quad -P(cs)\log_2 P(cs) \\
&= -\tfrac{1}{2}\log_2 \tfrac{1}{2} - \tfrac{1}{4}\log_2 \tfrac{1}{4} - \tfrac{1}{4}\log_2 \tfrac{1}{4} \\
&= -\tfrac{1}{2}(-1) - \tfrac{1}{4}(-2) - \tfrac{1}{4}(-2) \\
&= \tfrac{1}{2} + \tfrac{1}{2} + \tfrac{1}{2} \\
&= 1.5
\end{aligned}
$$

$$H(Y) = 1.0$$

Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

$$
\begin{aligned}
H(X) &= -\sum_{x \in X} P(x) \log_2 P(x) \\
&= -P(math) \log_2 P(math) \\
&\quad -P(history) \log_2 P(history) \\
&\quad -P(cs) \log_2 P(cs) \\
&= -\tfrac{1}{2} \log_2 \tfrac{1}{2} - \tfrac{1}{4} \log_2 \tfrac{1}{4} - \tfrac{1}{4} \log_2 \tfrac{1}{4} \\
&= -\tfrac{1}{2}(-1) - \tfrac{1}{4}(-2) - \tfrac{1}{4}(-2) \\
&= \tfrac{1}{2} + \tfrac{1}{2} + \tfrac{1}{2} \\
&= 1.5
\end{aligned}
$$

$$H(Y) = 1.0$$

Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

$$
\begin{aligned}
H(X) &= -\sum_{x \in X} P(x) \log_2 P(x) \\
&= -P(math) \log_2 P(math) \\
&\quad -P(history) \log_2 P(history) \\
&\quad -P(cs) \log_2 P(cs) \\
&= -\tfrac{1}{2} \log_2 \tfrac{1}{2} - \tfrac{1}{4} \log_2 \tfrac{1}{4} - \tfrac{1}{4} \log_2 \tfrac{1}{4} \\
&= -\tfrac{1}{2}(-1) - \tfrac{1}{4}(-2) - \tfrac{1}{4}(-2) \\
&= \tfrac{1}{2} + \tfrac{1}{2} + \tfrac{1}{2} \\
&= 1.5
\end{aligned}
$$

$$H(Y) = 1.0$$

Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

$$
\begin{aligned}
H(X) &= -\sum_{x \in X} P(x) \log_2 P(x) \\
&= -P(\mathit{math}) \log_2 P(\mathit{math}) \\
&\quad -P(\mathit{history}) \log_2 P(\mathit{history}) \\
&\quad -P(\mathit{cs}) \log_2 P(\mathit{cs}) \\
&= -\tfrac{1}{2} \log_2 \tfrac{1}{2} - \tfrac{1}{4} \log_2 \tfrac{1}{4} - \tfrac{1}{4} \log_2 \tfrac{1}{4} \\
&= -\tfrac{1}{2}(-1) - \tfrac{1}{4}(-2) - \tfrac{1}{4}(-2) \\
&= \tfrac{1}{2} + \tfrac{1}{2} + \tfrac{1}{2} \\
&= 1.5
\end{aligned}
$$

$$H(Y) = 1.0$$

Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

$$
\begin{aligned}
H(X) &= -\sum_{x \in X} P(x)\log_2 P(x) \\
&= -P(\mathit{math})\log_2 P(\mathit{math}) \\
&\quad -P(\mathit{history})\log_2 P(\mathit{history}) \\
&\quad -P(\mathit{cs})\log_2 P(\mathit{cs}) \\
&= -\tfrac{1}{2}\log_2 \tfrac{1}{2} - \tfrac{1}{4}\log_2 \tfrac{1}{4} - \tfrac{1}{4}\log_2 \tfrac{1}{4} \\
&= -\tfrac{1}{2}(-1) - \tfrac{1}{4}(-2) - \tfrac{1}{4}(-2) \\
&= \tfrac{1}{2} + \tfrac{1}{2} + \tfrac{1}{2} \\
&= 1.5
\end{aligned}
$$

$$H(Y) = 1.0$$

Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

$$\begin{aligned}
H(X) &= -\sum_{x \in X} P(x)\log_2 P(x) \\
&= -P(math)\log_2 P(math) \\
&\quad -P(history)\log_2 P(history) \\
&\quad -P(cs)\log_2 P(cs) \\
&= -\tfrac{1}{2}\log_2 \tfrac{1}{2} - \tfrac{1}{4}\log_2 \tfrac{1}{4} - \tfrac{1}{4}\log_2 \tfrac{1}{4} \\
&= -\tfrac{1}{2}(-1) - \tfrac{1}{4}(-2) - \tfrac{1}{4}(-2) \\
&= \tfrac{1}{2} + \tfrac{1}{2} + \tfrac{1}{2} \\
&= 1.5
\end{aligned}$$

$$H(Y) = 1.0$$

Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

$$
\begin{aligned}
H(X) &= -\sum_{x \in X} P(x) \log_2 P(x) \\
&= -P(math) \log_2 P(math) \\
&\quad -P(history) \log_2 P(history) \\
&\quad -P(cs) \log_2 P(cs) \\
&= -\tfrac{1}{2} \log_2 \tfrac{1}{2} - \tfrac{1}{4} \log_2 \tfrac{1}{4} - \tfrac{1}{4} \log_2 \tfrac{1}{4} \\
&= -\tfrac{1}{2}(-1) - \tfrac{1}{4}(-2) - \tfrac{1}{4}(-2) \\
&= \tfrac{1}{2} + \tfrac{1}{2} + \tfrac{1}{2} \\
&= 1.5
\end{aligned}
$$

$$H(Y) = 1.0$$

Specific Conditional Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Specific Conditional Entropy

$$H(Y|X=x) = -\sum_{y \in Y} P(y|x) \log_2 P(y|x)$$

$$\begin{aligned} H(Y|X=cs) &= -P(yes|cs)\log_2 P(yes|cs) \\ &\quad -P(no|cs)\log_2 P(no|cs) \\ &= -1\log_2 1 - 0\log_2 0 \\ &= -1\cdot 0 - 0\cdot\infty \\ &= 0 \end{aligned}$$

In other words, $X = cs$ is a great predictor of Y

Specific Conditional Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Specific Conditional Entropy

$$H(Y|X=x) = -\sum_{y \in Y} P(y|x) \log_2 P(y|x)$$

$$\begin{aligned} H(Y|X=cs) &= -P(yes|cs)\log_2 P(yes|cs) \\ &\quad - P(no|cs)\log_2 P(no|cs) \\ &= -1\log_2 1 - 0\log_2 0 \\ &= -1 \cdot 0 - 0 \cdot \infty \\ &= 0 \end{aligned}$$

In other words, $X = cs$ is a great predictor of Y

Specific Conditional Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Specific Conditional Entropy

$$H(Y|X=x) = -\sum_{y \in Y} P(y|x) \log_2 P(y|x)$$

$$\begin{aligned} H(Y|X=cs) = & -P(yes|cs)\log_2 P(yes|cs) \\ & -P(no|cs)\log_2 P(no|cs) \\ = & -1\log_2 1 - 0\log_2 0 \\ = & -1 \cdot 0 - 0 \cdot \infty \\ = & \ 0 \end{aligned}$$

In other words, $X = cs$ is a great predictor of Y

Specific Conditional Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Specific Conditional Entropy

$$H(Y|X=x) = -\sum_{y \in Y} P(y|x) \log_2 P(y|x)$$

$$\begin{aligned} H(Y|X=cs) &= -P(yes|cs)\log_2 P(yes|cs) \\ &\quad - P(no|cs)\log_2 P(no|cs) \\ &= -1\log_2 1 - 0\log_2 0 \\ &= -1\cdot 0 - 0\cdot\infty \\ &= 0 \end{aligned}$$

In other words, $X=cs$ is a great predictor of Y

Specific Conditional Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Specific Conditional Entropy

$$H(Y|X=x) = -\sum_{y\in Y} P(y|x)\log_2 P(y|x)$$

$$\begin{aligned} H(Y|X=cs) &= -P(yes|cs)\log_2 P(yes|cs) \\ &\quad -P(no|cs)\log_2 P(no|cs) \\ &= -1\log_2 1 - 0\log_2 0 \\ &= -1\cdot 0 - 0\cdot\infty \\ &= 0 \end{aligned}$$

In other words, $X=cs$ is a great predictor of Y

Specific Conditional Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Specific Conditional Entropy

$$H(Y|X=x) = -\sum_{y \in Y} P(y|x) \log_2 P(y|x)$$

$$\begin{aligned} H(Y|X=cs) &= -P(yes|cs)\log_2 P(yes|cs) \\ &\quad - P(no|cs)\log_2 P(no|cs) \\ &= -1\log_2 1 - 0\log_2 0 \\ &= -1\cdot 0 - 0\cdot\infty \\ &= 0 \end{aligned}$$

In other words, $X = cs$ is a great predictor of Y

Specific Conditional Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Specific Conditional Entropy

$$H(Y|X=x) = -\sum_{y \in Y} P(y|x) \log_2 P(y|x)$$

$$\begin{aligned} H(Y|X=cs) &= -P(yes|cs)\log_2 P(yes|cs) \\ &\quad -P(no|cs)\log_2 P(no|cs) \\ &= -1\log_2 1 - 0\log_2 0 \\ &= -1\cdot 0 - 0\cdot\infty \\ &= 0 \end{aligned}$$

In other words, $X = cs$ is a great predictor of Y

Specific Conditional Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Specific Conditional Entropy

$$H(Y|X=x) = -\sum_{y \in Y} P(y|x) \log_2 P(y|x)$$

$$\begin{aligned} H(Y|X=cs) &= -P(yes|cs) \log_2 P(yes|cs) \\ &\quad -P(no|cs) \log_2 P(no|cs) \\ &= -1 \log_2 1 - 0 \log_2 0 \\ &= -1 \cdot 0 - 0 \cdot \infty \\ &= 0 \end{aligned}$$

In other words, $X = cs$ is a great predictor of Y

Conditional Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Conditional Entropy

$$H(Y|X) = \sum_{x \in X} P(X=x)H(Y|X=x)$$

Given:

$$\begin{aligned} H(Y|X=math) &= 1 \\ H(Y|X=history) &= 0 \\ H(Y|X=cs) &= 0 \end{aligned}$$

$$\begin{aligned} H(Y|X) &= P(X=math)H(Y|X=math) + \\ &\quad P(X=history)H(Y|X=history) + \\ &\quad P(X=cs)H(Y|X=cs) \\ &= \tfrac{1}{2}\cdot 1 + \tfrac{1}{4}\cdot 0 + \tfrac{1}{4}\cdot 0 = \tfrac{1}{2} \end{aligned}$$

In other words, X is a good predictor of Y

Conditional Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Conditional Entropy

$$H(Y|X) = \sum_{x \in X} P(X=x) H(Y|X=x)$$

Given:

$$\begin{aligned} H(Y|X=math) &= 1 \\ H(Y|X=history) &= 0 \\ H(Y|X=cs) &= 0 \end{aligned}$$

$$\begin{aligned} H(Y|X) &= P(X=math)H(Y|X=math) + \\ &\quad P(X=history)H(Y|X=history) + \\ &\quad P(X=cs)H(Y|X=cs) \\ &= \tfrac{1}{2} \cdot 1 + \tfrac{1}{4} \cdot 0 + \tfrac{1}{4} \cdot 0 = \tfrac{1}{2} \end{aligned}$$

In other words, X is a good predictor of Y

Conditional Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Conditional Entropy

$$H(Y|X) = \sum_{x \in X} P(X=x)H(Y|X=x)$$

Given:

$$\begin{aligned} H(Y|X=math) &= 1 \\ H(Y|X=history) &= 0 \\ H(Y|X=cs) &= 0 \end{aligned}$$

$$\begin{aligned} H(Y|X) = \; & P(X=math)H(Y|X=math) + \\ & P(X=history)H(Y|X=history) + \\ & P(X=cs)H(Y|X=cs) \\ = \; & \tfrac{1}{2}\cdot 1 + \tfrac{1}{4}\cdot 0 + \tfrac{1}{4}\cdot 0 = \tfrac{1}{2} \end{aligned}$$

In other words, X is a good predictor of Y

Conditional Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Conditional Entropy

$$H(Y|X) = \sum_{x \in X} P(X=x) H(Y|X=x)$$

Given:

$$\begin{aligned} H(Y|X=math) &= 1 \\ H(Y|X=history) &= 0 \\ H(Y|X=cs) &= 0 \end{aligned}$$

$$\begin{aligned} H(Y|X) &= P(X=math)H(Y|X=math) + \\ &\quad P(X=history)H(Y|X=history) + \\ &\quad P(X=cs)H(Y|X=cs) \\ &= \tfrac{1}{2}\cdot 1 + \tfrac{1}{4}\cdot 0 + \tfrac{1}{4}\cdot 0 = \tfrac{1}{2} \end{aligned}$$

In other words, X is a good predictor of Y

Conditional Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Conditional Entropy

$$H(Y|X) = \sum_{x \in X} P(X=x) H(Y|X=x)$$

Given:

$$\begin{aligned} H(Y|X=math) &= 1 \\ H(Y|X=history) &= 0 \\ H(Y|X=cs) &= 0 \end{aligned}$$

$$\begin{aligned} H(Y|X) &= P(X=math)H(Y|X=math) + \\ &\quad P(X=history)H(Y|X=history) + \\ &\quad P(X=cs)H(Y|X=cs) \\ &= \tfrac{1}{2} \cdot 1 + \tfrac{1}{4} \cdot 0 + \tfrac{1}{4} \cdot 0 = \tfrac{1}{2} \end{aligned}$$

In other words, X is a good predictor of Y

Conditional Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Conditional Entropy

$$H(Y|X) = \sum_{x \in X} P(X=x) H(Y|X=x)$$

Given:

$$\begin{aligned} H(Y|X=math) &= 1 \\ H(Y|X=history) &= 0 \\ H(Y|X=cs) &= 0 \end{aligned}$$

$$\begin{aligned} H(Y|X) &= P(X=math)H(Y|X=math) + \\ &\quad P(X=history)H(Y|X=history) + \\ &\quad P(X=cs)H(Y|X=cs) \\ &= \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 0 + \frac{1}{4} \cdot 0 = \frac{1}{2} \end{aligned}$$

