

INTRODUCTION TO DATA SCIENCE

This lecture is
based on course by E. Fox and C. Guestrin, Univ of Washington

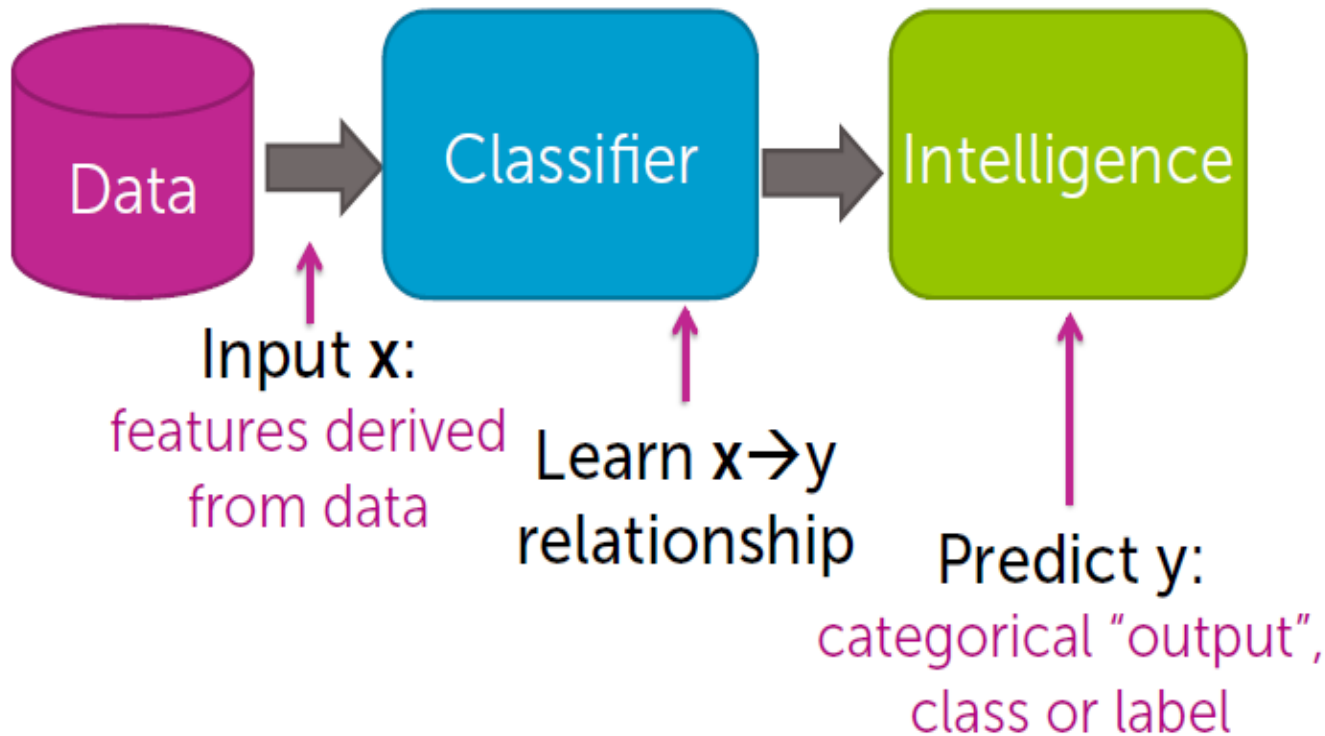
29/10, 5/11
2019

WFAiS UJ, Informatyka Stosowana
I stopień studiów

What is a classification?

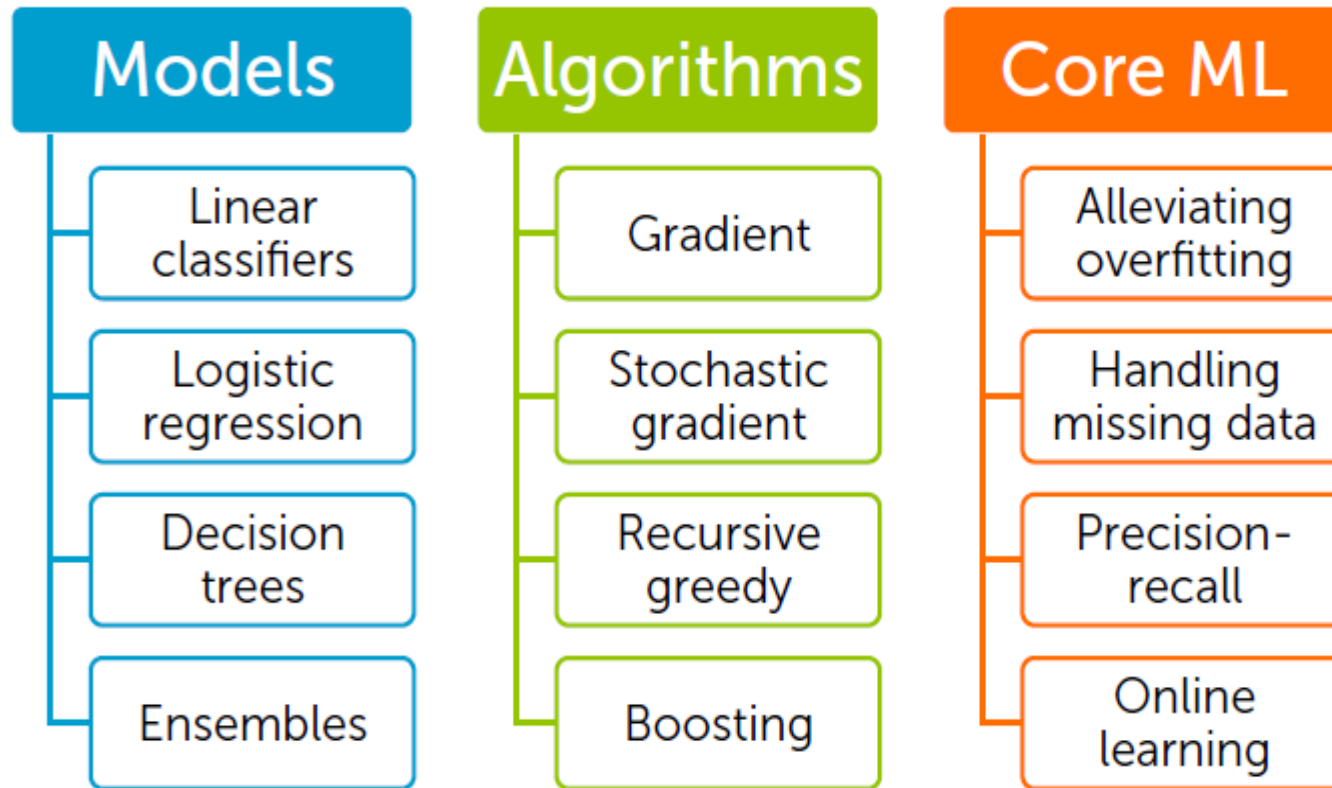
2

From features to predictions



Overview of the content

3



Linear classifier

An intelligent restaurant review system

5

It's a big day & I want to book a table at a nice Japanese restaurant

Seattle has many
★★★★
sushi restaurants



What are people
saying about
the food?
the ambiance?...



Reviews

6

★★★★☆ 428 reviews
\$\$ · Japanese, Sushi Bars



Sample review:

Watching the chefs create incredible edible art made the experience very unique.

My wife tried their ramen and it was pretty forgettable.

All the sushi was delicious!
Easily best sushi in Seattle.

Experience



Positive reviews not positive about everything

Classifying sentiment of review

7

Easily best sushi in Seattle.

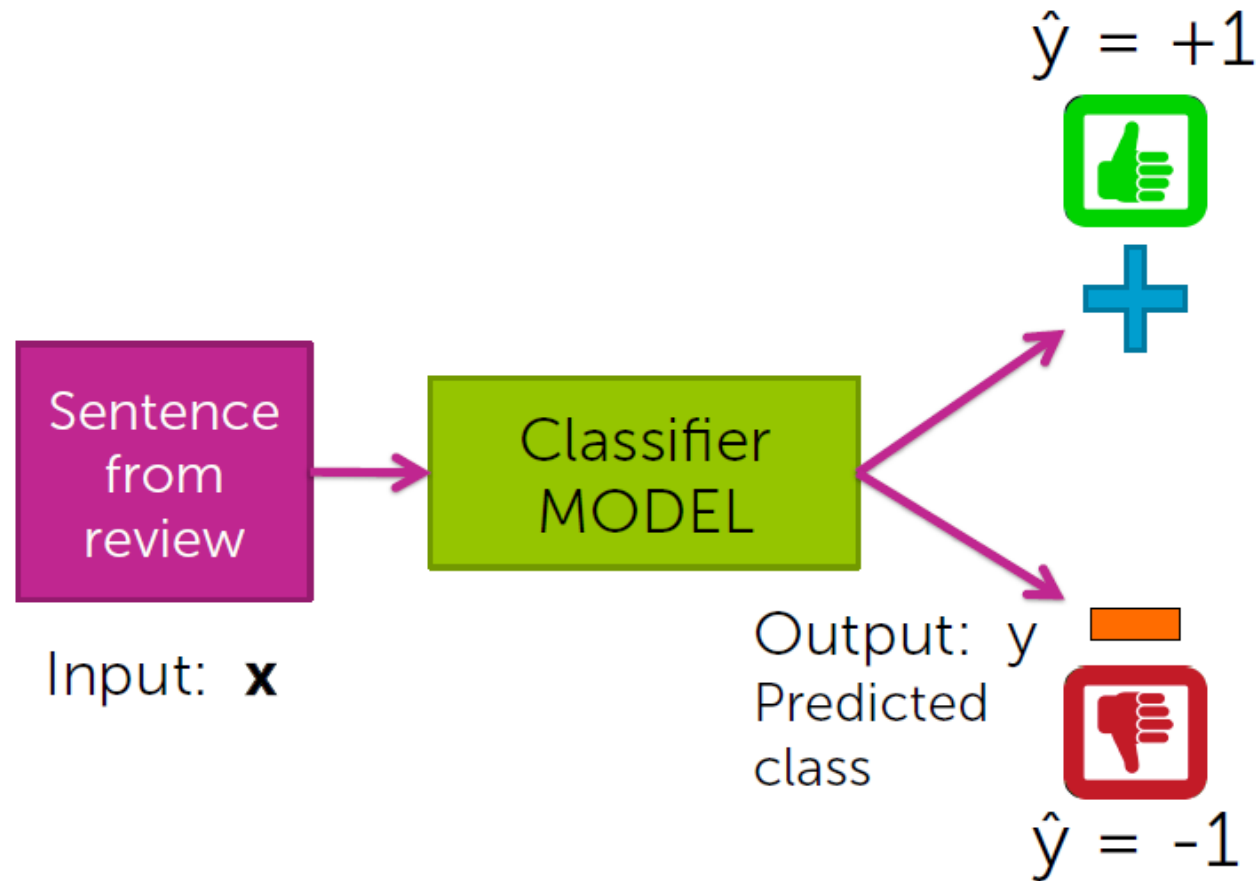


Sentence Sentiment
Classifier



Classifier

8



Note: we'll start talking about 2 classes, and address multiclass later

A (linear) classifier

9

Will use training data to learn a weight for each word

Word	Weight
good	1.0
great	1.5
awesome	2.7
bad	-1.0
terrible	-2.1
awful	-3.3
restaurant, the, we, where, ...	0.0
...	...

Scoring a sentence

10

Word	Coefficient
good	1.0
great	1.2
awesome	1.7
bad	-1.0
terrible	-2.1
awful	-3.3
restaurant, the, we, where, ...	0.0
...	...

Input \mathbf{x}_i :

Sushi was great,
the food was awesome,
but the service was terrible.

$$\text{Score}(\mathbf{x}_i) = 1.2 + 1.7 - 2.1 \\ = 0.8 > 0$$

$$\Rightarrow y = +1$$

positive review

Called a linear classifier, because output is weighted sum of input.

Simple linear classifier

11

Word	Coefficient
...	...



Simple linear classifier

$\text{Score}(\mathbf{x})$ = weighted count of words in sentence

If $\text{Score}(\mathbf{x}) > 0$:

$$\hat{y} = +1$$

Else:

$$\hat{y} = -1$$

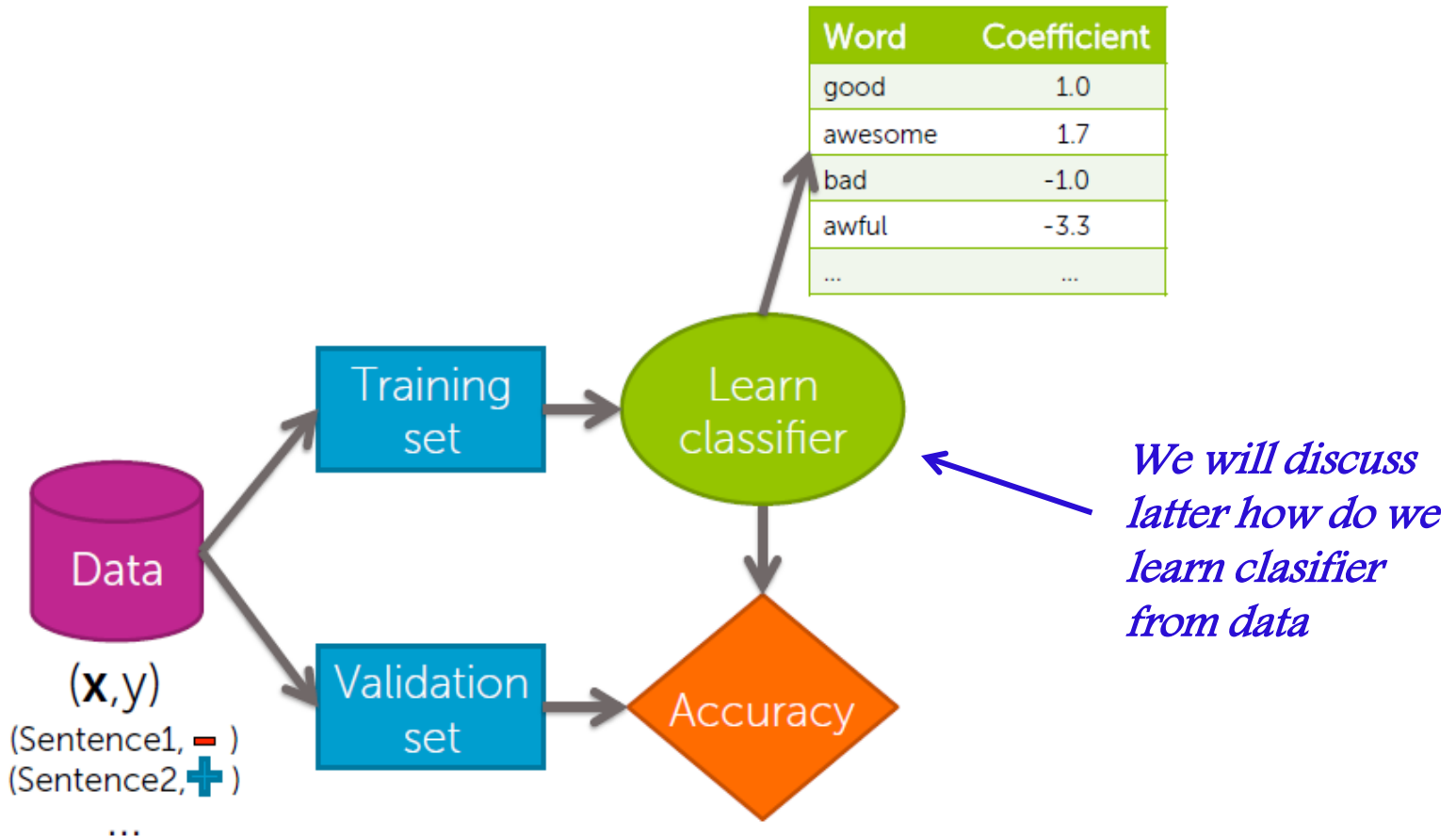
Sentence
from
review



Input: \mathbf{x}

Training a classifier = Learning the coefficients

12

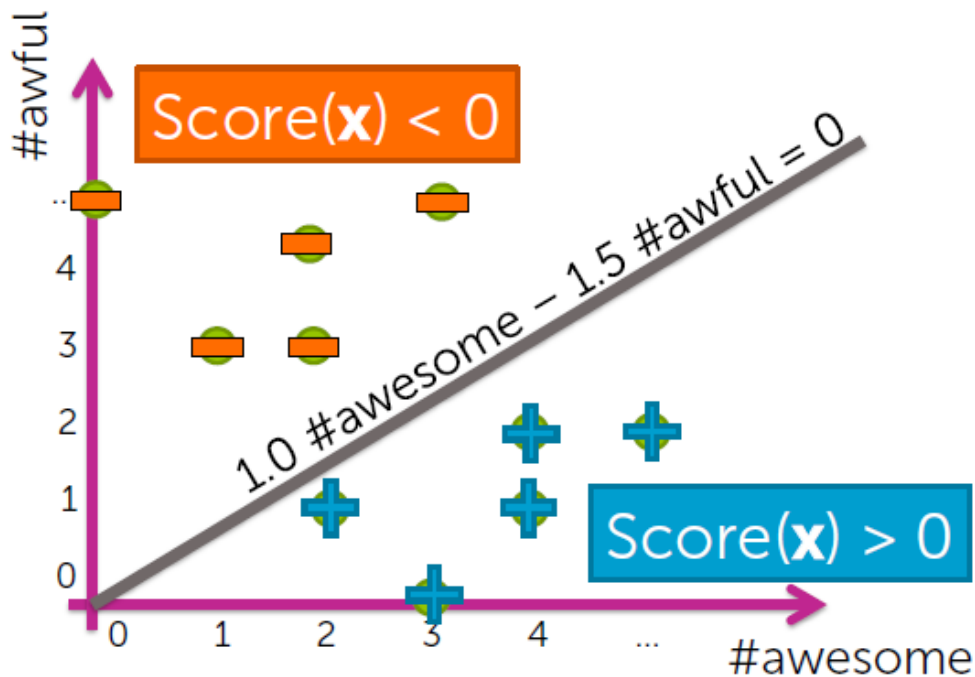


Decision boundary example

13

Word	Coefficient
#awesome	1.0
#awful	-1.5

→ $\text{Score}(x) = 1.0 \# \text{awesome} - 1.5 \# \text{awful}$

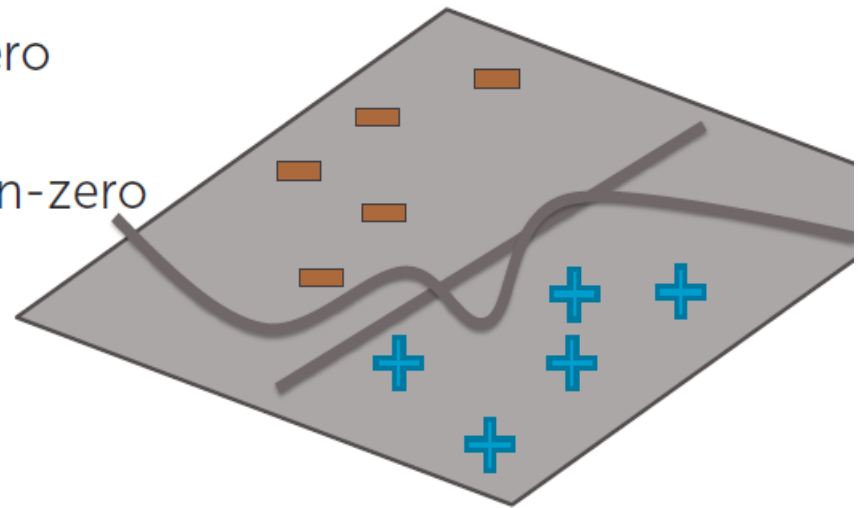


Decision boundary

14

Decision boundary separates positive & negative predictions

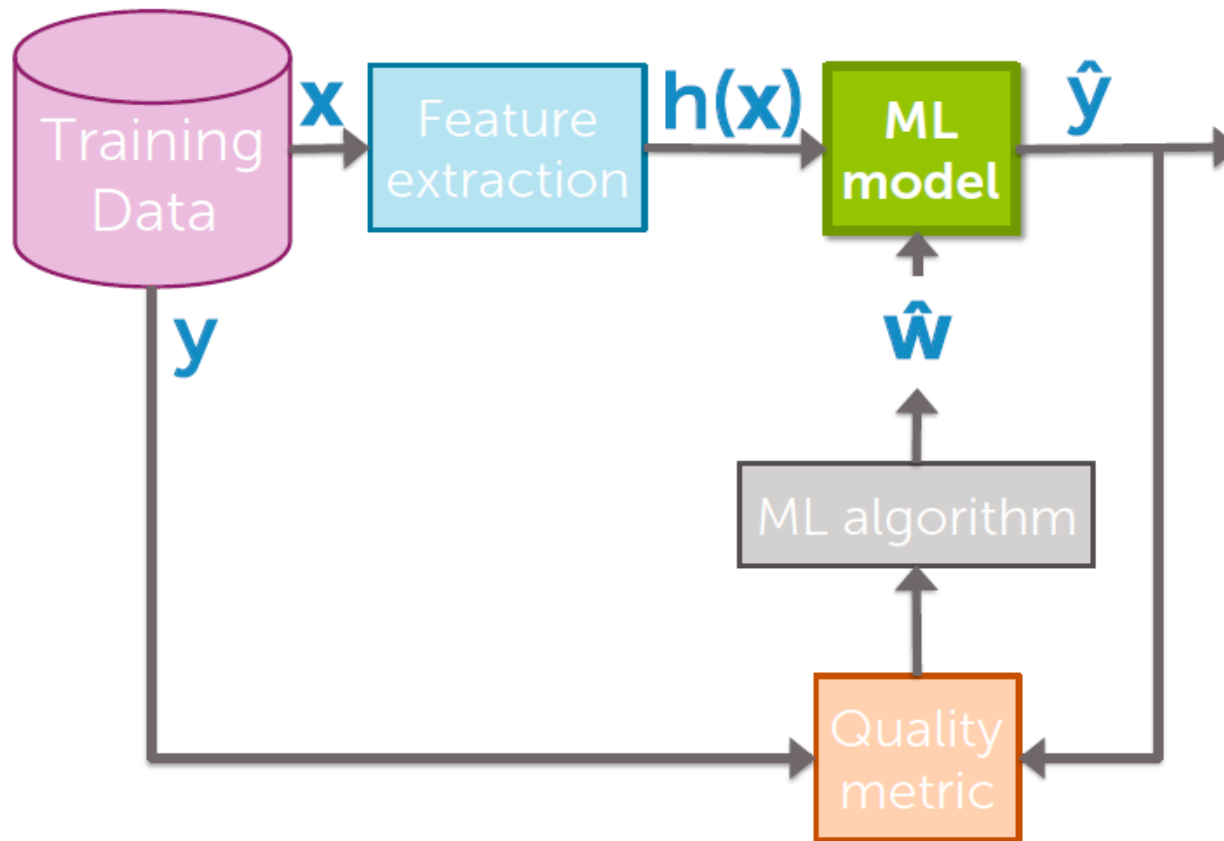
- For linear classifiers:
 - When 2 coefficients are non-zero
→ line
 - When 3 coefficients are non-zero
→ plane
 - When many coefficients are non-zero
→ hyperplane
- For more general classifiers
→ more complicated shapes



Flow chart:

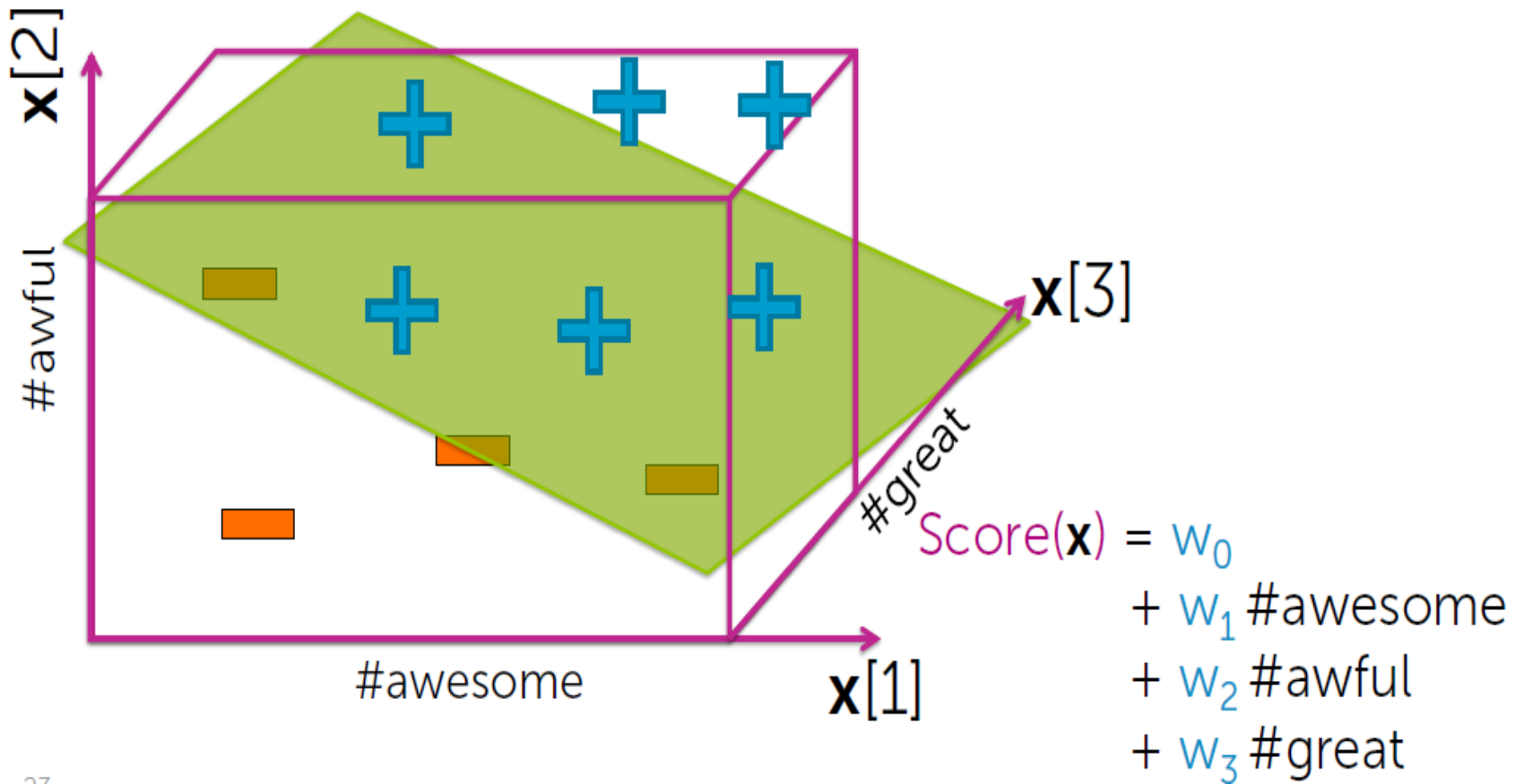


15



Coefficients of classifier

16



37

General notation

17

Output: $y \leftarrow \{-1, +1\}$

Inputs: $\mathbf{x} = (\mathbf{x}[1], \mathbf{x}[2], \dots, \mathbf{x}[d])$

\nwarrow
d-dim vector

Notational conventions:

$\mathbf{x}[j] = j^{\text{th}}$ input (*scalar*)

$h_j(\mathbf{x}) = j^{\text{th}}$ feature (*scalar*)

$\mathbf{x}_i =$ input of i^{th} data point (*vector*)

$\mathbf{x}_i[j] = j^{\text{th}}$ input of i^{th} data point (*scalar*)

Simple hyperplane

18

Model: $\hat{y}_i = \text{sign}(\text{Score}(\mathbf{x}_i))$

$$\text{Score}(\mathbf{x}_i) = w_0 + w_1 \mathbf{x}_i[1] + \dots + w_d \mathbf{x}_i[d] = \mathbf{w}^T \mathbf{x}_i$$

feature 1 = 1

feature 2 = $\mathbf{x}[1]$... e.g., #awesome

feature 3 = $\mathbf{x}[2]$... e.g., #awful

...

feature $d+1$ = $\mathbf{x}[d]$... e.g., #ramen

D-dimensional hyperplane

19

More generic features...

Model: $\hat{y}_i = \text{sign}(\text{Score}(\mathbf{x}_i))$

$\text{Score}(\mathbf{x}_i) = w_0 h_0(\mathbf{x}_i) + w_1 h_1(\mathbf{x}_i) + \dots + w_D h_D(\mathbf{x}_i)$

$$= \sum_{j=0}^D w_j h_j(\mathbf{x}_i) = \mathbf{w}^T \mathbf{h}(\mathbf{x}_i)$$

feature 1 = $h_0(\mathbf{x})$... e.g., 1

feature 2 = $h_1(\mathbf{x})$... e.g., $\mathbf{x}[1] = \text{\#awesome}$

feature 3 = $h_2(\mathbf{x})$... e.g., $\mathbf{x}[2] = \text{\#awful}$

or, $\log(\mathbf{x}[7]) \mathbf{x}[2] = \log(\text{\#bad}) \times \text{\#awful}$

or, $\text{tf-idf}(\text{"awful"})$

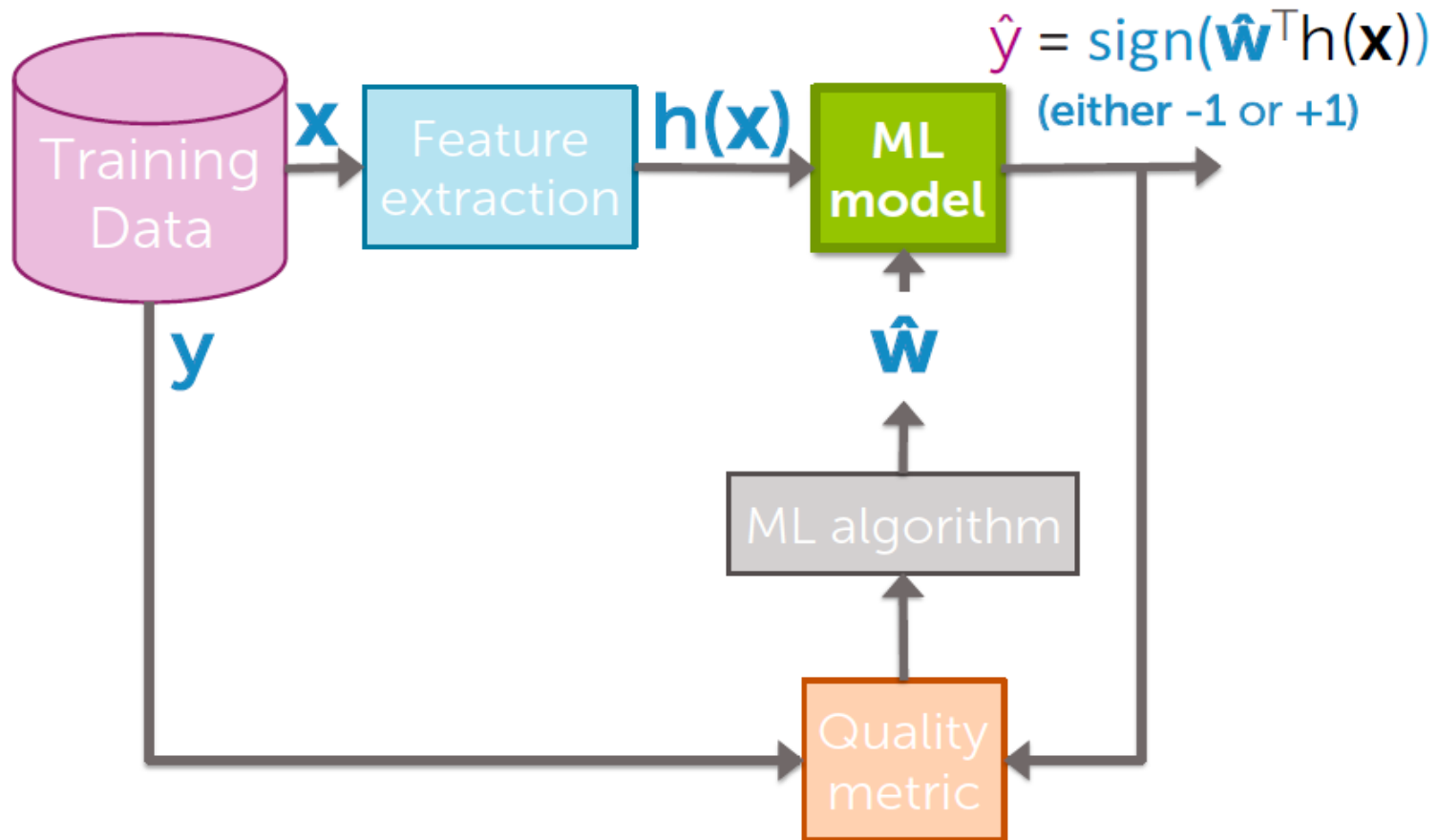
...

feature $D+1 = h_D(\mathbf{x})$... some other function of $\mathbf{x}[1], \dots, \mathbf{x}[d]$

Flow chart:

ML
model

20



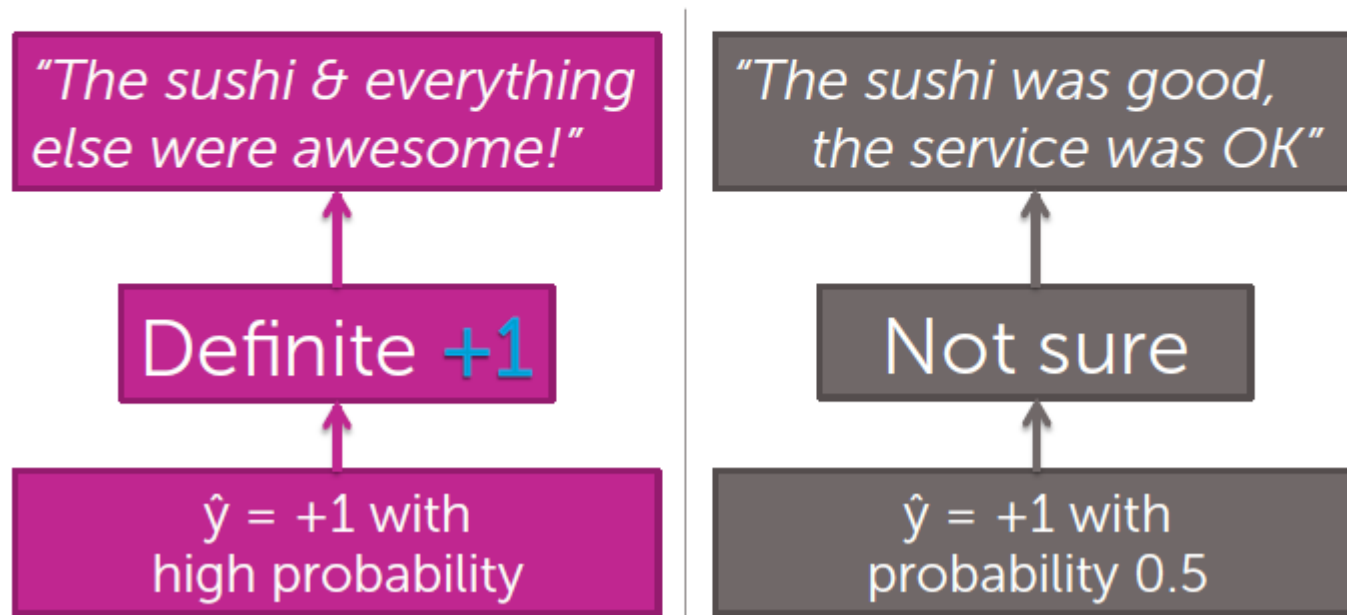
Linear classifier

▣ Class probability

How confident is your prediction?

22

- Thus far, we've outputted a prediction **+1** or **-1**
- But, how sure are you about the prediction?



Basics of probabilities

23

Probability a review is positive is 0.7

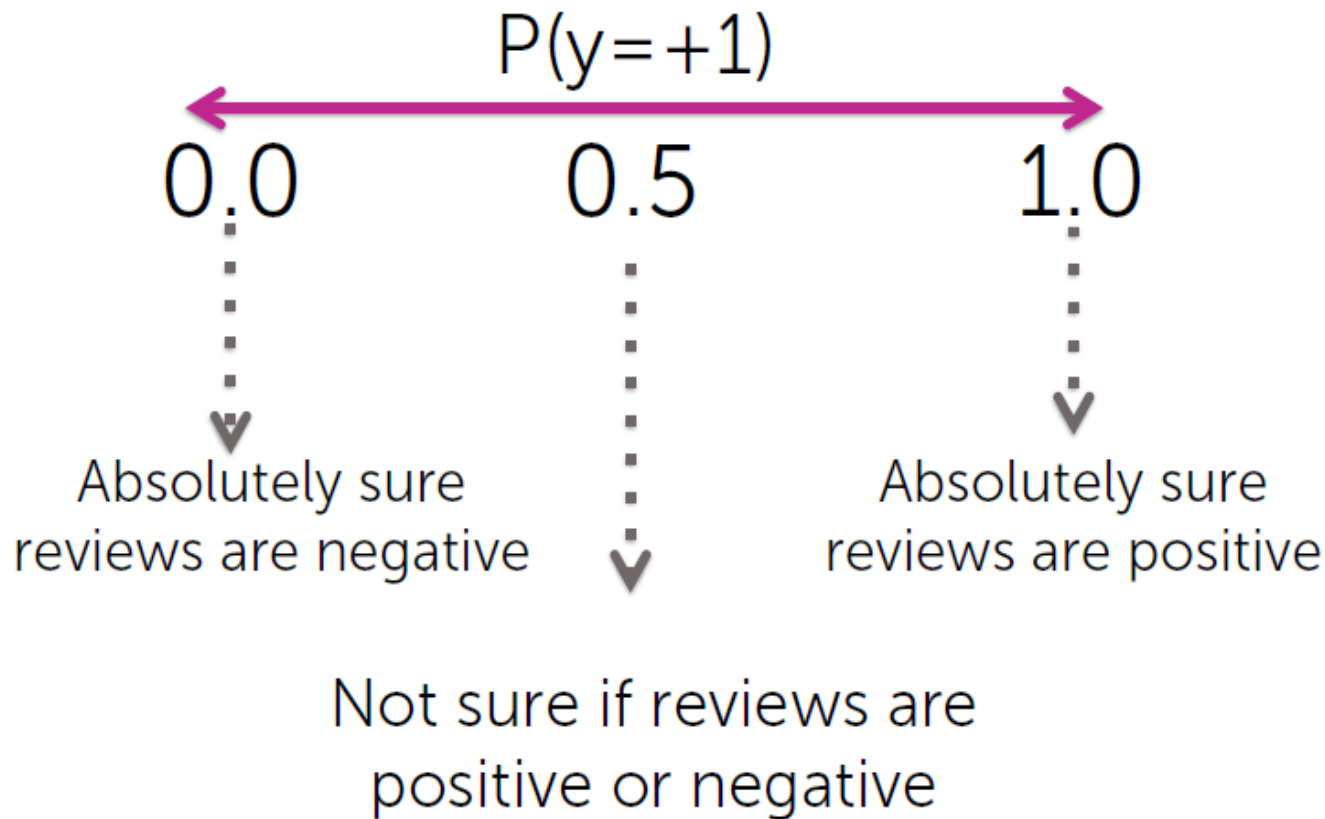


x = review text	y = sentiment
All the sushi was delicious! Easily best sushi in Seattle.	+1
The sushi & everything else were awesome!	+1
My wife tried their ramen, it was pretty forgettable.	-1
The sushi was good, the service was OK	+1
...	...