In other words, X is a good predictor of Y

Conditional Entropy

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Conditional Entropy

$$H(Y|X) = \sum_{x \in X} P(X=x)H(Y|X=x)$$

Given:

$$\begin{aligned} H(Y|X=math) &= 1 \\ H(Y|X=history) &= 0 \\ H(Y|X=cs) &= 0 \end{aligned}$$

$$\begin{aligned} H(Y|X) &= P(X=math)H(Y|X=math) + \\ &\quad P(X=history)H(Y|X=history) + \\ &\quad P(X=cs)H(Y|X=cs) \\ &= \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 0 + \frac{1}{4} \cdot 0 = \frac{1}{2} \end{aligned}$$

In other words, X is a good predictor of Y

Information Gain

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Information Gain

$IG(Y|X) = H(Y) - H(Y|X)$

Intuitive Explanation

How many bits would it save to know X?

$$H(Y|X) = 0.5$$
$$H(Y) = 1$$

$$IG(Y|X) = 1 - 0.5 = 0.5$$

Information Gain

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Information Gain

$IG(Y|X) = H(Y) - H(Y|X)$

Intuitive Explanation

How many bits would it save to know X?

$$H(Y|X) = 0.5$$
$$H(Y) = 1$$

$$IG(Y|X) = 1 - 0.5 = 0.5$$

Information Gain

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Information Gain

$IG(Y|X) = H(Y) - H(Y|X)$

Intuitive Explanation

How many bits would it save to know X?

$$
\begin{aligned}
H(Y|X) &= 0.5 \\
H(Y) &= 1
\end{aligned}
$$

$$IG(Y|X) = 1 - 0.5 = 0.5$$

Information Gain

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Information Gain

$IG(Y|X) = H(Y) - H(Y|X)$

Intuitive Explanation

How many bits would it save to know X?

$$\begin{aligned} H(Y|X) &= 0.5 \\ H(Y) &= 1 \end{aligned}$$

$$IG(Y|X) = 1 - 0.5 = 0.5$$

Information Gain

X	Y
math	yes
history	no
cs	yes
math	no
math	no
cs	yes
history	no
math	yes

Information Gain

$IG(Y|X) = H(Y) - H(Y|X)$

Intuitive Explanation

How many bits would it save to know X?

$$\begin{aligned} H(Y|X) &= 0.5 \\ H(Y) &= 1 \end{aligned}$$

$$IG(Y|X) = 1 - 0.5 = 0.5$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$H(Y) =$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$H(Y) = -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F})$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$\begin{aligned} H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F}) \\ &= -\tfrac{1}{2}\log_2 \tfrac{1}{2} - \tfrac{1}{2}\log_2 \tfrac{1}{2} \end{aligned}$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$\begin{aligned} H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F}) \\ &= -\tfrac{1}{2}\log_2 \tfrac{1}{2} - \tfrac{1}{2}\log_2 \tfrac{1}{2} = 1 \end{aligned}$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$
\begin{aligned}
H(Y) &= -P(\mathrm{T})\log_2 P(\mathrm{T}) - P(\mathrm{F})\log_2 P(\mathrm{F}) \\
&= -\tfrac{1}{2}\log_2 \tfrac{1}{2} - \tfrac{1}{2}\log_2 \tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &=
\end{aligned}
$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$
\begin{aligned}
H(Y) &= -P(\textsc{t})\log_2 P(\textsc{t}) - P(\textsc{f})\log_2 P(\textsc{f}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\textsc{t}|0)\log_2 P(\textsc{t}|0) -
\end{aligned}
$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$
\begin{aligned}
H(Y) &= -P(\text{T}) \log_2 P(\text{T}) - P(\text{F}) \log_2 P(\text{F}) \\
&= -\tfrac{1}{2} \log_2 \tfrac{1}{2} - \tfrac{1}{2} \log_2 \tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\text{T}|0) \log_2 P(\text{T}|0) - P(\text{F}|0) \log_2 P(\text{F}|0)
\end{aligned}
$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$\begin{aligned}
H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\text{T}|0)\log_2 P(\text{T}|0) - P(\text{F}|0)\log_2 P(\text{F}|0) \\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} -
\end{aligned}$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$\begin{aligned}
H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\text{T}|0)\log_2 P(\text{T}|0) - P(\text{F}|0)\log_2 P(\text{F}|0) \\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3}
\end{aligned}$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$
\begin{aligned}
H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\text{T}|0)\log_2 P(\text{T}|0) - P(\text{F}|0)\log_2 P(\text{F}|0) \\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3} = 0.92
\end{aligned}
$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$
\begin{aligned}
H(Y) &= -P(\mathrm{T})\log_2 P(\mathrm{T}) - P(\mathrm{F})\log_2 P(\mathrm{F}) \\
&= -\tfrac{1}{2}\log_2 \tfrac{1}{2} - \tfrac{1}{2}\log_2 \tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\mathrm{T}|0)\log_2 P(\mathrm{T}|0) - P(\mathrm{F}|0)\log_2 P(\mathrm{F}|0) \\
&= -\tfrac{1}{3}\log_2 \tfrac{1}{3} - \tfrac{2}{3}\log_2 \tfrac{2}{3} = 0.92 \\
H(Y|X_1{=}1) &=
\end{aligned}
$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$
\begin{aligned}
H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\text{T}|0)\log_2 P(\text{T}|0) - P(\text{F}|0)\log_2 P(\text{F}|0) \\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3} = 0.92 \\
H(Y|X_1{=}1) &= \ldots = 0.92
\end{aligned}
$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$
\begin{aligned}
H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F})\\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1\\
H(Y|X_1{=}0) &= -P(\text{T}|0)\log_2 P(\text{T}|0) - P(\text{F}|0)\log_2 P(\text{F}|0)\\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3} = 0.92\\
H(Y|X_1{=}1) &= \ldots = 0.92\\
H(Y|X_1) &=
\end{aligned}
$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$
\begin{aligned}
H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\text{T}|0)\log_2 P(\text{T}|0) - P(\text{F}|0)\log_2 P(\text{F}|0) \\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3} = 0.92 \\
H(Y|X_1{=}1) &= \ldots = 0.92 \\
H(Y|X_1) &= P(X_1{=}0)H(Y|X_1{=}0) + P(X_1{=}1)H(Y|X_1{=}1)
\end{aligned}
$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$\begin{aligned}
H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\text{T}|0)\log_2 P(\text{T}|0) - P(\text{F}|0)\log_2 P(\text{F}|0) \\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3} = 0.92 \\
H(Y|X_1{=}1) &= \ldots = 0.92 \\
H(Y|X_1) &= P(X_1{=}0)H(Y|X_1{=}0) + P(X_1{=}1)H(Y|X_1{=}1) \\
&= 0.5 \cdot 0.92 +
\end{aligned}$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$\begin{aligned}
H(Y) &= -P(\textsc{t})\log_2 P(\textsc{t}) - P(\textsc{f})\log_2 P(\textsc{f}) \\
&= -\tfrac{1}{2}\log_2 \tfrac{1}{2} - \tfrac{1}{2}\log_2 \tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\textsc{t}|0)\log_2 P(\textsc{t}|0) - P(\textsc{f}|0)\log_2 P(\textsc{f}|0) \\
&= -\tfrac{1}{3}\log_2 \tfrac{1}{3} - \tfrac{2}{3}\log_2 \tfrac{2}{3} = 0.92 \\
H(Y|X_1{=}1) &= \ldots = 0.92 \\
H(Y|X_1) &= P(X_1{=}0)H(Y|X_1{=}0) + P(X_1{=}1)H(Y|X_1{=}1) \\
&= 0.5 \cdot 0.92 + 0.5 \cdot 0.92
\end{aligned}$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$
\begin{aligned}
H(Y) &= -P(\textsc{t})\log_2 P(\textsc{t}) - P(\textsc{f})\log_2 P(\textsc{f}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\textsc{t}|0)\log_2 P(\textsc{t}|0) - P(\textsc{f}|0)\log_2 P(\textsc{f}|0) \\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3} = 0.92 \\
H(Y|X_1{=}1) &= \ldots = 0.92 \\
H(Y|X_1) &= P(X_1{=}0)H(Y|X_1{=}0) + P(X_1{=}1)H(Y|X_1{=}1) \\
&= 0.5 \cdot 0.92 + 0.5 \cdot 0.92 = 0.92
\end{aligned}
$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$
\begin{aligned}
H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\text{T}|0)\log_2 P(\text{T}|0) - P(\text{F}|0)\log_2 P(\text{F}|0) \\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3} = 0.92 \\
H(Y|X_1{=}1) &= \ldots = 0.92 \\
H(Y|X_1) &= P(X_1{=}0)H(Y|X_1{=}0) + P(X_1{=}1)H(Y|X_1{=}1) \\
&= 0.5\cdot 0.92 + 0.5\cdot 0.92 = 0.92 \\
IG(Y|X_1) &=
\end{aligned}
$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$\begin{aligned}
H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\text{T}|0)\log_2 P(\text{T}|0) - P(\text{F}|0)\log_2 P(\text{F}|0) \\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3} = 0.92 \\
H(Y|X_1{=}1) &= \ldots = 0.92 \\
H(Y|X_1) &= P(X_1{=}0)H(Y|X_1{=}0) + P(X_1{=}1)H(Y|X_1{=}1) \\
&= 0.5\cdot 0.92 + 0.5\cdot 0.92 = 0.92 \\
IG(Y|X_1) &= H(Y) - H(Y|X_1)
\end{aligned}$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$
\begin{aligned}
H(Y) &= -P(\text{T}) \log_2 P(\text{T}) - P(\text{F}) \log_2 P(\text{F}) \\
&= -\tfrac{1}{2} \log_2 \tfrac{1}{2} - \tfrac{1}{2} \log_2 \tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\text{T}|0) \log_2 P(\text{T}|0) - P(\text{F}|0) \log_2 P(\text{F}|0) \\
&= -\tfrac{1}{3} \log_2 \tfrac{1}{3} - \tfrac{2}{3} \log_2 \tfrac{2}{3} = 0.92 \\
H(Y|X_1{=}1) &= \ldots = 0.92 \\
H(Y|X_1) &= P(X_1{=}0)H(Y|X_1{=}0) + P(X_1{=}1)H(Y|X_1{=}1) \\
&= 0.5 \cdot 0.92 + 0.5 \cdot 0.92 = 0.92 \\
IG(Y|X_1) &= H(Y) - H(Y|X_1) = 1 - 0.92
\end{aligned}
$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$\begin{aligned}
H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\text{T}|0)\log_2 P(\text{T}|0) - P(\text{F}|0)\log_2 P(\text{F}|0) \\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3} = 0.92 \\
H(Y|X_1{=}1) &= \ldots = 0.92 \\
H(Y|X_1) &= P(X_1{=}0)H(Y|X_1{=}0) + P(X_1{=}1)H(Y|X_1{=}1) \\
&= 0.5 \cdot 0.92 + 0.5 \cdot 0.92 = 0.92 \\
IG(Y|X_1) &= H(Y) - H(Y|X_1) = 1 - 0.92 = 0.08
\end{aligned}$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$
\begin{aligned}
H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\text{T}|0)\log_2 P(\text{T}|0) - P(\text{F}|0)\log_2 P(\text{F}|0) \\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3} = 0.92 \\
H(Y|X_1{=}1) &= \ldots = 0.92 \\
H(Y|X_1) &= P(X_1{=}0)H(Y|X_1{=}0) + P(X_1{=}1)H(Y|X_1{=}1) \\
&= 0.5\cdot 0.92 + 0.5\cdot 0.92 = 0.92 \\
IG(Y|X_1) &= H(Y) - H(Y|X_1) = 1 - 0.92 = 0.08 \\
IG(Y|X_2) &=
\end{aligned}
$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$
\begin{aligned}
H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\text{T}|0)\log_2 P(\text{T}|0) - P(\text{F}|0)\log_2 P(\text{F}|0) \\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3} = 0.92 \\
H(Y|X_1{=}1) &= \ldots = 0.92 \\
H(Y|X_1) &= P(X_1{=}0)H(Y|X_1{=}0) + P(X_1{=}1)H(Y|X_1{=}1) \\
&= 0.5\cdot 0.92 + 0.5\cdot 0.92 = 0.92 \\
IG(Y|X_1) &= H(Y) - H(Y|X_1) = 1 - 0.92 = 0.08 \\
IG(Y|X_2) &= \text{Your turn!}
\end{aligned}
$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$\begin{aligned}
H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\text{T}|0)\log_2 P(\text{T}|0) - P(\text{F}|0)\log_2 P(\text{F}|0) \\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3} = 0.92 \\
H(Y|X_1{=}1) &= \ldots = 0.92 \\
H(Y|X_1) &= P(X_1{=}0)H(Y|X_1{=}0) + P(X_1{=}1)H(Y|X_1{=}1) \\
&= 0.5\cdot 0.92 + 0.5\cdot 0.92 = 0.92 \\
IG(Y|X_1) &= H(Y) - H(Y|X_1) = 1 - 0.92 = 0.08 \\
IG(Y|X_2) &= \text{Your turn!} = 0.67
\end{aligned}$$

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$
\begin{aligned}
H(Y) &= -P(\textsc{t})\log_2 P(\textsc{t}) - P(\textsc{f})\log_2 P(\textsc{f}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\textsc{t}|0)\log_2 P(\textsc{t}|0) - P(\textsc{f}|0)\log_2 P(\textsc{f}|0) \\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3} = 0.92 \\
H(Y|X_1{=}1) &= \ldots = 0.92 \\
H(Y|X_1) &= P(X_1{=}0)H(Y|X_1{=}0) + P(X_1{=}1)H(Y|X_1{=}1) \\
&= 0.5\cdot 0.92 + 0.5\cdot 0.92 = 0.92 \\
IG(Y|X_1) &= H(Y) - H(Y|X_1) = 1 - 0.92 = 0.08 \\
IG(Y|X_2) &= \text{Your turn!} \quad = 0.67
\end{aligned}
$$

So X_2 is a much better feature to split on

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$
\begin{aligned}
H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\text{T}|0)\log_2 P(\text{T}|0) - P(\text{F}|0)\log_2 P(\text{F}|0) \\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3} = 0.92 \\
H(Y|X_1{=}1) &= \ldots = 0.92 \\
H(Y|X_1) &= P(X_1{=}0)H(Y|X_1{=}0) + P(X_1{=}1)H(Y|X_1{=}1) \\
&= 0.5\cdot 0.92 + 0.5\cdot 0.92 = 0.92 \\
IG(Y|X_1) &= H(Y) - H(Y|X_1) = 1 - 0.92 = 0.08 \\
IG(Y|X_2) &= \text{Your turn!} = 0.67
\end{aligned}
$$

So X_2 is a much better feature to split on

Repeat at each node

Decision Trees and Information Gain

Select the feature with the highest information gain:

X_1	X_2	Y
0	a	T
0	b	F
1	c	T
0	c	F
1	b	F
1	a	T

$$
\begin{aligned}
H(Y) &= -P(\text{T})\log_2 P(\text{T}) - P(\text{F})\log_2 P(\text{F}) \\
&= -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1 \\
H(Y|X_1{=}0) &= -P(\text{T}|0)\log_2 P(\text{T}|0) - P(\text{F}|0)\log_2 P(\text{F}|0) \\
&= -\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3} = 0.92 \\
H(Y|X_1{=}1) &= \ldots = 0.92 \\
H(Y|X_1) &= P(X_1{=}0)H(Y|X_1{=}0) + P(X_1{=}1)H(Y|X_1{=}1) \\
&= 0.5 \cdot 0.92 + 0.5 \cdot 0.92 = 0.92 \\
IG(Y|X_1) &= H(Y) - H(Y|X_1) = 1 - 0.92 = 0.08 \\
IG(Y|X_2) &= \text{Your turn!} \quad = 0.67
\end{aligned}
$$

So X_2 is a much better feature to split on

Repeat at each node; different examples⇒different entropy

Random Forests

Algorithm:

1. Generate a bootstrap sample of n examples
2. For each node, select k features at random
3. Split on the feature with the highest information gain
4. Repeat steps 1, 2 and 3 to generate n decision trees
5. Classify by taking the majority vote

Why random forests?

- Combining independent classifiers $\Rightarrow$ better model
- Partially random trees should be more independent

Random Forests

Algorithm:

1. Generate a bootstrap sample of n examples
2. For each node, select k features at random
3. Split on the feature with the highest information gain
4. Repeat steps 1, 2 and 3 to generate n decision trees
5. Classify by taking the majority vote

Why random forests?

- Combining independent classifiers $\Rightarrow$ better model
- Partially random trees should be more independent

Random Forests

Algorithm:

1. Generate a bootstrap sample of n examples
2. For each node, select k features at random
3. Split on the feature with the highest information gain
4. Repeat steps 1, 2 and 3 to generate n decision trees
5. Classify by taking the majority vote

Why random forests?

- Combining independent classifiers $\Rightarrow$ better model
- Partially random trees should be more independent

Random Forest Example

x_0	x_1	x_2	$f(x)$
0	0	0	F
0	0	1	F
0	1	1	F
1	0	1	F
1	0	0	T
1	1	1	T

Bootstrap sample:

$[0, 5, 0, 1, 5, 4]$

Random feature subsets:

$[1, 2]$, $[2, 0]$

Random Forest Example

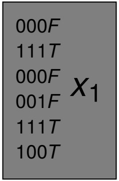
x_0	x_1	x_2	$f(x)$
0	0	0	F
0	0	1	F
0	1	1	F
1	0	1	F
1	0	0	T
1	1	1	T

000F
111T
000F
001F
111T
100T

Bootstrap sample:
$[0, 5, 0, 1, 5, 4]$
Random feature subsets:
$[1, 2]$, $[2, 0]$

Random Forest Example

x_0	x_1	x_2	$f(x)$
0	0	0	F
0	0	1	F
0	1	1	F
1	0	1	F
1	0	0	T
1	1	1	T

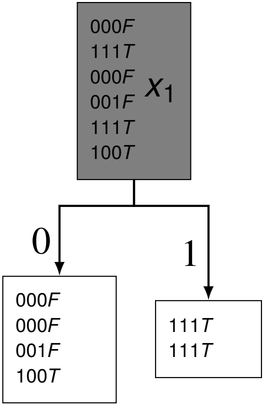


Bootstrap sample:
[0, 5, 0, 1, 5, 4]

Random feature subsets:
[1, 2], [2, 0]

Random Forest Example

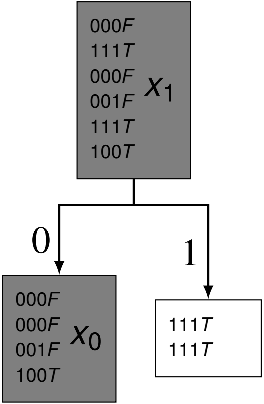
x_0	x_1	x_2	$f(x)$
0	0	0	F
0	0	1	F
0	1	1	F
1	0	1	F
1	0	0	T
1	1	1	T



Bootstrap sample:
[0, 5, 0, 1, 5, 4]
Random feature subsets:
[1, 2], [2, 0]

Random Forest Example

x_0	x_1	x_2	$f(x)$
0	0	0	F
0	0	1	F
0	1	1	F
1	0	1	F
1	0	0	T
1	1	1	T



Bootstrap sample:

[0, 5, 0, 1, 5, 4]

Random feature subsets:

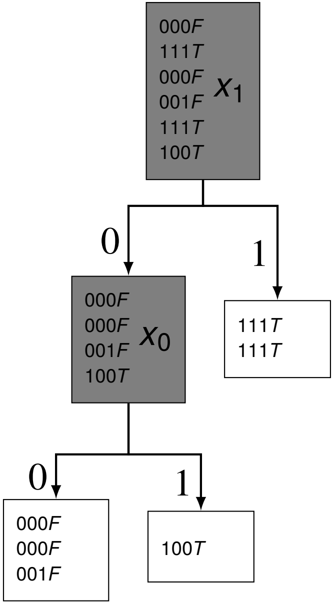
[1, 2], [2, 0]

Random Forest Example

x_0	x_1	x_2	$f(x)$
0	0	0	F
0	0	1	F
0	1	1	F
1	0	1	F
1	0	0	T
1	1	1	T

Bootstrap sample:
$[0, 5, 0, 1, 5, 4]$

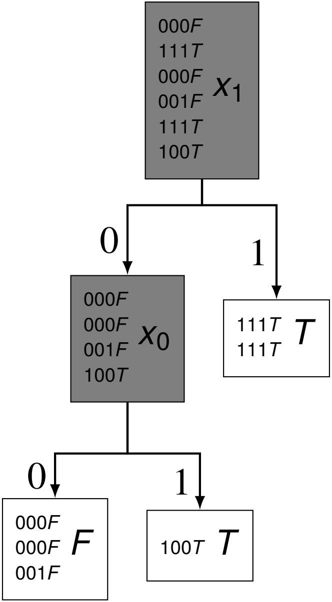
Random feature subsets:
$[1, 2], [2, 0]$



Random Forest Example

x_0	x_1	x_2	$f(x)$
0	0	0	F
0	0	1	F
0	1	1	F
1	0	1	F
1	0	0	T
1	1	1	T

Bootstrap sample:
  [0, 5, 0, 1, 5, 4]
Random feature subsets:
  [1, 2], [2, 0]



Random Forest Properties

Disadvantages

- Hard to estimate complexity due to random factor
- k (# of features) must be determined experimentally

Advantages

- State of the art performance on many datasets
- Relatively simple to implement
- Expected error can be determined while training
  - Bootstrap sample leaves out about $\frac{1}{3}$ of examples
  - Use these out-of-bag examples to test individual trees
  - Collect votes from all trees to get overall accuracy

Random Forest Properties

Disadvantages

- Hard to estimate complexity due to random factor
- k (# of features) must be determined experimentally

Advantages

- State of the art performance on many datasets
- Relatively simple to implement
- Expected error can be determined while training
  - Bootstrap sample leaves out about $\frac{1}{3}$ of examples
  - Use these out-of-bag examples to test individual trees
  - Collect votes from all trees to get overall accuracy

Random Forest Properties

Disadvantages

- Hard to estimate complexity due to random factor
- k (# of features) must be determined experimentally

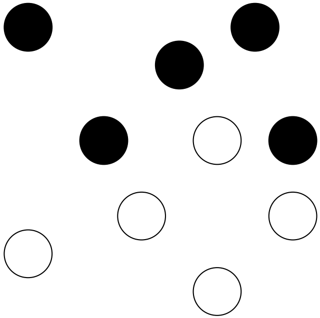
Advantages

- State of the art performance on many datasets
- Relatively simple to implement
- Expected error can be determined while training
  - Bootstrap sample leaves out about $\frac{1}{3}$ of examples
  - Use these out-of-bag examples to test individual trees
  - Collect votes from all trees to get overall accuracy

k-Nearest Neighbor Classifiers

To classify a new example, p

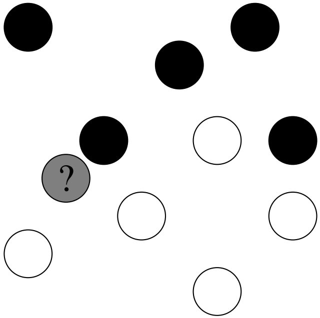
- Find the k training examples closest to p
- Classify p with the most common $f(x)$



k-Nearest Neighbor Classifiers

To classify a new example, p

- Find the k training examples closest to p
- Classify p with the most common $f(x)$

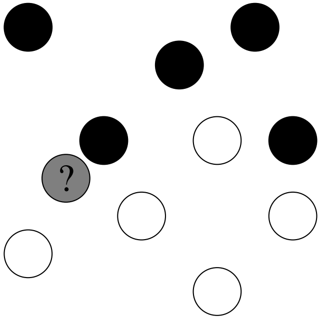


k-Nearest Neighbor Classifiers

To classify a new example, p

- Find the k training examples closest to p
- Classify p with the most common $f(x)$

$$k = 1$$

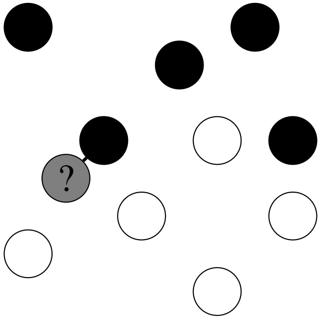


k-Nearest Neighbor Classifiers

To classify a new example, p

- Find the k training examples closest to p
- Classify p with the most common $f(x)$

$$k = 1$$

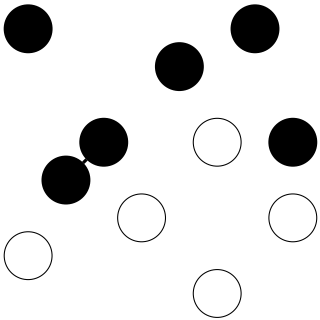


k-Nearest Neighbor Classifiers

To classify a new example, p

- Find the k training examples closest to p
- Classify p with the most common $f(x)$

$k = 1$

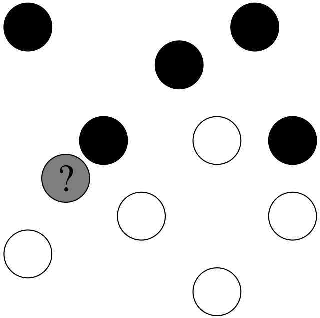


k-Nearest Neighbor Classifiers

To classify a new example, p

- Find the k training examples closest to p
- Classify p with the most common $f(x)$

$k = 3$

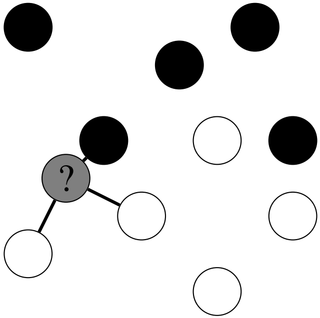


k-Nearest Neighbor Classifiers

To classify a new example, p

- Find the k training examples closest to p
- Classify p with the most common $f(x)$

$k = 3$

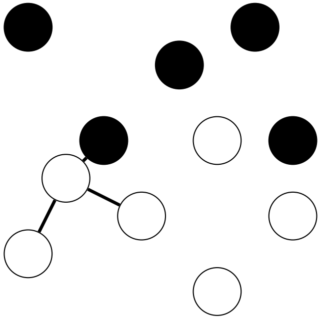


k-Nearest Neighbor Classifiers

To classify a new example, p

- Find the k training examples closest to p
- Classify p with the most common $f(x)$