I expect 70% of rows
to have $y = +1$
(Exact number will vary
for each specific dataset)

Interpreting probabilities as degrees of belief


24




Conditional probability

25

Probability a review with
3 "awesome" and 1 "awful" is positive is 0.9



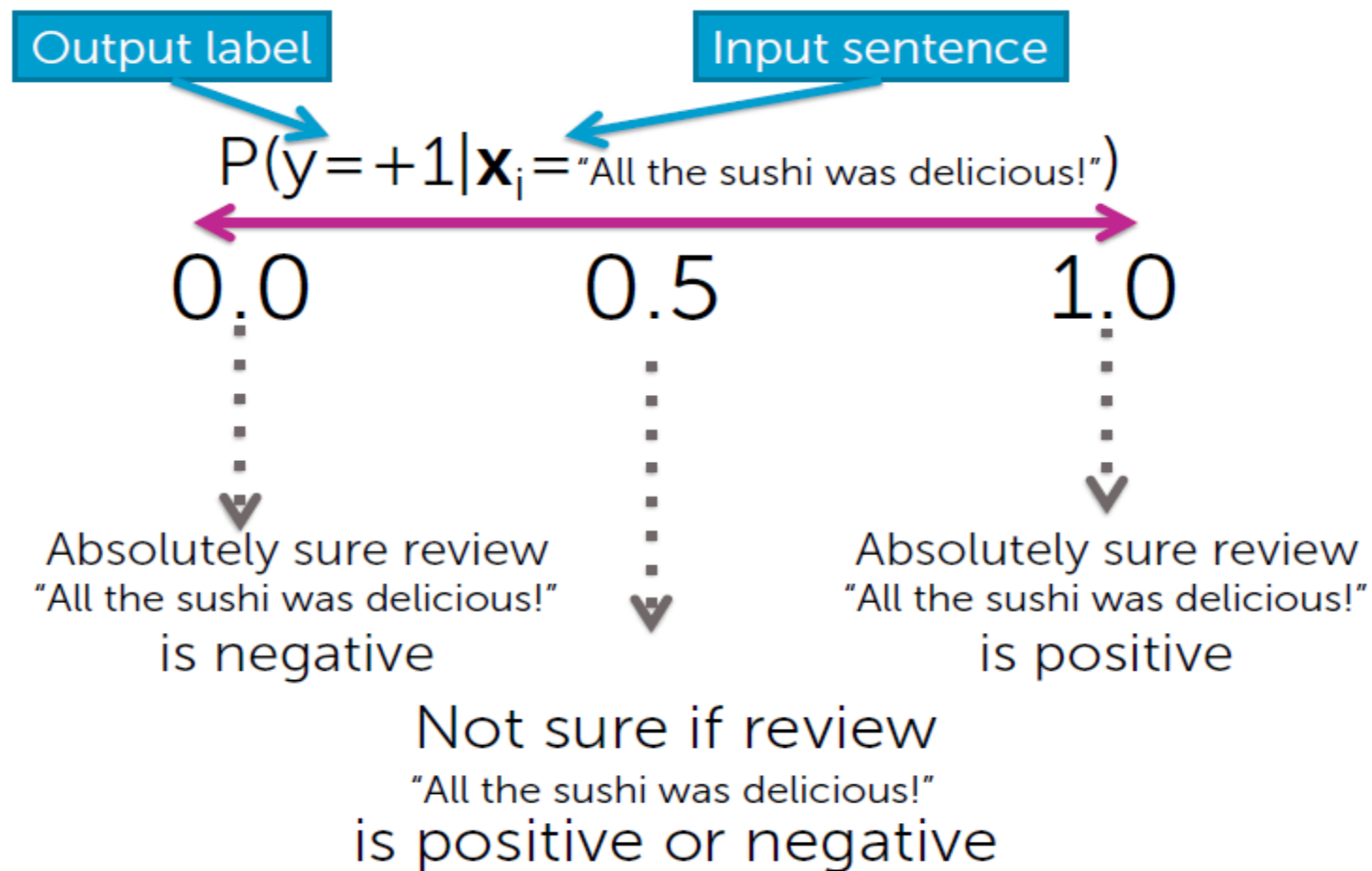
x = review text	y = sentiment
All the sushi was delicious! Easily best sushi in Seattle.	+1
Sushi was awesome & everything else was awesome ! The service was awful , but overall awesome place!	+1
My wife tried their ramen, it was pretty forgettable.	-1
The sushi was good, the service was OK	+1
...	...
awesome ... awesome ... awful ... awesome	+1
...	...
awesome ... awesome ... awful ... awesome	-1
...	...
...	...
awesome ... awesome ... awful ... awesome	+1



I expect 90% of rows with
reviews containing
3 "awesome" & 1 "awful"
to have $y = +1$
(Exact number will vary
for each specific dataset)

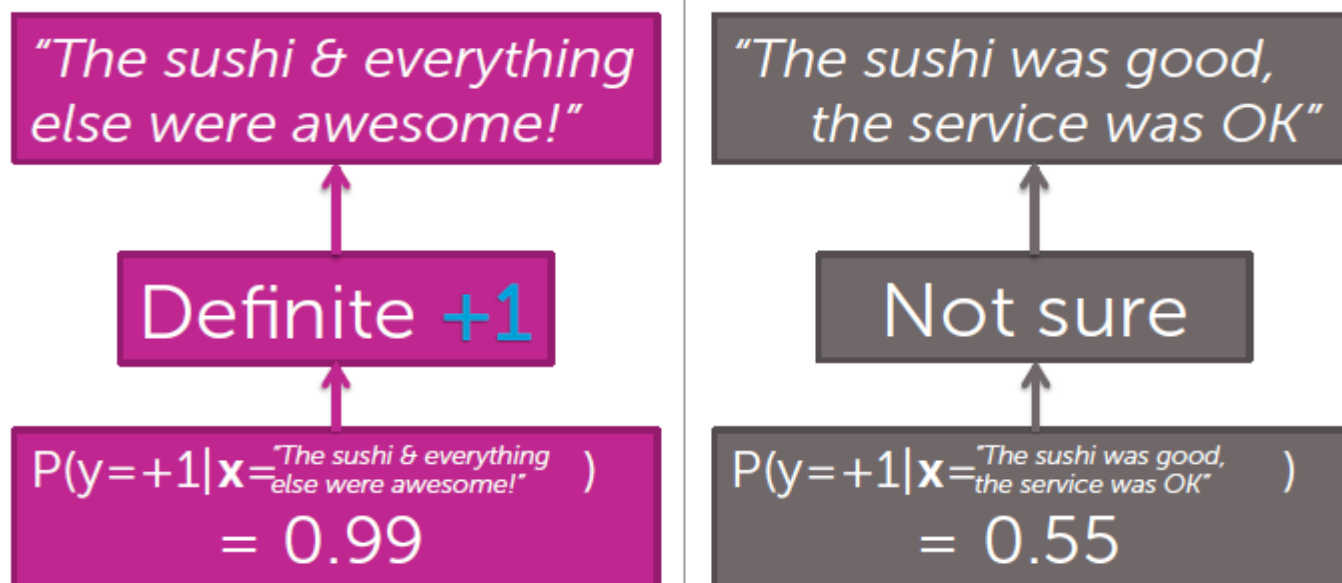
Interpreting conditional probabilities

26



How confident is your prediction?

27



Many classifiers provide a degree of certainty:



Learn conditional probabilities from data

28

Training data: N observations (\mathbf{x}_i, y_i)

$x[1] = \text{\#awesome}$	$x[2] = \text{\#awful}$	$y = \text{sentiment}$
2	1	+1
0	2	-1
3	3	-1
4	1	+1
...

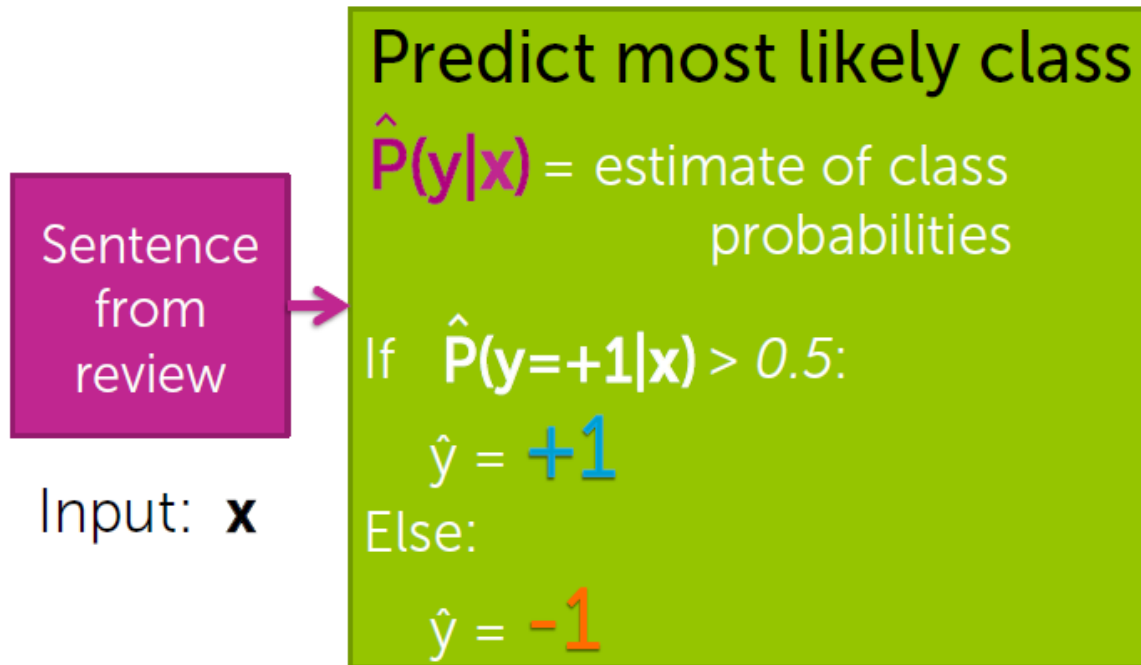
Optimize **quality metric**
on training data

Find best model $\hat{\mathbf{p}}$
by finding best $\hat{\mathbf{w}}$

Useful for
predicting \hat{y}

Predicting class probabilities

29

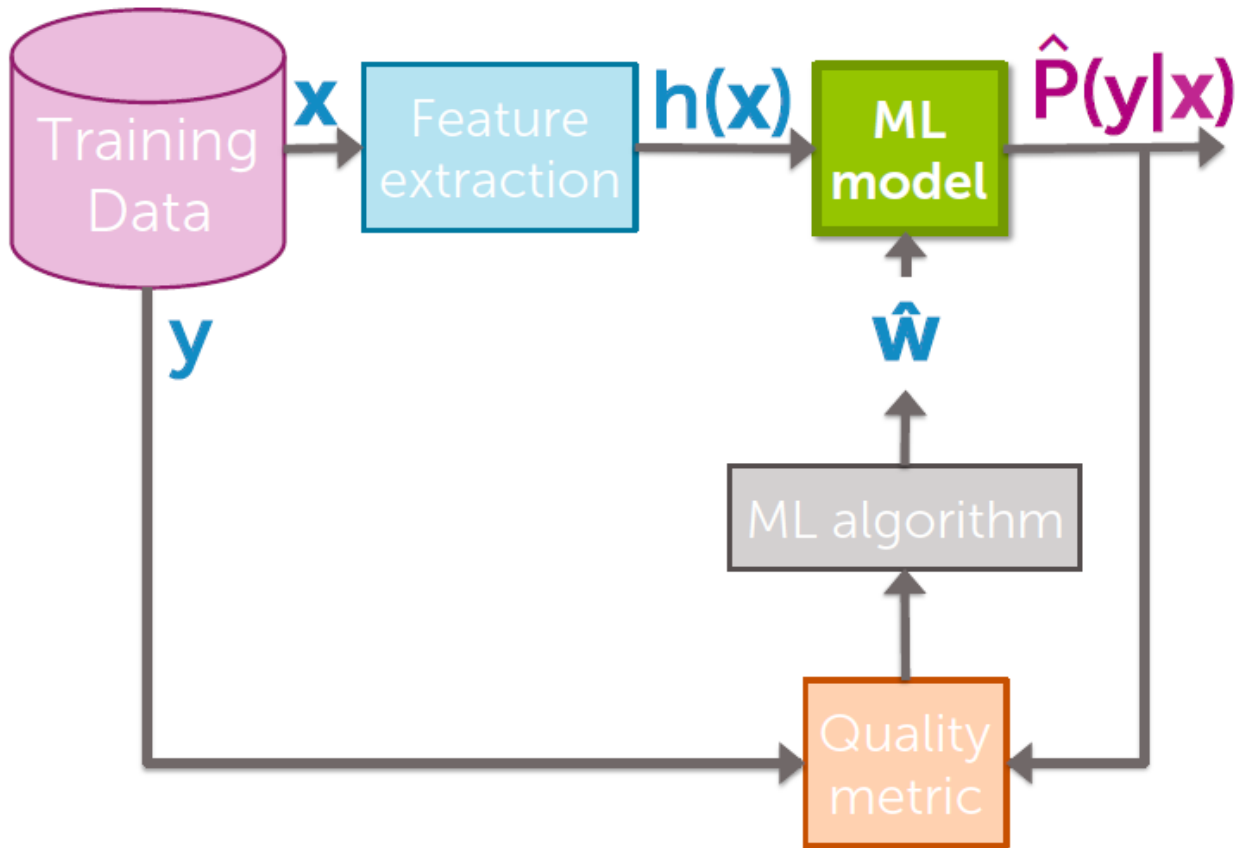


- Estimating $\hat{\mathbf{P}}(\mathbf{y}|\mathbf{x})$ improves **interpretability**:
 - Predict $\hat{\mathbf{y}} = +1$ **and** tell me how sure you are

Flow chart:

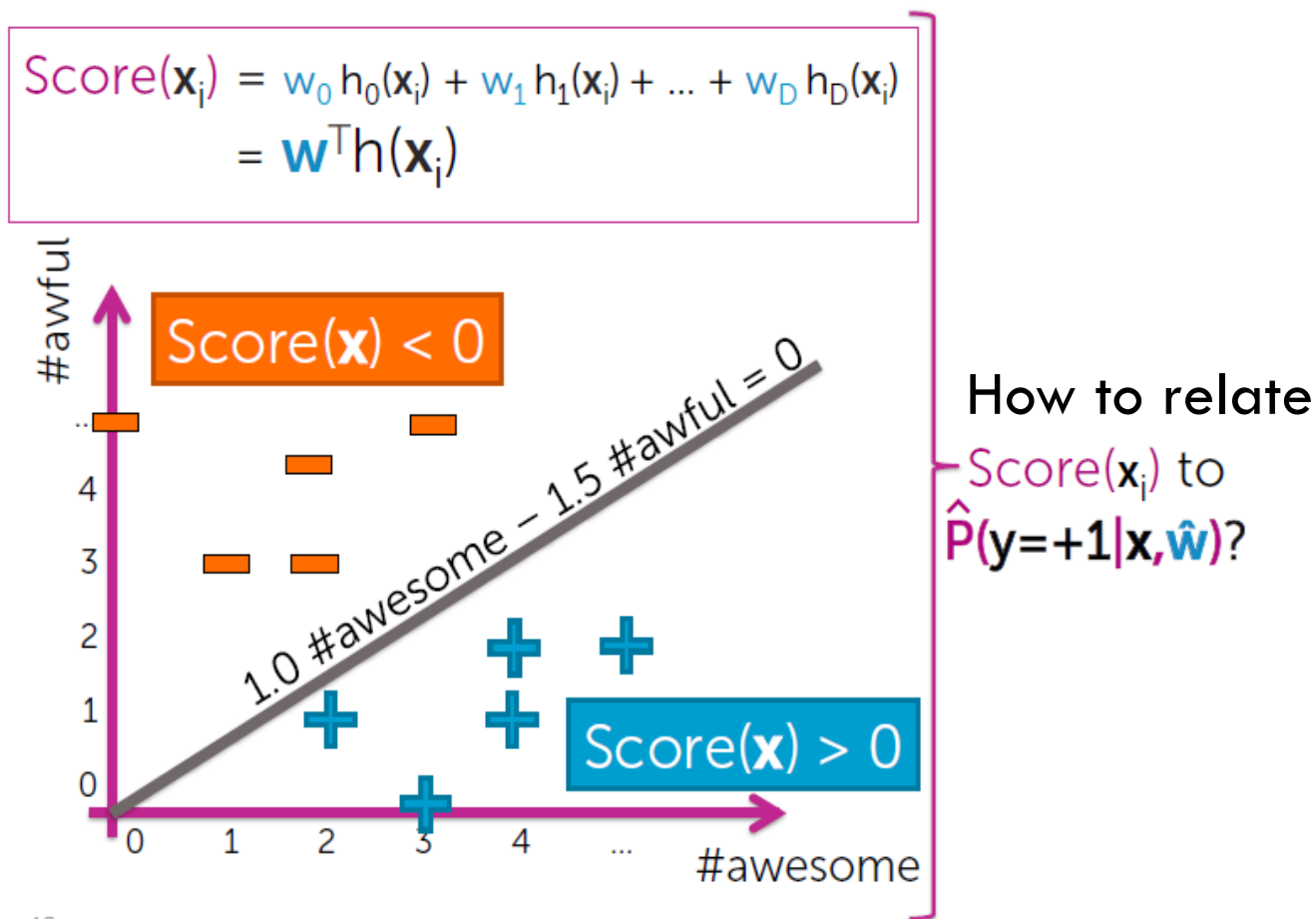
ML
model

30



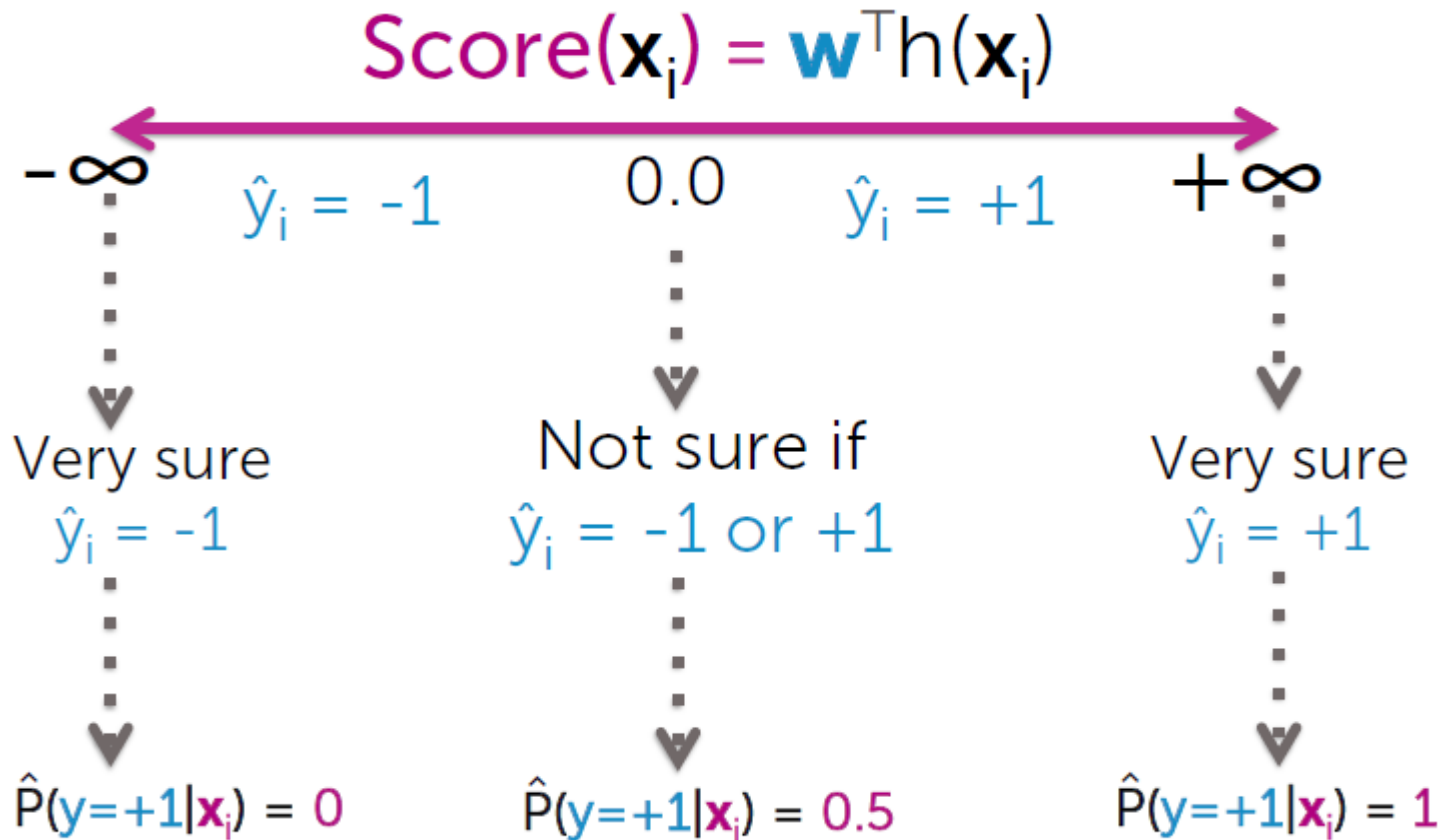
Thus far we focused on decision boundaries

31



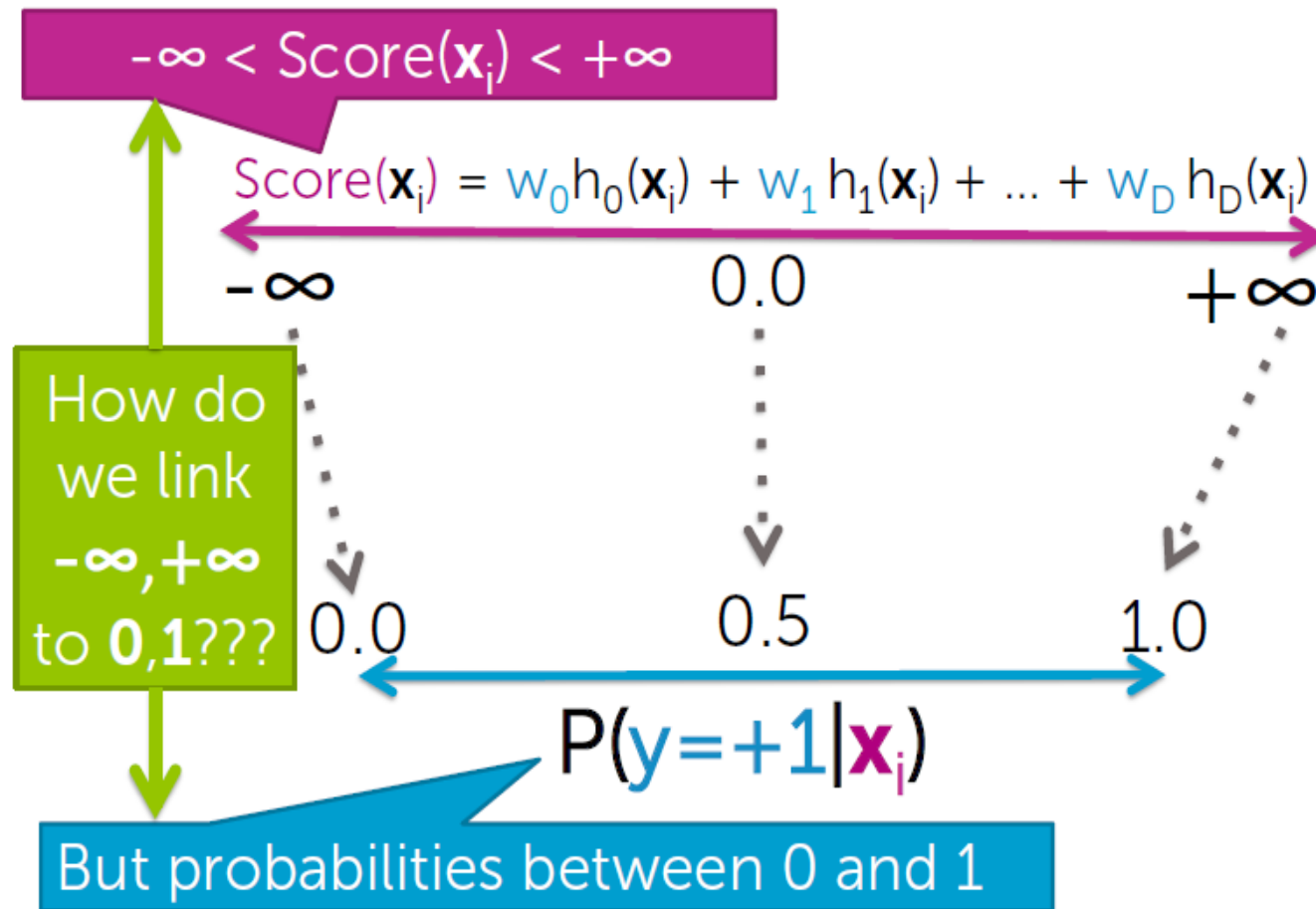
Interpreting Score(\mathbf{x}_i)

32



Why not just use regression to build classifier?

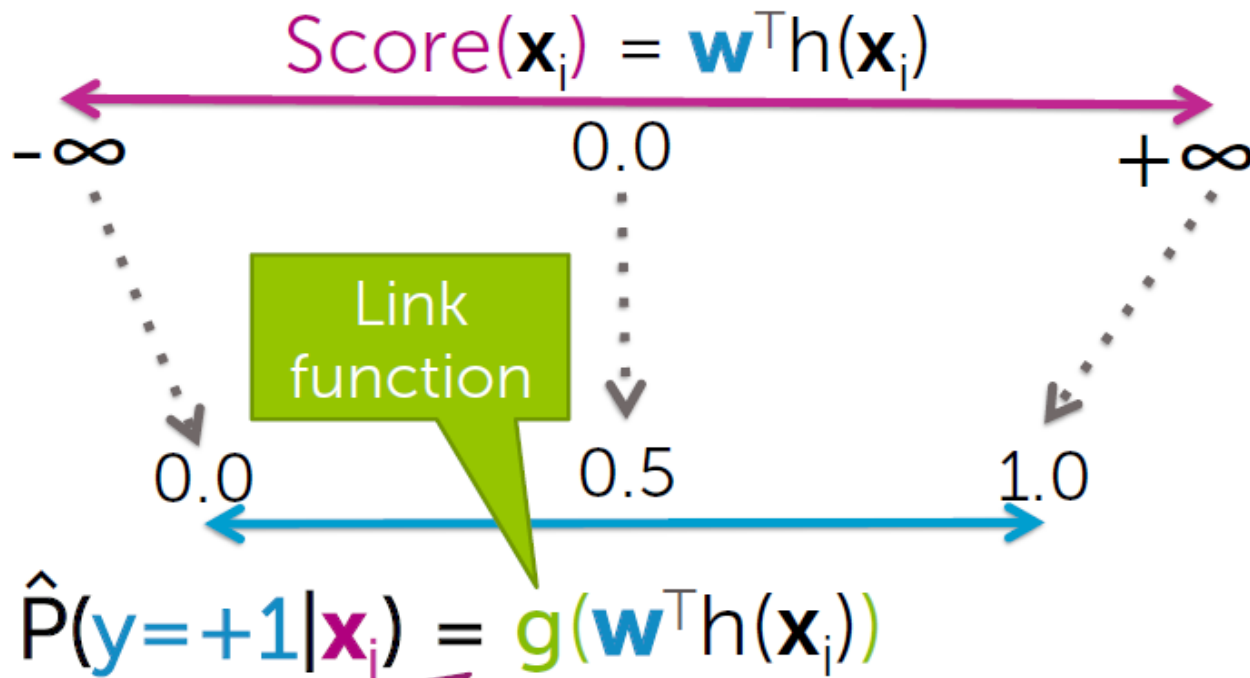
33



Link function

34

Link function: squeeze real line into [0,1]

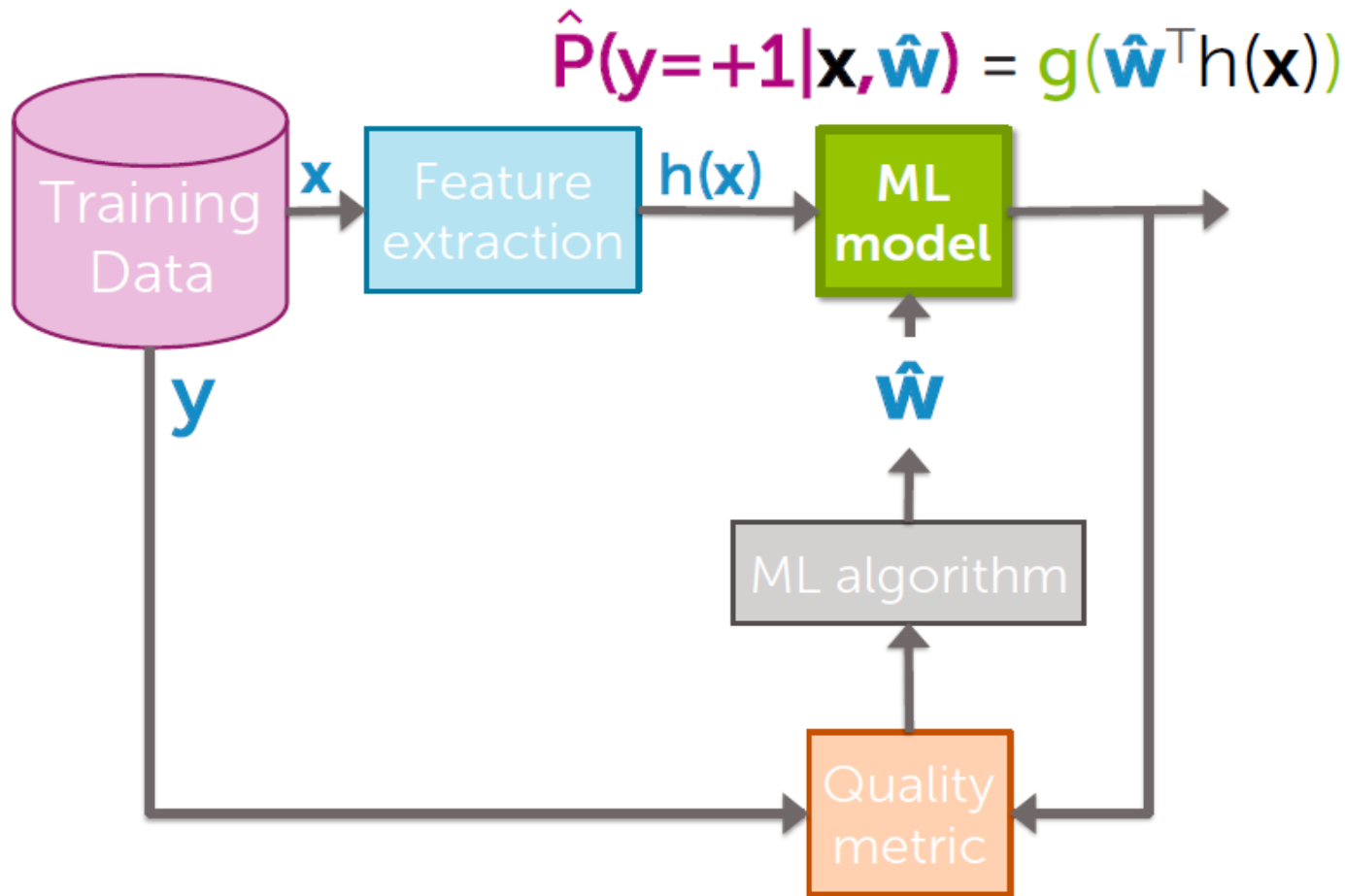


Generalized linear model

Flow chart:

ML
model

35

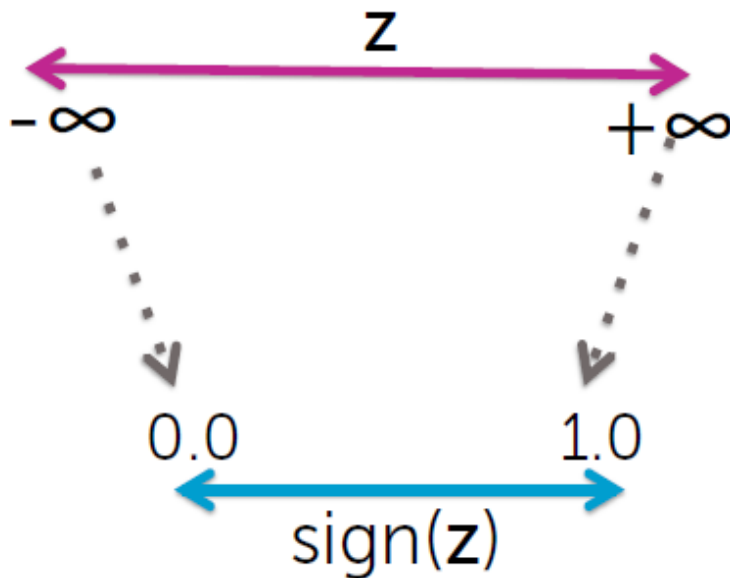


Logistic regression classifier:

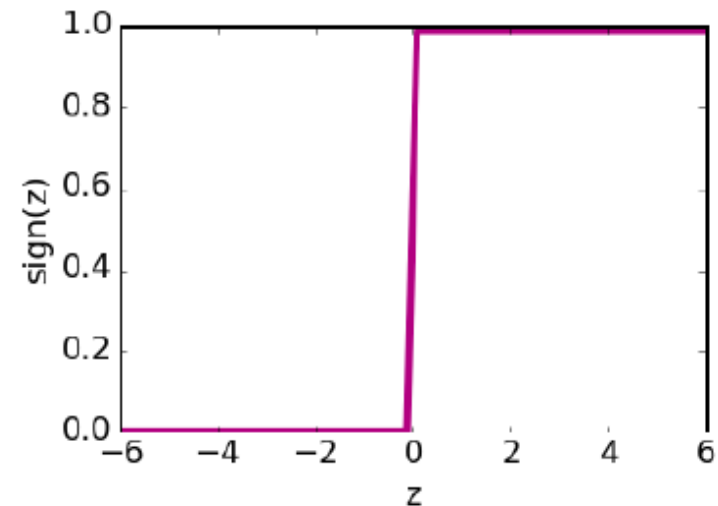
- ▣ linear score with logistic link function

Simplest link function: $\text{sign}(z)$

37



$$\text{sign}(z) = \begin{cases} +1 & \text{if } z \geq 0 \\ -1 & \text{otherwise} \end{cases}$$



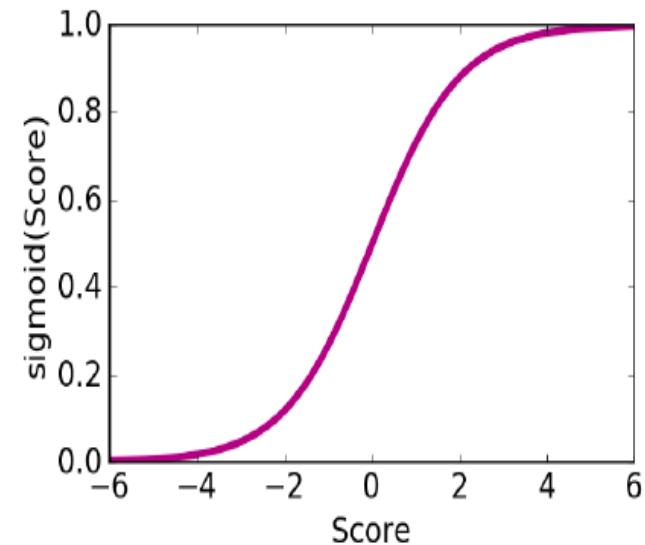
But, $\text{sign}(z)$ only outputs -1 or +1,
no probabilities in between

Logistic function (sigmoid, logit)

38

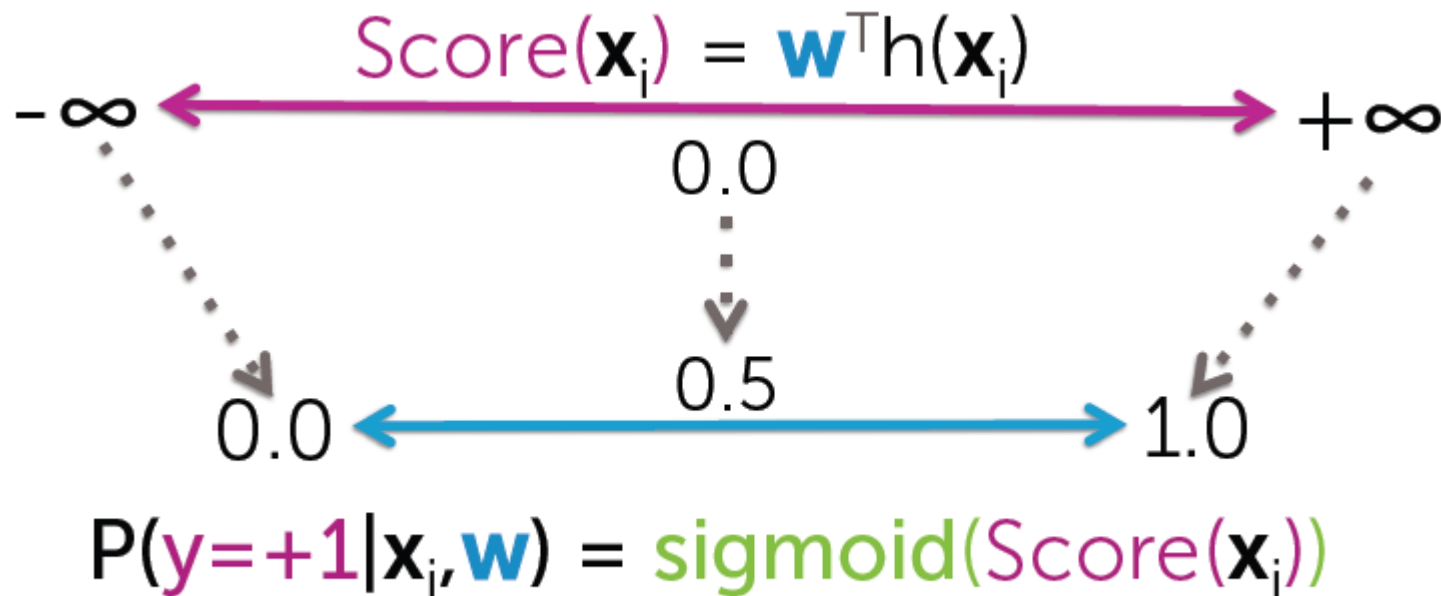
$$\text{sigmoid}(\text{Score}) = \frac{1}{1 + e^{-\text{Score}}}$$

Score	$-\infty$	-2	0.0	+2	$+\infty$
sigmoid(Score)	0.0	0.12	0.5	0.88	1.0



Logistic regression model

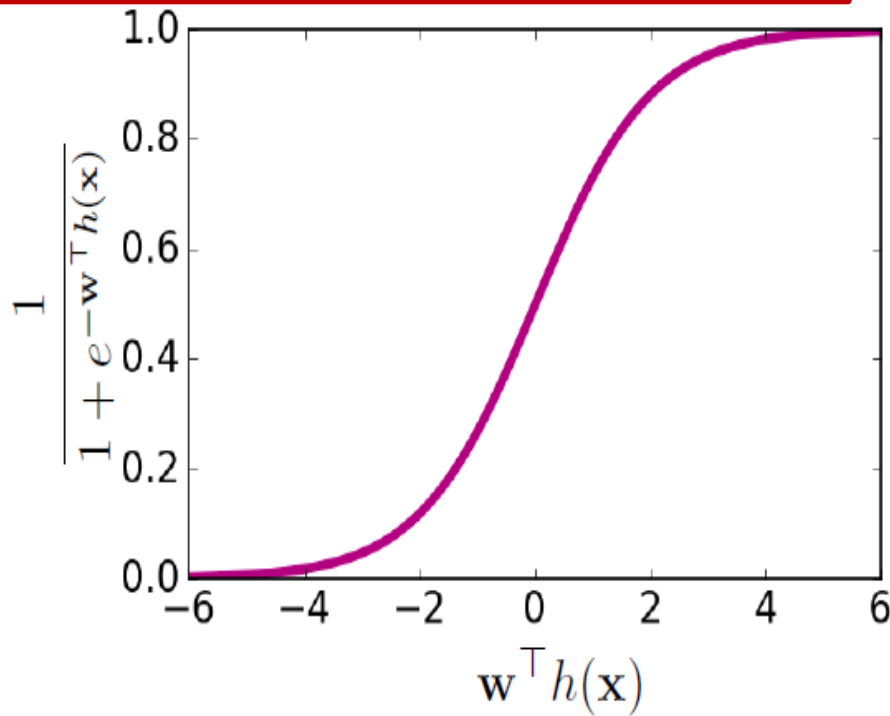
39



Understanding the logistic regression model

40

$$P(y = +1 \mid \mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{h}(\mathbf{x})}}$$