$k = 3$



Distance Metrics

Euclidean Distance

$$\sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

close 0

far ∞

Cosine Similarity

$$\frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

close 1

far -1

Distance Metrics

Euclidean Distance

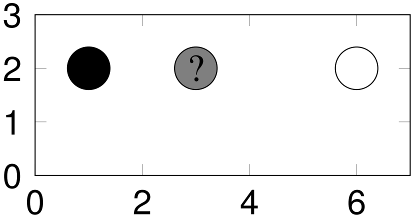
$$\sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

close	0
far	∞

Cosine Similarity

$$\frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

close	1
far	-1



Distance Metrics

Euclidean Distance

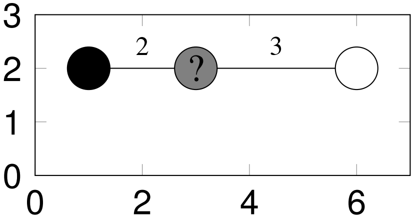
$$\sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

close 0
far ∞

Cosine Similarity

$$\frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

close 1
far -1



Distance Metrics

Euclidean Distance

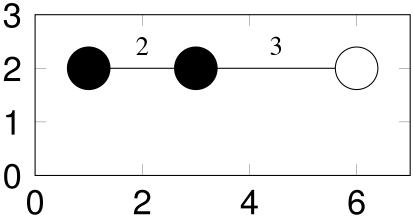
$$\sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

close 0
far ∞

Cosine Similarity

$$\frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

close 1
far -1



Distance Metrics

Euclidean Distance

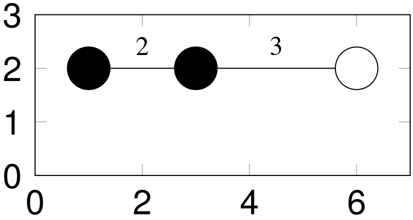
$$\sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

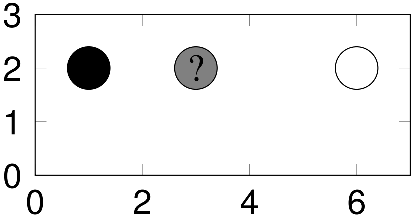
close	0
far	∞

Cosine Similarity

$$\frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

close	1
far	-1





Distance Metrics

Euclidean Distance

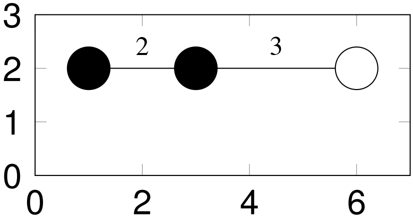
$$\sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

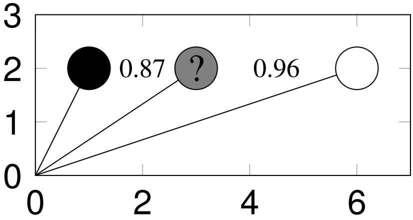
close 0
far ∞

Cosine Similarity

$$\frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

close 1
far -1





Distance Metrics

Euclidean Distance

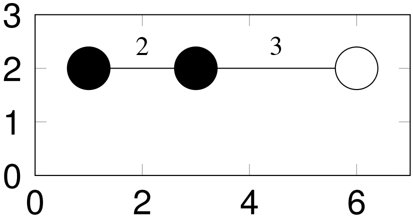
$$\sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

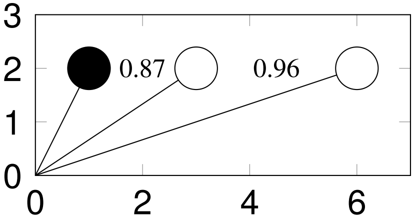
close 0
far ∞

Cosine Similarity

$$\frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

close 1
far -1





k-Nearest Neighbor Exercise

Training data:

x_1	x_2	$f(x)$
-1	1	A
0	1	A
0	2	A
2	2	B
3	2	B
3	3	B

Classify $x = [1, 1]$ with 3-NN

Euclidean? Cosine?

Euclidean Distance

$$\sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

close 0

far ∞

Cosine Similarity

$$\frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

close 1

far -1

k-Nearest Neighbor Exercise

Training data:

x_1	x_2	$f(x)$
-1	1	A
0	1	A
0	2	A
2	2	B
3	2	B
3	3	B

Classify $x = [1, 1]$ with 3-NN

Euclidean? A

0	1	A
0	2	A
2	2	B

Cosine? B

2	2	B
3	2	B
3	3	B

Euclidean Distance

$$\sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

close 0

far ∞

Cosine Similarity

$$\frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

close 1

far -1

k-Nearest Neighbor Properties

Theoretical Properties

- Given enough data and the right k, minimizes error
- Able to approximate many kinds of functions

Empirical Issues

- Simple to implement given a distance function
- Large training data means long search times
- Not always clear what distance function to use

Linear Models

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

Where:

- $x_1, x_2, x_3, \ldots$ are the features of x
- θ are feature weights

Learning linear models:

- Select parameters (weights) to minimize *loss*

Linear Models

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

Where:

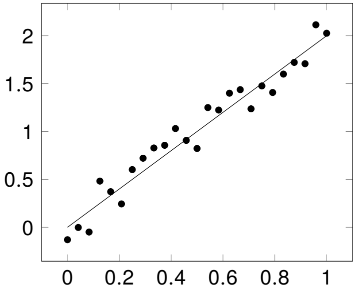
- $x_1, x_2, x_3, \ldots$ are the features of x
- θ are feature weights

Learning linear models:

- Select parameters (weights) to minimize *loss*

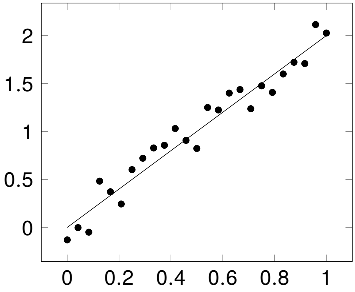
Loss

How "wrong" is this $h(x)$?



Loss

How "wrong" is this $h(x)$?

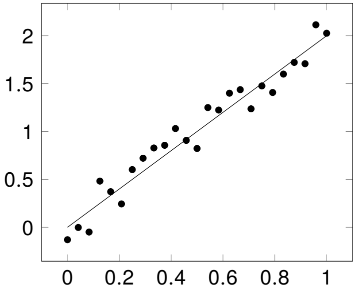


Loss at one point

$$L_{0/1}(\theta, x) = \begin{cases} 0 & \text{if } h_\theta(x) = f(x) \\ 1 & \text{otherwise} \end{cases}$$

Loss

How “wrong” is this $h(x)$?



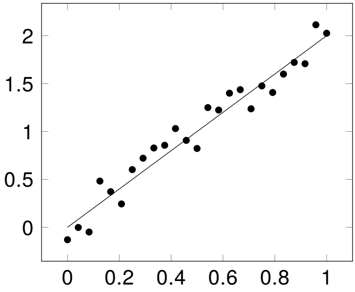
Loss at one point

$$L_{0/1}(\theta, x) = \begin{cases} 0 & \text{if } h_\theta(x) = f(x) \\ 1 & \text{otherwise} \end{cases}$$

$$L_1(\theta, x) = |f(x) - h_\theta(x)|$$

Loss

How "wrong" is this $h(x)$?



Loss at one point

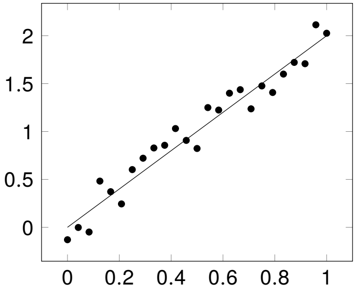
$$L_{0/1}(\theta, x) = \begin{cases} 0 & \text{if } h_\theta(x) = f(x) \\ 1 & \text{otherwise} \end{cases}$$

$$L_1(\theta, x) = |f(x) - h_\theta(x)|$$

$$L_2(\theta, x) = (f(x) - h_\theta(x))^2$$

Loss

How "wrong" is this $h(x)$?



Loss at one point

$$L_{0/1}(\theta, x) = \begin{cases} 0 & \text{if } h_\theta(x) = f(x) \\ 1 & \text{otherwise} \end{cases}$$

$$L_1(\theta, x) = |f(x) - h_\theta(x)|$$

$$L_2(\theta, x) = (f(x) - h_\theta(x))^2$$

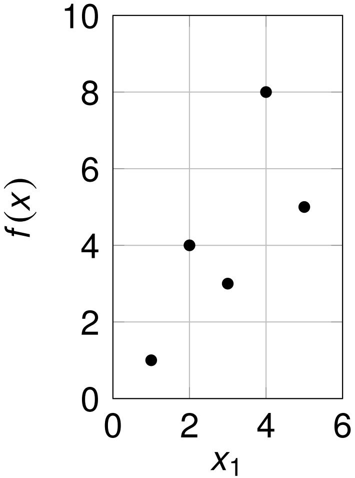
Empirical Loss

$$L(\theta) = \frac{1}{|X|} \sum_{x \in X} L(\theta, x)$$

Loss Exercise

Recall that $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots$ and consider:

1. $\boldsymbol{\theta}_a = [0, 1]$
2. $\boldsymbol{\theta}_b = [0.6, 1.2]$



Which θ is better by:

- $L_{0/1}(\theta) = \frac{1}{|X|}\sum_{x\in X} h_\theta(x) \neq f(x)$?
- $L_1(\theta) = \frac{1}{|X|}\sum_{x\in X} |f(x) - h_\theta(x)|$?
- $L_2(\theta) = \frac{1}{|X|}\sum_{x\in X} (f(x) - h_\theta(x))^2$?

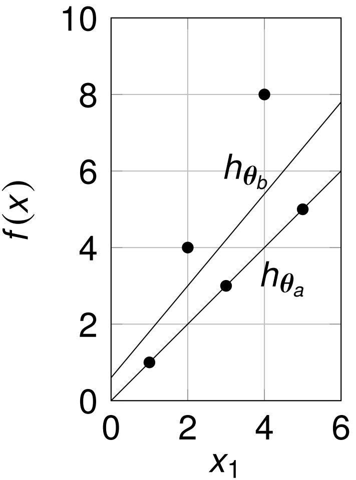
Loss Exercise

Recall that $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots$ and consider:

1. $\boldsymbol{\theta}_a = [0, 1]$
2. $\boldsymbol{\theta}_b = [0.6, 1.2]$

Which θ is better by:

- $L_{0/1}(\theta) = \dfrac{1}{|X|} \sum_{x \in X} h_\theta(x) \neq f(x)$?
  $\boldsymbol{\theta}_a \left(\frac{2}{5} \text{ vs. } \frac{5}{5} \right)$
- $L_1(\theta) = \dfrac{1}{|X|} \sum_{x \in X} |f(x) - h_\theta(x)|$?
  $\boldsymbol{\theta}_a \left(\frac{6}{5} \text{ vs. } \frac{7.2}{5} \right)$
- $L_2(\theta) = \dfrac{1}{|X|} \sum_{x \in X} (f(x) - h_\theta(x))^2$?
  $\boldsymbol{\theta}_b \left(\frac{20}{5} \text{ vs. } \frac{12.4}{5} \right)$



Learning Parameters as Minimizing Loss

Key Ideas

- Best model parameters $\theta = \underset{\theta'}{\operatorname{argmin}}\, Loss(h_{\theta'})$
- Minimized function has partial derivatives = 0

Learning Parameters as Minimizing Loss

Key Ideas

- Best model parameters $\theta = \operatorname*{argmin}_{\theta'} Loss(h_{\theta'})$
- Minimized function has partial derivatives = 0

Example: $x = [x_1]$ with L_2 loss

$$Loss(h_\theta) = \frac{1}{|X|} \sum_{x \in X} (f(x) - (\theta_0 + \theta_1 x_1))^2$$

Learning Parameters as Minimizing Loss

Key Ideas

- Best model parameters $\theta = \underset{\theta'}{\operatorname{argmin}} \mathit{Loss}(h_{\theta'})$
- Minimized function has partial derivatives = 0

Example: $x = [x_1]$ with L_2 loss

$$\begin{aligned} \mathit{Loss}(h_\theta) &= \frac{1}{|X|} \sum_{x \in X} (f(x) - (\theta_0 + \theta_1 x_1))^2 \\ &= \frac{1}{|X|} \sum_{x \in X} f(x)^2 - 2\theta_0 f(x) - 2\theta_1 x_1 f(x) + \theta_0^2 + 2\theta_0\theta_1 x_1 + \theta_1^2 x_1^2 \end{aligned}$$

Learning Parameters as Minimizing Loss

Key Ideas

- Best model parameters $\theta = \underset{\theta'}{\operatorname{argmin}} Loss(h_{\theta'})$
- Minimized function has partial derivatives = 0

Example: $x = [x_1]$ with L_2 loss

$$\begin{aligned}
Loss(h_\theta) &= \frac{1}{|X|}\sum_{x\in X}(f(x) - (\theta_0 + \theta_1 x_1))^2 \\
&= \frac{1}{|X|}\sum_{x\in X} f(x)^2 - 2\theta_0 f(x) - 2\theta_1 x_1 f(x) + \theta_0^2 + 2\theta_0\theta_1 x_1 + \theta_1^2 x_1^2 \\
\frac{\delta}{\delta\theta_0} Loss(h_\theta) &= \frac{1}{|X|}\sum_{x\in X} -2f(x) + 2\theta_0 + 2\theta_1 x_1 = 0
\end{aligned}$$

Learning Parameters as Minimizing Loss

Key Ideas

- Best model parameters $\theta = \operatorname*{argmin}_{\theta'} Loss(h_{\theta'})$
- Minimized function has partial derivatives = 0

Example: $x = [x_1]$ with L_2 loss

$$
\begin{aligned}
Loss(h_\theta) &= \frac{1}{|X|}\sum_{x\in X}(f(x) - (\theta_0 + \theta_1 x_1))^2 \\
&= \frac{1}{|X|}\sum_{x\in X} f(x)^2 - 2\theta_0 f(x) - 2\theta_1 x_1 f(x) + \theta_0^2 + 2\theta_0\theta_1 x_1 + \theta_1^2 x_1^2 \\
\frac{\delta}{\delta\theta_0} Loss(h_\theta) &= \frac{1}{|X|}\sum_{x\in X} -2f(x) + 2\theta_0 + 2\theta_1 x_1 = 0 \\
\theta_0 &= \frac{1}{|X|}\sum_{x\in X} f(x) - \theta_1 x_1
\end{aligned}
$$

Learning Parameters as Minimizing Loss

Key Ideas

- Best model parameters $\theta = \operatorname*{argmin}_{\theta'} Loss(h_{\theta'})$
- Minimized function has partial derivatives = 0

Example: $x = [x_1]$ with L_2 loss

$$
\begin{aligned}
Loss(h_\theta) &= \frac{1}{|X|}\sum_{x\in X}\left(f(x) - (\theta_0 + \theta_1 x_1)\right)^2 \\
&= \frac{1}{|X|}\sum_{x\in X} f(x)^2 - 2\theta_0 f(x) - 2\theta_1 x_1 f(x) + \theta_0^2 + 2\theta_0\theta_1 x_1 + \theta_1^2 x_1^2 \\
\frac{\delta}{\delta\theta_0} Loss(h_\theta) &= \frac{1}{|X|}\sum_{x\in X} -2f(x) + 2\theta_0 + 2\theta_1 x_1 = 0 \\
\theta_0 &= \frac{1}{|X|}\sum_{x\in X} f(x) - \theta_1 x_1 \\
\frac{\delta}{\delta\theta_1} Loss(h_\theta) &= \ldots
\end{aligned}
$$

Learning Parameters as Minimizing Loss

Formal approach:

- Given: function h_θ and function $Loss(\theta)$
- Derive $\nabla Loss(\theta) = [\frac{\delta}{\delta\theta_0}Loss(\theta), \frac{\delta}{\delta\theta_1}Loss(\theta), \ldots]$
- Solve for θ in $\nabla Loss(\theta) = 0$

But there may be no closed form solution!