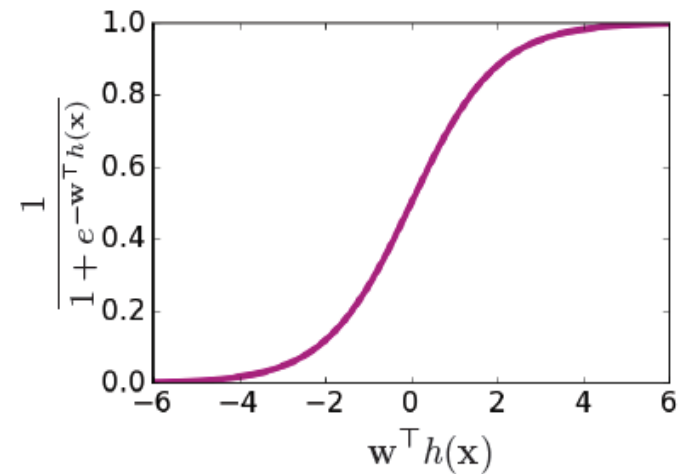
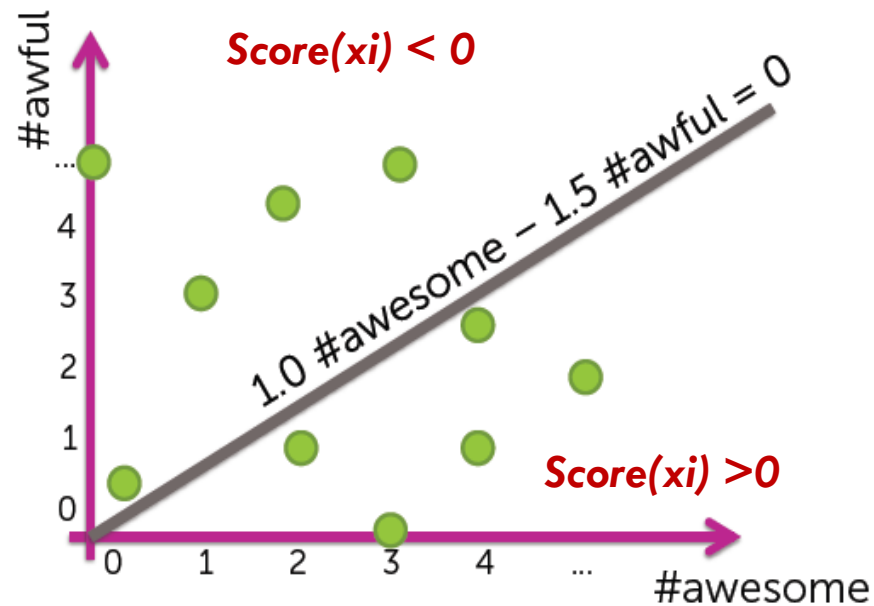


Score(\mathbf{x}_i)	$P(y=+1 \mathbf{x}_i, \mathbf{w})$
0	0.5
-2	0.12
2	0.88
4	0.98

Logistic regression

41

Logistic regression →
Linear decision boundary

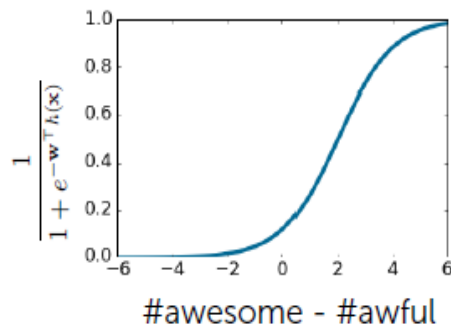


Effect of coefficients

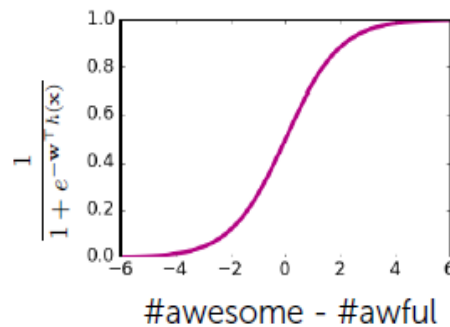
42

Effect of coefficients on logistic regression model

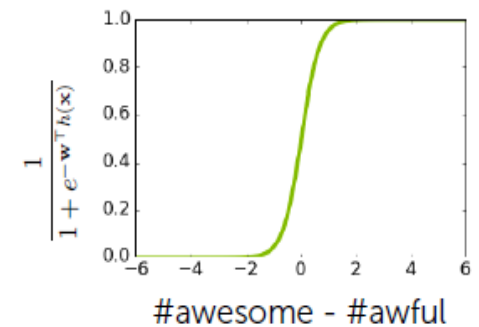
w_0	-2
$w_{\text{\#awesome}}$	+1
$w_{\text{\#awful}}$	-1



w_0	0
$w_{\text{\#awesome}}$	+1
$w_{\text{\#awful}}$	-1



w_0	0
$w_{\text{\#awesome}}$	+3
$w_{\text{\#awful}}$	-3

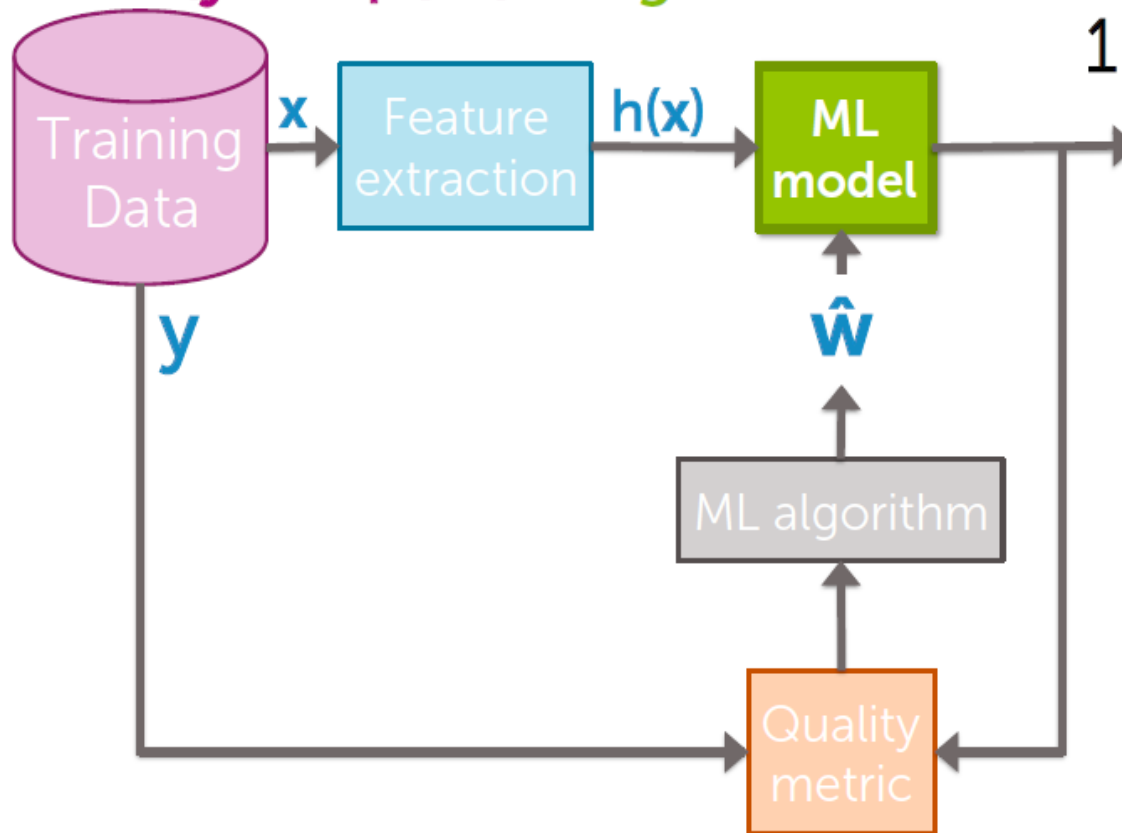


Flow chart:

ML
model

43

$$\hat{P}(y=+1|\mathbf{x},\hat{\mathbf{w}}) = \text{sigmoid}(\hat{\mathbf{w}}^T \mathbf{h}(\mathbf{x})) = \frac{1}{1 + e^{-\hat{\mathbf{w}}^T \mathbf{h}(\mathbf{x})}}$$



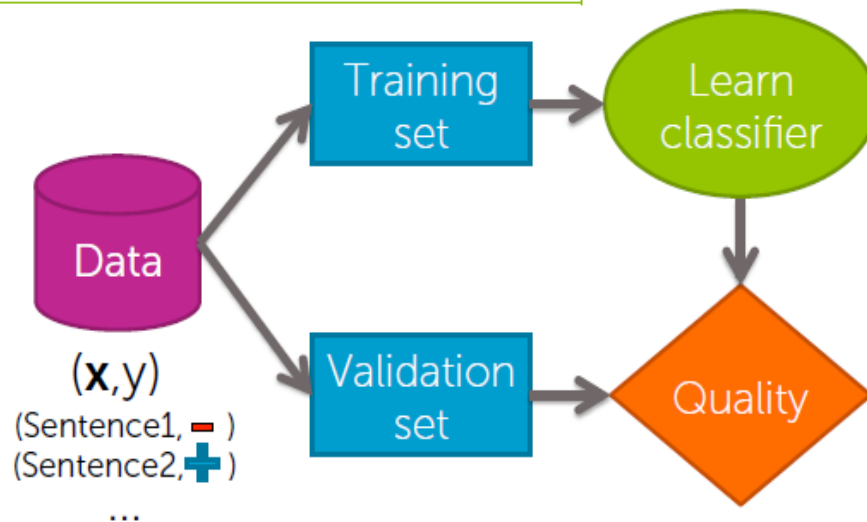
Learning logistic regression model

44

Training a classifier = Learning the coefficients

Word	Coefficient	Value
	\hat{w}_0	-2.0
good	\hat{w}_1	1.0
awesome	\hat{w}_2	1.7
bad	\hat{w}_3	-1.0
awful	\hat{w}_4	-3.3
...

$$\hat{P}(y=+1|\mathbf{x},\hat{\mathbf{w}}) = \frac{1}{1 + e^{-\hat{\mathbf{w}}^T \mathbf{h}(\mathbf{x})}}$$



Categorical inputs

45

- Numeric inputs:
 - #awesome, age, salary,...
 - Intuitive when multiplied by coefficient
 - e.g., 1.5 #awesome

Numeric value, but should be interpreted as category
(98195 not about 9x larger than 10005)

- Categorical inputs:



Gender
(Male, Female,...)



Country of birth
(Argentina, Brazil, USA,...)



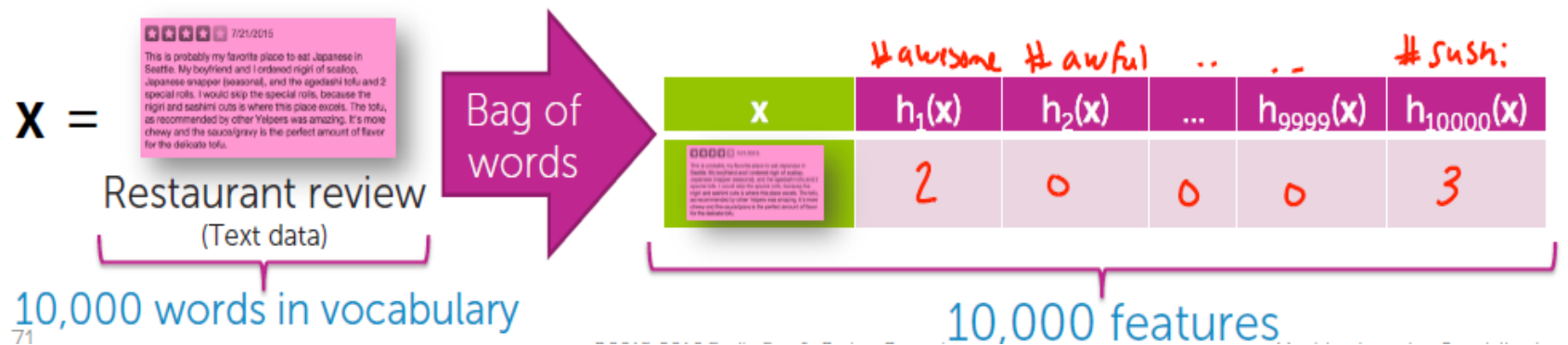
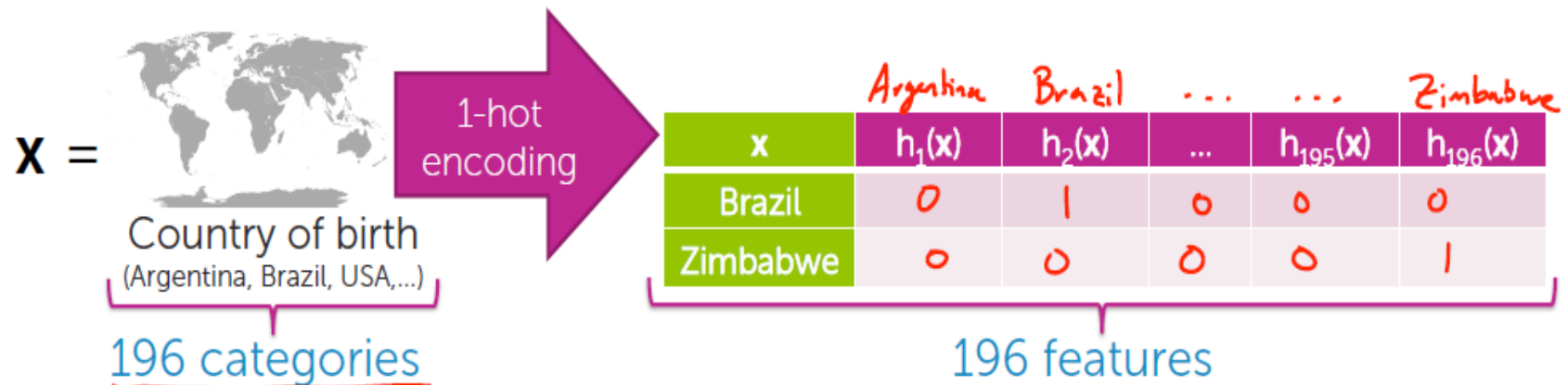
Zipcode
(10005, 98195,...)

How do we multiply category by coefficient???

Must convert categorical inputs into numeric features

Encoding categories as numeric features

46



Multiclass classification

47



Input: x
Image pixels



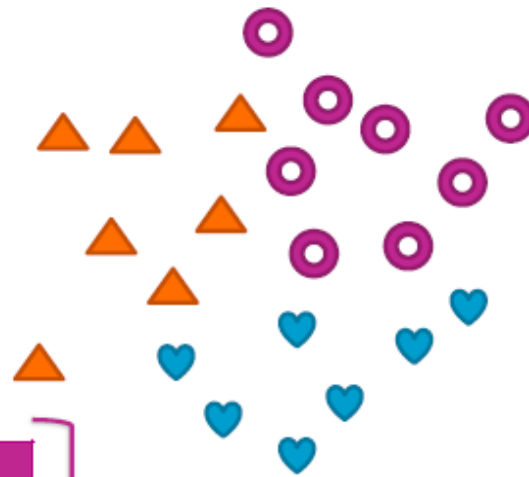
Output: y
Object in image

Multiclass classification

48

- C possible classes:
 - y can be 1, 2,..., C
- N datapoints:

Data point	x[1]	x[2]	y
\mathbf{x}_1, y_1	2	1	▲
\mathbf{x}_2, y_2	0	2	♥
\mathbf{x}_3, y_3	3	3	◯
\mathbf{x}_4, y_4	4	1	◯



Learn:

$$\hat{P}(y = \text{▲} | \mathbf{x})$$

$$\hat{P}(y = \text{♥} | \mathbf{x})$$

$$\hat{P}(y = \text{◯} | \mathbf{x})$$

1 versus all

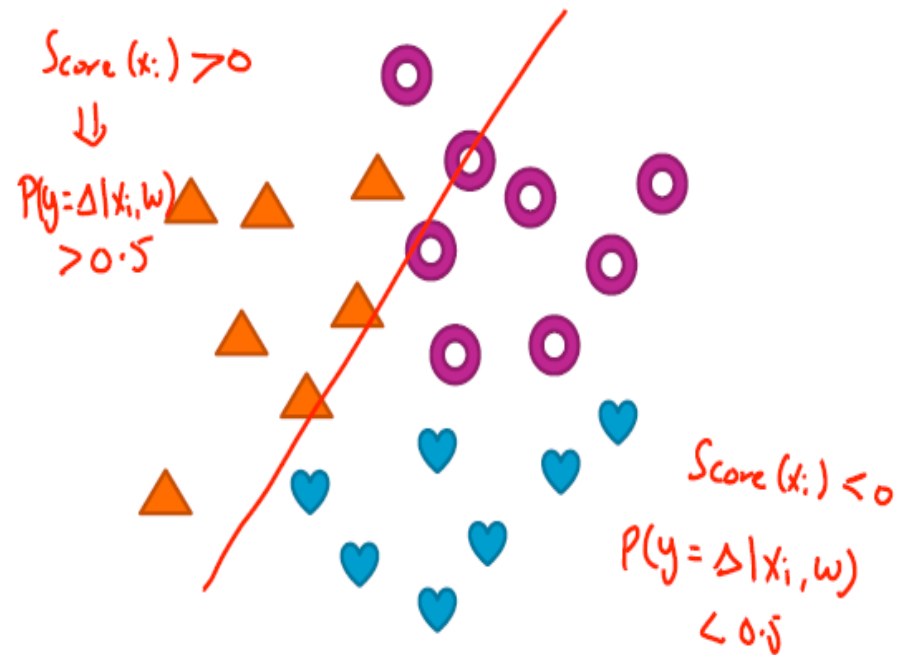
49

Estimate $\hat{P}(y=\triangle | \mathbf{x})$ using 2-class model

+1 class: points with $y_i = \triangle$
-1 class: points with $y_i = \heartsuit$ OR \bigcirc

Train classifier: $\hat{P}(y=\triangle | \mathbf{x})$

Predict: $\hat{P}(y=\triangle | \mathbf{x}_i) = \hat{P}(y=\triangle | \mathbf{x}_i)$

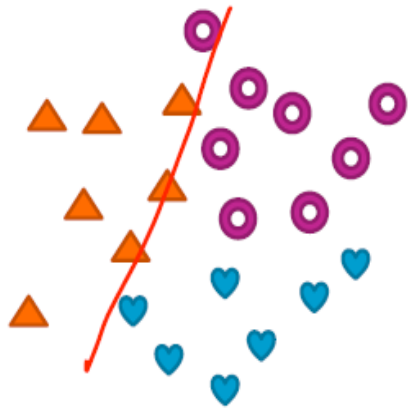


1 versus all

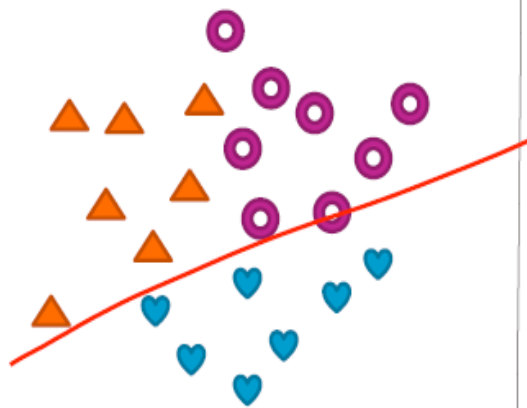
50

1 versus all: simple multiclass classification
using C 2-class models

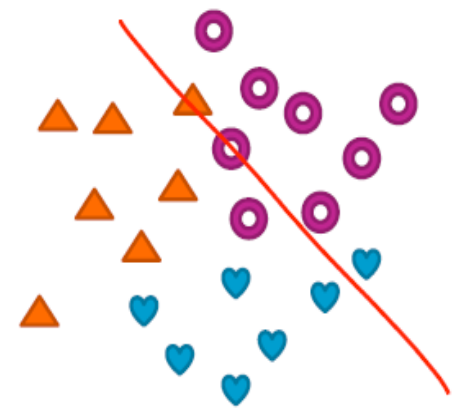
$$\hat{P}(y=\triangle | \mathbf{x}_i) = \hat{P}_\triangle(y=\triangle | \mathbf{x}_i, \mathbf{w})$$

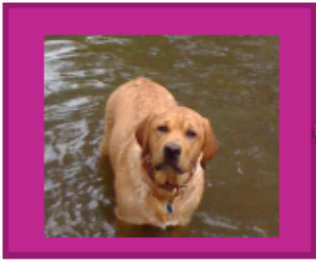


$$\hat{P}(y=\heartsuit | \mathbf{x}_i) = \hat{P}_\heartsuit(y=\heartsuit | \mathbf{x}_i, \mathbf{w})$$



$$\hat{P}(y=\circ | \mathbf{x}_i) = \hat{P}_\circ(y=\circ | \mathbf{x}_i, \mathbf{w})$$





Input: \mathbf{x}_i

Multiclass training

$\hat{P}_c(y=+1|\mathbf{x})$ = estimate of
1 vs all model for each class

Predict most likely class

$\text{max_prob} = 0; \hat{y} = 0$

For $c = 1, \dots, C$:

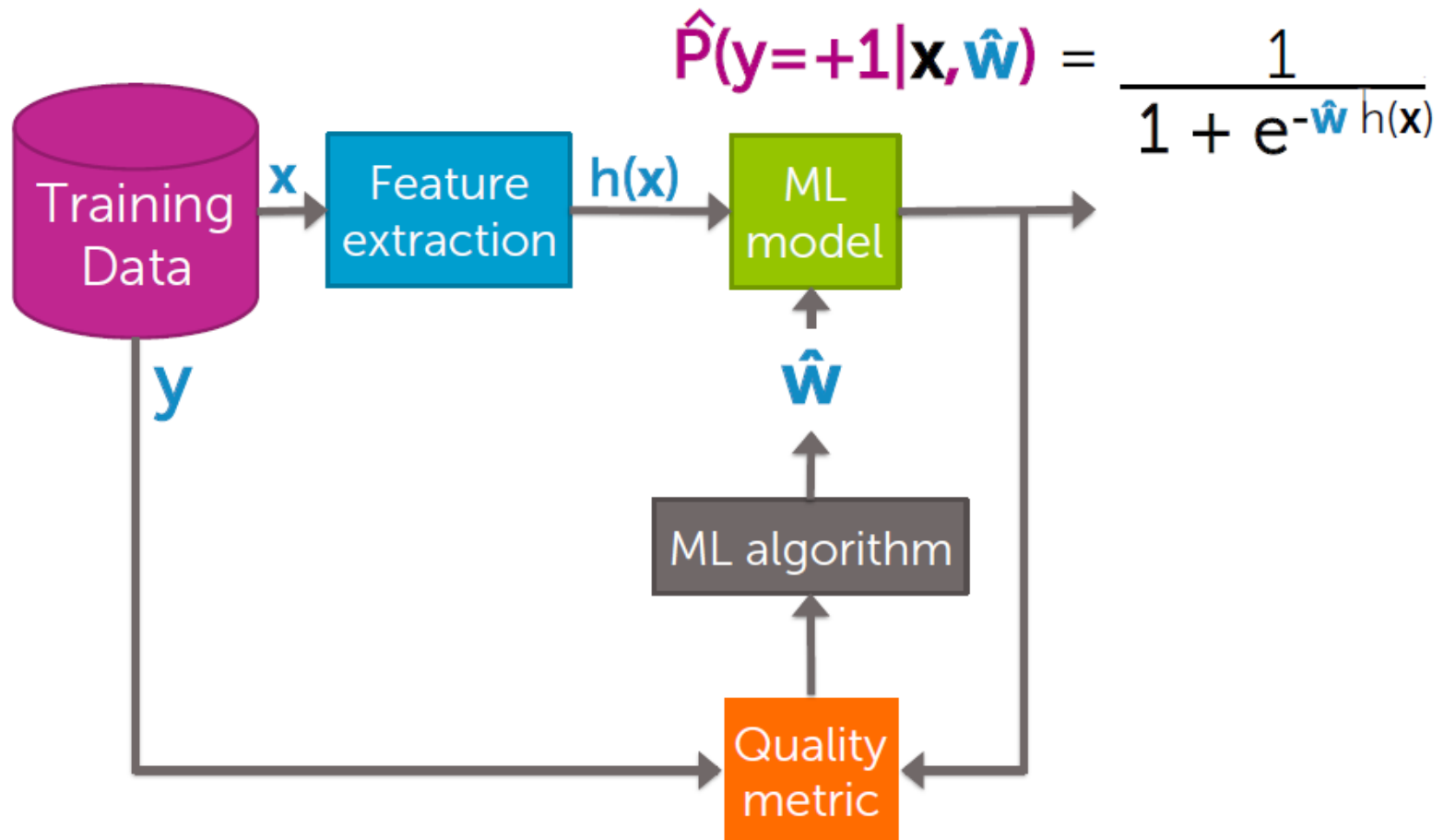
If $\hat{P}_c(y=+1|\mathbf{x}_i) > \text{max_prob}$:

$\hat{y} = c$

$\text{max_prob} = \hat{P}_c(y=+1|\mathbf{x}_i)$

Summary: Logistic regression classifier

52



What you can do now...

53

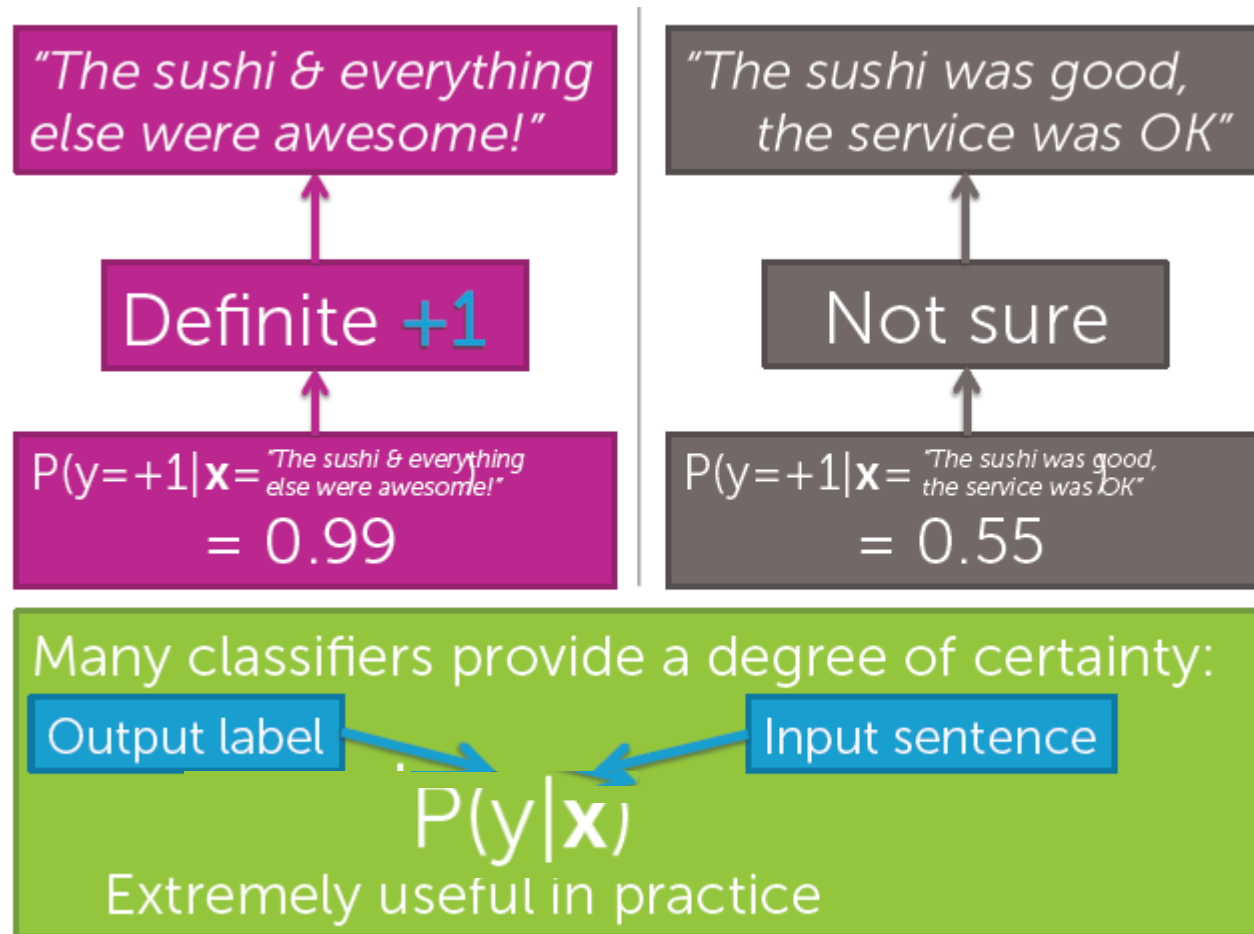
- Describe decision boundaries and linear classifiers
- Use class probability to express degree of confidence in prediction
- Define a logistic regression model
- Interpret logistic regression outputs as class probabilities
- Describe impact of coefficient values on logistic regression output
- Use 1-hot encoding to represent categorical inputs
- Perform multiclass classification using the 1-versus-all approach

Linear classifier

▣ Parameters learning

Learn a probabilistic classification model

55



A (linear) classifier

56

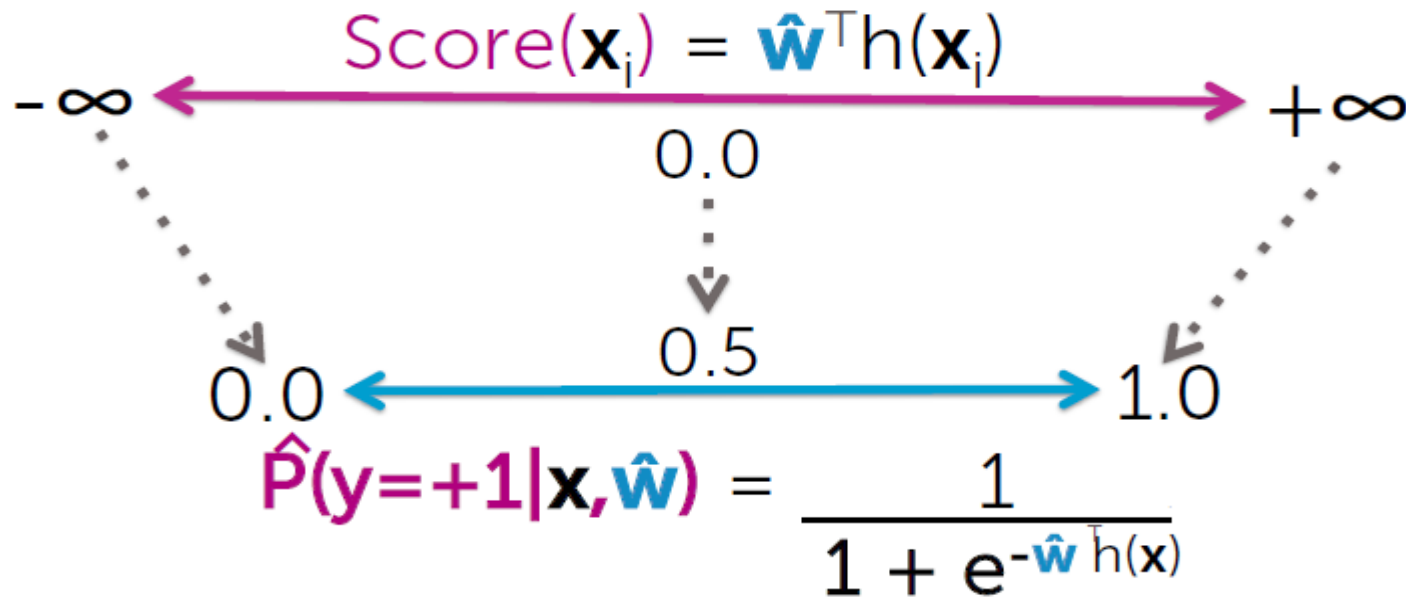
- Will use training data to learn a weight or coefficient for each word

Word	Coefficient	Value
	\hat{w}_0	-2.0
good	\hat{w}_1	1.0
great	\hat{w}_2	1.5
awesome	\hat{w}_3	2.7
bad	\hat{w}_4	-1.0
terrible	\hat{w}_5	-2.1
awful	\hat{w}_6	-3.3
restaurant, the, we, ...	$\hat{w}_7, \hat{w}_8, \hat{w}_9, \dots$	0.0
...		...

Logistic regression

57

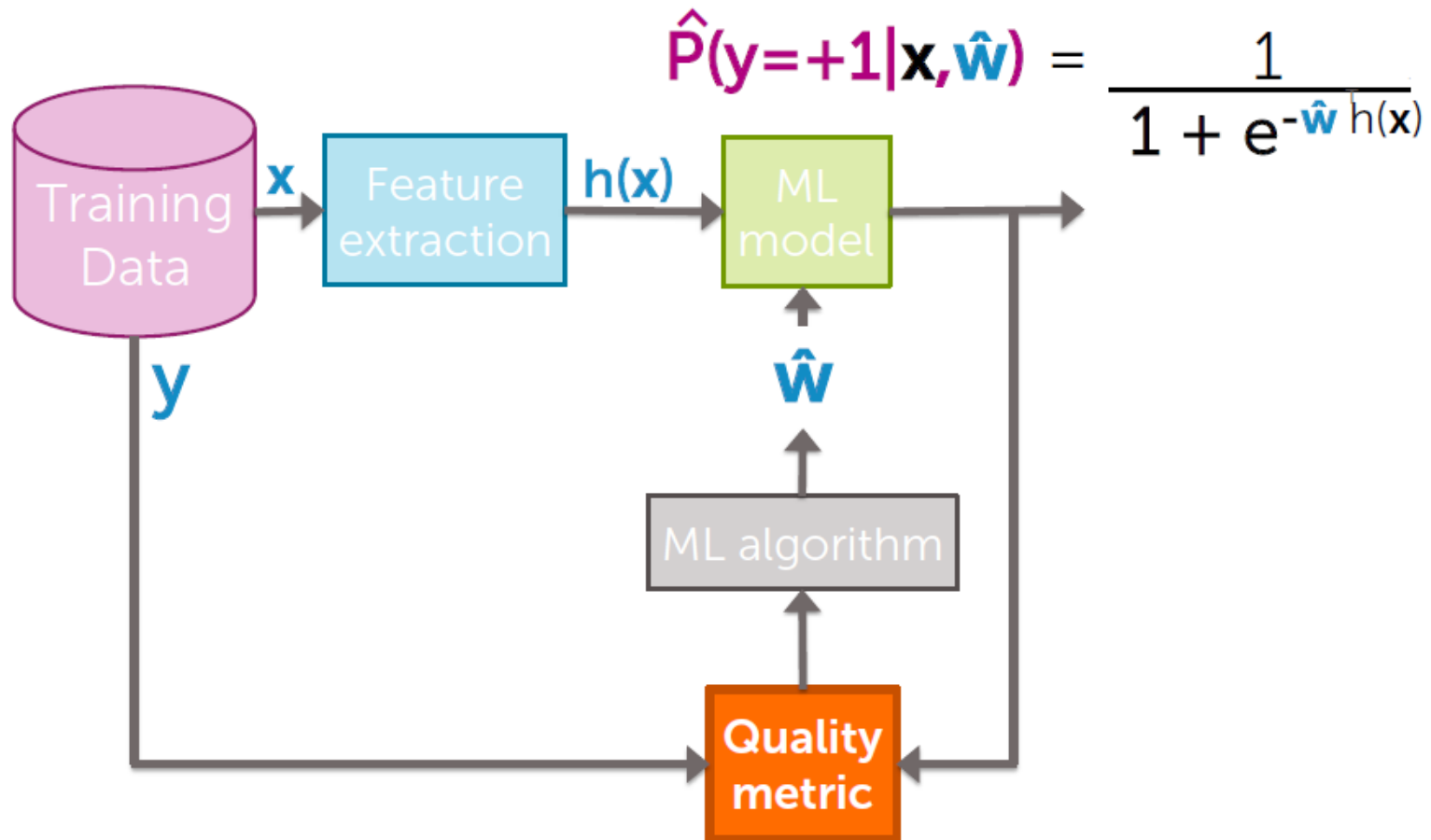
Logistic regression model



Flow chart:

Quality
metric

58



Learning problem

59

Training data:

N observations (\mathbf{x}_i, y_i)

$x[1] = \text{\#awesome}$	$x[2] = \text{\#awful}$	$y = \text{sentiment}$
2	1	+1
0	2	-1
3	3	-1
4	1	+1
1	1	+1
2	4	-1
0	3	-1
0	1	-1
2	1	+1

Optimize
quality metric
on training
data

$\hat{\mathbf{W}}$

Finding best coefficients

60

x[1] = #awesome	x[2] = #awful	y = sentiment
0	2	-1
3	3	-1
2	4	-1
0	3	-1
0	1	-1

$$P(y=+1|x_i, \mathbf{w}) = 0.0$$

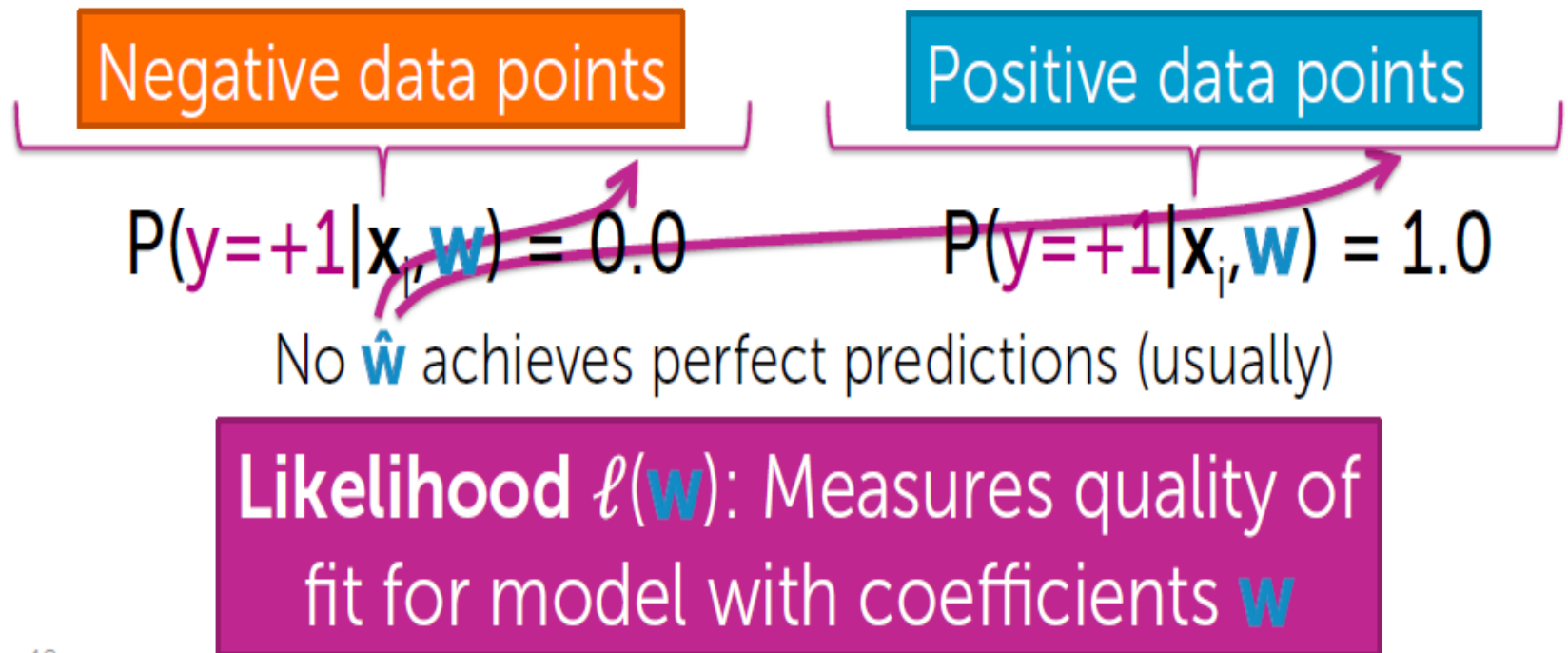
x[1] = #awesome	x[2] = #awful	y = sentiment
2	1	+1
4	1	+1
1	1	+1
2	1	+1

$$P(y=+1|x_i, \mathbf{w}) = 1.0$$

Pick $\hat{\mathbf{w}}$ that makes

Quality metric: likelihood function

61



12

Quality metric: probability of data

62

$x[1] = \text{\#awesome}$	$x[2] = \text{\#awful}$	$y = \text{sentiment}$
2	1	+1

x_1

y_1

If model good, should predict:

$\hat{y}_1 = +1$

Pick w to maximize:

$$P(y = +1 | x_1, w) = P(y = +1 | x[1]=2, x[2]=1, w)$$

$x[1] = \text{\#awesome}$	$x[2] = \text{\#awful}$	$y = \text{sentiment}$
0	2	-1

x_2

y_2

If model good, should predict:

$\hat{y}_2 = -1$

Pick w to maximize:

$$P(y = -1 | x_2, w)$$

15

Maximizing likelihood (probability of data)

63

Data point	x[1]	x[2]	y	Choose w to maximize
x_1, y_1	2	1	+1	$P(y=+1 x_1, w) = P(y=+1 x_0]=2, x_0]=1, w)$
x_2, y_2	0	2	-1	$P(y=-1 x_2, w)$
x_3, y_3	3	3	<u>-1</u>	$P(y=-1 x_3, w)$
x_4, y_4	4	1	<u>+1</u>	$P(y=+1 x_4, w)$
x_5, y_5	1	1	+1	
x_6, y_6	2	4	-1	
x_7, y_7	0	3	-1	
x_8, y_8	0	1	-1	
x_9, y_9	2	1	+1	

Must combine into single measure of quality ?

Multiply probabilities

$P(y=+1|x_1, w) P(y=-1|x_2, w) P(y=-1|x_3, w) \dots$

Maximum likelihood estimation (MLE)

64

Learn logistic regression model with MLE

Data point	x[1]	x[2]	y	Choose \mathbf{w} to maximize
\mathbf{x}_1, y_1	2	1	$y_1 = +1$	$P(y=+1 x[1]=2, x[2]=1, \mathbf{w})$
\mathbf{x}_2, y_2	0	2	-1	$P(y=-1 x[1]=0, x[2]=2, \mathbf{w})$
\mathbf{x}_3, y_3	3	3	-1	$P(y=-1 x[1]=3, x[2]=3, \mathbf{w})$
\mathbf{x}_4, y_4	4	1	$+1$	$P(y=+1 x[1]=4, x[2]=1, \mathbf{w})$

$$\ell(\mathbf{w}) = \underbrace{P(y=+1|x[1]=2, x[2]=1, \mathbf{w})}_{P(y_1|\mathbf{x}_1, \mathbf{w})} \underbrace{P(y=-1|x[1]=0, x[2]=2, \mathbf{w})}_{P(y_2|\mathbf{x}_2, \mathbf{w})} \underbrace{P(y=-1|x[1]=3, x[2]=3, \mathbf{w})}_{P(y_3|\mathbf{x}_3, \mathbf{w})} \underbrace{P(y=+1|x[1]=4, x[2]=1, \mathbf{w})}_{P(y_4|\mathbf{x}_4, \mathbf{w})}$$

Num. of data points N

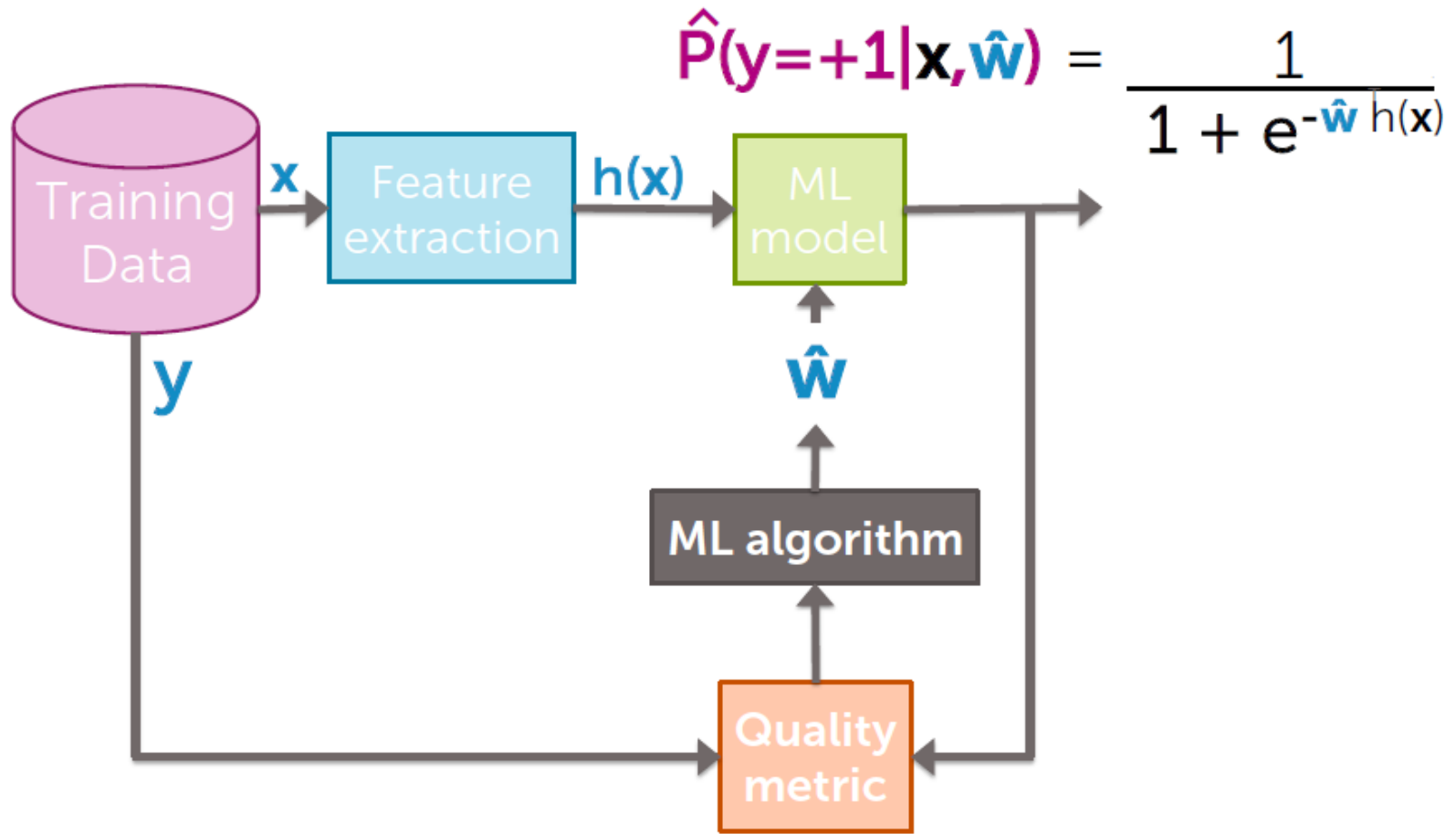
$$\ell(\mathbf{w}) = \prod_{i=1}^N P(y_i | \mathbf{x}_i, \mathbf{w}) \quad \leftarrow \text{pick } \mathbf{w} \text{ to make this fn. as large as possible}$$

17

Flow chart:

ML algorithm

65



Find „best” classifier

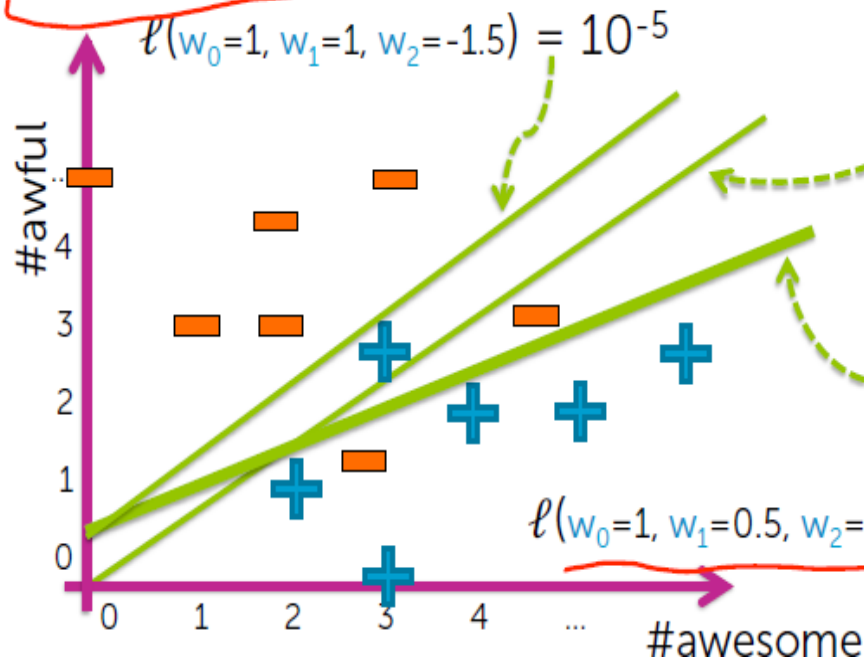
66

Maximize likelihood over all possible w_0, w_1, w_2

$$\ell(\mathbf{w}) = \prod_{i=1}^N P(y_i \mid \mathbf{x}_i, \mathbf{w})$$

$$\ell(w_0=0, w_1=1, w_2=-1.5) = 10^{-6}$$

$$\ell(w_0=1, w_1=1, w_2=-1.5) = 10^{-5}$$



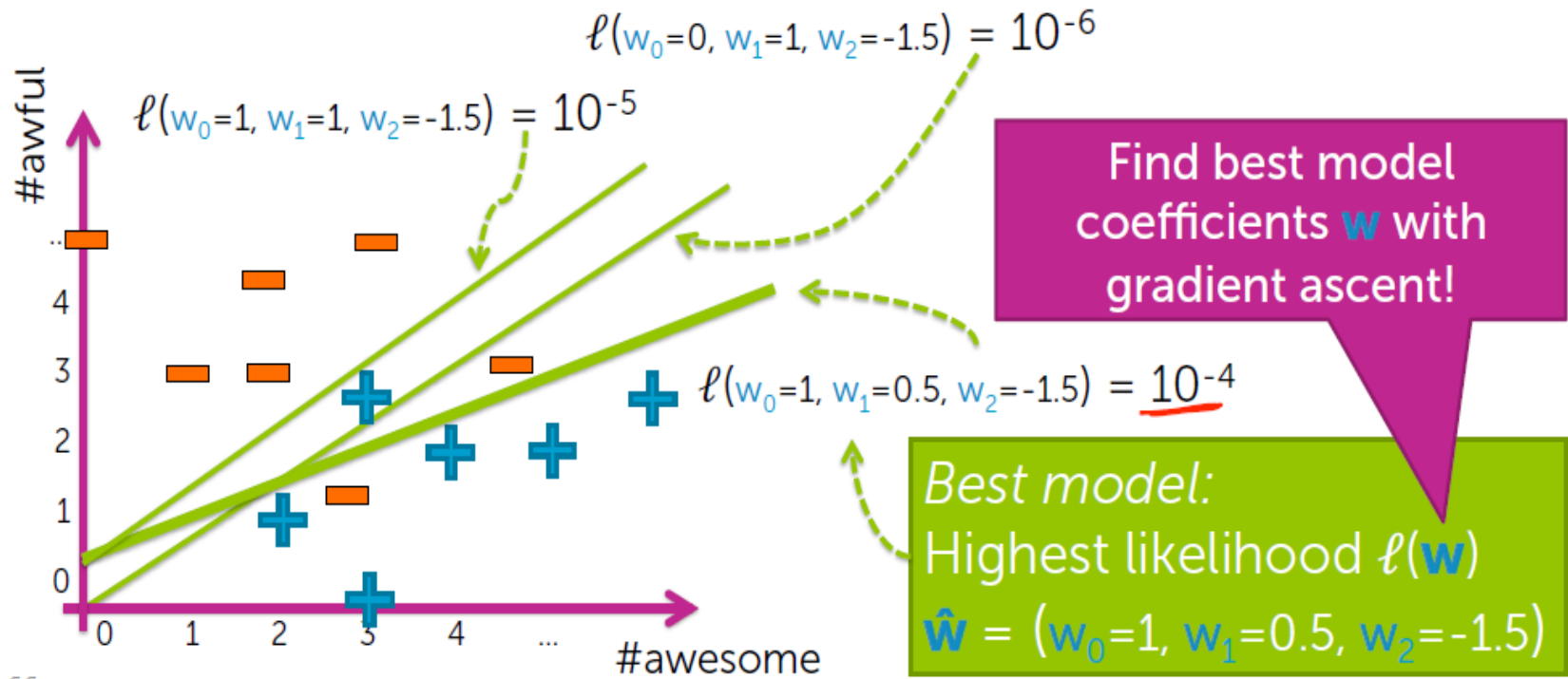
Best model:
Highest likelihood $\ell(\mathbf{w})$
 $\hat{\mathbf{w}} = (w_0=1, w_1=0.5, w_2=-1.5)$

optimize with
gradient ascent

Find best classifier

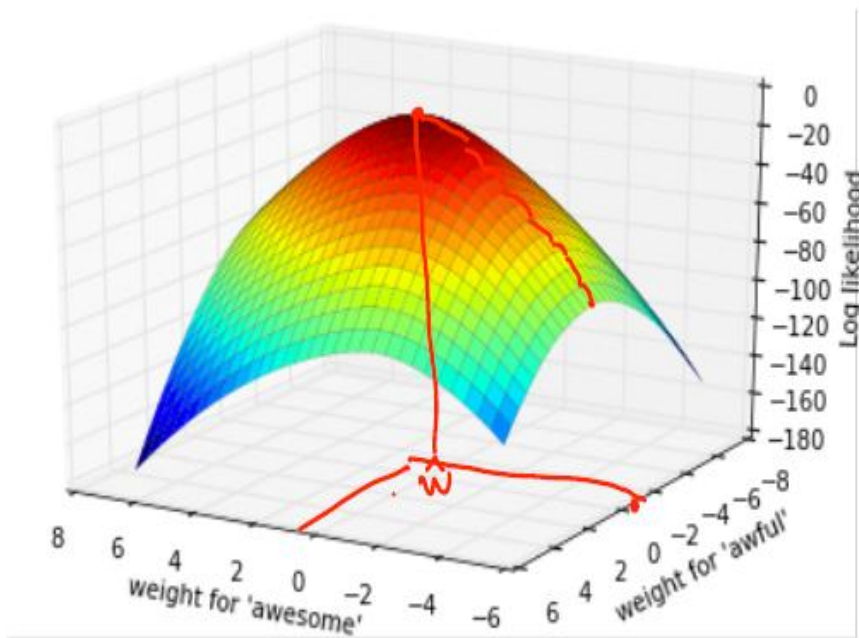
67

Maximize quality metric over all possible w_0, w_1, w_2
Likelihood $\ell(\mathbf{w})$



Maximizing likelihood

68



No closed-form solution \rightarrow use gradient ascent

Maximize function over all possible w_0, w_1, w_2

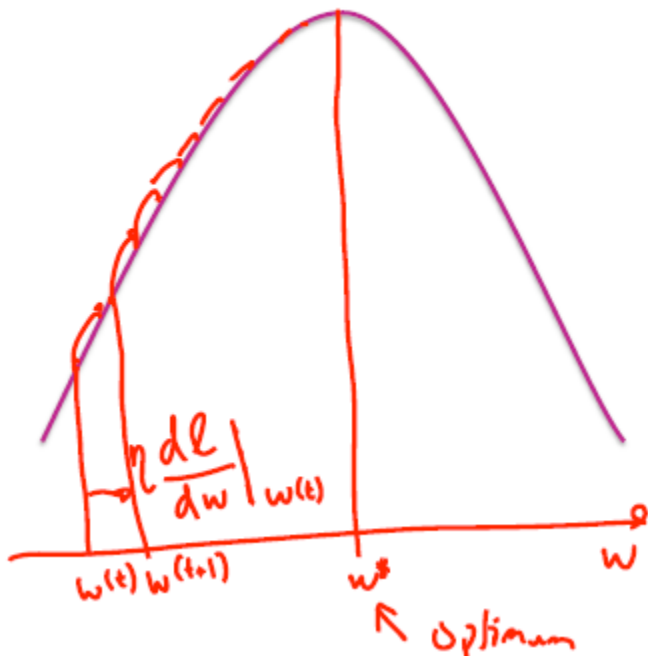
$$\max_{w_0, w_1, w_2} \prod_{i=1}^N P(y_i \mid \mathbf{x}_i, \mathbf{w})$$

$\ell(w_0, w_1, w_2)$ is a function of 3 variables

Gradient ascent

69

Finding the max via hill climbing



Algorithm:

while not converged

$$w^{(t+1)} \leftarrow w^{(t)} + \eta \frac{d\ell}{dw} \bigg|_{w^{(t)}}$$

step size

Gradient ascent

70

Convergence criteria

For convex functions,
optimum occurs when

$$\frac{d\ell}{dw} = 0$$

In practice, stop when

$$\left. \frac{d\ell}{dw} \right|_{w^{(t)}} < \epsilon$$

↑
tolerance



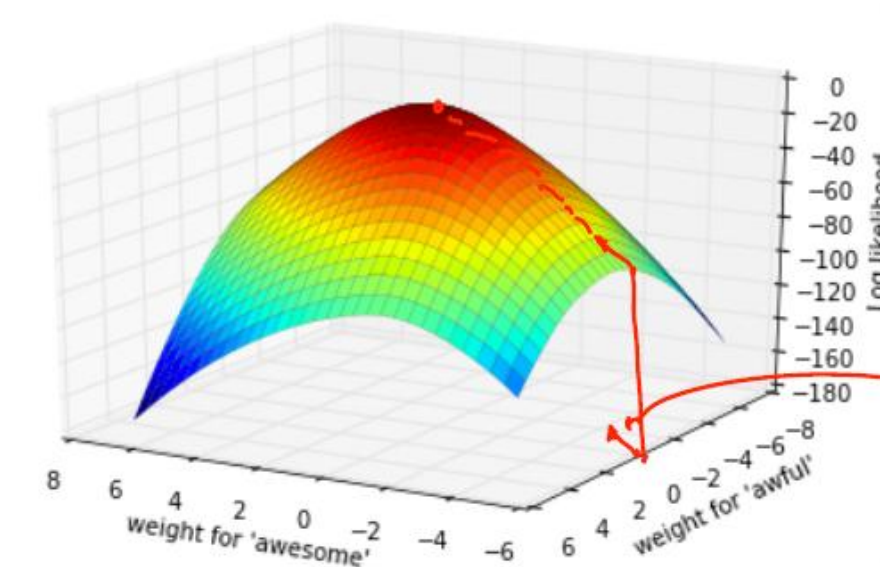
Algorithm:

while not converged
 $w^{(t+1)} \leftarrow w^{(t)} + \eta \left. \frac{d\ell}{dw} \right|_{w^{(t)}}$

Gradient ascent

71

Moving to multiple dimensions:
Gradients

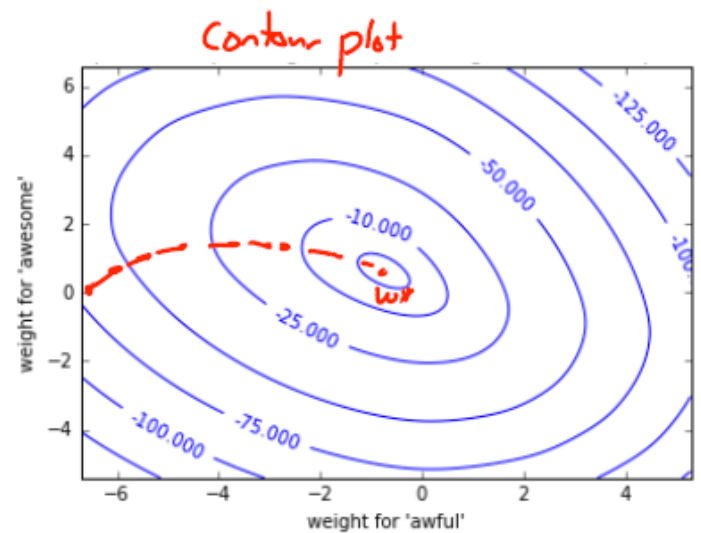
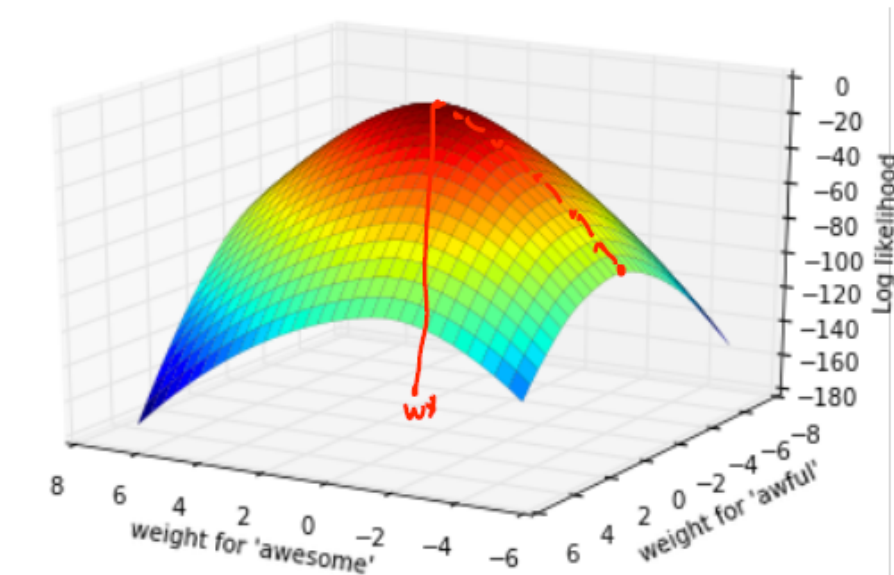


$$\nabla \ell(\mathbf{w}) = \begin{bmatrix} \frac{\partial \ell}{\partial w_0} \\ \frac{\partial \ell}{\partial w_1} \\ \vdots \\ \frac{\partial \ell}{\partial w_D} \end{bmatrix} \leftarrow D+1 \text{ dim vector}$$

Gradient ascent

72

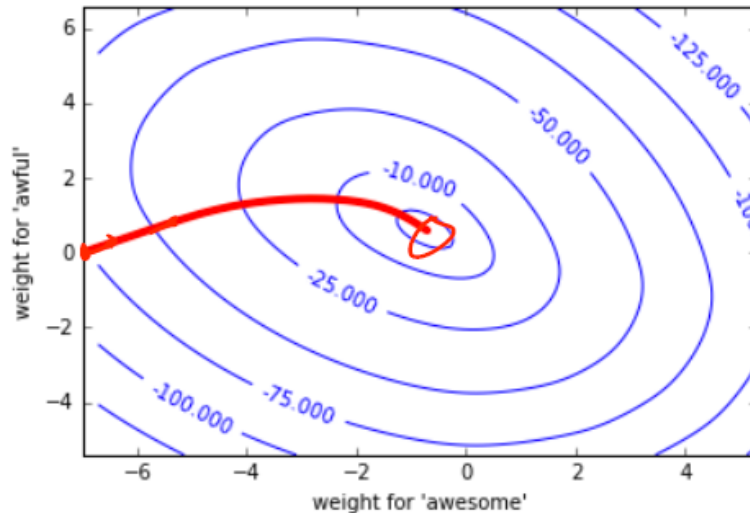
Contour plots



Gradient ascent

73

Gradient ascent



Algorithm:

$w^{(0)} = 0$, random , or something smart.

while not converged

$$w^{(t+1)} \leftarrow w^{(t)} + \eta \nabla \ell(w^{(t)})$$

step size

Details

- ▣ Derivative of likelihood for logistic regression

The log trick, often used in ML...

75

- Products become sums:
 $\ln a \cdot b = \ln a + \ln b$ | $\ln \frac{a}{b} = \ln a - \ln b$
- Doesn't change maximum!
 - If $\hat{\mathbf{w}}$ maximizes $f(\mathbf{w})$:

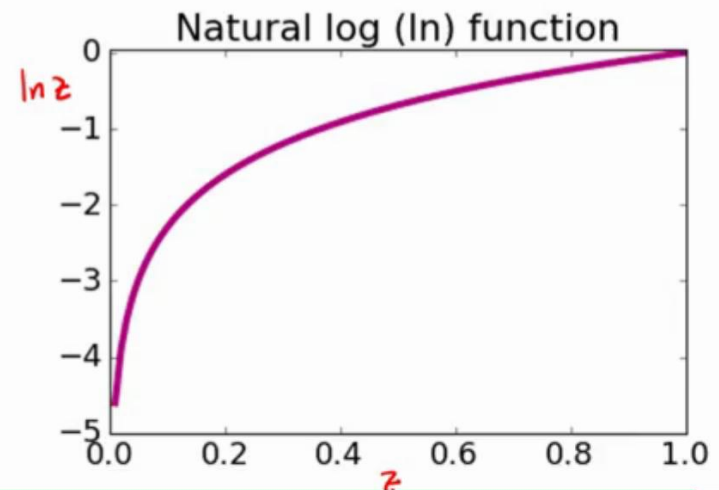
$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} f(\mathbf{w})$$

the \mathbf{w} that makes $f(\mathbf{w})$ largest

- Then $\hat{\mathbf{w}}_{\ln}$ maximizes $\ln(f(\mathbf{w}))$:

$$\hat{\mathbf{w}}_{\ln} = \arg \max_{\mathbf{w}} \ln(f(\mathbf{w}))$$

$$\hat{\mathbf{w}} = \hat{\mathbf{w}}_{\ln}$$



Log-likelihood function

76

- Goal: choose coefficients \mathbf{w} maximizing likelihood:

$$\ell(\mathbf{w}) = \prod_{i=1}^N P(y_i \mid \mathbf{x}_i, \mathbf{w})$$

- Math simplified by using log-likelihood – taking (natural) log:

$$\ell\ell(\mathbf{w}) = \ln \prod_{i=1}^N P(y_i \mid \mathbf{x}_i, \mathbf{w})$$

natural log

Log-likelihood function

77

Using log to turn products into sums

$$\ln \prod_{i=1}^N f_i = \sum_{i=1}^N \ln f_i$$

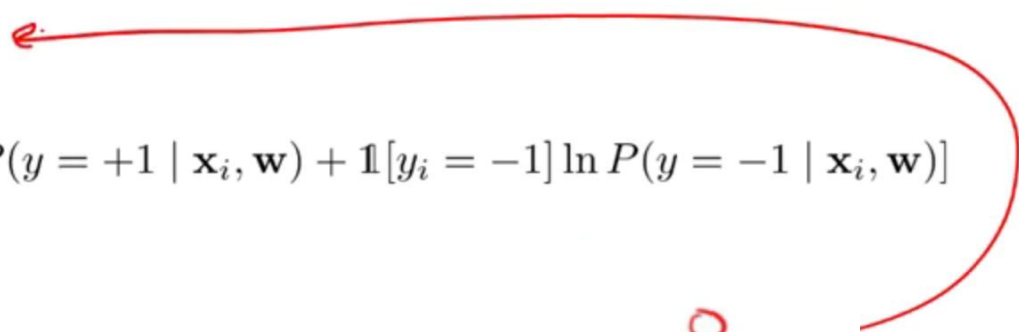
- The log of the product of likelihoods becomes the sum of the logs:

$$\begin{aligned} \ell\ell(\mathbf{w}) &= \ln \prod_{i=1}^N P(y_i \mid \mathbf{x}_i, \mathbf{w}) \\ &= \sum_{i=1}^N \ln P(y_i \mid \mathbf{x}_i, \mathbf{w}) \end{aligned}$$

Rewriting log-likelihood

78

- For simpler math, we'll rewrite likelihood with indicators:

$$\begin{aligned}\ell\ell(\mathbf{w}) &= \sum_{i=1}^N \ln P(y_i | \mathbf{x}_i, \mathbf{w}) \\ &= \sum_{i=1}^N [\mathbb{1}[y_i = +1] \ln P(y = +1 | \mathbf{x}_i, \mathbf{w}) + \mathbb{1}[y_i = -1] \ln P(y = -1 | \mathbf{x}_i, \mathbf{w})]\end{aligned}$$


Indicator function

if $y_i = +1$

if $y_i = -1$

✓

0

0

✓

Logistic regression

79

Logistic regression model: $P(y=-1|\mathbf{x}, \mathbf{w})$

- Probability model predicts $y=+1$:

$$P(y=+1|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{h}(\mathbf{x})}}$$

- Probability model predicts $y=-1$:

$$\begin{aligned} P(y=-1|\mathbf{x}, \mathbf{w}) &= 1 - P(y=+1|\mathbf{x}, \mathbf{w}) = 1 - \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{h}(\mathbf{x})}} \\ &= \frac{1 + e^{-\mathbf{w}^T \mathbf{h}(\mathbf{x})} - 1}{1 + e^{-\mathbf{w}^T \mathbf{h}(\mathbf{x})}} = \frac{e^{-\mathbf{w}^T \mathbf{h}(\mathbf{x})}}{1 + e^{-\mathbf{w}^T \mathbf{h}(\mathbf{x})}} \end{aligned}$$

Logistic regression

80

Plugging in logistic function for 1 data point

$$P(y = +1 \mid \mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^\top h(\mathbf{x})}} \quad P(y = -1 \mid \mathbf{x}, \mathbf{w}) = \frac{e^{-\mathbf{w}^\top h(\mathbf{x})}}{1 + e^{-\mathbf{w}^\top h(\mathbf{x})}}$$

$$\ell\ell(\mathbf{w}) = \mathbb{1}[y_i = +1] \ln P(y = +1 \mid \mathbf{x}_i, \mathbf{w}) + \mathbb{1}[y_i = -1] \ln P(y = -1 \mid \mathbf{x}_i, \mathbf{w})$$

$$= \mathbb{1}[y_i = +1] \ln \frac{1}{1 + e^{-\mathbf{w}^\top h(\mathbf{x}_i)}} + (1 - \mathbb{1}[y_i = +1]) \ln \frac{e^{-\mathbf{w}^\top h(\mathbf{x}_i)}}{1 + e^{-\mathbf{w}^\top h(\mathbf{x}_i)}}$$

$$= -\mathbb{1}[y_i = +1] \ln(1 + e^{-\mathbf{w}^\top h(\mathbf{x}_i)}) + (1 - \mathbb{1}[y_i = +1]) [-\mathbf{w}^\top h(\mathbf{x}_i) - \ln(1 + e^{-\mathbf{w}^\top h(\mathbf{x}_i)})]$$

$$= - (1 - \mathbb{1}[y_i = +1]) \mathbf{w}^\top h(\mathbf{x}_i) - \ln(1 + e^{-\mathbf{w}^\top h(\mathbf{x}_i)})$$

Simpler form

$$\ln e^a = a$$

$$\mathbb{1}[y_i = -1] = 1 - \mathbb{1}[y_i = +1]$$

$$\ln \frac{1}{1 + e^{-\mathbf{w}^\top h(\mathbf{x}_i)}} = -\ln(1 + e^{-\mathbf{w}^\top h(\mathbf{x}_i)})$$

$$\ln \frac{e^{-\mathbf{w}^\top h(\mathbf{x}_i)}}{1 + e^{-\mathbf{w}^\top h(\mathbf{x}_i)}} = \frac{\ln e^{-\mathbf{w}^\top h(\mathbf{x}_i)}}{1 + e^{-\mathbf{w}^\top h(\mathbf{x}_i)}} = \frac{-\mathbf{w}^\top h(\mathbf{x}_i)}{1 + e^{-\mathbf{w}^\top h(\mathbf{x}_i)}}$$

Logistic regression

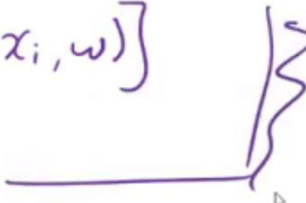
81

Gradient for 1 data point

$$\ell\ell(\mathbf{w}) = -(1 - \mathbb{1}[y_i = +1])\mathbf{w}^\top h(\mathbf{x}_i) - \ln(1 + e^{-\mathbf{w}^\top h(\mathbf{x}_i)})$$

$$\frac{\partial \ell\ell}{\partial w_j} = -(1 - \mathbb{1}[y_i = +1]) \frac{\partial w^\top h(\mathbf{x}_i)}{\partial w_j} - \frac{\partial \ln(1 + e^{-w^\top h(\mathbf{x}_i)})}{\partial w_j}$$

$$= -(1 - \mathbb{1}[y_i = +1]) h_j(\mathbf{x}_i) + h_j(\mathbf{x}_i) P(y = -1 | \mathbf{x}_i, \mathbf{w})$$

$$= h_j(\mathbf{x}_i) [\mathbb{1}[y_i = +1] - P(y = +1 | \mathbf{x}_i, \mathbf{w})]$$


$$\begin{aligned} \frac{\partial w^\top h(\mathbf{x}_i)}{\partial w_j} &= h_j(\mathbf{x}_i) \\ \hline \frac{\partial \ln(1 + e^{-w^\top h(\mathbf{x}_i)})}{\partial w_j} &= -h_j(\mathbf{x}_i) \underbrace{\frac{e^{-w^\top h(\mathbf{x}_i)}}{1 + e^{-w^\top h(\mathbf{x}_i)}}}_{P(y = -1 | \mathbf{x}_i, \mathbf{w})} \end{aligned}$$

Logistic regression

82

Finally, gradient for all data points

- Gradient for one data point:

$$h_j(\mathbf{x}_i) \left(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w}) \right)$$

- Adding over data points:

$$\frac{\partial \ell}{\partial w_j} = \sum_{i=1}^N h_j(\mathbf{x}_i) \left(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w}) \right)$$

Linear classifier (continue)

▣ Parameters learning

Derivative for logistic regression

84

Derivative of (log-)likelihood

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^N h_j(\mathbf{x}_i) \left(\mathbb{1}[y_i = +1] - \underbrace{P(y = +1 \mid \mathbf{x}_i, \mathbf{w})}_{\text{predict } \mathbf{x}_i \text{ is positive}} \right)$$

Indicator function:

$$\mathbb{1}[y_i = +1] = \begin{cases} 1 & \text{if } y_i = +1 \\ 0 & \text{if } y_i = -1 \end{cases}$$

Derivative for logistic regression

85

Computing derivative

$$\frac{\partial \ell(\mathbf{w}^{(t)})}{\partial \mathbf{w}_j} = \sum_{i=1}^N h_j(\mathbf{x}_i) \left(\mathbb{1}[y_i = +1] - P(y = +1 | \mathbf{x}_i, \mathbf{w}^{(t)}) \right)$$

$\mathbf{w}^{(t)}$:

$w_0^{(t)}$	0
$w_1^{(t)}$	1
$w_2^{(t)}$	-2

$$\frac{\partial \ell}{\partial w_1}$$

$h_i(\mathbf{x}) = \text{awesome}$

x[1]	x[2]	y	P(y=+1 x,w)	Contribution to derivative for w_1
2	1	+1	0.5	$2(1 - 0.5) = 1$
0	2	-1	0.02	$0(0 - 0.02) = 0$
3	3	-1	0.05	$3(0 - 0.05) = -0.15$
4	1	+1	0.88	$4(1 - 0.88) = 0.48$

Total derivative:

$$\frac{\partial \ell(\mathbf{w}^{(t)})}{\partial w_1} = 1 + 0 - 0.15 + 0.48 = 1.33$$

$$w_1^{(t+1)} = w_1^{(t)} + \eta \frac{\partial \ell(\mathbf{w}^{(t)})}{\partial w_1} \quad | \quad \eta = 0.1$$

$$= 1 + 0.1 \cdot 1.33 = 1.133$$

Derivative for logistic regression

86

Derivative of (log-)likelihood: Interpretation

Sum over data points Feature value Difference between truth and prediction

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^N h_j(\mathbf{x}_i) \left(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w}) \right)$$

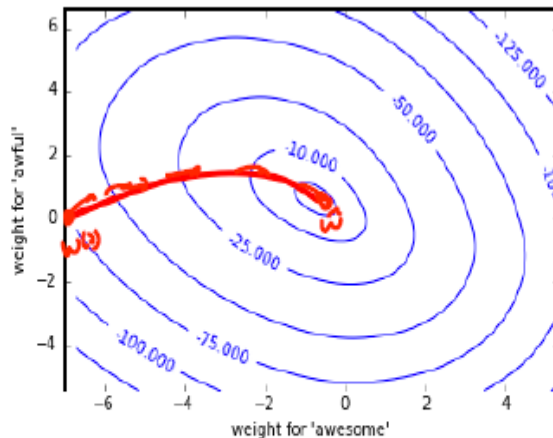
Δ_i

If $h_j(\mathbf{x}_i)=1$:

	<u>$P(y=+1 \mathbf{x}_i, \mathbf{w}) \approx 1$</u>	<u>$P(y=+1 \mathbf{x}_i, \mathbf{w}) \approx 0$</u>
<u>$y_i=+1$</u>	$\Delta_i = (1 - 1) \approx 0$ \hookrightarrow don't change anything!	$\Delta_i \approx 1 \Rightarrow$ increase w_j \Rightarrow Score(\mathbf{x}_i) becomes larger $\Rightarrow P(y=+1 \mathbf{x}_i, \mathbf{w})$ increases
<u>$y_i=-1$</u>	$\Delta_i = -1 \Rightarrow w_j$ to decrease \Rightarrow Score(\mathbf{x}_i) decreases $\Rightarrow P(y=+1 \mathbf{x}_i, \mathbf{w})$ decrease	$\Delta_i \approx 0$ \Rightarrow don't change anything

Gradient ascent for logistic regression

87



init $\mathbf{w}^{(1)} = 0$ (or randomly, or smartly), $t = 1$

while $\|\nabla \ell(\mathbf{w}^{(t)})\| > \epsilon$

for $j = 0, \dots, D$

$$\text{partial}[j] = \sum_{i=1}^N h_j(\mathbf{x}_i) \left(\mathbb{1}[y_i = +1] - \underbrace{P(y = +1 \mid \mathbf{x}_i, \mathbf{w}^{(t)})}_{\frac{1}{1 + e^{-\mathbf{w}^{(t)T} \mathbf{h}(\mathbf{x}_i)}}} \right)$$

$$\mathbf{w}_j^{(t+1)} \leftarrow \mathbf{w}_j^{(t)} + \eta \text{partial}[j]$$

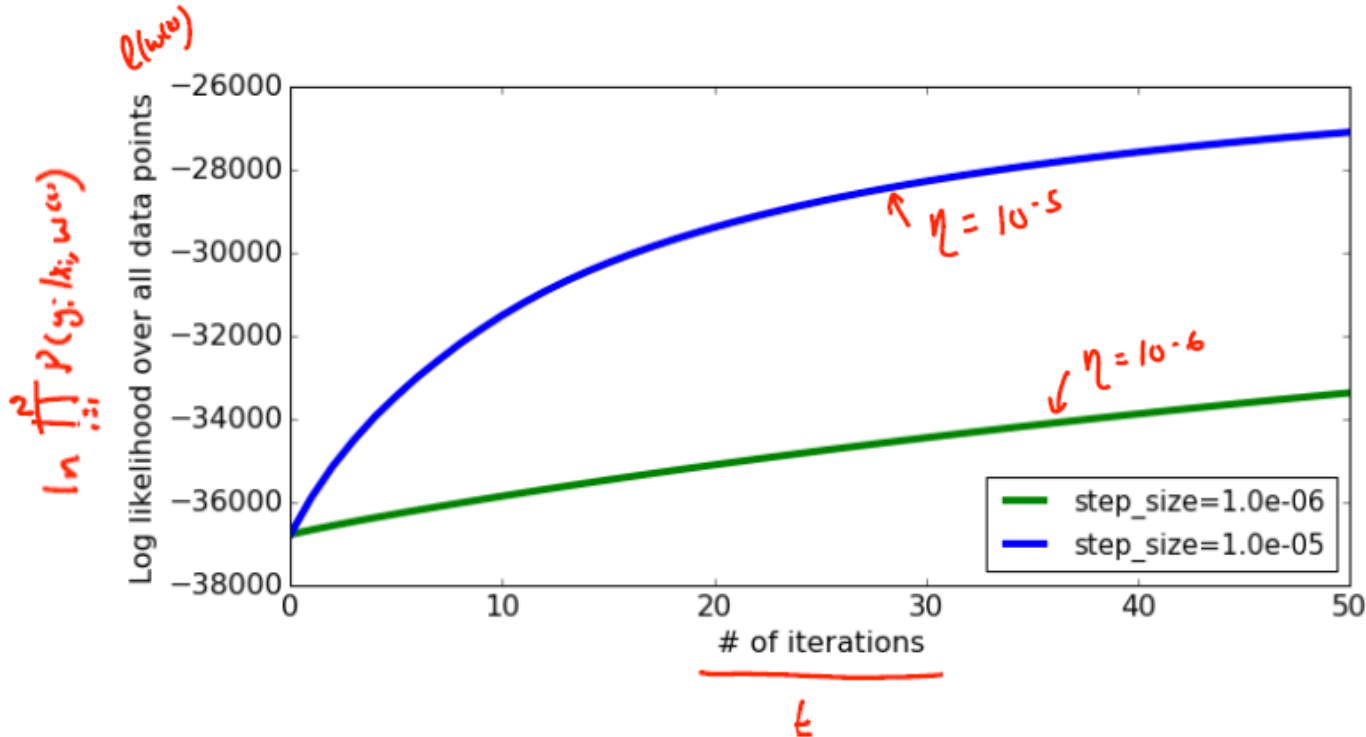
$t \leftarrow t + 1$

step size $\frac{\partial \ell(\mathbf{w}^{(t)})}{\partial \mathbf{w}_j}$

Choosing the step size

88

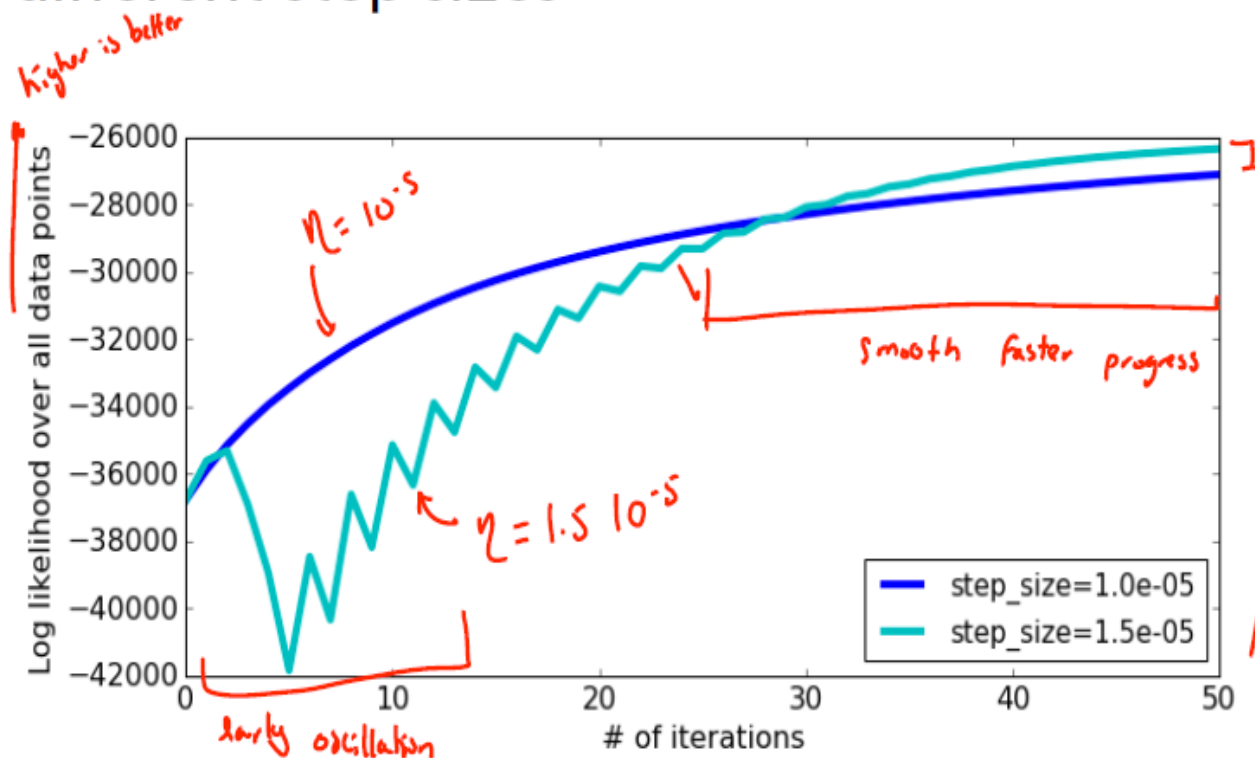
If step size is too small, can take a long time to converge