Gradient Descent

```
θ = any setting of all parameters
while θ has not converged:
  for i in 0...|θ|:
```

$$\theta_i = \theta_i - \alpha \frac{\delta}{\delta\theta_i} Loss(\theta)$$

Learning Parameters as Minimizing Loss

Formal approach:

- Given: function h_θ and function $Loss(\theta)$
- Derive $\nabla Loss(\theta) = [\frac{\delta}{\delta\theta_0}Loss(\theta), \frac{\delta}{\delta\theta_1}Loss(\theta), \ldots]$
- Solve for θ in $\nabla Loss(\theta) = 0$

But there may be no closed form solution!

Gradient Descent

```
θ = any setting of all parameters
while θ has not converged:
  for i in 0...|θ|:
```

$$\theta_i = \theta_i - \alpha \frac{\delta}{\delta\theta_i} Loss(\theta)$$

Learning Parameters as Minimizing Loss

Formal approach:

- Given: function h_θ and function $Loss(\theta)$
- Derive $\nabla Loss(\theta) = [\frac{\delta}{\delta\theta_0}Loss(\theta), \frac{\delta}{\delta\theta_1}Loss(\theta), \ldots]$
- Solve for θ in $\nabla Loss(\theta) = 0$

But there may be no closed form solution!

Gradient Descent

θ = any setting of all parameters
while θ has not converged:
  for i in 0 $\ldots |\theta|$:

$$\theta_i = \theta_i - \alpha \frac{\delta}{\delta\theta_i} Loss(\theta)$$

Gradient Descent Properties

For convex functions:

- Given small enough α, converges to global minimum
- May be slow: scans entire training data every step

Alternative: stochastic gradient descent:

- Calculate loss for each x and update θ accordingly
- Often faster than batch gradient descent
- Not guaranteed to converge to global minimum

For non-convex functions:

- Only converges to a local minimum

Gradient Descent Properties

For convex functions:

- Given small enough α, converges to global minimum
- May be slow: scans entire training data every step

Alternative: stochastic gradient descent:

- Calculate loss for each x and update θ accordingly
- Often faster than batch gradient descent
- Not guaranteed to converge to global minimum

For non-convex functions:

- Only converges to a local minimum

Gradient Descent Properties

For convex functions:

- Given small enough α, converges to global minimum
- May be slow: scans entire training data every step

Alternative: stochastic gradient descent:

- Calculate loss for each x and update θ accordingly
- Often faster than batch gradient descent
- Not guaranteed to converge to global minimum

For non-convex functions:

- Only converges to a local minimum

Avoiding Overfitting with Regularization

Recall overfitting solutions:

- Use simpler model
- Use fewer features

Learning as optimization allows another: *Regularization*

- Instead of minimizing $\mathit{Loss}(\theta)$
- Minimize $\mathit{Loss}(\theta) + \lambda\, \mathit{Complexity}(\theta)$
- Where λ can be tuned

Common choices for $\mathit{Complexity}(\theta)$

- $L_1(\theta) = \sum_i |\theta_i|$, encourages weights of 0 (sparsity)
- $L_2(\theta) = \sum_i |\theta_i|^2$, often makes math easier

Avoiding Overfitting with Regularization

Recall overfitting solutions:

- Use simpler model
- Use fewer features

Learning as optimization allows another: *Regularization*

- Instead of minimizing $\mathit{Loss}(\theta)$
- Minimize $\mathit{Loss}(\theta) + \lambda\, \mathit{Complexity}(\theta)$
- Where λ can be tuned

Common choices for $\mathit{Complexity}(\theta)$

- $L_1(\theta) = \sum_i |\theta_i|$, encourages weights of 0 (sparsity)
- $L_2(\theta) = \sum_i |\theta_i|^2$, often makes math easier

Avoiding Overfitting with Regularization

Recall overfitting solutions:

- Use simpler model
- Use fewer features

Learning as optimization allows another: *Regularization*

- Instead of minimizing $\mathit{Loss}(\theta)$
- Minimize $\mathit{Loss}(\theta) + \lambda\, \mathit{Complexity}(\theta)$
- Where λ can be tuned

Common choices for $\mathit{Complexity}(\theta)$

- $L_1(\theta) = \sum_i |\theta_i|$, encourages weights of 0 (sparsity)
- $L_2(\theta) = \sum_i |\theta_i|^2$, often makes math easier

Avoiding Overfitting with Regularization

Recall overfitting solutions:

- Use simpler model
- Use fewer features

Learning as optimization allows another: *Regularization*

- Instead of minimizing $\mathit{Loss}(\theta)$
- Minimize $\mathit{Loss}(\theta) + \lambda\,\mathit{Complexity}(\theta)$
- Where λ can be tuned

Common choices for $\mathit{Complexity}(\theta)$

- $L_1(\theta) = \sum_i |\theta_i|$, encourages weights of 0 (sparsity)
- $L_2(\theta) = \sum_i |\theta_i|^2$, often makes math easier

Regularization Exercise

x_1	x_2	$f(x)$
2	1	4
3	2	9
4	3	13

Given L_2 loss, which is better:

1. $\boldsymbol{\theta}_a = [-2, 3, 1]$
2. $\boldsymbol{\theta}_b = [0, 3, 0]$

for the regularization:

- None?
- $L_1,\ \lambda = \frac{1}{3}$?
- $L_2,\ \lambda = \frac{1}{3}$?

Loss:

$$L_2(\theta) = \frac{1}{|X|}\sum_{x\in X}(f(x) - h_\theta(x))^2$$

Regularizers:

$$L_1(\theta) = \sum_i |\theta_i| \quad L_2(\theta) = \sum_i |\theta_i|^2$$

Regularization:

$$L(\theta) = Loss(\theta) + \lambda\, Complexity(\theta)$$

Regularization Exercise

x_1	x_2	$f(x)$
2	1	4
3	2	9
4	3	13

Given L_2 loss, which is better:

1. $\boldsymbol{\theta}_a = [-2, 3, 1]$
2. $\boldsymbol{\theta}_b = [0, 3, 0]$

for the regularization:

- None? $\boldsymbol{\theta}_a$ $\left(\frac{1}{3} \text{ vs. } \frac{5}{3}\right)$
- L_1, $\lambda = \frac{1}{3}$? $\boldsymbol{\theta}_a$ $\left(\frac{7}{3} \text{ vs. } \frac{8}{3}\right)$
- L_2, $\lambda = \frac{1}{3}$? $\boldsymbol{\theta}_b$ $\left(\frac{15}{3} \text{ vs. } \frac{14}{3}\right)$

Loss:

$$L_2(\theta) = \frac{1}{|X|} \sum_{x \in X} (f(x) - h_\theta(x))^2$$

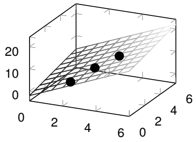
Regularizers:

$$L_1(\theta) = \sum_i |\theta_i| \quad L_2(\theta) = \sum_i |\theta_i|^2$$

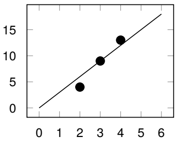
Regularization:

$$L(\theta) = Loss(\theta) + \lambda\, Complexity(\theta)$$

$h(x) = -2 + 3x_1 + x_2$

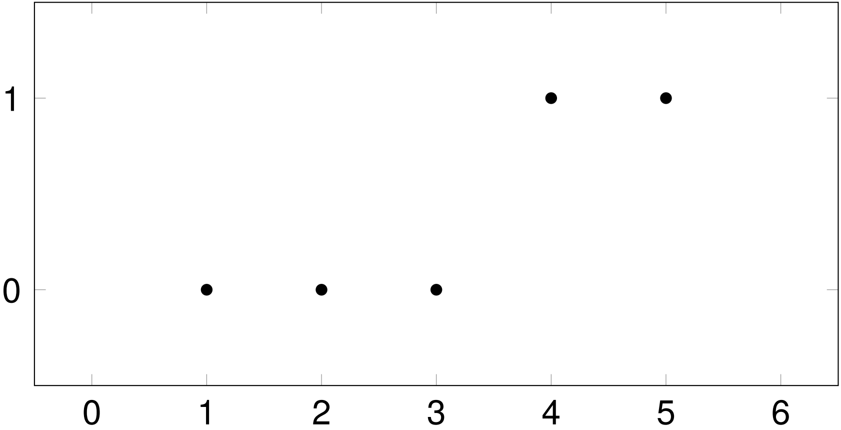


$h(x) = 3x_1$



Linear vs. Logistic Regression

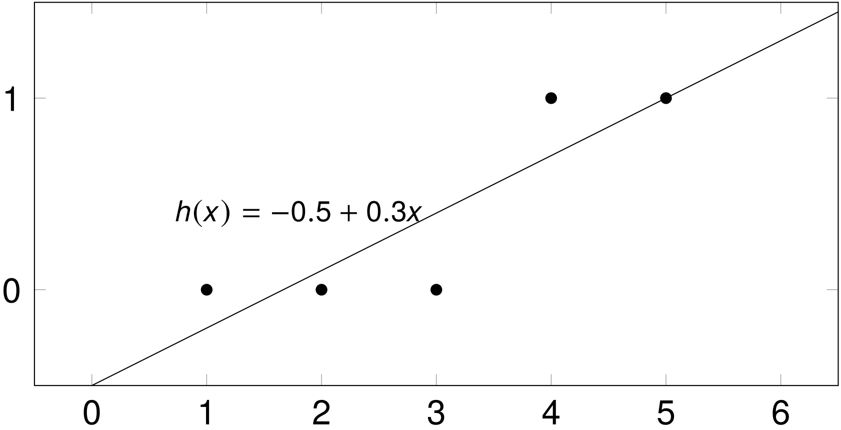
Linear regression is bad for classification:



Instead, use *logistic regression*

Linear vs. Logistic Regression

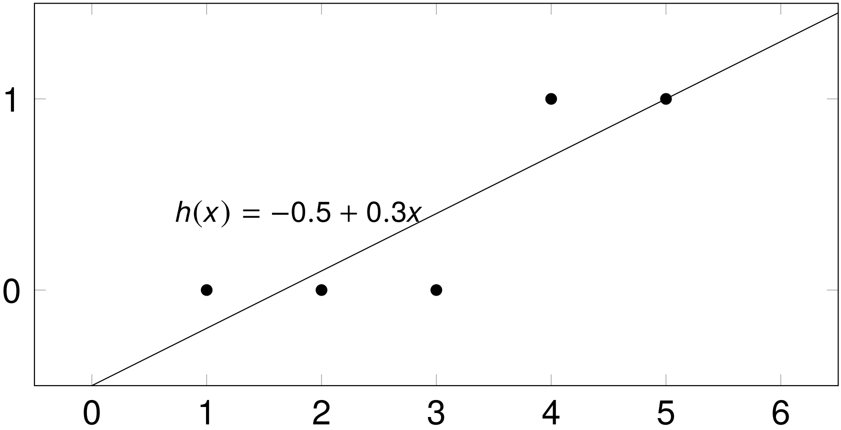
Linear regression is bad for classification:



Instead, use *logistic regression*

Linear vs. Logistic Regression

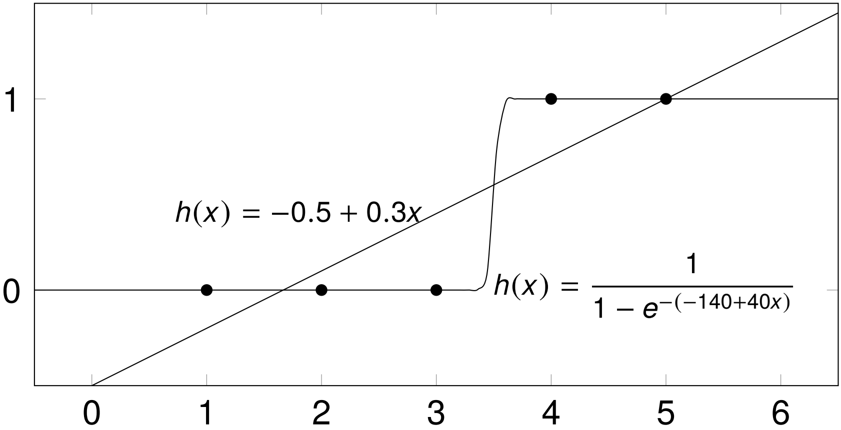
Linear regression is bad for classification:



Instead, use *logistic regression*

Linear vs. Logistic Regression

Linear regression is bad for classification:



Instead, use *logistic regression*

Linear vs. Logistic Regression

Linear regression:

- $h(x) : \mathbb{R}^n \Rightarrow \mathbb{R}$
- $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots$
- $L_2(h_\theta) = \frac{1}{|X|} \sum_{x \in X} (f(x) - (\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots))^2$