Choosing the step size

89

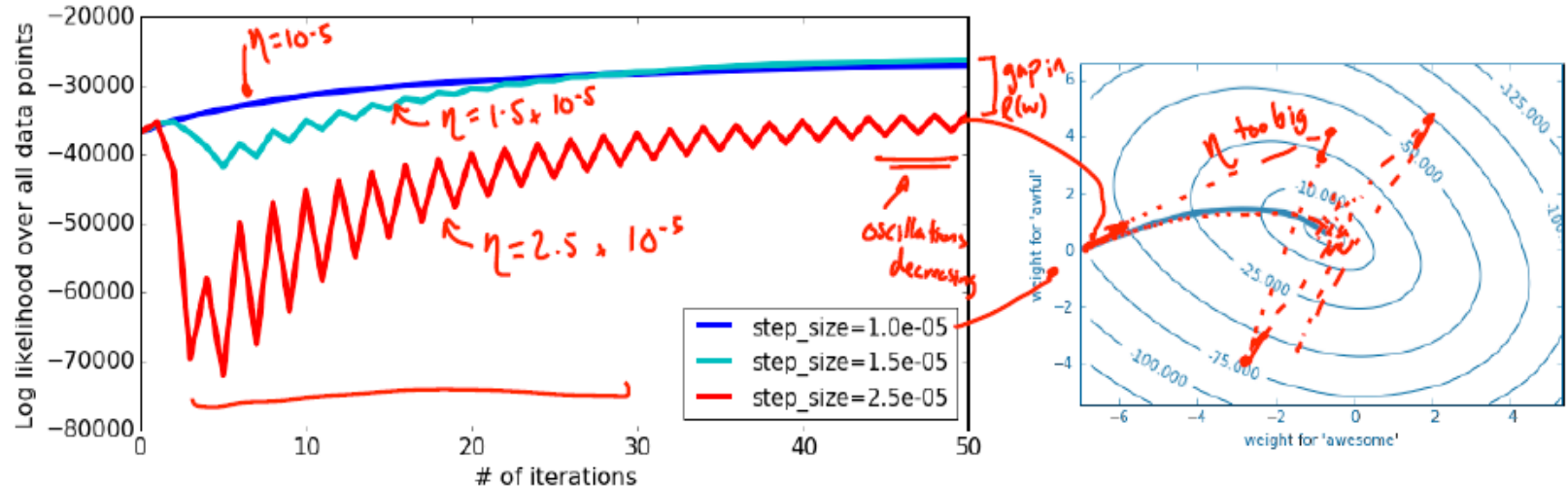
Compare converge with different step sizes



Choosing the step size

90

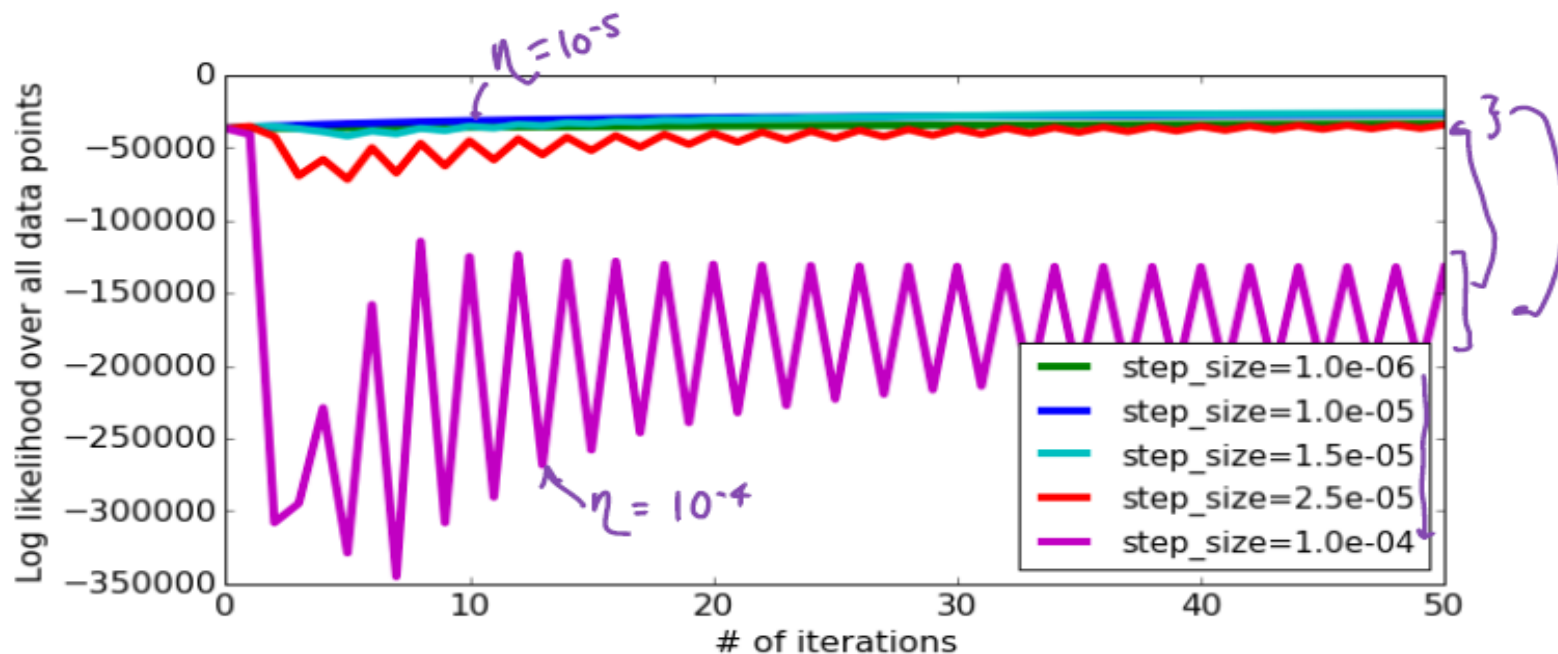
Careful with step sizes that are too large



Choosing the step size

91

Very large step sizes can even cause divergence or wild oscillations



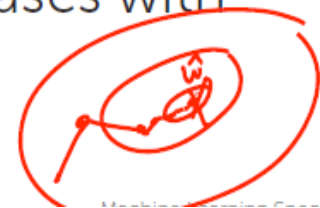
Choosing the step size

92

Simple rule of thumb for picking step size η

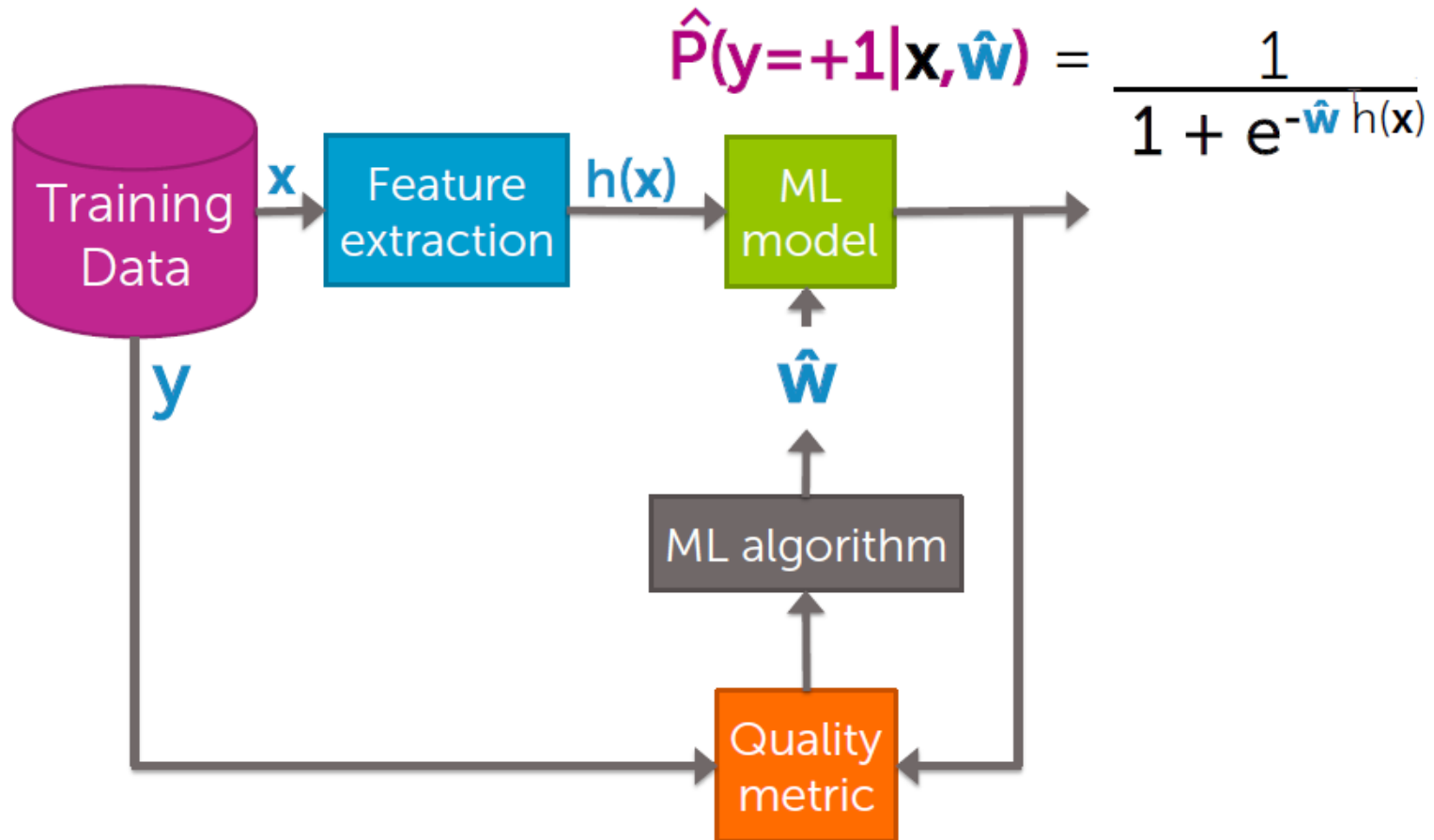
- Unfortunately, picking step size requires a lot of trial and error ☹
- Try a several values, exponentially spaced
 - Goal: plot learning curves to
 - find one η that is too small (smooth but moving too slowly)
 - find one η that is too large (oscillation or divergence)
- Try values in between to find “best” η
 - ↳ exponentially space, pick one that leads best training data likelihood
- Advanced tip: can also try step size that decreases with iterations, e.g.,

$$\eta_t = \frac{\eta_0}{t}$$



Flow chart

93



What you can do now

94

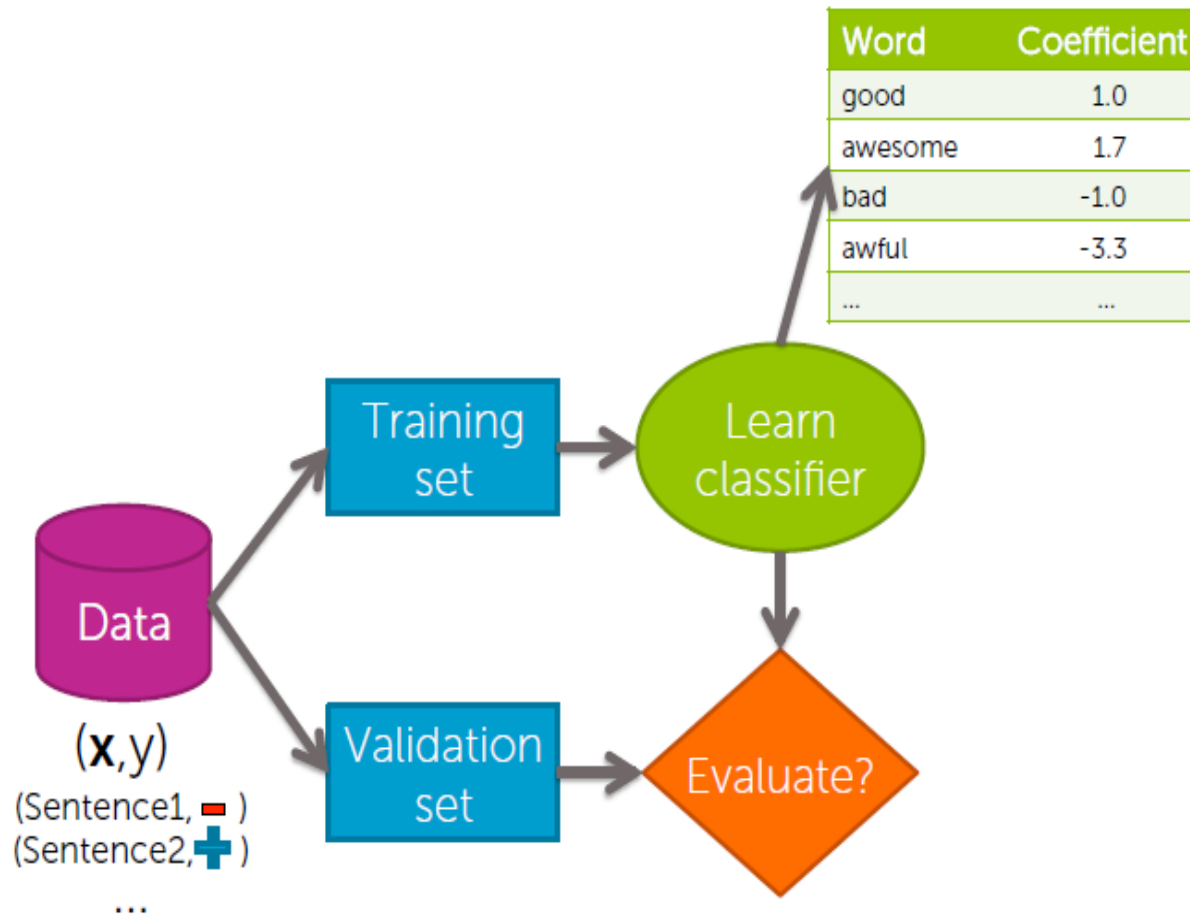
- Measure quality of a classifier using the likelihood function
- Interpret the likelihood function as the probability of the observed data
- Learn a logistic regression model with gradient descent
- (Optional) Derive the gradient descent update rule for logistic regression

Linear classifier

▣ Overfitting & regularization

Training a classifier = Learning the coefficients

96



Classification error & accuracy

97

- Error measures fraction of mistakes

$$\text{error} = \frac{\# \text{ Mistakes}}{\text{Total number of data points}}$$

- Best possible value is 0.0

- Often, measure **accuracy**

- Fraction of correct predictions

$$\text{accuracy} = \frac{\# \text{ Correct}}{\text{Total number of data points}}$$

- Best possible value is 1.0

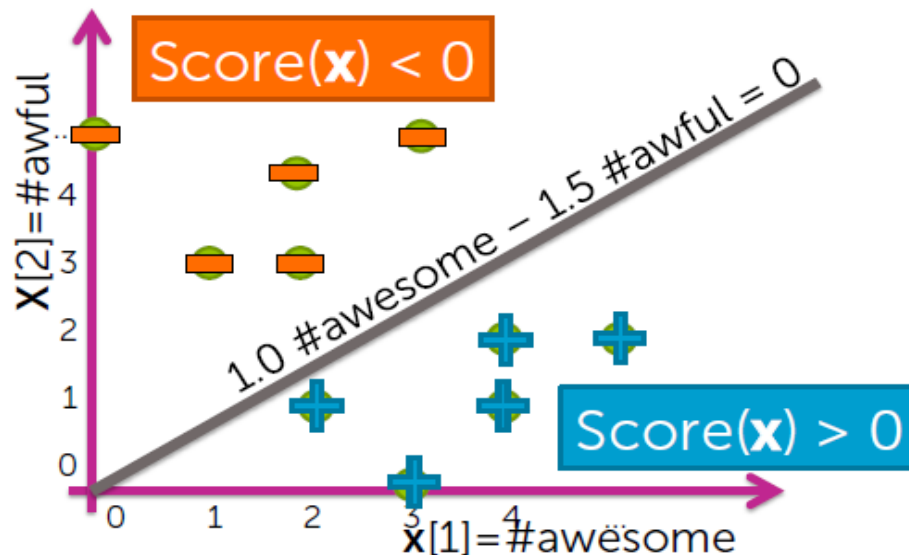
Overfitting in classification

98

Decision boundary example

Word	Coefficient
#awesome	1.0
#awful	-1.5

→ $\text{Score}(\mathbf{x}) = 1.0 \text{ \#awesome} - 1.5 \text{ \#awful}$

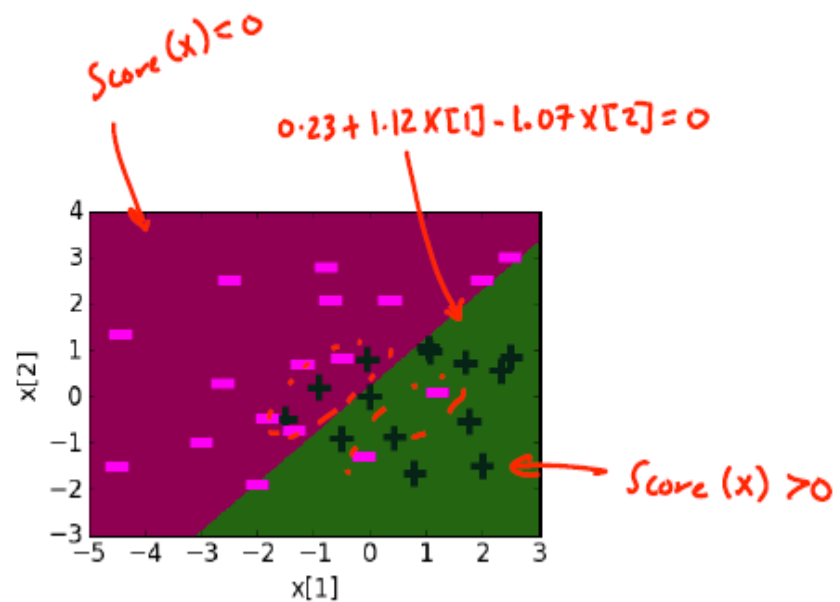
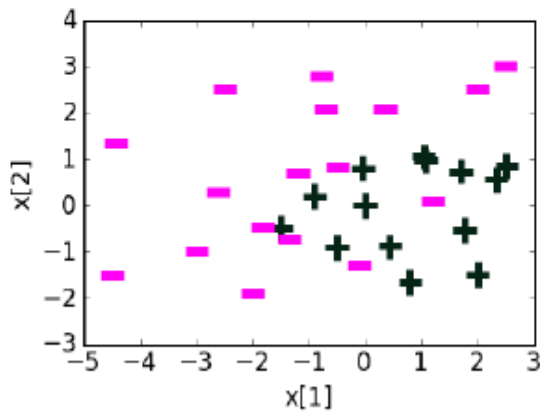


Overfitting in classification

99

Learned decision boundary

Feature	Value	Coefficient learned
$h_0(x)$	$w_0 \cdot 1$	0.23
$h_1(x)$	$w_1 \cdot x[1]$	1.12
$h_2(x)$	$w_2 \cdot x[2]$	-1.07

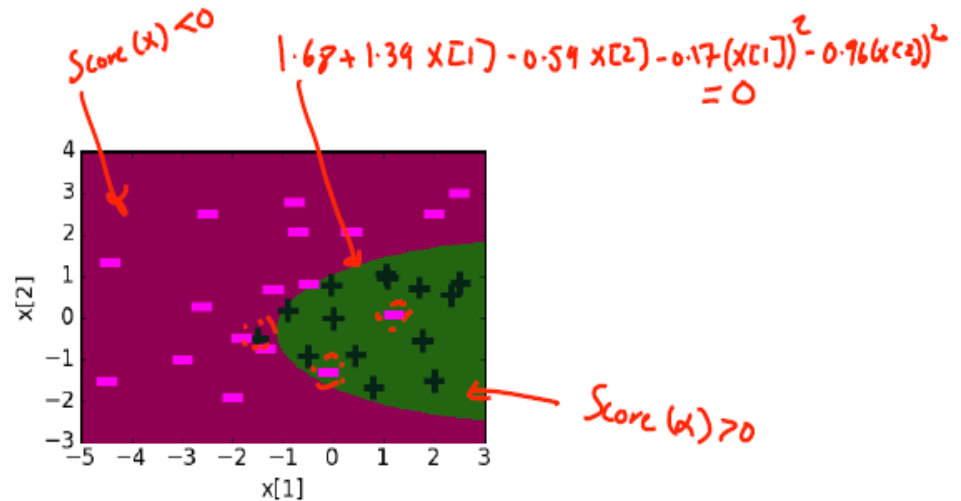
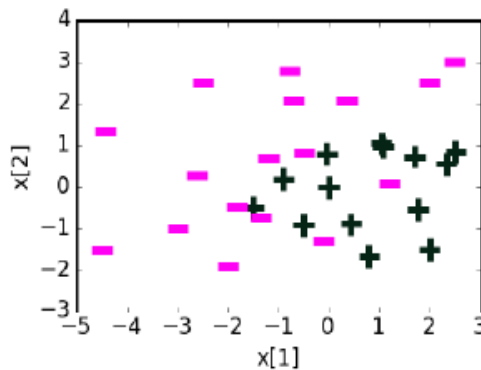


Overfitting in classification

100

Quadratic features (in 2d)

Feature	Value	Coefficient learned
$h_0(x)$	1	1.68
$h_1(x)$	$x[1]$	1.39
$h_2(x)$	$x[2]$	-0.59
$h_3(x)$	$(x[1])^2$	-0.17
$h_4(x)$	$(x[2])^2$	-0.96



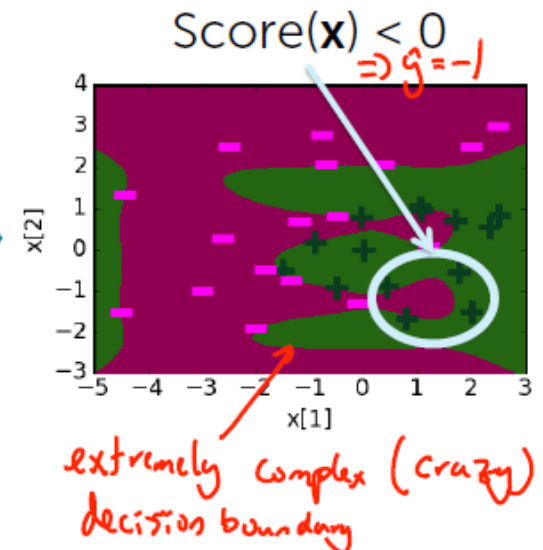
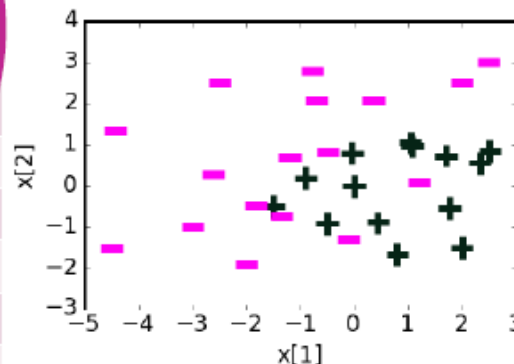
Overfitting in classification

101

Degree 6 features (in 2d)

Feature	Value	Coefficient learned
$h_0(\mathbf{x})$	1	21.6
$h_1(\mathbf{x})$	$x[1]$	5.3
$h_2(\mathbf{x})$	$x[2]$	-42.7
$h_3(\mathbf{x})$	$(x[1])^2$	-15.9
$h_4(\mathbf{x})$	$(x[2])^2$	-48.6
$h_5(\mathbf{x})$	$(x[1])^3$	-11.0
$h_6(\mathbf{x})$	$(x[2])^3$	67.0
$h_7(\mathbf{x})$	$(x[1])^4$	1.5
$h_8(\mathbf{x})$	$(x[2])^4$	48.0
$h_9(\mathbf{x})$	$(x[1])^5$	4.4
$h_{10}(\mathbf{x})$	$(x[2])^5$	-14.2
$h_{11}(\mathbf{x})$	$(x[1])^6$	0.8
$h_{12}(\mathbf{x})$	$(x[2])^6$	-8.6

Coefficient values getting large



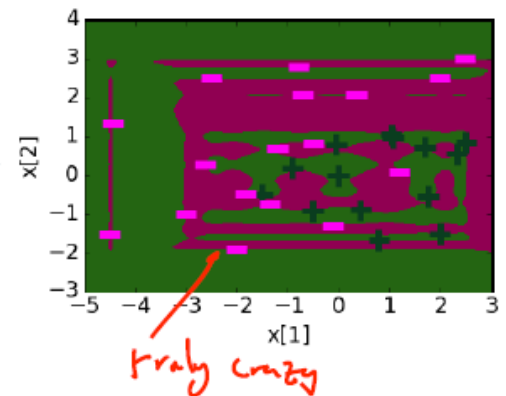
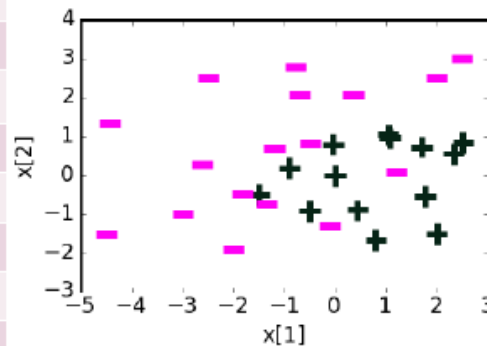
Overfitting in classification

102

Degree 20 features (in 2d)

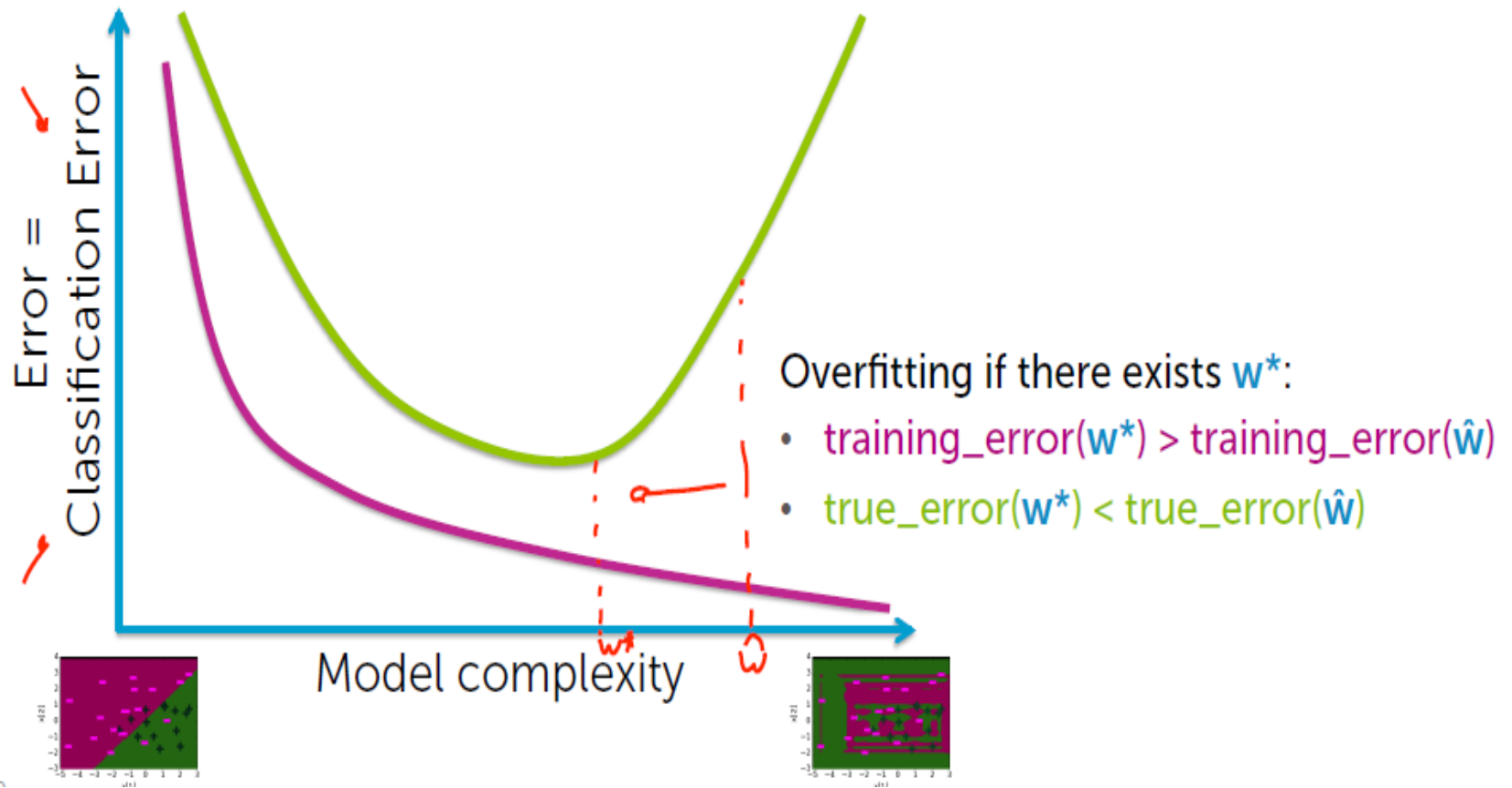
Feature	Value	Coefficient learned
$h_0(x)$	1	8.7
$h_1(x)$	$x[1]$	5.1
$h_2(x)$	$x[2]$	78.7
...
$h_{11}(x)$	$(x[1])^6$	-7.5
$h_{12}(x)$	$(x[2])^6$	3803
$h_{13}(x)$	$(x[1])^7$	21.1
$h_{14}(x)$	$(x[2])^7$	-2406
...
$h_{37}(x)$	$(x[1])^{19}$	$-2 \cdot 10^{-6}$
$h_{38}(x)$	$(x[2])^{19}$	-0.15
$h_{39}(x)$	$(x[1])^{20}$	$-2 \cdot 10^{-8}$
$h_{40}(x)$	$(x[2])^{20}$	0.03

Often, overfitting associated with very large estimated coefficients \hat{w}



Overfitting in classification

103



Overfitting in logistic regression

104

The subtle (negative) consequence of overfitting in logistic regression

Overfitting \rightarrow Large coefficient values

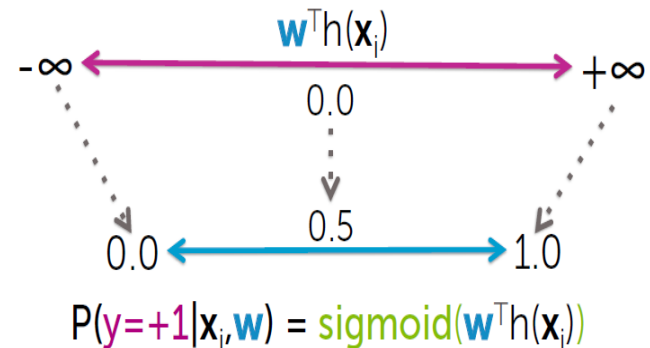


$\hat{\mathbf{w}}^T \mathbf{x}_i$ is very positive (or very negative)
 $\rightarrow \text{sigmoid}(\hat{\mathbf{w}}^T \mathbf{x}_i)$ goes to 1 (or to 0)



Model becomes extremely overconfident of predictions

Logistic regression model



Remember about this probability interpretation

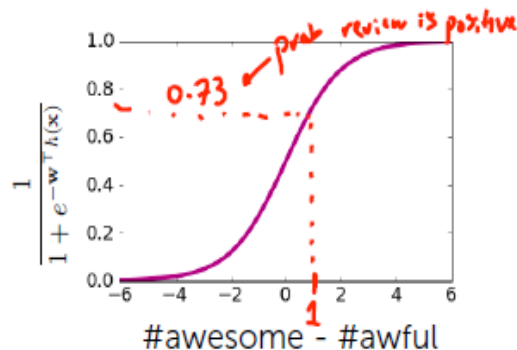
Effect of coefficients on logistic regression model

105

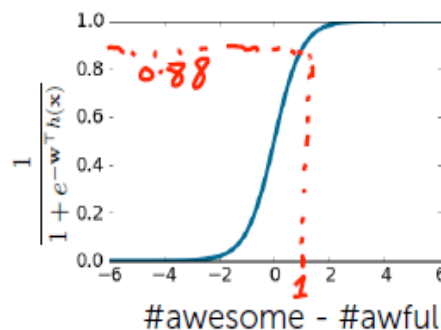
With increasing coefficients model becomes overconfident on predictions

Input x : #awesome=2, #awful=1

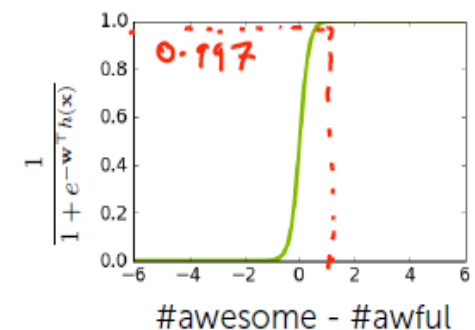
w_0	0
$w_{\text{\#awesome}}$	+1
$w_{\text{\#awful}}$	-1



w_0	0
$w_{\text{\#awesome}}$	+2
$w_{\text{\#awful}}$	-2



w_0	0
$w_{\text{\#awesome}}$	+6
$w_{\text{\#awful}}$	-6



Learned probabilities

106

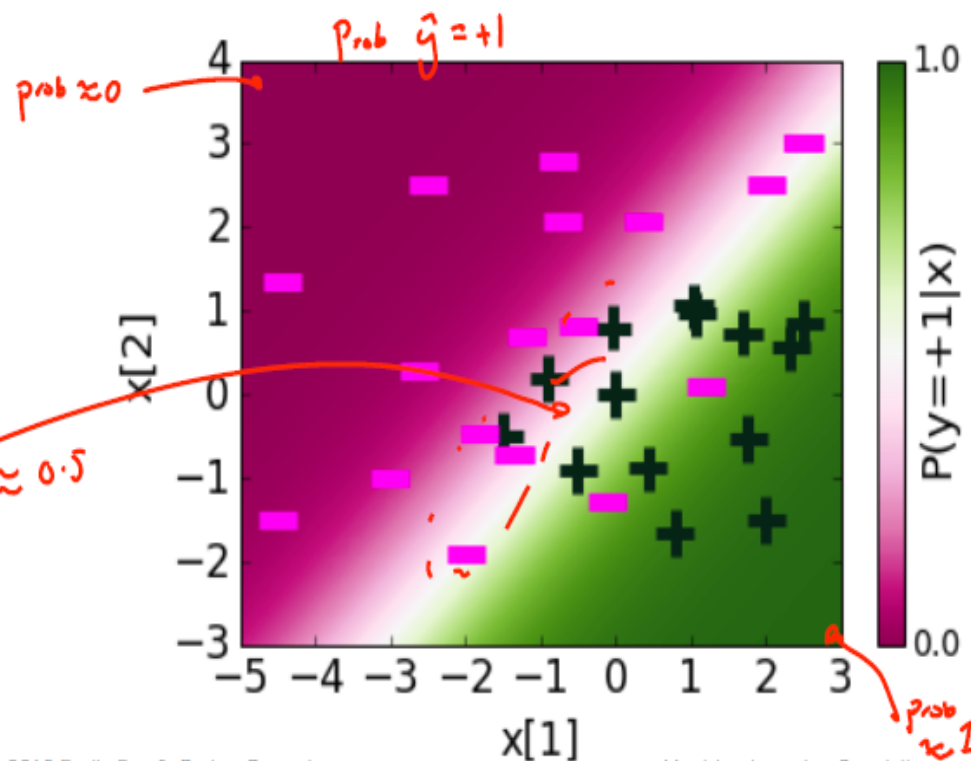
Feature	Value	Coefficient learned
$h_0(\mathbf{x})$	1	0.23
$h_1(\mathbf{x})$	$x[1]$	1.12
$h_2(\mathbf{x})$	$x[2]$	-1.07

$$P(y = +1 \mid \mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{h}(\mathbf{x})}}$$

Make sense

wide region
of uncertainty

prob ≈ 0.5



27

©2015-2016 Emily Fox & Carlos Guestrin

Machine Learning Specialization

29/10, 5/11 2019

Quadratic features: learned probabilities

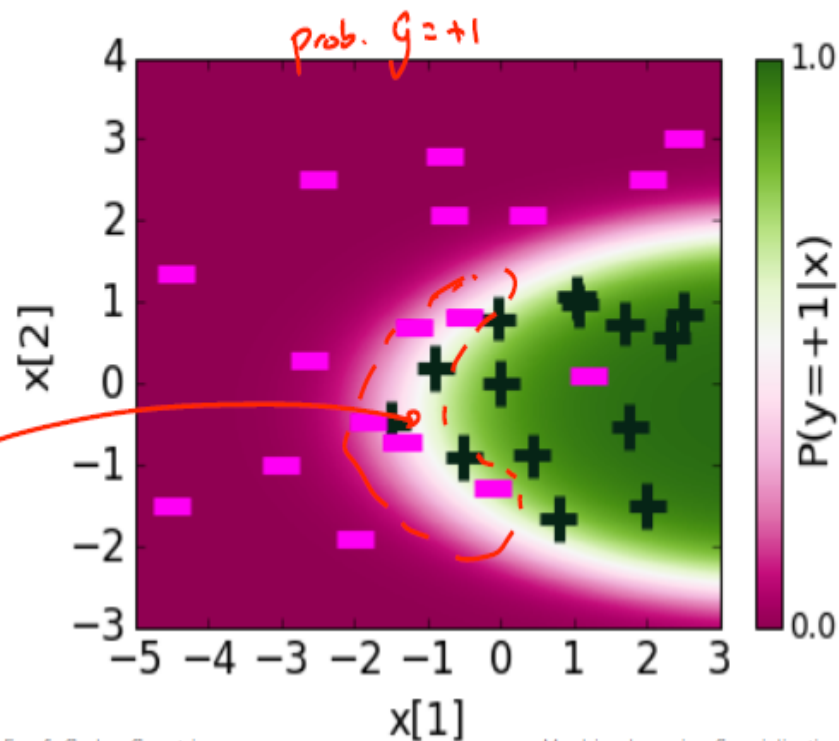
107

Feature	Value	Coefficient learned
$h_0(\mathbf{x})$	1	1.68
$h_1(\mathbf{x})$	$x[1]$	1.39
$h_2(\mathbf{x})$	$x[2]$	-0.58
$h_3(\mathbf{x})$	$(x[1])^2$	-0.17
$h_4(\mathbf{x})$	$(x[2])^2$	-0.96

$$P(y = +1 \mid \mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{h}(\mathbf{x})}}$$

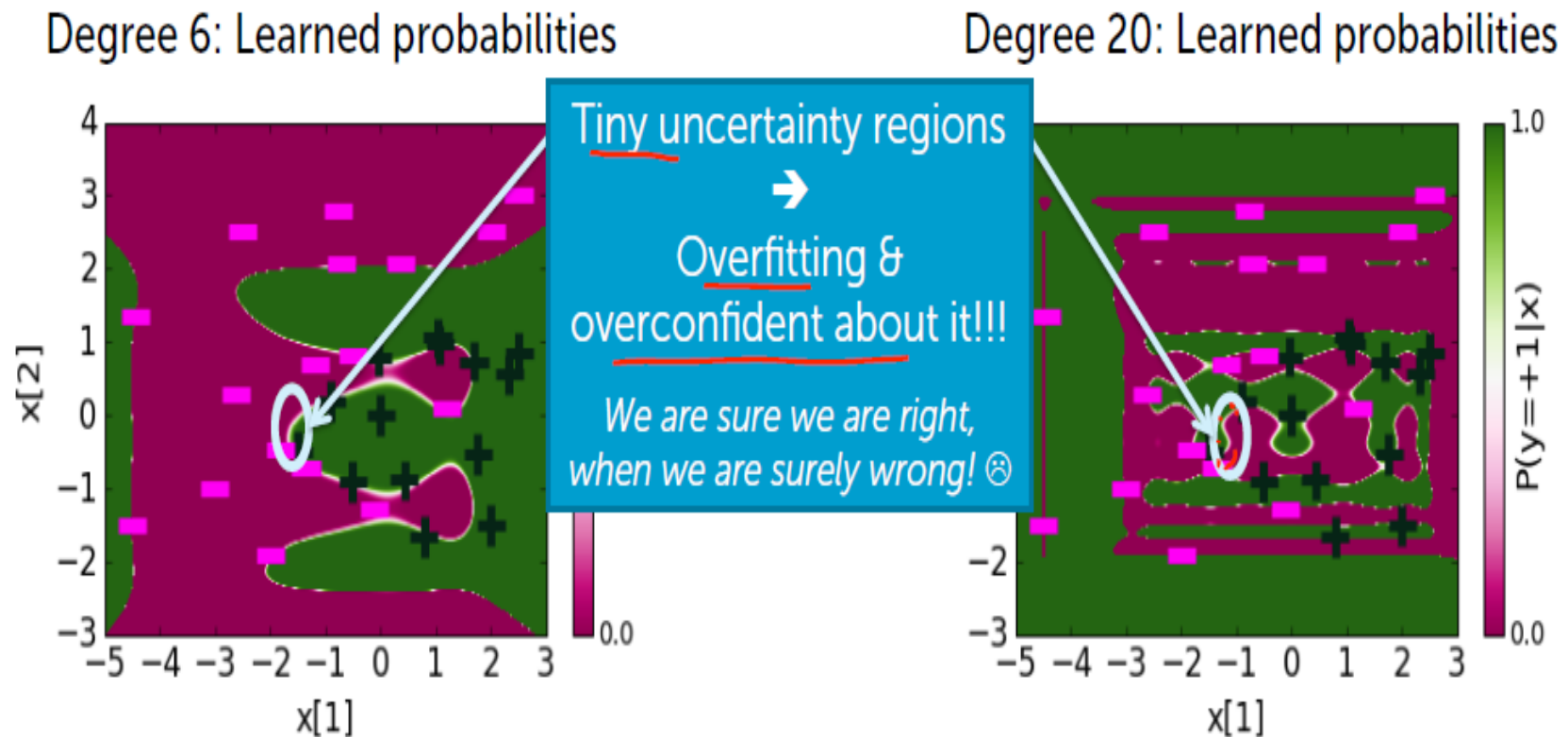
better fit to data

uncertainty region narrower



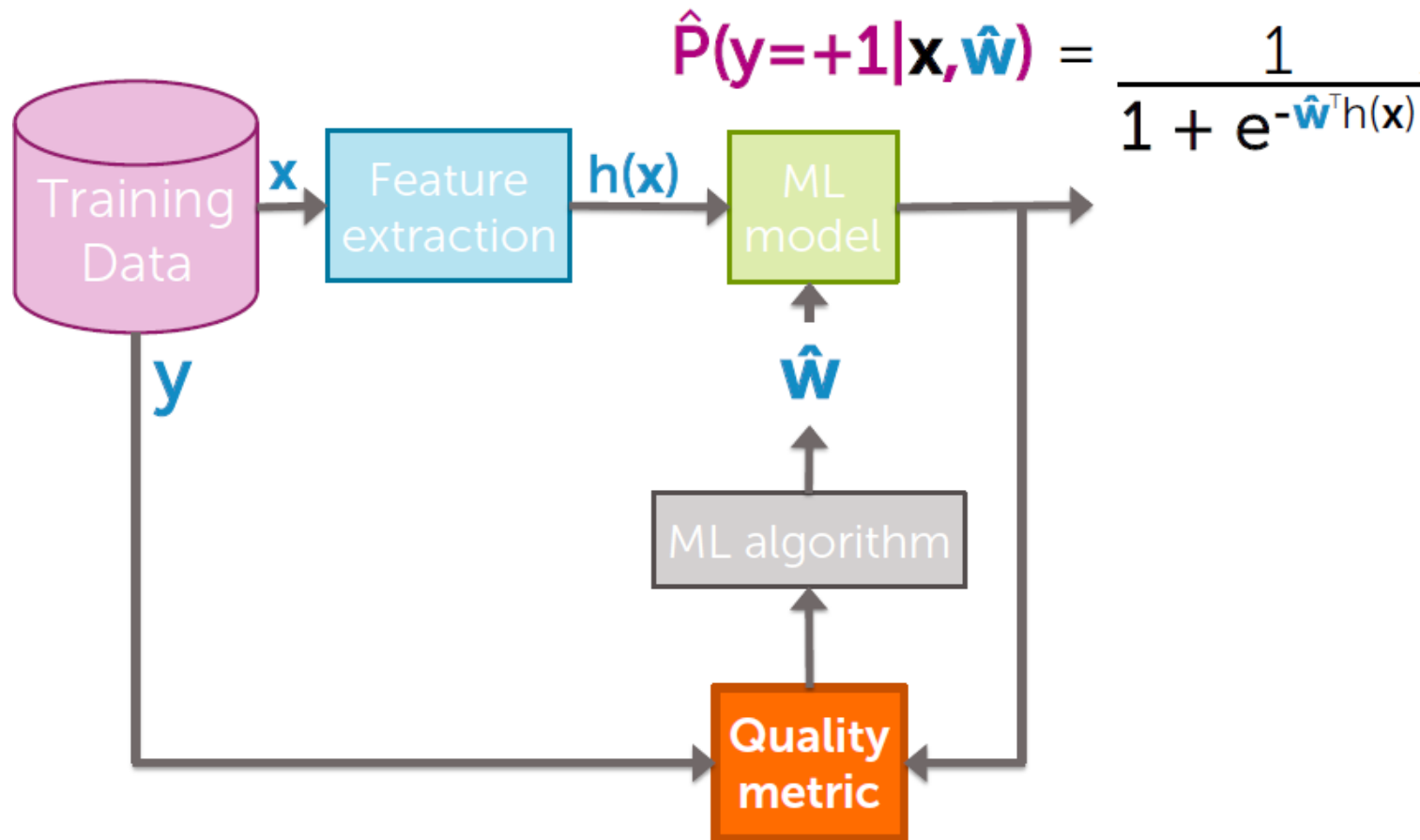
Overfitting → overconfident predictions

108



Quality metric → penalizing large coefficients

109

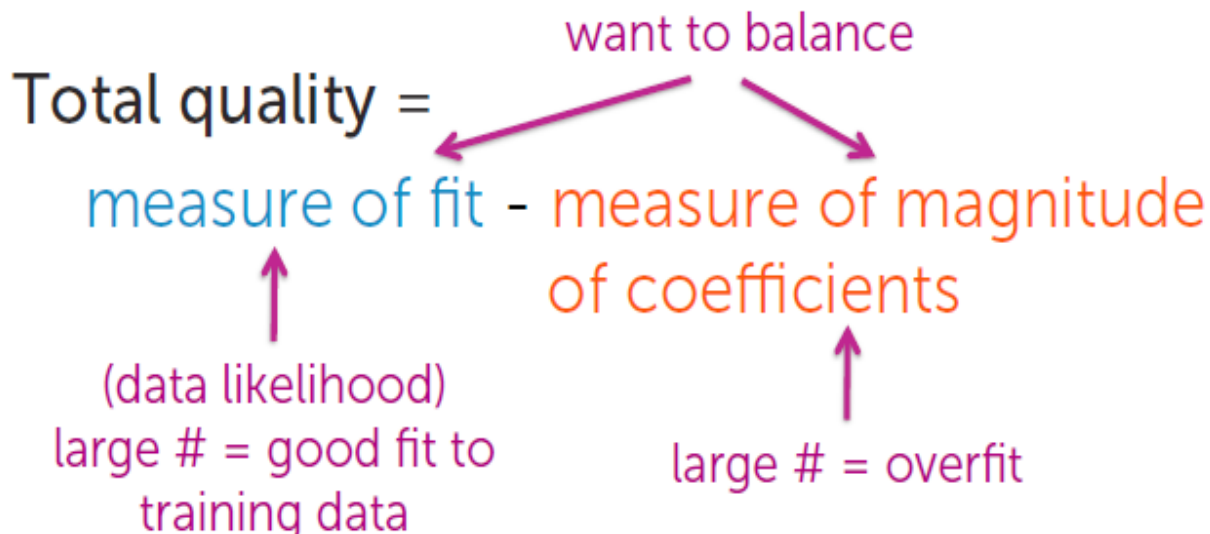


Desired total cost format

110

Want to balance:

- i. How well function fits data
- ii. Magnitude of coefficients



Maximum likelihood estimation (MLE)

111

□ Measure of fit = Data likelihood

- Choose coefficients \mathbf{w} that maximize likelihood:

$$\prod_{i=1}^N P(y_i \mid \mathbf{x}_i, \mathbf{w})$$

- Typically, we use the log of likelihood function
(simplifies math and has better convergence properties) ← !!!

$$\ell(\mathbf{w}) = \ln \prod_{i=1}^N P(y_i \mid \mathbf{x}_i, \mathbf{w})$$

Measure of magnitude of logistic regression coefficients

112

What summary # is indicative of size of logistic regression coefficients?

- Sum of squares (L_2 norm)

$$\|w\|_2^2 = w_0^2 + w_1^2 + w_2^2 + \dots + w_D^2$$

Penalize large Coefficients

- Sum of absolute value (L_1 norm)

$$\|w\|_1 = |w_0| + |w_1| + |w_2| + \dots + |w_D|$$

Sparse solution

Consider specific total cost

113

max
w

Total quality =

measure of fit - measure of magnitude
of coefficients

The diagram illustrates the components of the total quality function. A horizontal purple line is divided into two segments by a vertical purple line. The left segment is labeled $\ell(\mathbf{w})$ and the right segment is labeled $\|\mathbf{w}\|_2^2$. Below the left segment, a red arrow points to the text "log data likelihood". Below the right segment, a red arrow points to the text "L2 penalty".

↑
log data
likelihood

↑
L2 penalty

Consider resulting objectives

114

What if $\hat{\mathbf{w}}$ selected to minimize

$$\ell(\mathbf{w}) - \lambda \|\mathbf{w}\|_2^2$$

↖ tuning parameter = balance of fit and magnitude

If $\lambda=0$:

→ Reduces $\max_{\mathbf{w}} \ell(\mathbf{w}) \rightarrow$ Standard (unpenalized) MLE solution

If $\lambda=\infty$:

→ $\max_{\mathbf{w}} \ell(\mathbf{w}) - \infty \|\mathbf{w}\|_2^2 \rightarrow$ only care about penalizing \mathbf{w} , large coefficients $\rightarrow \mathbf{w}=0$

If λ in between:

→ Balance data fit against the magnitude of the coefficients

Consider resulting objectives

115

What if $\hat{\mathbf{w}}$ selected to minimize

$$\ell(\mathbf{w}) - \lambda \|\mathbf{w}\|_2^2$$

 tuning parameter = balance of fit and magnitude

L_2 regularized
logistic regression

Pick λ using:

- Validation set (for large datasets)
- Cross-validation (for smaller datasets)
(see regression course)

Bias-variance tradeoff

116

Large λ :

high bias, low variance

(e.g., $\hat{\mathbf{w}} = 0$ for $\lambda = \infty$)

In essence, λ
controls model
complexity

Small λ :

low bias, high variance

(e.g., maximum likelihood (MLE) fit of
high-order polynomial for $\lambda = 0$)

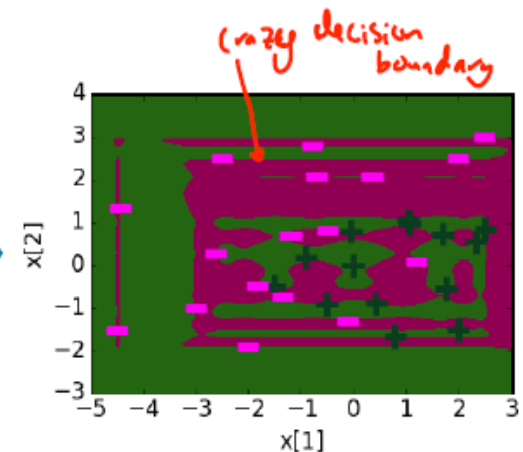
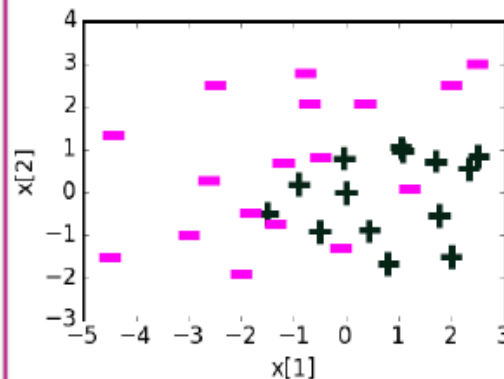
Visualizing effect of regularisation

117

Degree 20 features, $\lambda=0$

Feature	Value	Coefficient learned
$h_0(\mathbf{x})$	1	8.7
$h_1(\mathbf{x})$	$x[1]$	5.1
$h_2(\mathbf{x})$	$x[2]$	78.7
...
$h_{11}(\mathbf{x})$	$(x[1])^6$	-7.5
$h_{12}(\mathbf{x})$	$(x[2])^6$	<u>3803</u>
$h_{13}(\mathbf{x})$	$(x[1])^7$	21.1
$h_{14}(\mathbf{x})$	$(x[2])^7$	<u>-2406</u>
...
$h_{37}(\mathbf{x})$	$(x[1])^{19}$	$-2 \cdot 10^{-6}$
$h_{38}(\mathbf{x})$	$(x[2])^{19}$	-0.15
$h_{39}(\mathbf{x})$	$(x[1])^{20}$	$-2 \cdot 10^{-8}$
$h_{40}(\mathbf{x})$	$(x[2])^{20}$	0.03

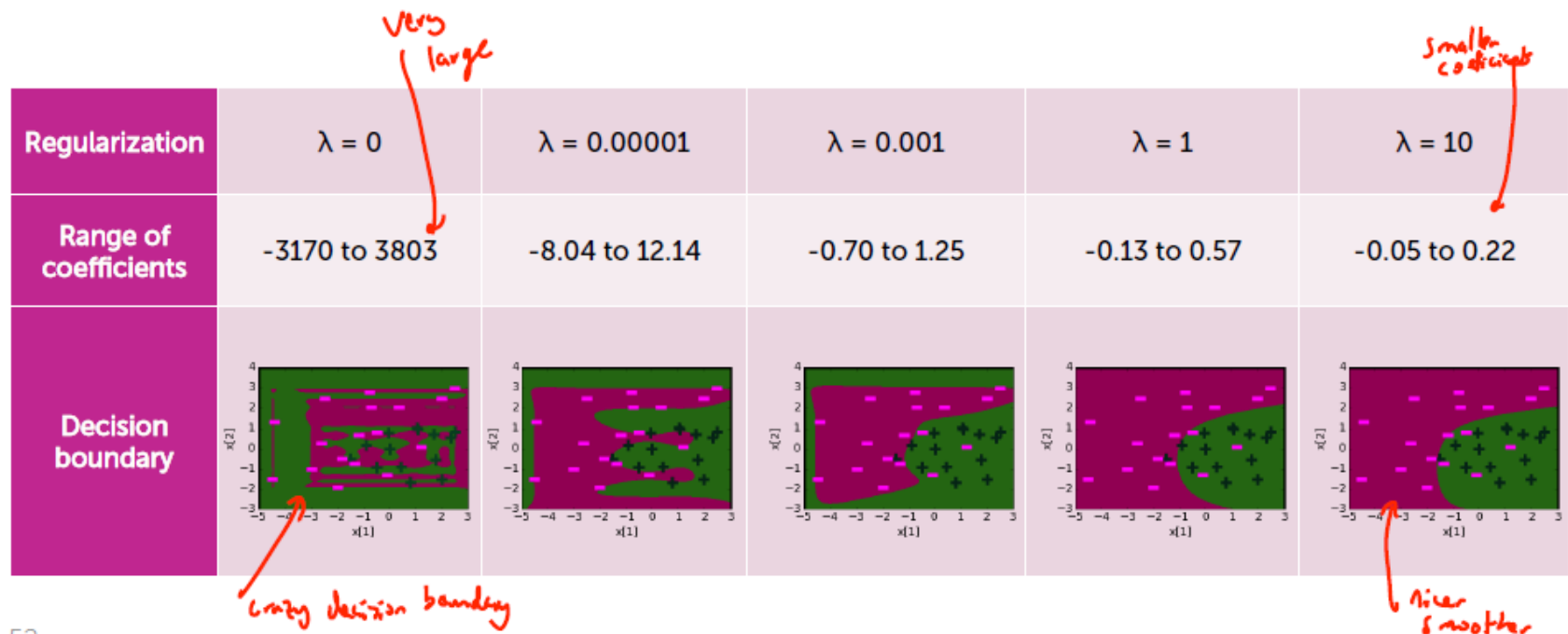
Coefficients range from -3170 to 3803



Visualizing effect of regularisation

118

Degree 20 features,
effect of regularization penalty λ

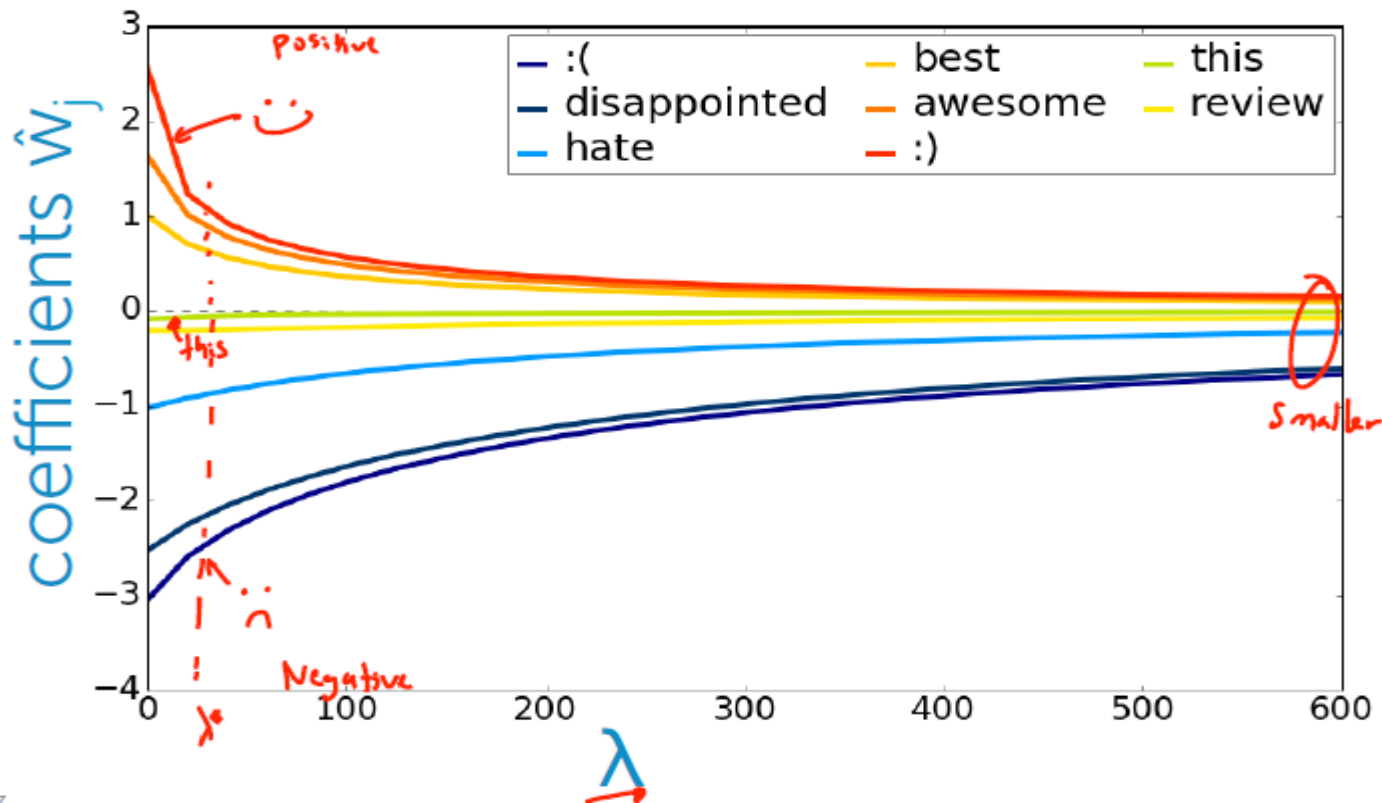


52

Effect of regularisation

119

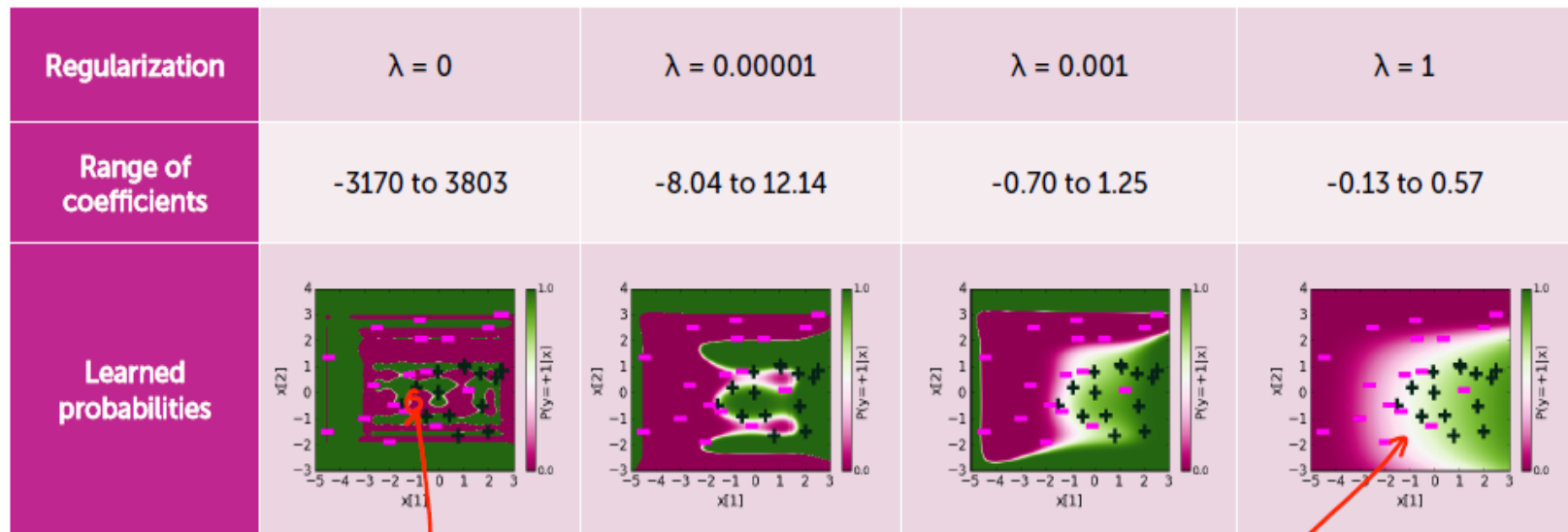
Coefficient path



Visualizing effect of regularisation

120

Degree 20 features:
regularization reduces "overconfidence"



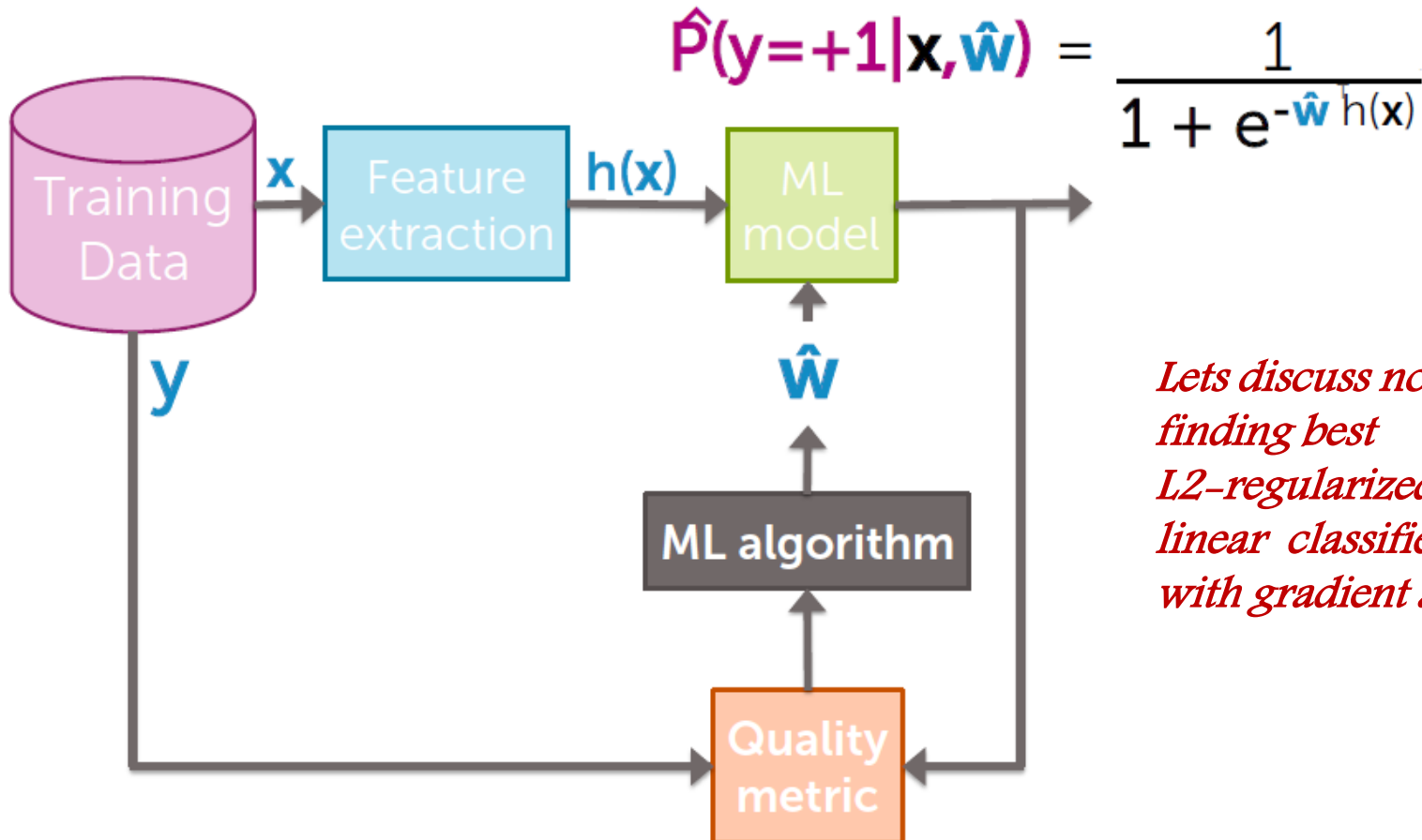
highly
over confident

very natural uncertainty
region

Flow chart:

ML algorithm

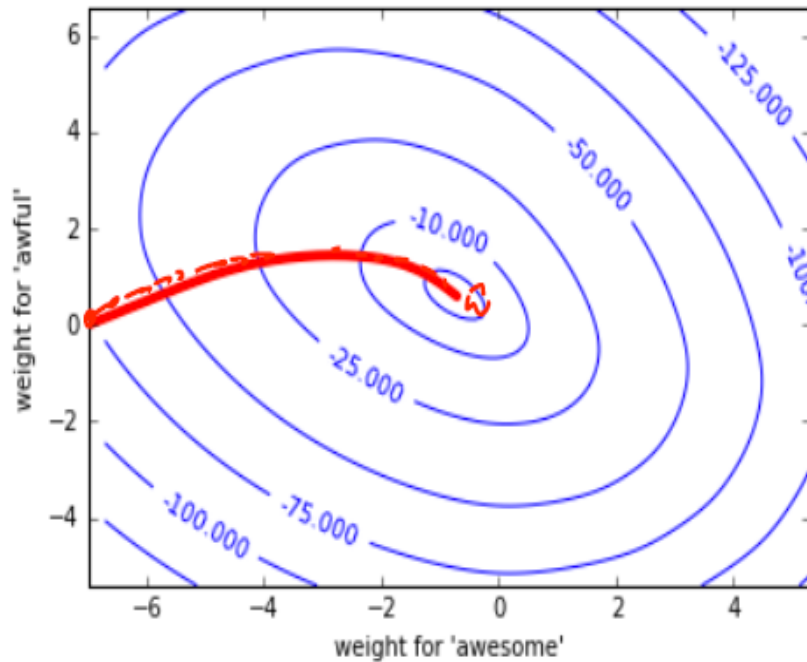
121



*Lets discuss now
finding best
L2-regularized
linear classifier
with gradient ascent*

Gradient ascent

122



Algorithm:

while not converged

$$\underline{\mathbf{w}}^{(t+1)} \leftarrow \underline{\mathbf{w}}^{(t)} + \eta \nabla \ell(\underline{\mathbf{w}}^{(t)})$$

need the gradient of
regularized log likelihood

Gradient of L2 regularized log-likelihood

123

Total quality =

measure of fit - measure of magnitude
of coefficients

The diagram shows two terms, $\ell(\mathbf{w})$ and $\lambda \|\mathbf{w}\|_2^2$, each enclosed in a purple bracket. A purple arrow points from $\ell(\mathbf{w})$ down to the first term of the derivative equation, and another purple arrow points from $\lambda \|\mathbf{w}\|_2^2$ down to the second term of the derivative equation.

$$\text{Total derivative} = \frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} - \lambda \frac{\partial \|\mathbf{w}\|_2^2}{\partial \mathbf{w}_j}$$

Gradient of L2 regularized log-likelihood

124

Derivative of (log-)likelihood

Diagram illustrating the derivative of the log-likelihood function. The equation is:

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^N h_j(\mathbf{x}_i) \left(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w}) \right)$$

Annotations:

- Sum over data points (points to the summation index i)
- Feature value (points to $h_j(\mathbf{x}_i)$)
- Difference between truth and prediction (points to the term in parentheses)

Derivative of L_2 penalty

$$\frac{\partial \|\mathbf{w}\|_2^2}{\partial \mathbf{w}_j} = \frac{\partial}{\partial w_j} [w_0^2 + w_1^2 + w_2^2 + \dots + w_j^2 + \dots + w_D^2] = 2w_j$$

Gradient of L2 regularized log-likelihood

125

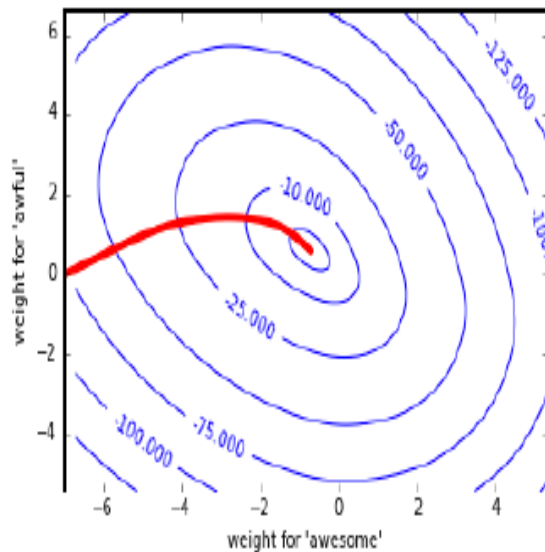
Understanding contribution of L_2 regularization

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} \quad \overbrace{- 2\lambda \mathbf{w}_j}^{\text{Term from } L_2 \text{ penalty}}$$

	$- 2 \lambda w_j$	Impact on w_j
$w_j > 0$	< 0	decreases $w_j \Rightarrow w_j$ becomes closer to 0
$w_j < 0$	> 0	increases $w_j \Rightarrow w_j$ becomes closer to 0

Gradient ascent with L2 regularization

126



init $\mathbf{w}^{(1)} = 0$ (or randomly, or smartly), $t=1$

while not converged:

for $j=0, \dots, D$

$$\text{partial}[j] = \sum_{i=1}^N h_j(\mathbf{x}_i) \left(\mathbb{1}[y_i = +1] - P(y = +1 | \mathbf{x}_i, \mathbf{w}^{(t)}) \right)$$

$$\mathbf{w}_j^{(t+1)} \leftarrow \mathbf{w}_j^{(t)} + \eta \left(\text{partial}[j] - 2\lambda \mathbf{w}_j^{(t)} \right)$$

$$t \leftarrow t + 1$$

step
size

$\frac{\partial \ell(\mathbf{w})}{\partial w_j}$

only change \mathbf{w}_j !!

Logistic regression with L1 regularization

127

Recall **sparsity** (many $\hat{\mathbf{w}}_j=0$)
gives efficiency and interpretability

Efficiency:

- If $\text{size}(\mathbf{w}) = 100\text{B}$, each prediction is expensive
- If $\hat{\mathbf{w}}$ **sparse**, computation only depends on # of non-zeros



$$\hat{y}_i = \text{sign} \left(\sum_{\hat{\mathbf{w}}_j \neq 0} \hat{\mathbf{w}}_j h_j(\mathbf{x}_i) \right)$$

Interpretability:

- Which features are relevant for prediction?

Sparse logistic regression

128

Total quality =

measure of fit - measure of magnitude
of coefficients

The diagram shows two horizontal curly braces. The first brace is under the text 'measure of fit' and is labeled $\ell(\mathbf{w})$. The second brace is under the text 'measure of magnitude of coefficients' and is labeled $\|\mathbf{w}\|_1 = |w_0| + \dots + |w_D|$.

L_1 regularized
logistic regression

Leads to
sparse
solutions!

L1 regularised logistic regression

129

Just like L2 regularization, solution is governed by a continuous parameter λ

$$\ell(\mathbf{w}) - \lambda \|\mathbf{w}\|_1$$

tuning parameter =
balance of fit and sparsity

If $\lambda=0$:

→ No regularization → standard MLE solution

If $\lambda=\infty$:

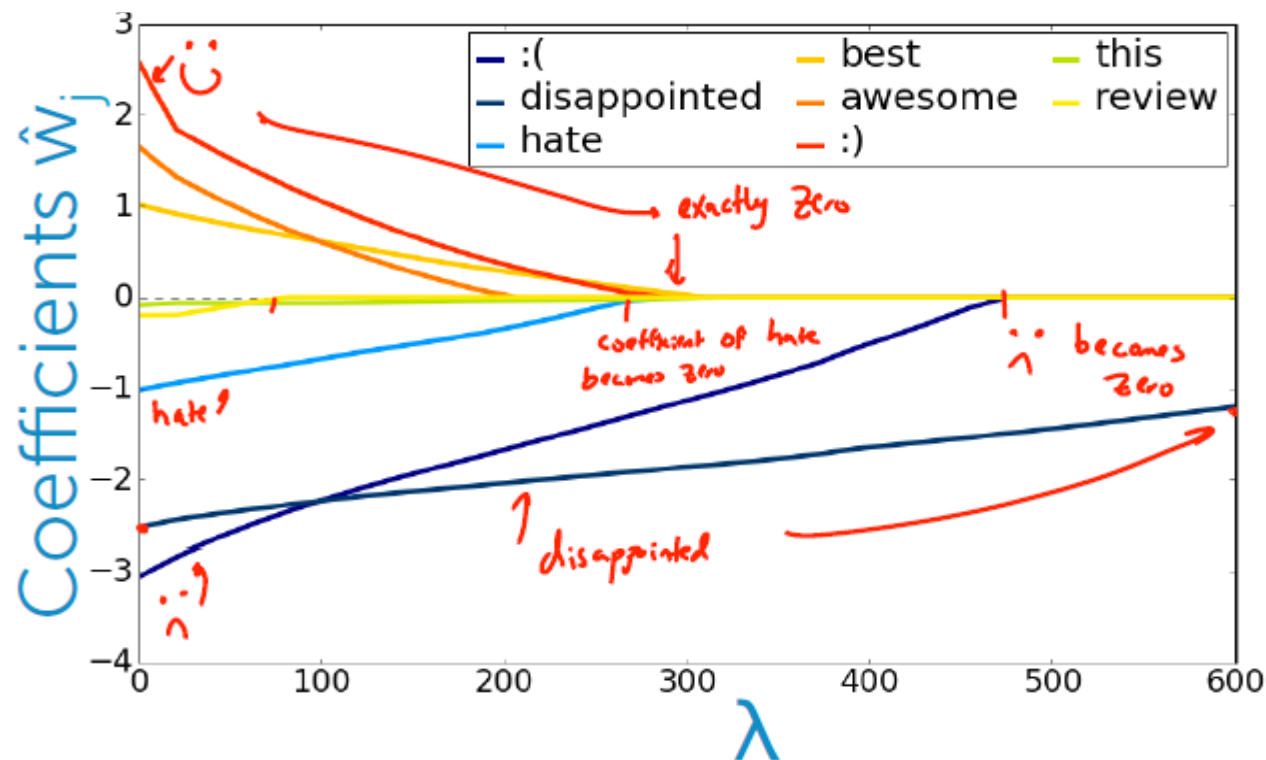
→ all weight is on regularization → $\hat{\mathbf{w}} = \mathbf{0}$

If λ in between:

→ Sparse solutions: some $\hat{w}_i \neq 0$, many other $\hat{w}_i = 0$

L1 regularised logistic regression

130



What you can do now...