Logistic regression:

- $h(x) : \mathbb{R}^n \Rightarrow [0, 1]$
- $h_\theta(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots)}}$
- $L_2(h_\theta) = \frac{1}{|X|} \sum_{x \in X} \left(f(x) - \left(\frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots)}} \right) \right)^2$

Linear vs. Logistic Regression

Linear regression:

- $h(x) : \mathbb{R}^n \Rightarrow \mathbb{R}$
- $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots$
- $L_2(h_\theta) = \frac{1}{|X|} \sum_{x \in X} (f(x) - (\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots))^2$

Logistic regression:

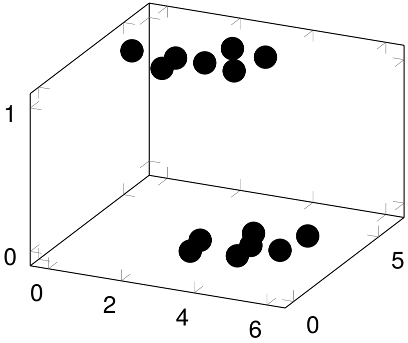
- $h(x) : \mathbb{R}^n \Rightarrow [0, 1]$
- $h_\theta(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots)}}$
- $L_2(h_\theta) = \frac{1}{|X|} \sum_{x \in X} \left(f(x) - \left(\frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots)}} \right) \right)^2$

Logistic Regression Properties

- $h_\theta(x)$ can be interpreted as $P(f(x) = 1)$
- Can be generalized for multi-class classification
- With regularization, state-of-the-art on many problems

Max Margin Classification

Classification problem:



Classification hyperplanes:

- $x_2 = 3.7 \quad \Rightarrow \frac{13}{14}$
- $2x_1 - x_2 = 3 \quad \Rightarrow \frac{12}{14}$
- $x_1 - x_2 = 0 \quad \Rightarrow \frac{14}{14}$

Maximizing margin:

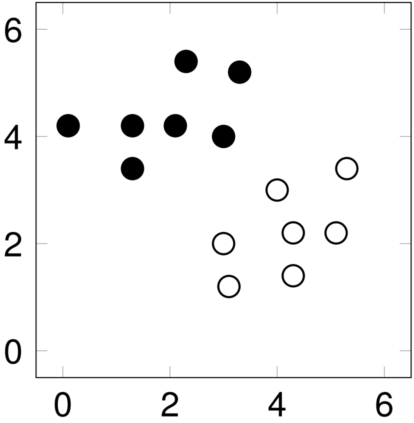
$$2x_1 - x_2 = 3 \Rightarrow 0.35$$

$$x_1 - x_2 = 0 \Rightarrow 0.50$$

Data points at margin are called support vectors

Max Margin Classification

Classification problem:



Classification hyperplanes:

- $x_2 = 3.7 \quad \Rightarrow \frac{13}{14}$
- $2x_1 - x_2 = 3 \quad \Rightarrow \frac{14}{14}$
- $x_1 - x_2 = 0 \quad \Rightarrow \frac{14}{14}$

Maximizing margin:

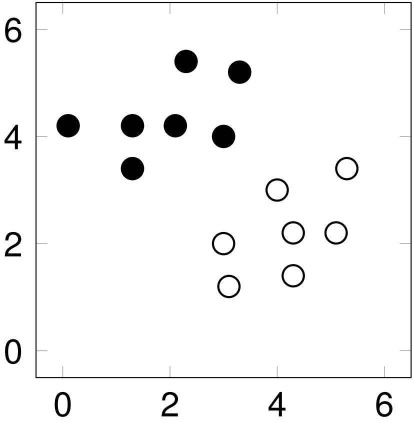
$$2x_1 - x_2 = 3 \quad \Rightarrow 0.35$$

$$x_1 - x_2 = 0 \quad \Rightarrow 0.50$$

Data points at margin are called *support vectors*

Max Margin Classification

Classification problem:



Classification hyperplanes:

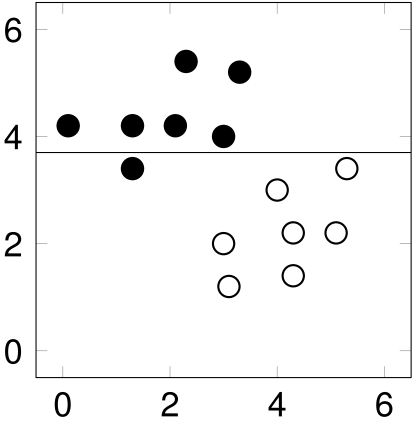
- $x_2 = 3.7$
- $2x_1 - x_2 = 3$
- $x_1 - x_2 = 0$

Maximizing margin:

Data points at margin are called *support vectors*

Max Margin Classification

Classification problem:



Classification hyperplanes:

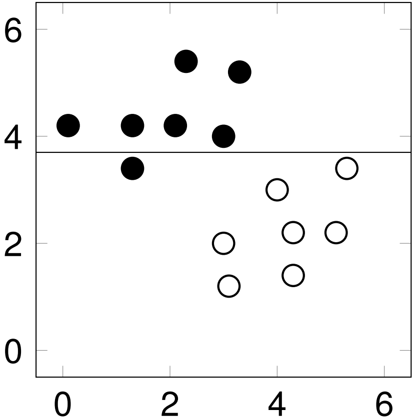
- $x_2 = 3.7$ $\Rightarrow \frac{13}{14}$
- $2x_1 - x_2 = 3$
- $x_1 - x_2 = 0$

Maximizing margin:

Data points at margin are called *support vectors*

Max Margin Classification

Classification problem:



Classification hyperplanes:

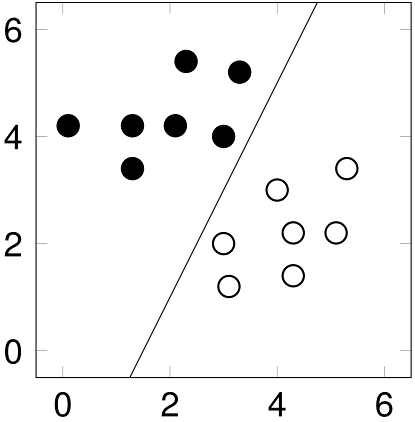
- $x_2 = 3.7 \quad \Rightarrow \frac{13}{14}$
- $2x_1 - x_2 = 3$
- $x_1 - x_2 = 0$

Maximizing margin:

Data points at margin are called *support vectors*

Max Margin Classification

Classification problem:



Classification hyperplanes:

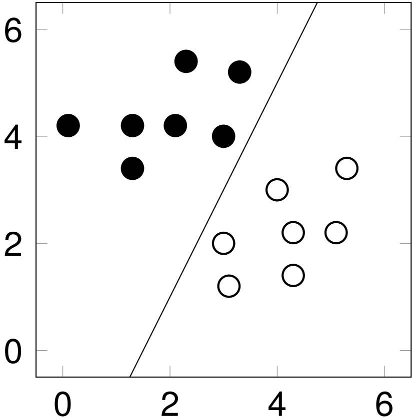
- $x_2 = 3.7$ $\Rightarrow \frac{13}{14}$
- $2x_1 - x_2 = 3$ $\Rightarrow \frac{14}{14}$
- $x_1 - x_2 = 0$

Maximizing margin:

Data points at margin are called *support vectors*

Max Margin Classification

Classification problem:

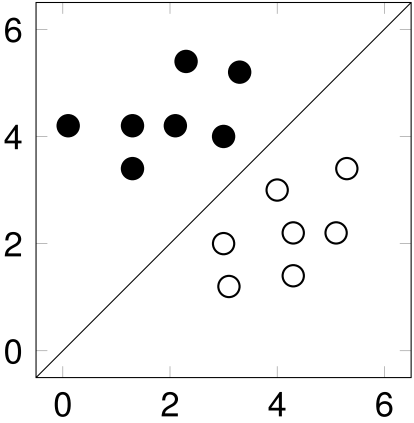


Classification hyperplanes:

- $x_2 = 3.7 \quad \Rightarrow \frac{13}{14}$
- $2x_1 - x_2 = 3 \quad \Rightarrow \frac{14}{14}$

Max Margin Classification

Classification problem:



Classification hyperplanes:

- $x_2 = 3.7 \Rightarrow \frac{13}{14}$
- $2x_1 - x_2 = 3 \Rightarrow \frac{14}{14}$
- $x_1 - x_2 = 0 \Rightarrow \frac{14}{14}$

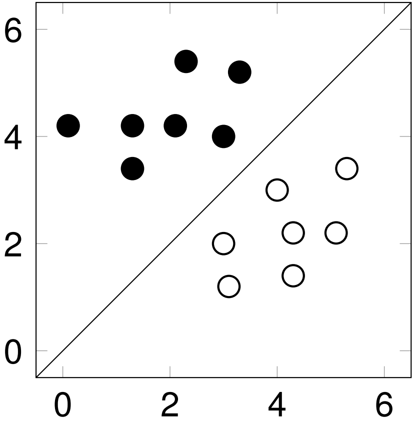
Maximizing margin:

- $2x_1 - x_2 = 3 \Rightarrow 0.35$
- $x_1 - x_2 = 0 \Rightarrow 0.50$

Data points at margin are called *support vectors*

Max Margin Classification

Classification problem:



Classification hyperplanes:

- $x_2 = 3.7 \Rightarrow \frac{13}{14}$
- $2x_1 - x_2 = 3 \Rightarrow \frac{14}{14}$
- $x_1 - x_2 = 0 \Rightarrow \frac{14}{14}$

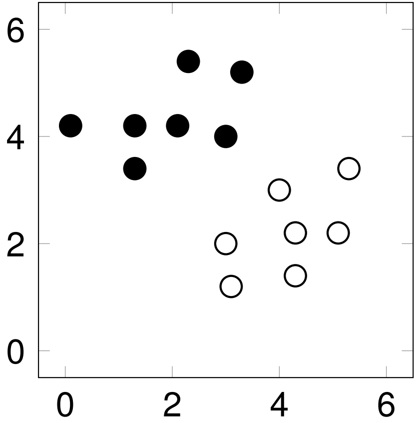
Maximizing margin:

- $2x_1 - x_2 = 3 \Rightarrow 0.35$
- $x_1 - x_2 = 0 \Rightarrow 0.50$

Data points at margin are called *support vectors*

Max Margin Classification

Classification problem:



Classification hyperplanes:

- $x_2 = 3.7 \quad \Rightarrow \frac{13}{14}$
- $2x_1 - x_2 = 3 \quad \Rightarrow \frac{14}{14}$
- $x_1 - x_2 = 0 \quad \Rightarrow \frac{14}{14}$

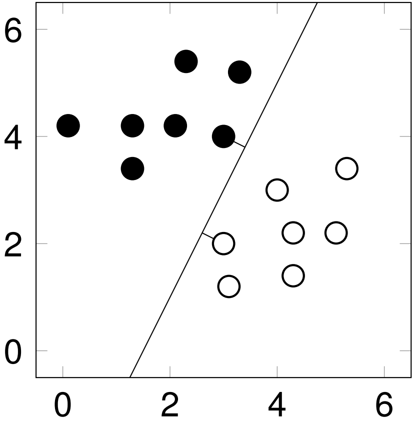
Maximizing margin:

- $2x_1 - x_2 = 3 \quad \Rightarrow 0.35$
- $x_1 - x_2 = 0 \quad \Rightarrow 0.50$

Data points at margin are called *support vectors*

Max Margin Classification

Classification problem:



Classification hyperplanes:

- $x_2 = 3.7 \quad \Rightarrow \frac{13}{14}$
- $2x_1 - x_2 = 3 \quad \Rightarrow \frac{14}{14}$
- $x_1 - x_2 = 0 \quad \Rightarrow \frac{14}{14}$

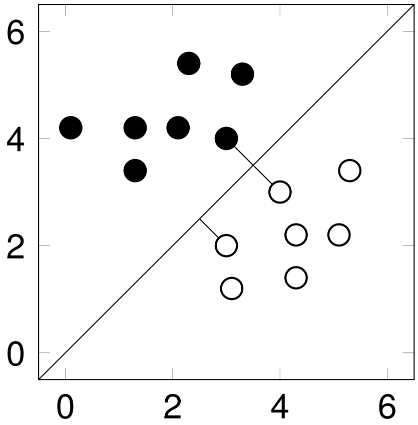
Maximizing margin:

- $2x_1 - x_2 = 3 \quad \Rightarrow 0.35$
- $x_1 - x_2 = 0 \quad \Rightarrow 0.50$

Data points at margin are called *support vectors*

Max Margin Classification

Classification problem:



Classification hyperplanes:

- $x_2 = 3.7 \quad \Rightarrow \frac{13}{14}$
- $2x_1 - x_2 = 3 \quad \Rightarrow \frac{14}{14}$
- $x_1 - x_2 = 0 \quad \Rightarrow \frac{14}{14}$

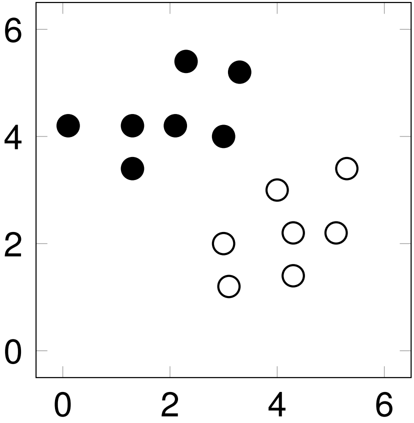
Maximizing margin:

- $2x_1 - x_2 = 3 \quad \Rightarrow 0.35$
- $x_1 - x_2 = 0 \quad \Rightarrow 0.50$

Data points at margin are called *support vectors*

Max Margin Classification

Classification problem:



Classification hyperplanes:

- $x_2 = 3.7 \quad \Rightarrow \frac{13}{14}$
- $2x_1 - x_2 = 3 \quad \Rightarrow \frac{14}{14}$
- $x_1 - x_2 = 0 \quad \Rightarrow \frac{14}{14}$