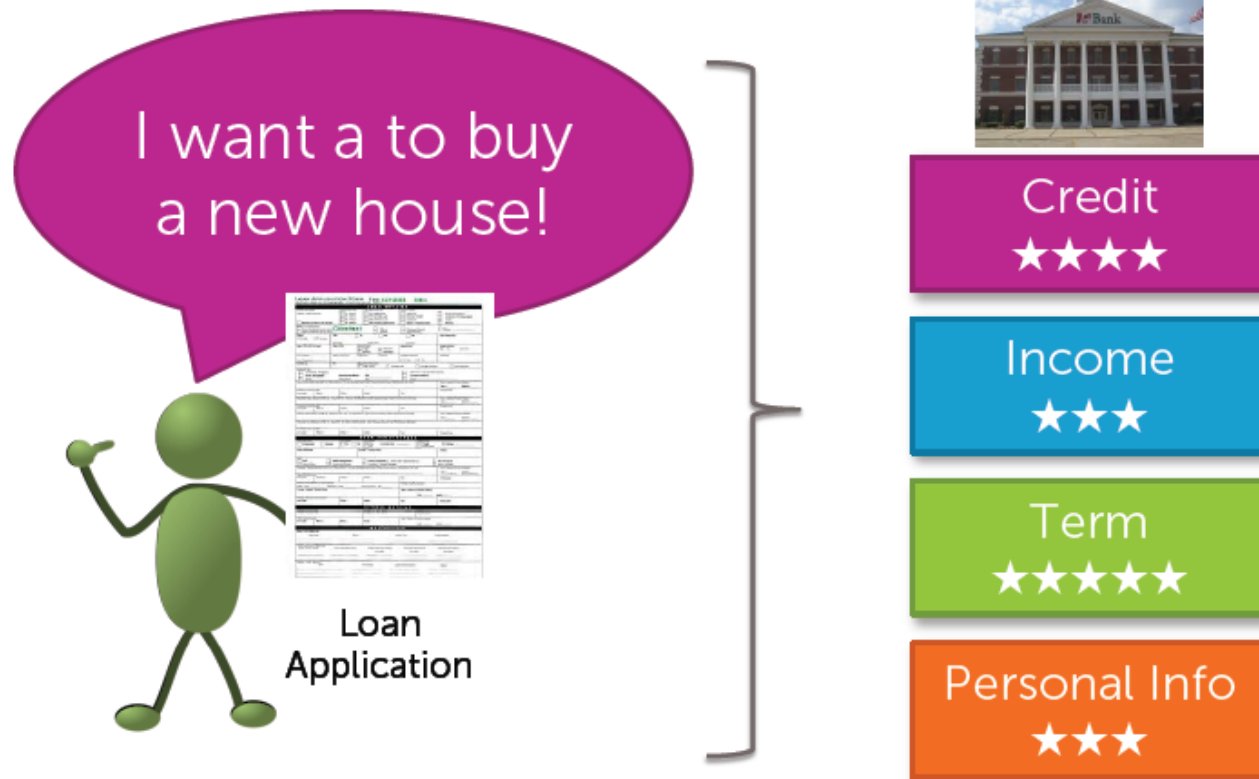
131

- Identify when overfitting is happening
- Relate large learned coefficients to overfitting
- Describe the impact of overfitting on decision boundaries and predicted probabilities of linear classifiers
- Motivate the form of L_2 regularized logistic regression quality metric
- Describe what happens to estimated coefficients as tuning parameter λ is varied
- Interpret coefficient path plot
- Estimate L_2 regularized logistic regression coefficients using gradient ascent
- Describe the use of L_1 regularization to obtain sparse logistic regression solutions

Decision trees

What makes a loan risky?

133



Credit history explained

134

Did I pay previous
loans on time?



Example: excellent,
good, or fair

Credit History



Income



Term



Personal Info



Income

135

What's my income?

Example:
\$80K per year



Credit History

★★★★

Income

★★★

Term

★★★★★

Personal Info

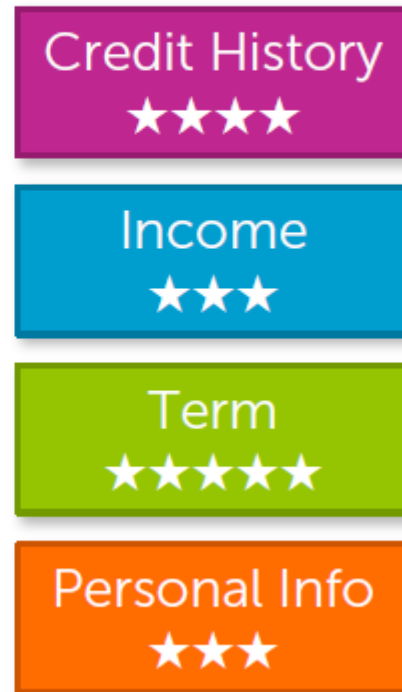
★★★

Loan terms

136

How soon do I need to
pay the loan?

Example: 3 years,
5 years,...

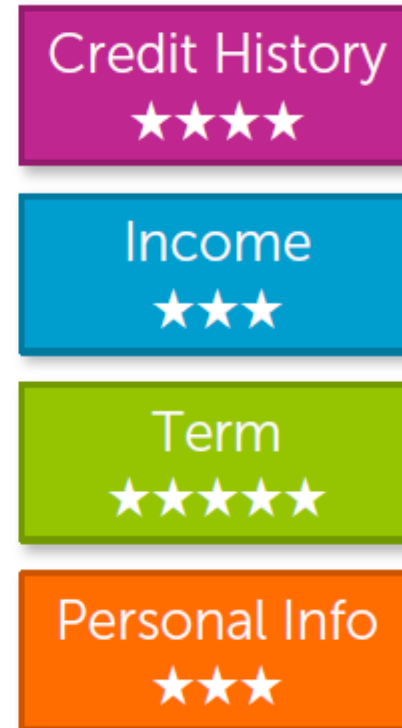


Personal information

137

Age, reason for the loan, marital status,...

Example: Home loan for a married couple



Intelligent application

138

Loan
Applications

A sample loan application form with a pink border and header.A sample loan application form with a blue border and header.A sample loan application form with a green border and header.

Intelligent loan application
review system

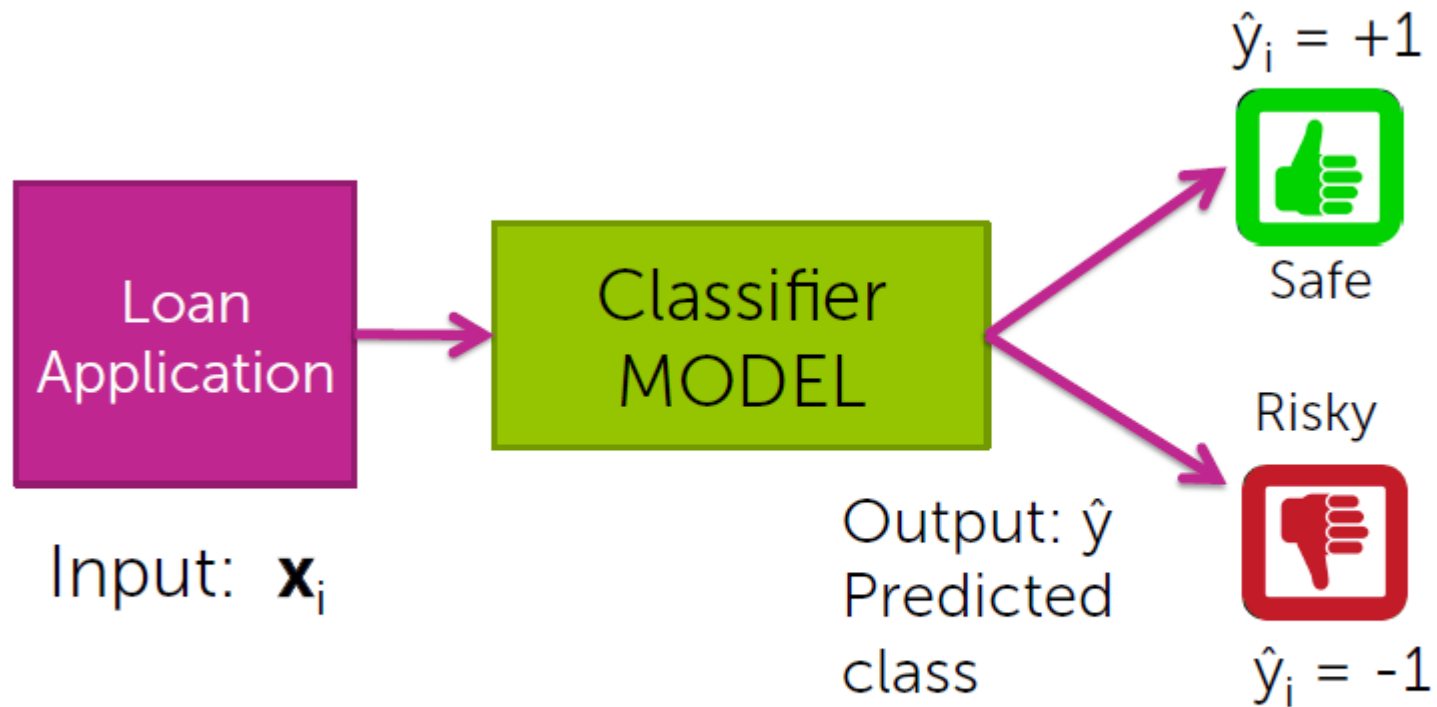
Safe
✓

Risky
✗

Risky
✗

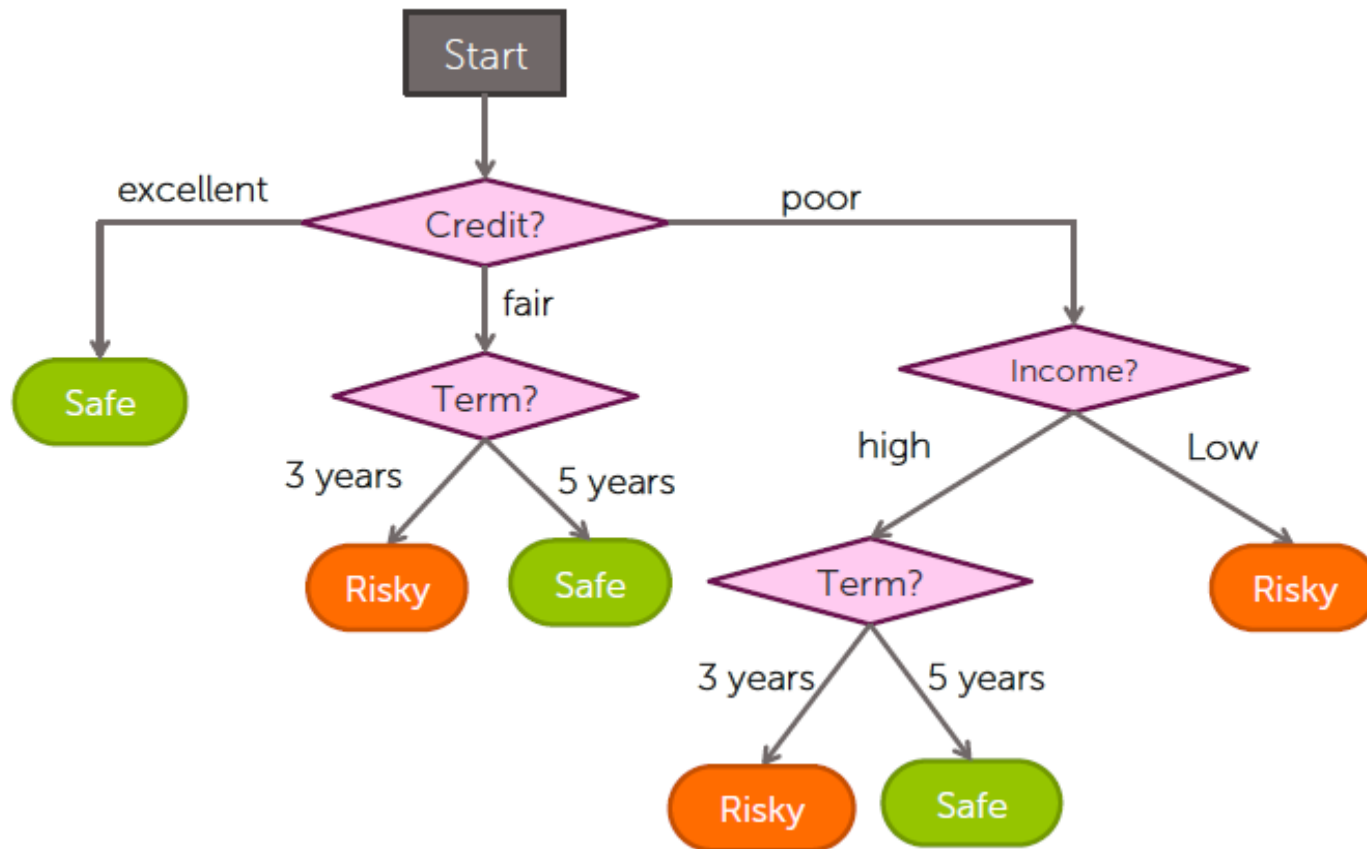
Classifier: review type

139



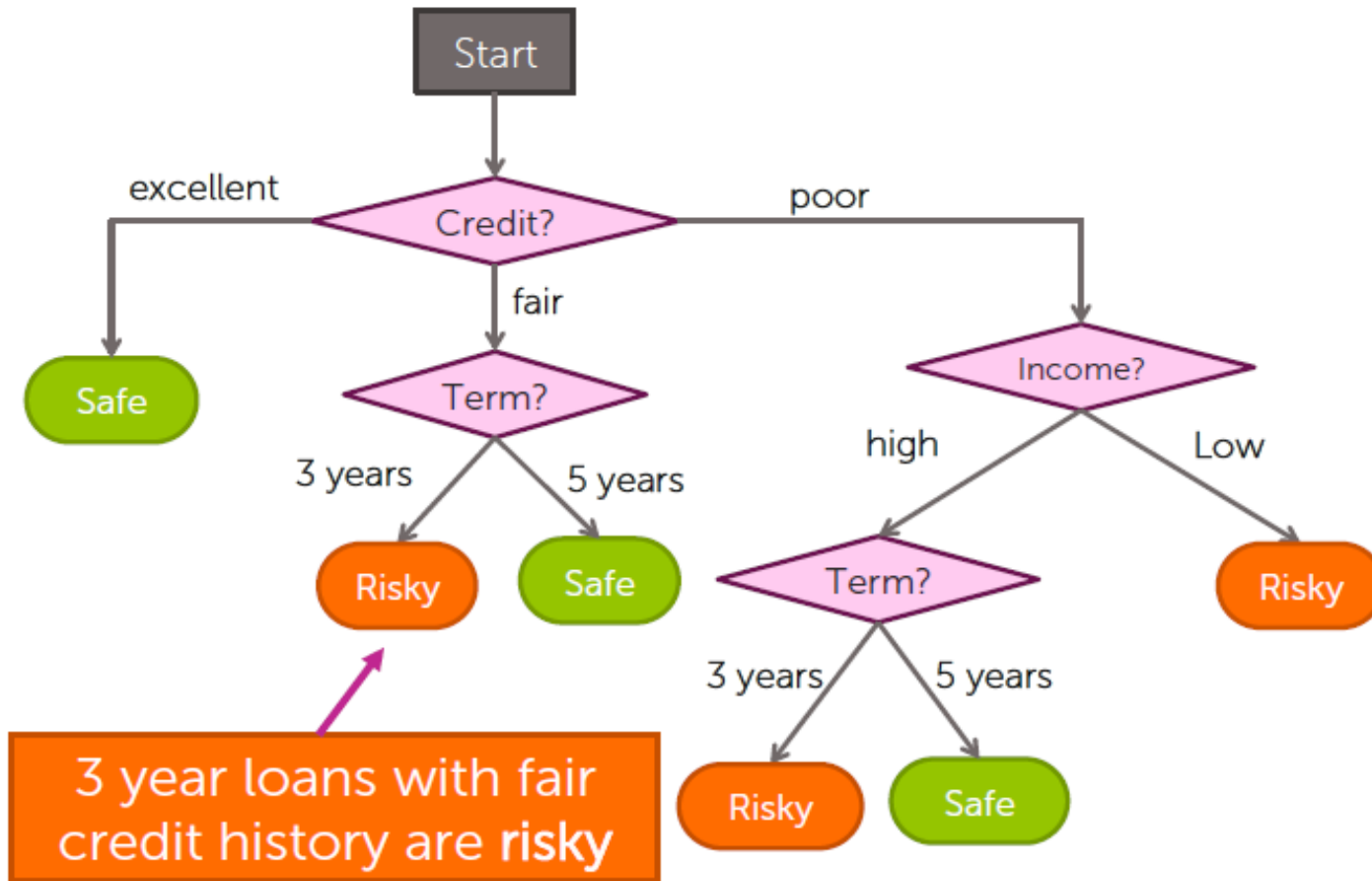
Classifier: decision trees

140



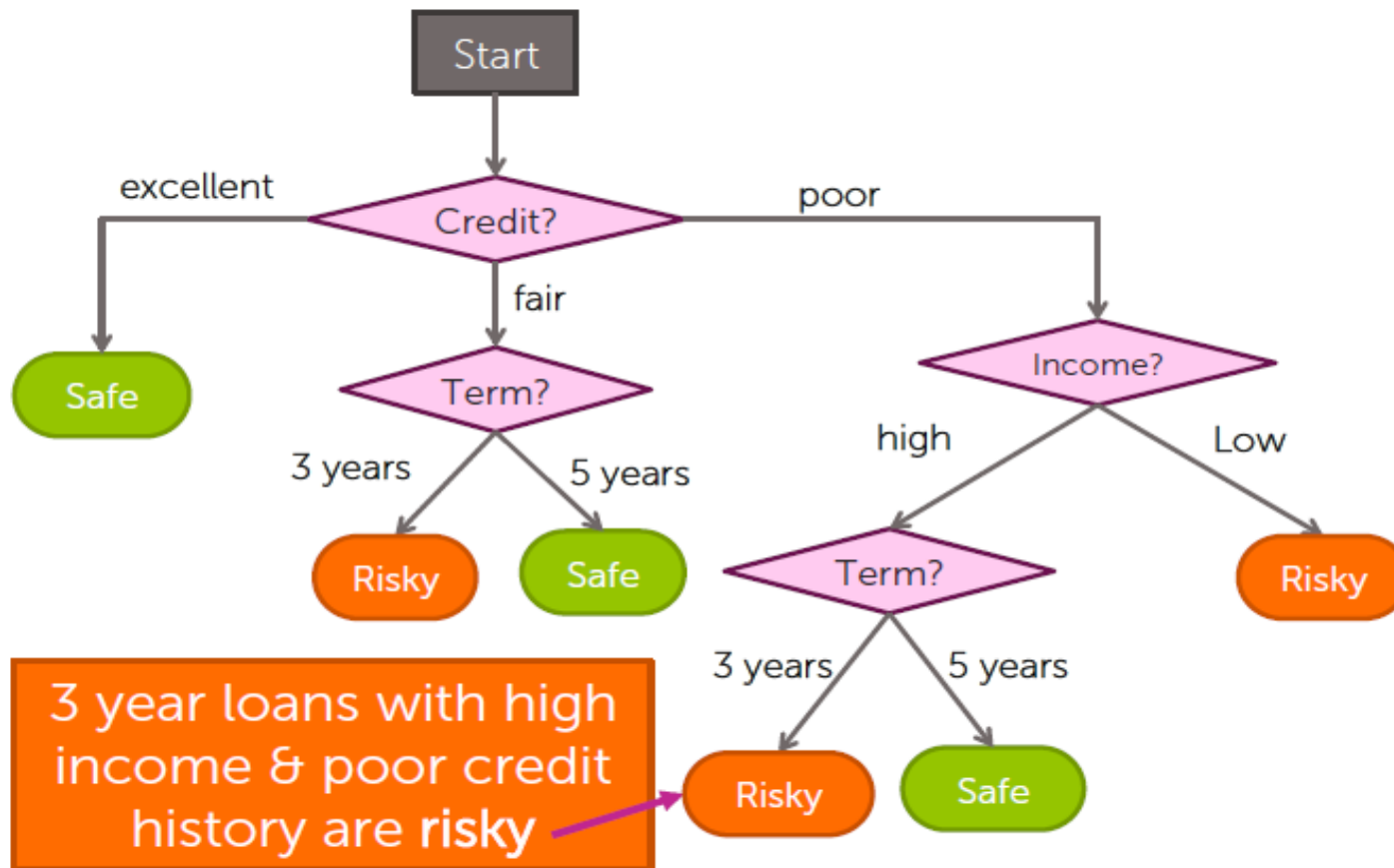
Scoring a loan application

141



Scoring a loan application

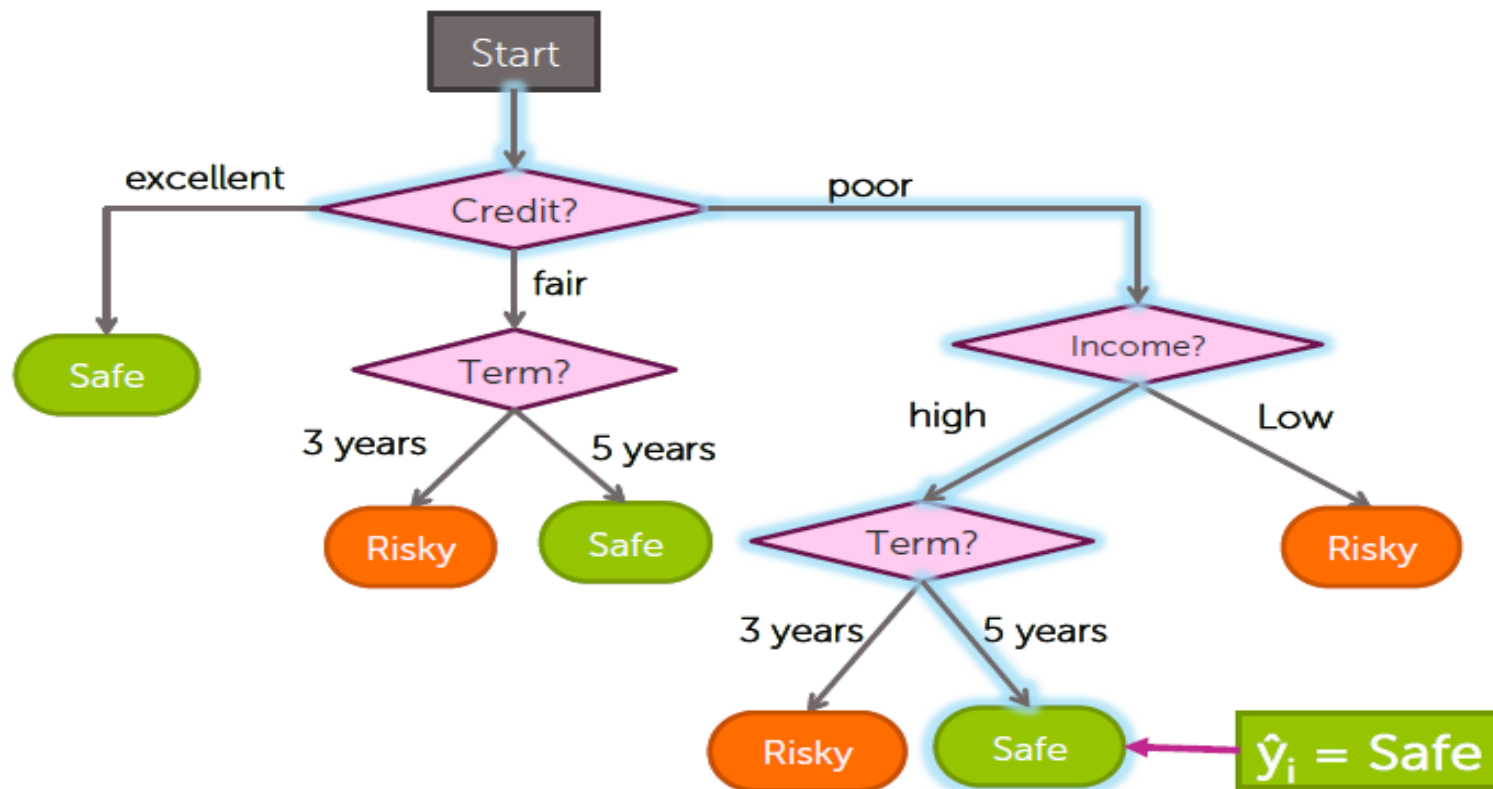
142



Scoring a loan application

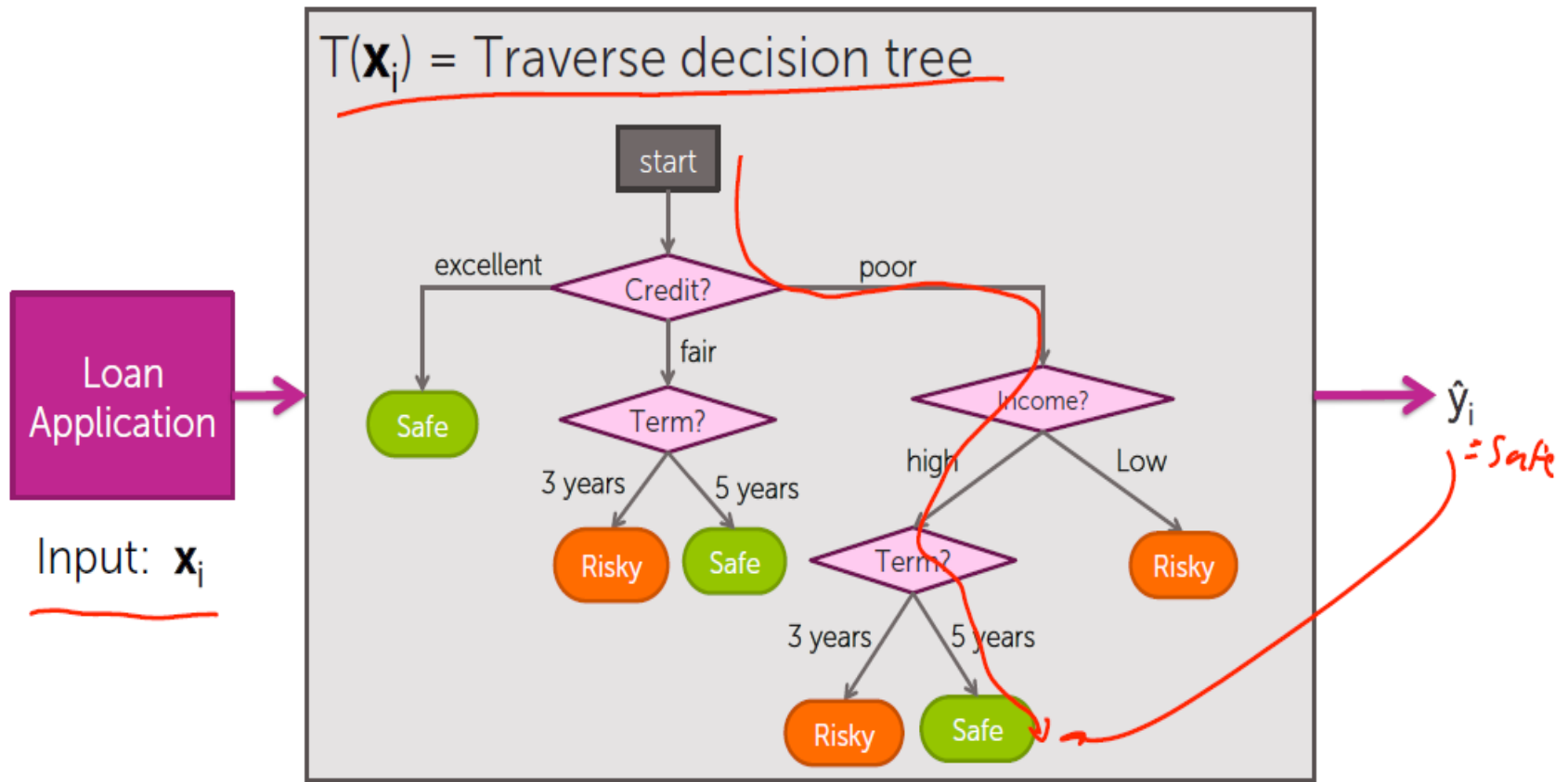
143

$\mathbf{x}_i = (\text{Credit} = \text{poor}, \text{Income} = \text{high}, \text{Term} = 5 \text{ years})$



Decision tree model

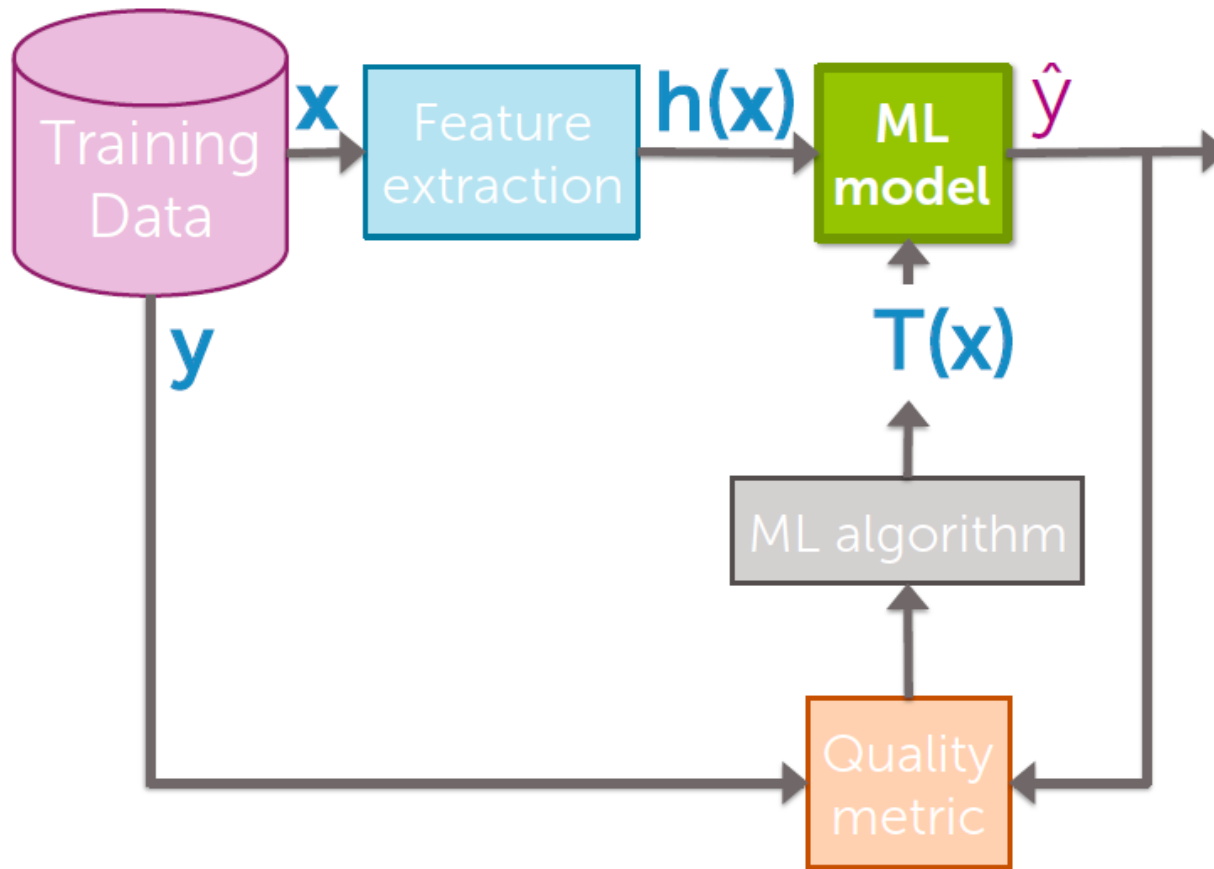
144



Flow chart:

ML
model

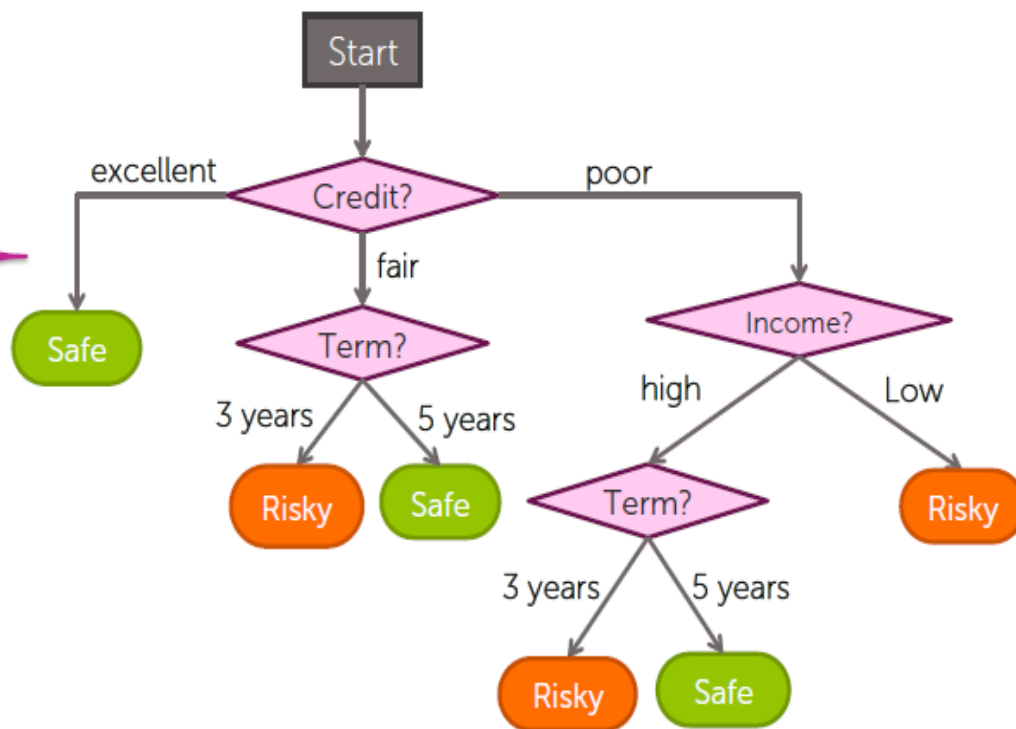
145



Learn decision tree from data

146

$h_1(x)$	$h_2(x)$	$h_3(x)$	Loan status
Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

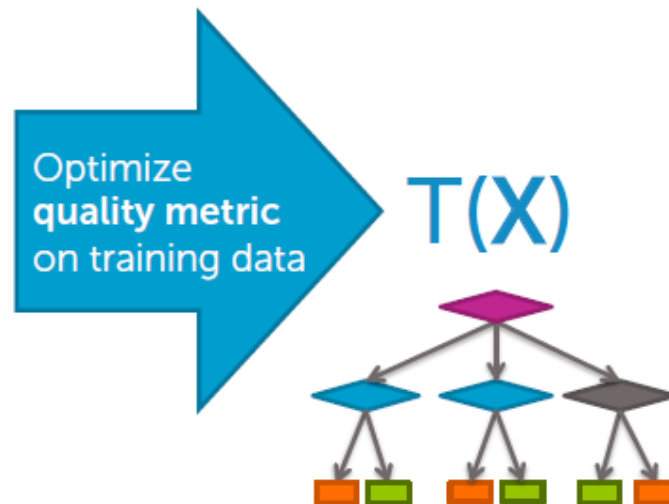


Learn decision tree from data

147

Training data: N observations (\mathbf{x}_i, y_i)

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe



Quality metric: Classification error

148

- Error measures fraction of mistakes

$$\text{Error} = \frac{\text{\# incorrect predictions}}{\text{\# examples}}$$

- Best possible value : 0.0
- Worst possible value: 1.0

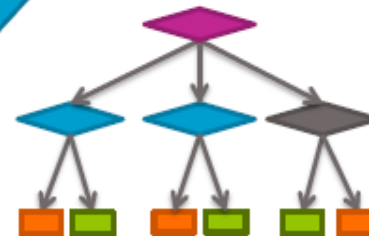
Find the tree with lowest classification error

149

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

Minimize
classification error
on training data

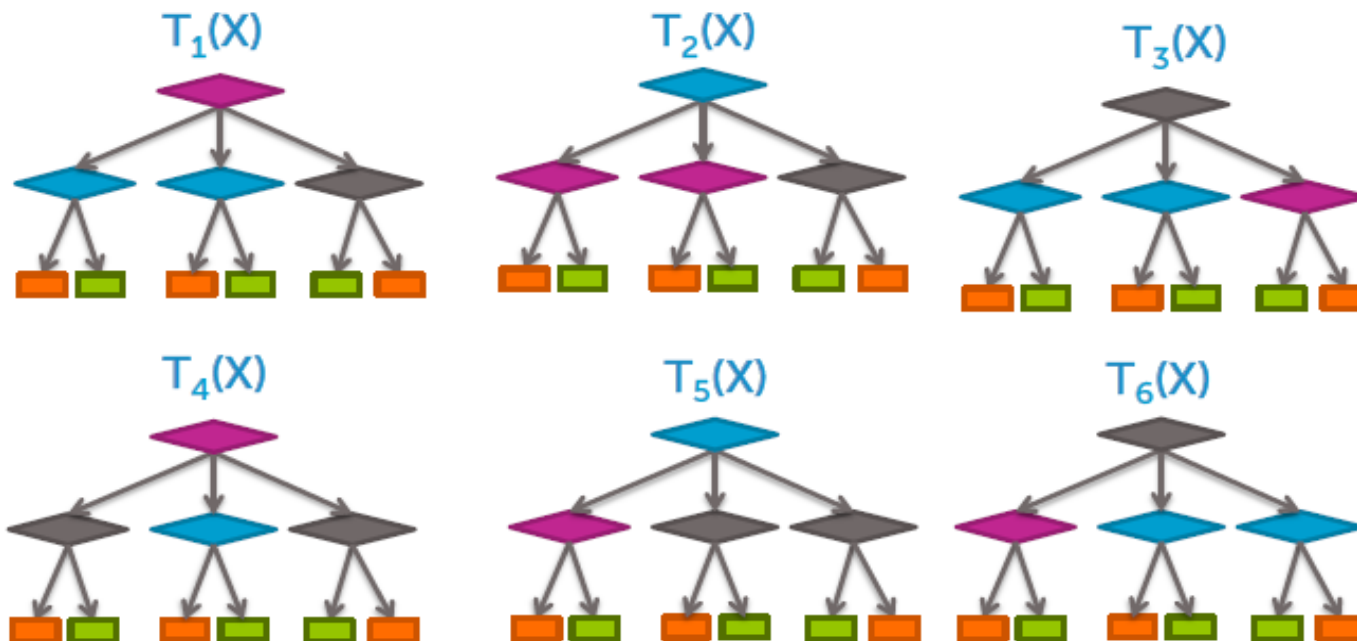
$T(X)$



How do we find the best tree?