Maximizing margin:

- $2x_1 - x_2 = 3 \quad \Rightarrow 0.35$
- $x_1 - x_2 = 0 \quad \Rightarrow 0.50$

Data points at margin are called *support vectors*

Support Vector Machine Classifiers

Support vector machine loss, with $f(x) : \mathbb{R} \Rightarrow \{-1, +1\}$:

$$L_2(\theta) = \underbrace{\frac{1}{2} \sum_i \theta_i^2}_{\text{regularizer}} + \underbrace{C}_{\substack{\text{misclassify} \\ \text{cost}}} \sum_{x \in X} \max\left(0, 1 - f(x) \underbrace{\sum_i \theta_i x_i}_{\text{linear model}}\right)$$

Compare to L2-regularized logistic regression:

$$L_2(\theta) = \frac{1}{2} \sum_i \theta_i^2 + \frac{1}{|X|} \sum_{x \in X} \left(f(x) - \left(\frac{1}{1 + e^{-\sum_i \theta_i x_i}} \right) \right)^2$$

Support Vector Machine Classifiers

Support vector machine loss, with $f(x) : \mathbb{R} \Rightarrow \{-1, +1\}$:

$$L_2(\theta) = \underbrace{\frac{1}{2}\sum_i \theta_i^2}_{\text{regularizer}} + \underbrace{C}_{\substack{\text{misclassify}\\ \text{cost}}} \sum_{x \in X} \max\left(0, 1 - f(x) \underbrace{\sum_i \theta_i x_i}_{\text{linear model}}\right)$$

Compare to L2-regularized logistic regression:

$$L_2(\theta) = \frac{1}{2}\sum_i \theta_i^2 + \frac{1}{|X|} \sum_{x \in X} \left(f(x) - \left(\frac{1}{1 + e^{-\sum_i \theta_i x_i}}\right)\right)^2$$

Kernel Methods

Support vector machine dual form:

$$L(\alpha) = \sum_{x \in X} \alpha_x - \frac{1}{2} \sum_{x \in X,\, x' \in X} \alpha_x \alpha_{x'} f(x) f(x') \sum_i x_i x_i'$$

Common kernels $k(x, x')$:

- Linear – $(\sum_i x_i x_i')$
- Polynomial – $(\sum_i x_i x_i')^d$
- Radial basis function (RBF) – $e^{-\gamma \|x - x'\|^2}$

Demo: kernels allow non-linear classification boundaries

- http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html?js=1

Kernel Methods

Support vector machine dual form:

$$L(\alpha) = \sum_{x \in X} \alpha_x - \frac{1}{2} \sum_{x \in X,\, x' \in X} \alpha_x \alpha_{x'} f(x) f(x') k(x, x')$$

Common kernels $k(x, x')$:

- Linear – $(\sum_i x_i x_i')$
- Polynomial – $(\sum_i x_i x_i')^d$
- Radial basis function (RBF) – $e^{-\gamma \|x - x'\|^2}$

Demo: kernels allow non-linear classification boundaries

- `http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html?js=1`

Kernel Methods

Support vector machine dual form:

$$L(\alpha) = \sum_{x \in X} \alpha_x - \frac{1}{2} \sum_{x \in X,\, x' \in X} \alpha_x \alpha_{x'} f(x) f(x') k(x, x')$$

Common kernels $k(x, x')$:

- Linear – $(\sum_i x_i x_i')$
- Polynomial – $(\sum_i x_i x_i')^d$
- Radial basis function (RBF) – $e^{-\gamma \|x - x'\|^2}$

Demo: kernels allow non-linear classification boundaries

- http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html?js=1

Kernel Methods

Support vector machine dual form:

$$L(\alpha) = \sum_{x \in X} \alpha_x - \frac{1}{2} \sum_{x \in X,\, x' \in X} \alpha_x \alpha_{x'} f(x) f(x') k(x, x')$$

Common kernels $k(x, x')$:

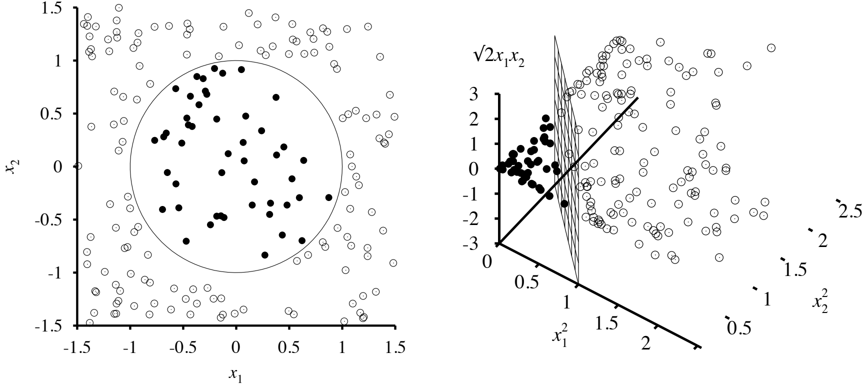
- Linear – $(\sum_i x_i x_i')$
- Polynomial – $(\sum_i x_i x_i')^d$
- Radial basis function (RBF) – $e^{-\gamma \|x - x'\|^2}$

Demo: kernels allow non-linear classification boundaries

- `http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html?js=1`

Kernel Alternative: Feature Engineering

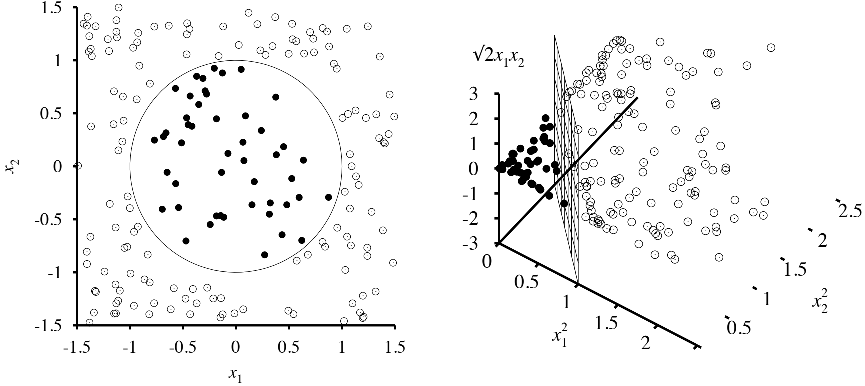
Nonlinear classification via feature transformation:



Kernels: similar effect, but may be more efficient

Kernel Alternative: Feature Engineering

Nonlinear classification via feature transformation:



Kernels: similar effect, but may be more efficient

Support Vector Machine Properties

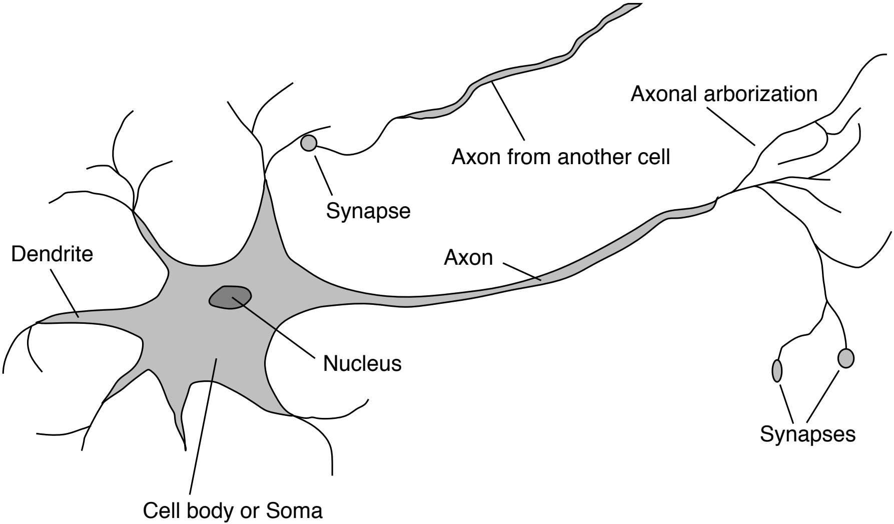
Theoretical Properties

- Efficient optimal separators in huge feature spaces
- Can approximate essentially any function

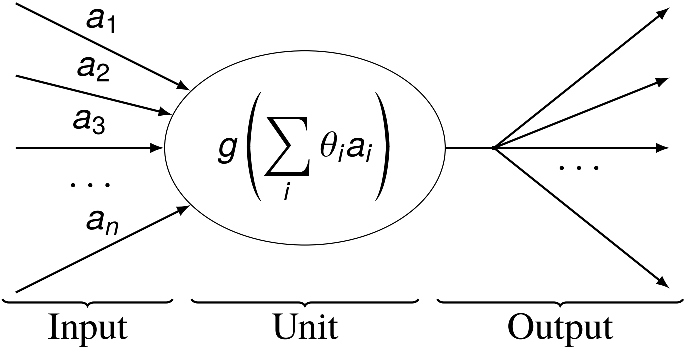
Empirical Issues

- Classification usually fast, but training often slow
- Kernel functions (and parameters) chosen empirically

Neurons in the Brain



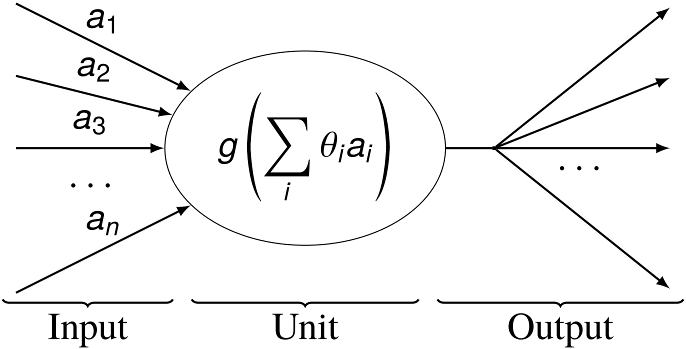
Neurons in a Neural Network



Common choices for *activation function* g:

$$threshold(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \qquad sigmoid(x) = \frac{1}{1 + e^{-x}}$$

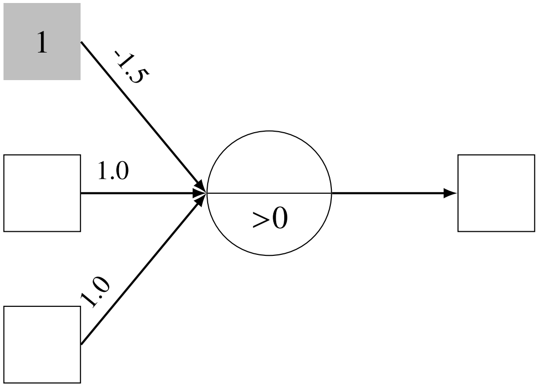
Neurons in a Neural Network



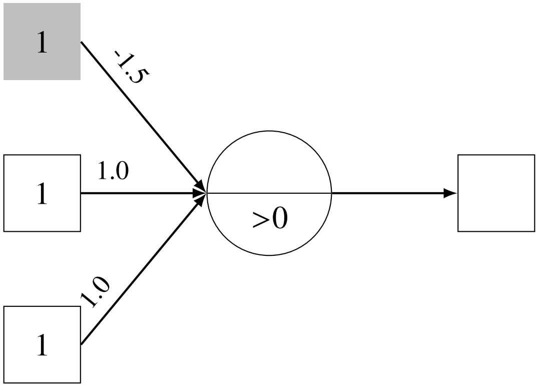
Common choices for *activation function* g:

$$\mathit{threshold}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \qquad \mathit{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

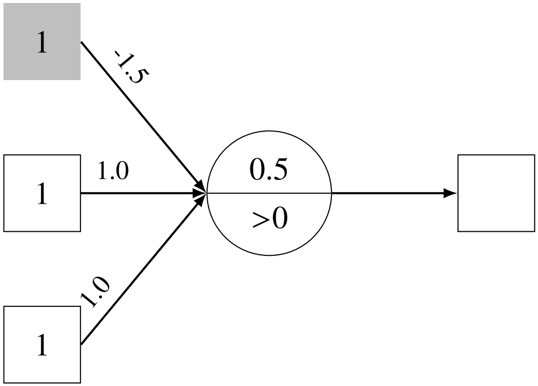
Neural Network Example: Logical And



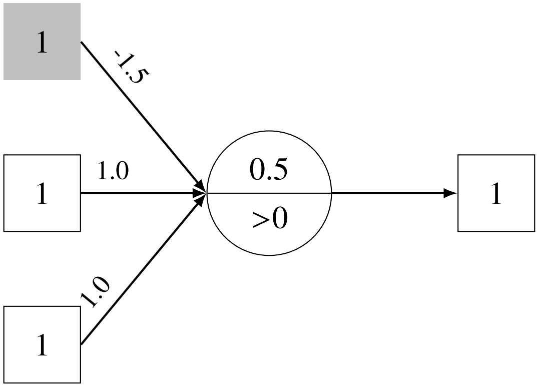
Neural Network Example: Logical And



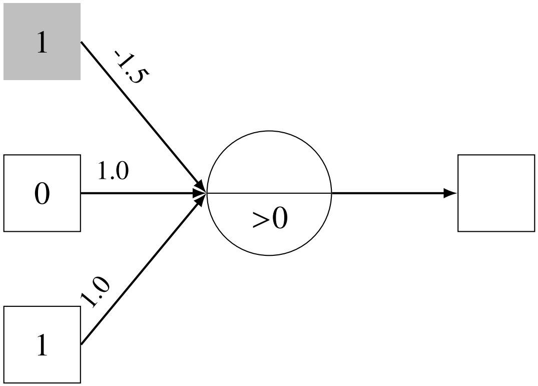
Neural Network Example: Logical And



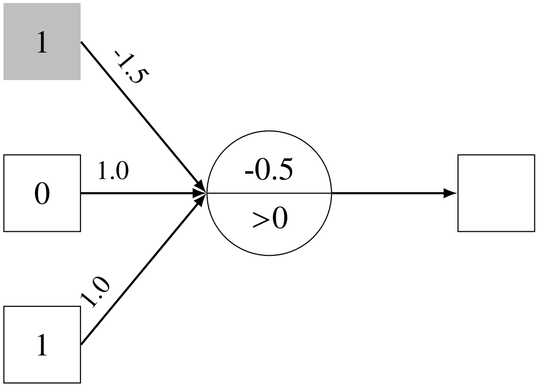
Neural Network Example: Logical And



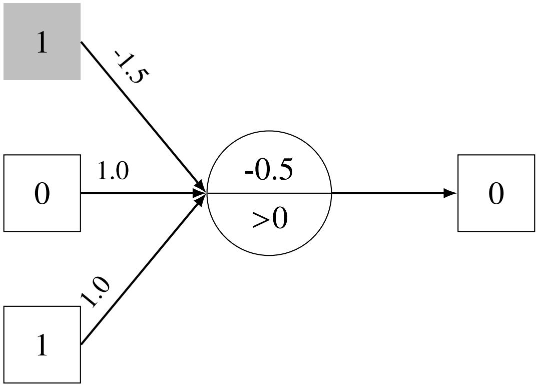
Neural Network Example: Logical And



Neural Network Example: Logical And

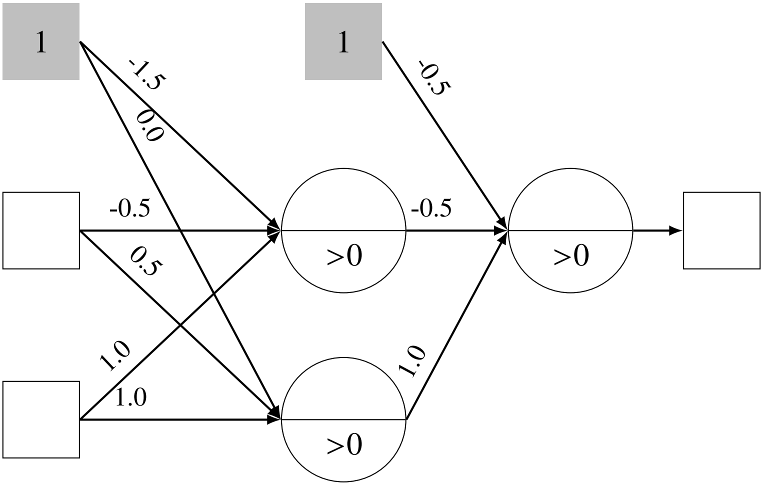


Neural Network Example: Logical And



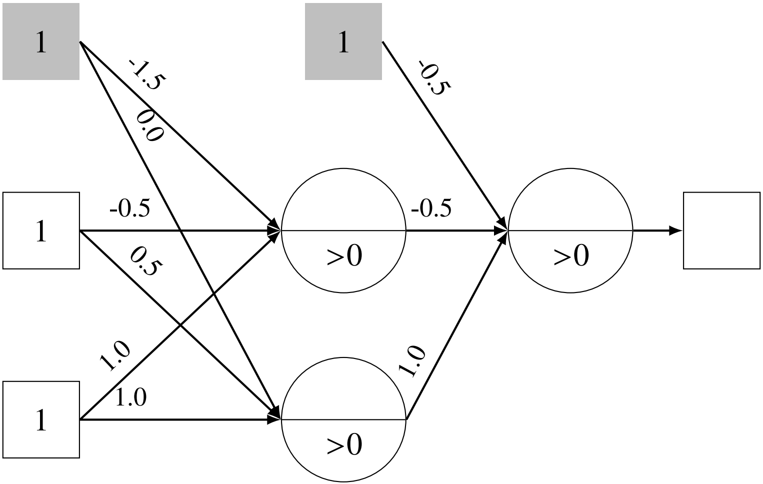
Neural Network Classification

Sum inputs, apply activation function, repeat



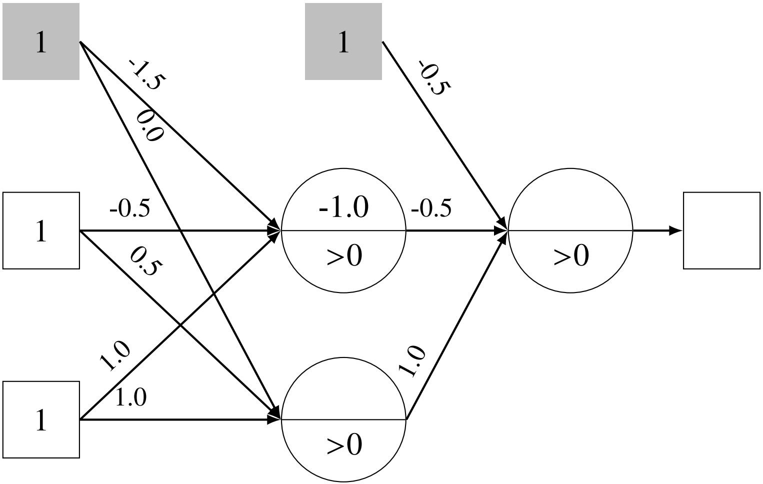
Neural Network Classification

Sum inputs, apply activation function, repeat



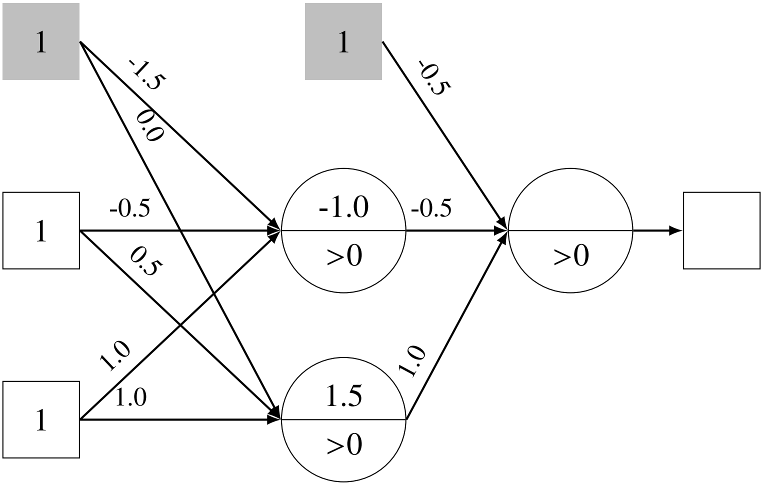
Neural Network Classification

Sum inputs, apply activation function, repeat



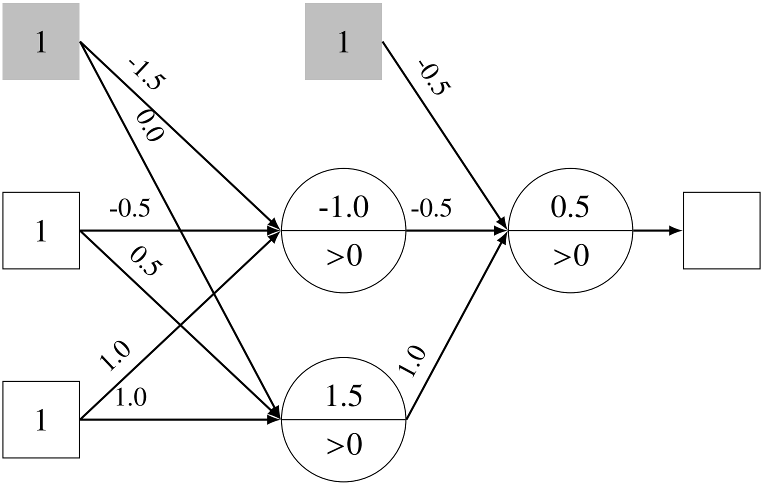
Neural Network Classification

Sum inputs, apply activation function, repeat



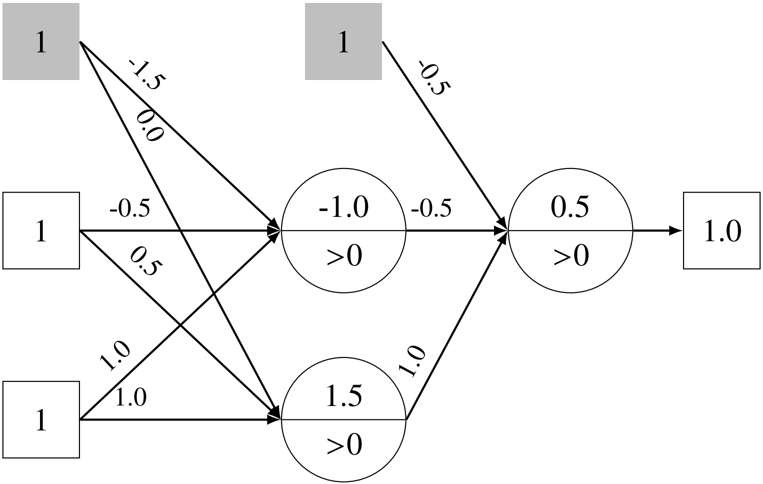
Neural Network Classification

Sum inputs, apply activation function, repeat

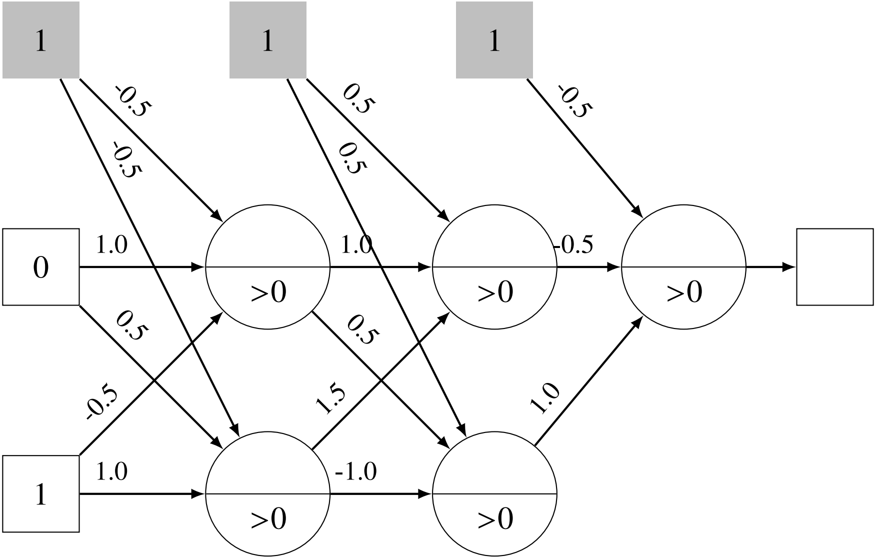


Neural Network Classification

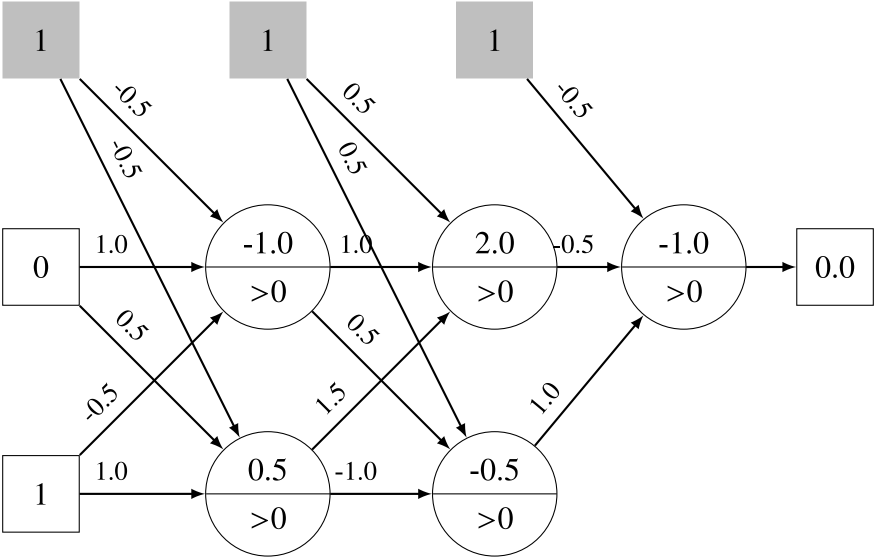
Sum inputs, apply activation function, repeat



Neural Network Exercise



Neural Network Exercise



Learning Neural Networks

Gradient descent over the θs in all nodes:

1. Forward-propagate activation, get output a_o
2. $L(\theta) = \frac{1}{|X|} \sum_{x \in X} -f(x) \log(a_o) - (1 - f(x)) \log(1 - a_o)$
3. Calculate $\nabla L(\theta)$ by *back propagation*:
    1. Measure how much each unit was "responsible" for errors
    2. For output unit, $(a_o - f(x))$
    3. For hidden layer k, weighted average of layer $k + 1$
4. Use $L(\theta)$ and $\nabla L(\theta)$ to take a gradient descent step
5. Goto 1

Neural Net Properties

Expressive Power

- Single layer networks: linearly separable functions
- Multi-layer networks: essentially any function

Empirical Issues

- How many hidden layers?
- How many nodes in each layer?
- How to initialize weights?
- Has gradient descent gotten stuck at local minimum?

Neural Net Properties

Expressive Power

- Single layer networks: linearly separable functions
- Multi-layer networks: essentially any function

Empirical Issues

- How many hidden layers?
- How many nodes in each layer?
- How to initialize weights?
- Has gradient descent gotten stuck at local minimum?

Key Ideas

Supervised Learning:

- Input: $(x, f(x))$ examples; Output: h, a guess at f
- Representation: x decomposed into features
- Evaluation: train, development, test
- Learning curves: reveal underfitting, overfitting

Supervised Learning Algorithms:

- Decision trees and random forests
- Linear and logistic regression
- Support vector machines
- k-nearest neighbors
- Neural networks