150

Exponentially large number of possible trees makes decision tree learning **hard**!
(NP-hard problem)



Simple (greedy) algorithm finds good tree

151

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

Approximately
minimize
classification error
on training data

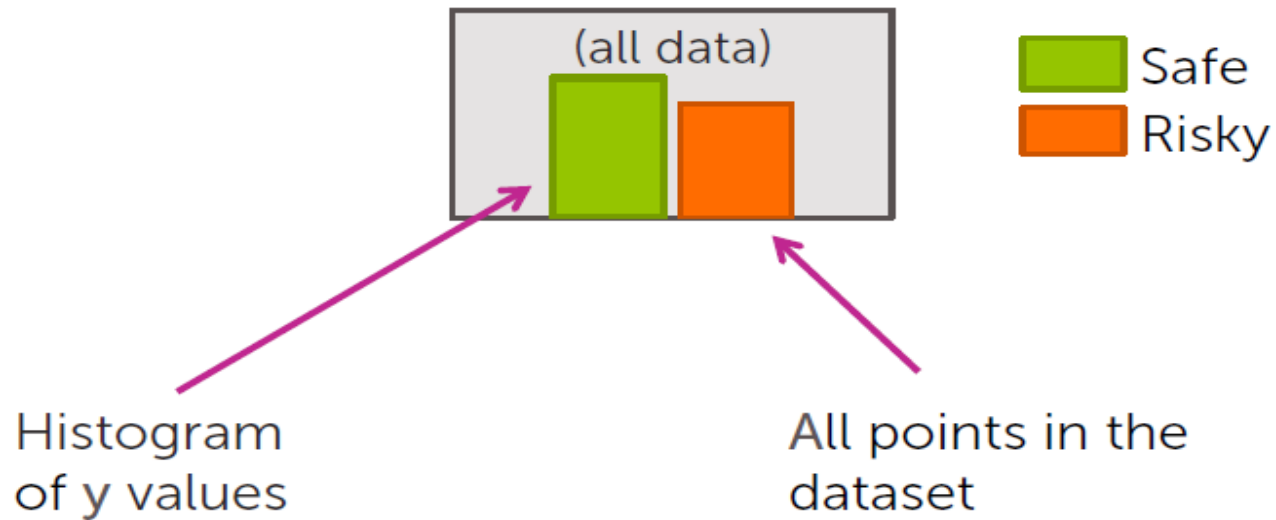
$T(X)$



Greedy algorithm

152

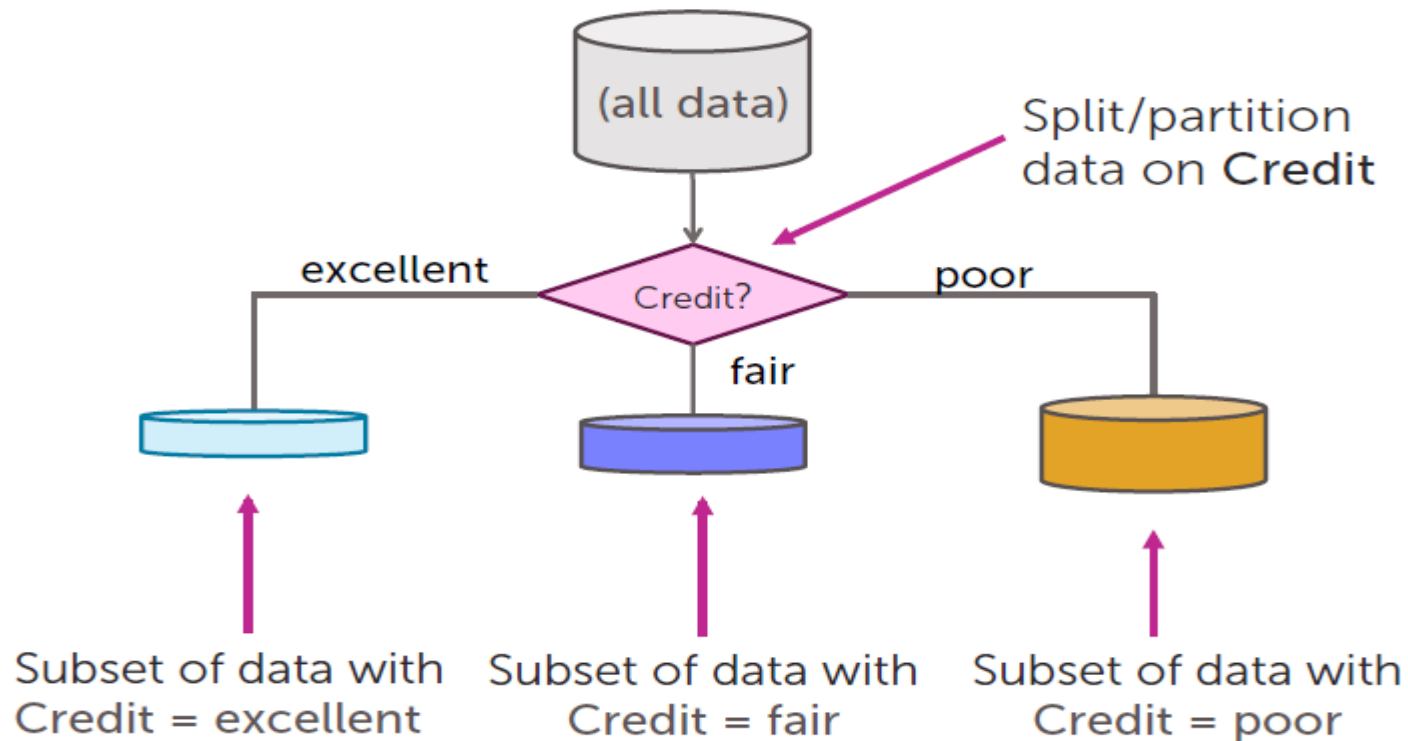
Step 1: Start with an empty tree



Greedy algorithm

153

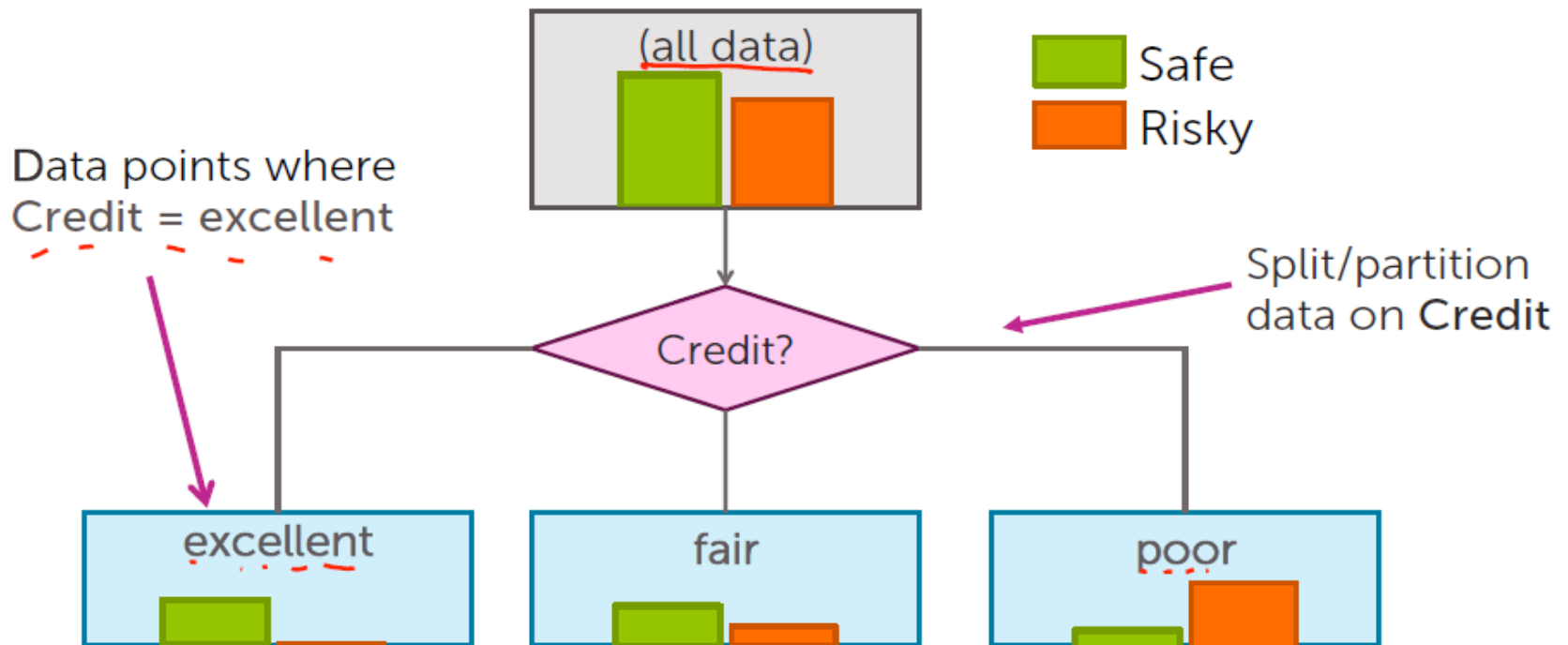
Step 2: Split on a feature



Greedy algorithm

154

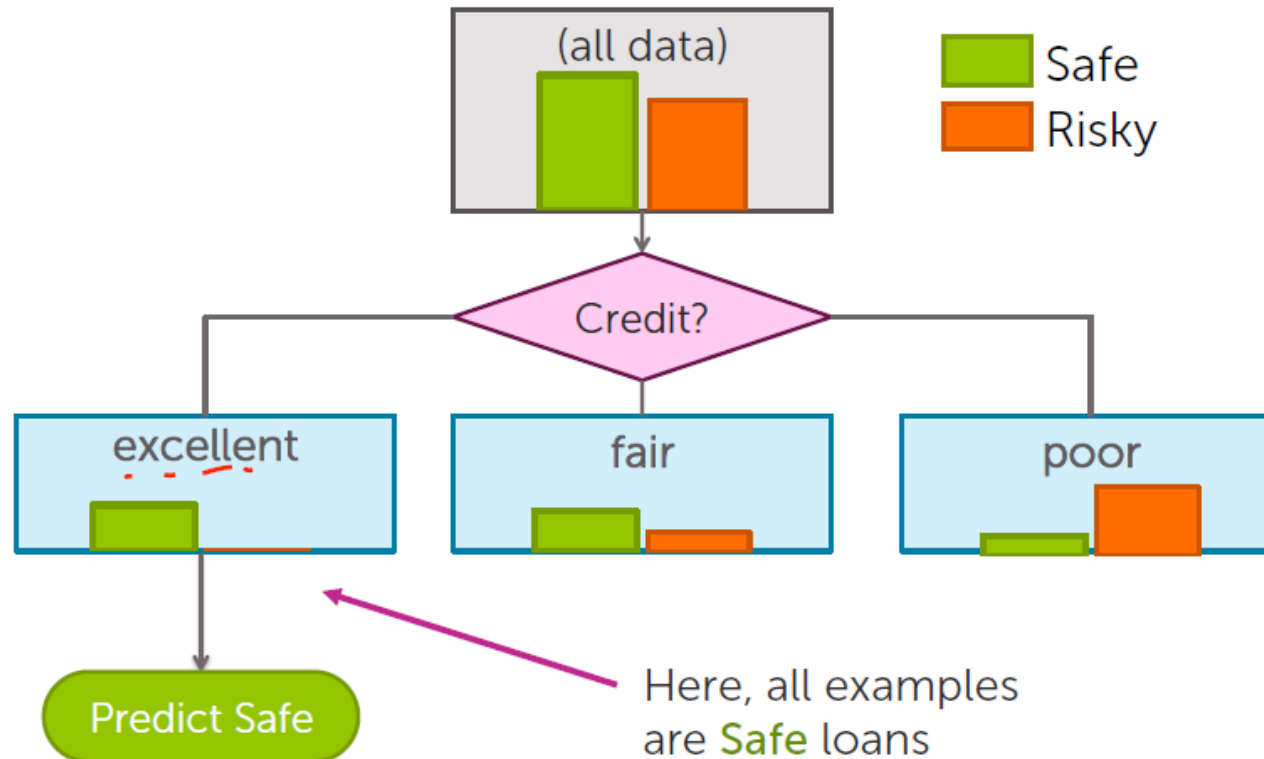
Feature split explained



Greedy algorithm

155

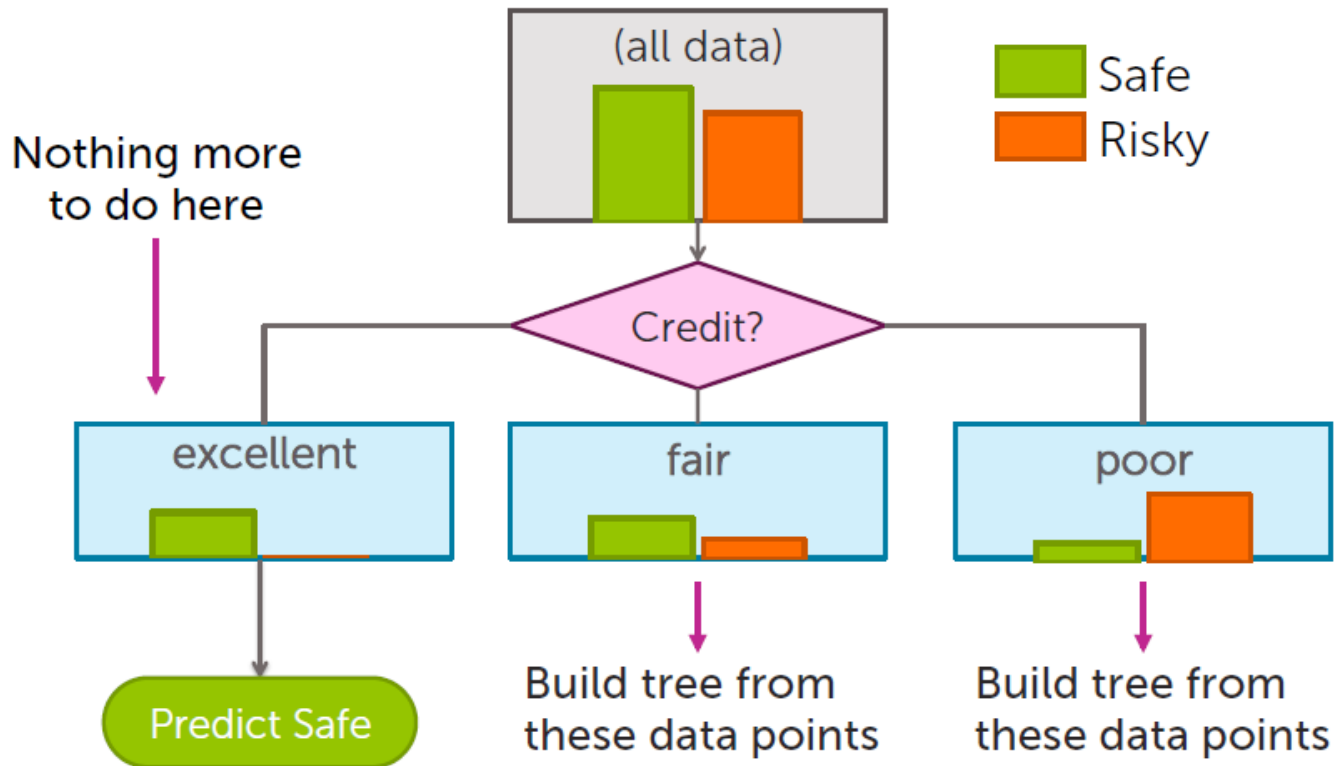
Step 3: Making predictions



Greedy algorithm

156

Step 4: Recursion



Greedy decision tree learning

157

- Step 1: Start with an empty tree

- Step 2: Select a feature to split data

- For each split of the tree:

- Step 3: If nothing more to, make predictions

- Step 4: Otherwise, go to Step 2 & continue (recurse) on this split

Problem 1: Feature split selection

Problem 2: Stopping condition

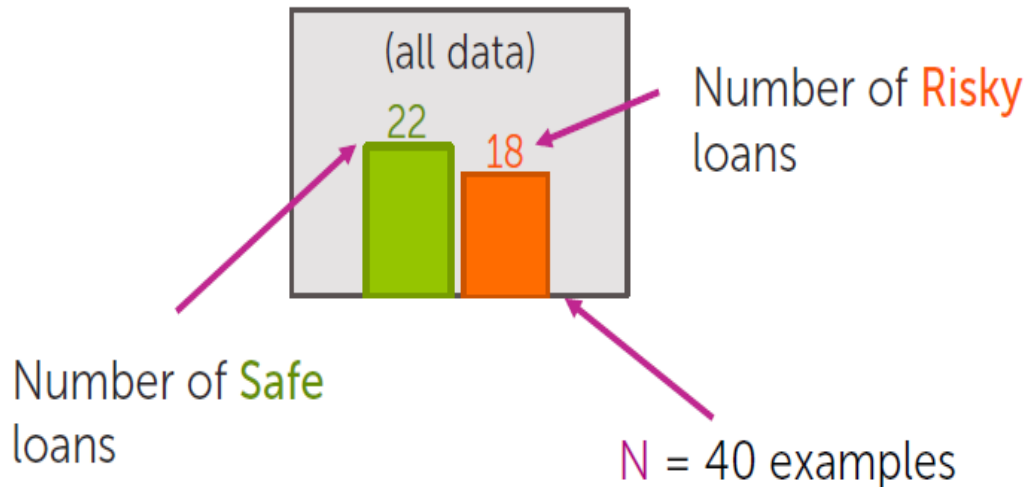
Recursion

Feature split learning

158

Start with all the data

Loan status: Safe Risky



Assume $N = 40$, 3 features

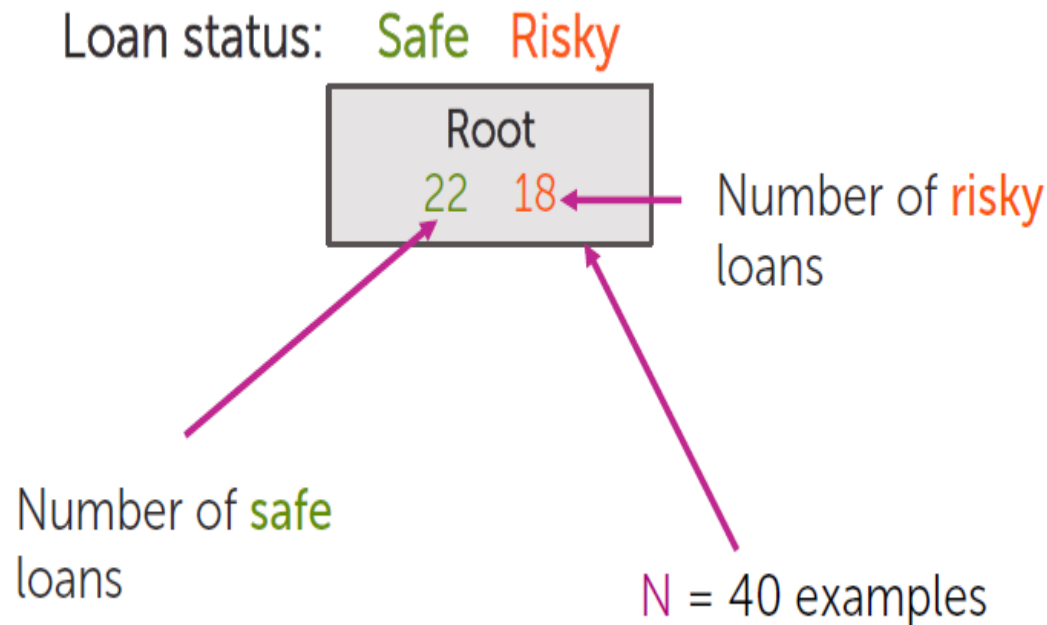
Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

Feature split learning

159

Start with all the data

Assume $N = 40$, 3 features

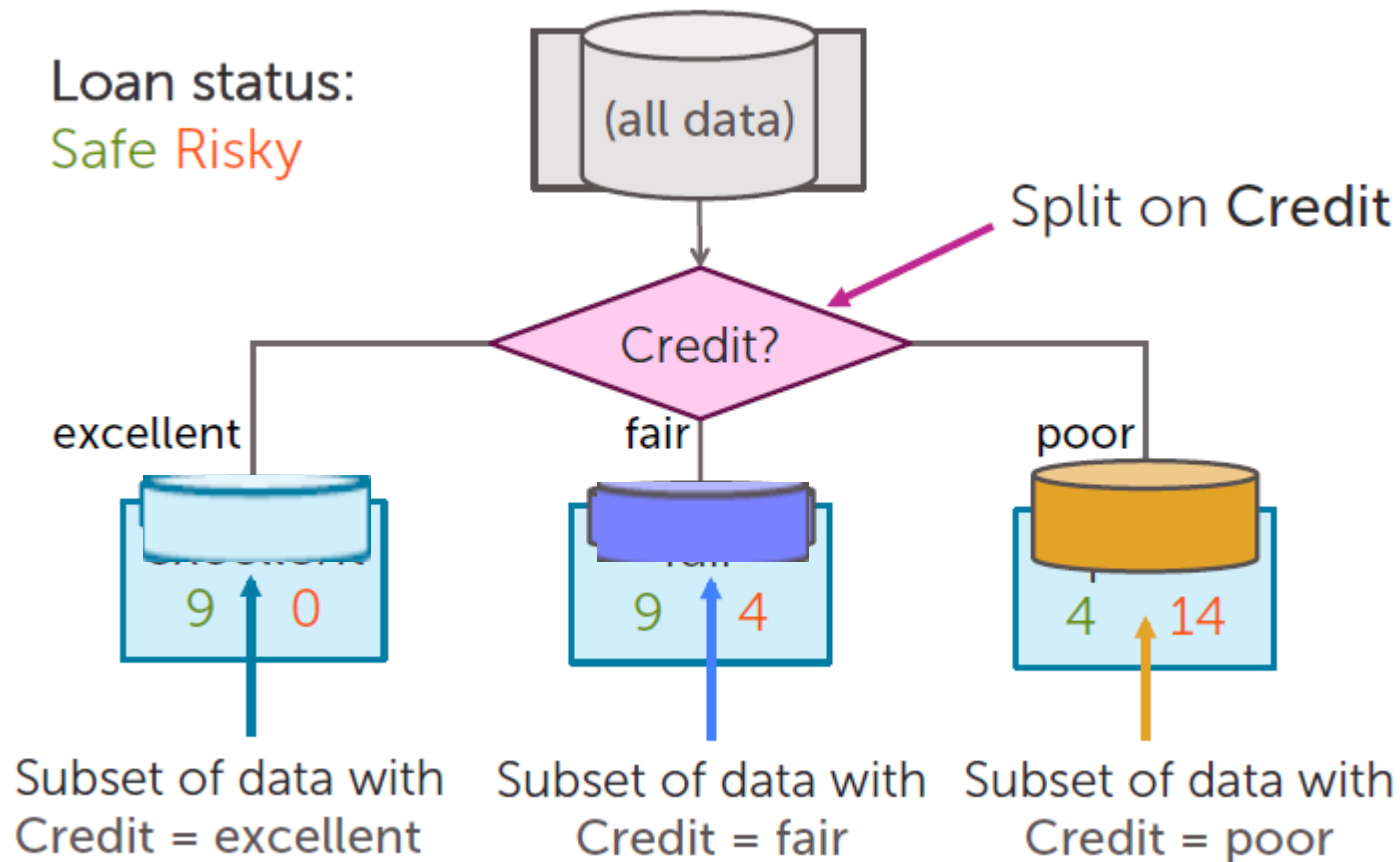


Compact notation

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

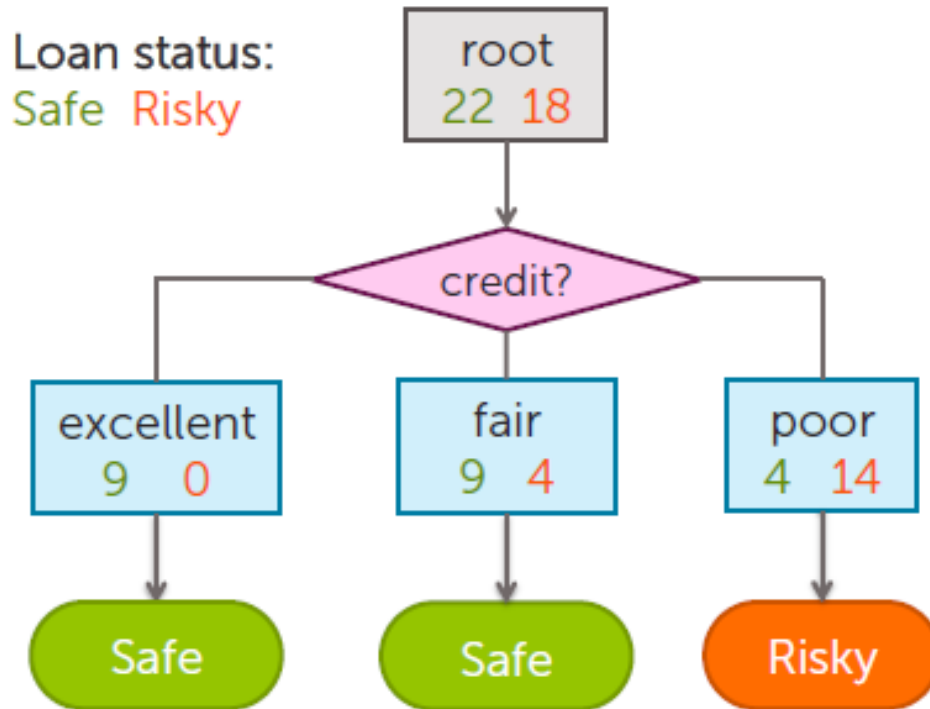
Decision stump: single level tree

160



Making predictions with a decision stump

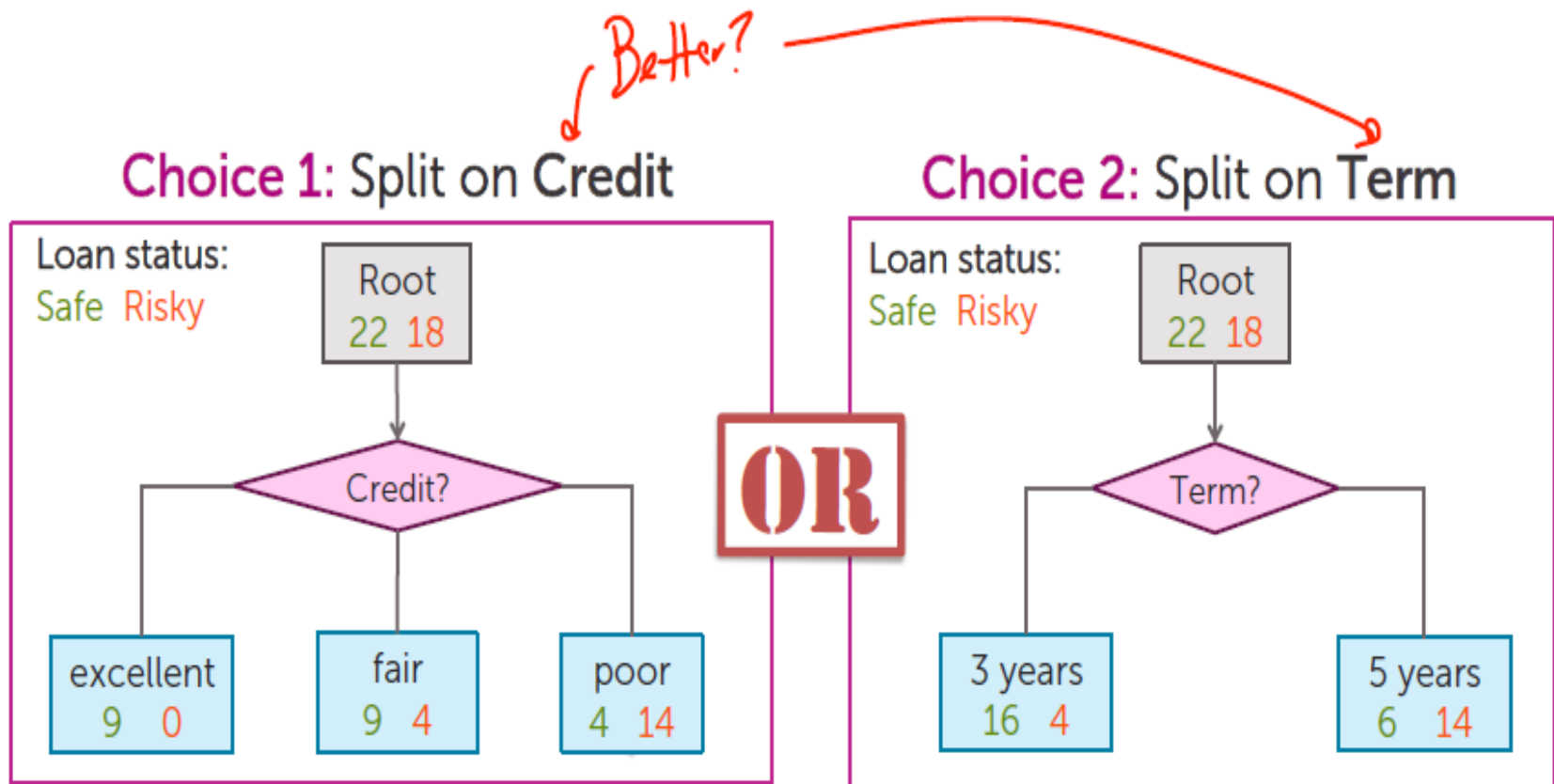
161



For each intermediate node,
set \hat{y} = majority value

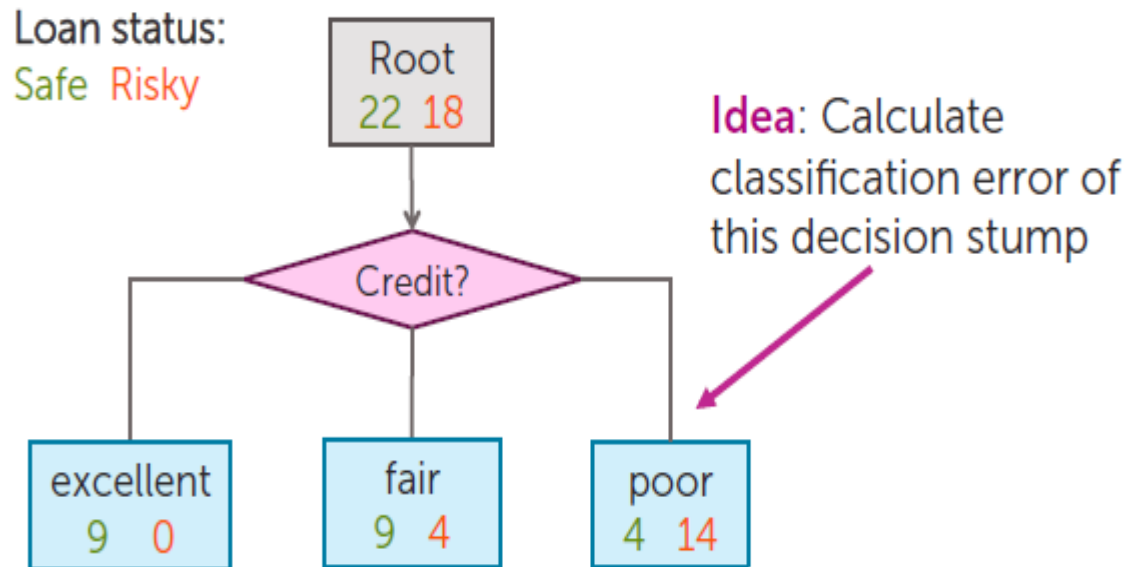
How do we select the best feature to split on?

162



How do we measure effectiveness of a split?

163

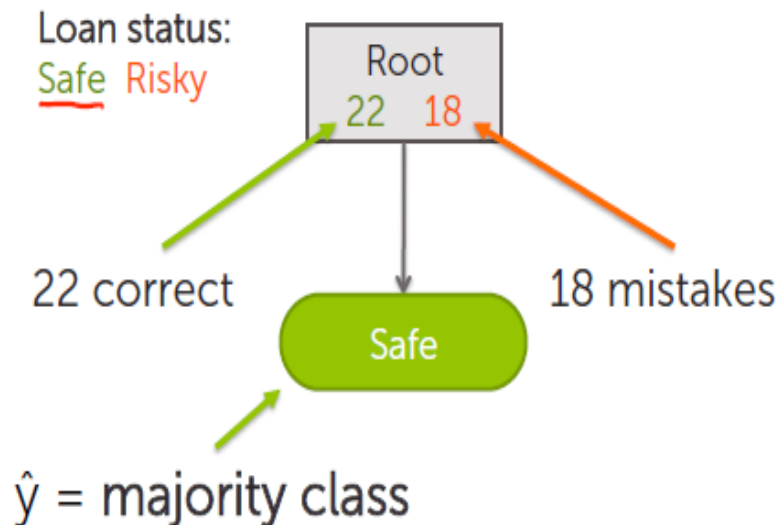


$$\text{Error} = \frac{\text{\# mistakes}}{\text{\# data points}}$$

Calculating classification error

164

- **Step 1:** \hat{y} = class of majority of data in node
- **Step 2:** Calculate classification error of predicting \hat{y} for this data



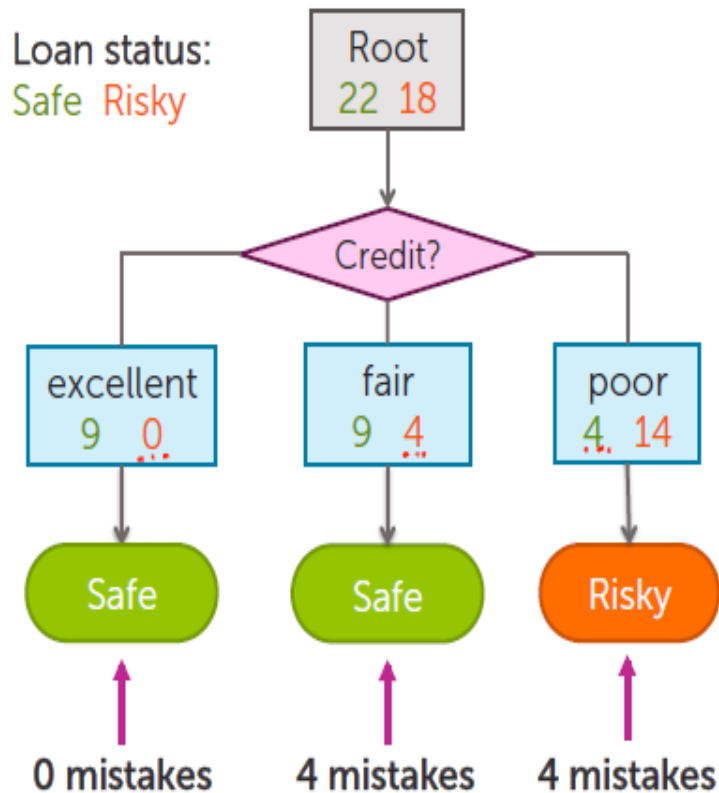
$$\text{Error} = \frac{18}{22+18}$$
$$= 0.45$$

Tree	Classification error
(root)	0.45

Classification error

165

Choice 1: Split on Credit



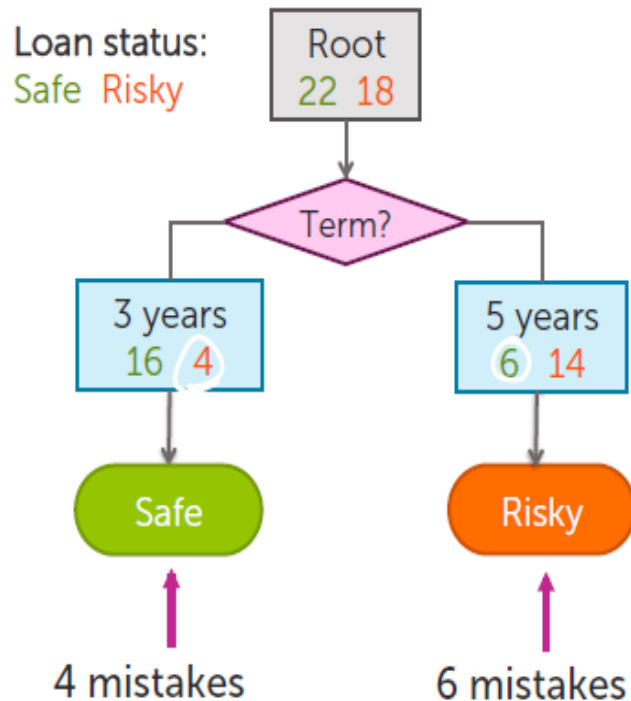
$$\text{Error} = \frac{4 + 4}{40} = 0.20$$

Tree	Classification error
(root)	0.45
Split on credit	0.2

Classification error

166

Choice 2: Split on Term



$$\text{Error} = \frac{4+6}{40} = 0.25$$

Tree	Classification error
(root)	0.45
Split on credit	0.2
Split on term	0.25

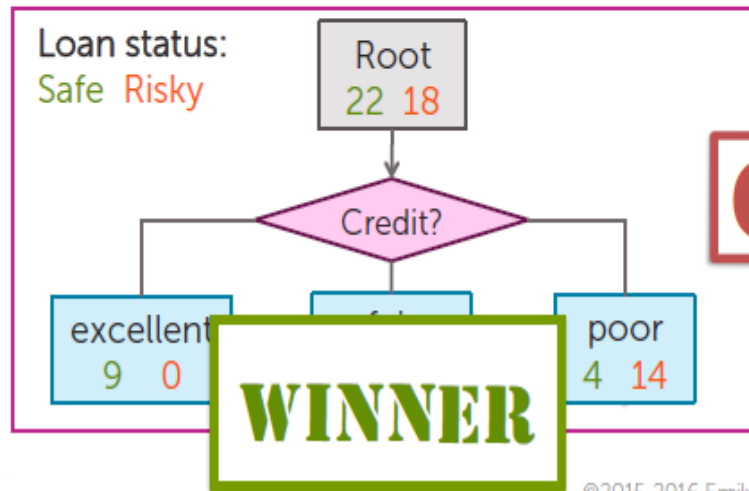
Choice 1 vs Choice 2

167

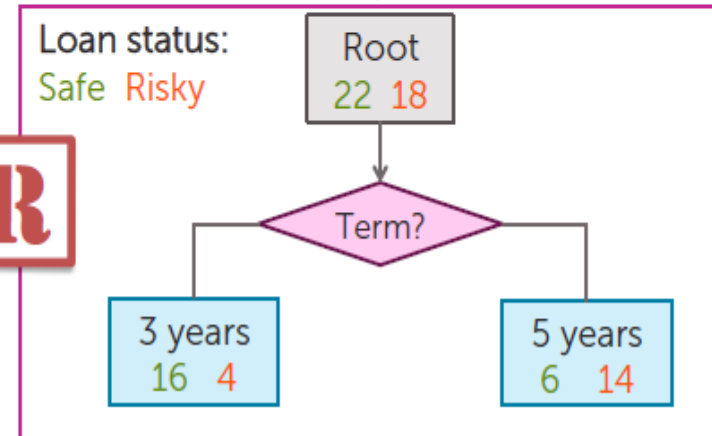
Tree	Classification error
(root)	0.45
split on <u>credit</u>	0.2
split on loan term	0.25

← First split!

Choice 1: Split on Credit



Choice 2: Split on Term



OR

Feature split selection algorithm


168

- Given a subset of data M (a node in a tree)
- For each feature $h_i(x)$: *credit, term, income*
 1. Split data of M according to feature $h_i(x)$
 2. Compute classification error split
- Chose feature $h^*(x)$ with lowest classification error *credit*

Greedy decision tree learning algorithm

169

- Step 1: Start with an empty tree
- Step 2: Select a feature to split data
- For each split of the tree:
 - Step 3: If nothing more to, make predictions
 - Step 4: Otherwise, go to Step 2 & continue (recurse) on this split

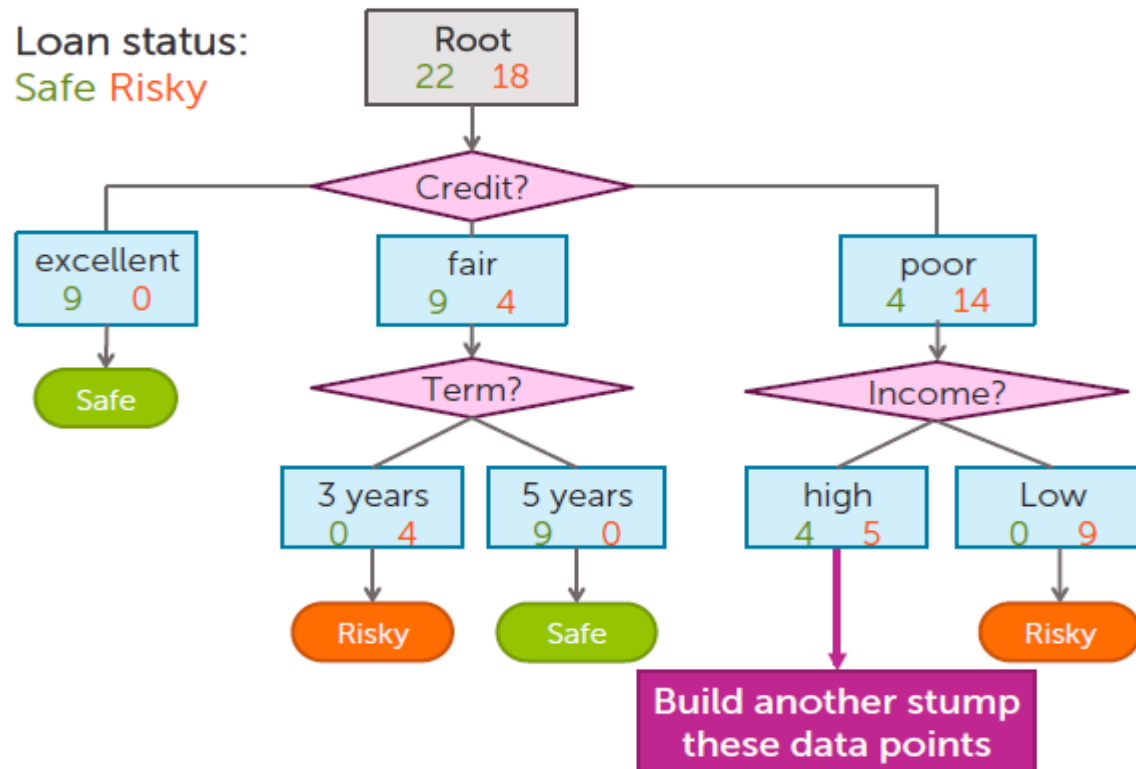


Pick feature split
leading to lowest
classification error

Recursive stump learning

170

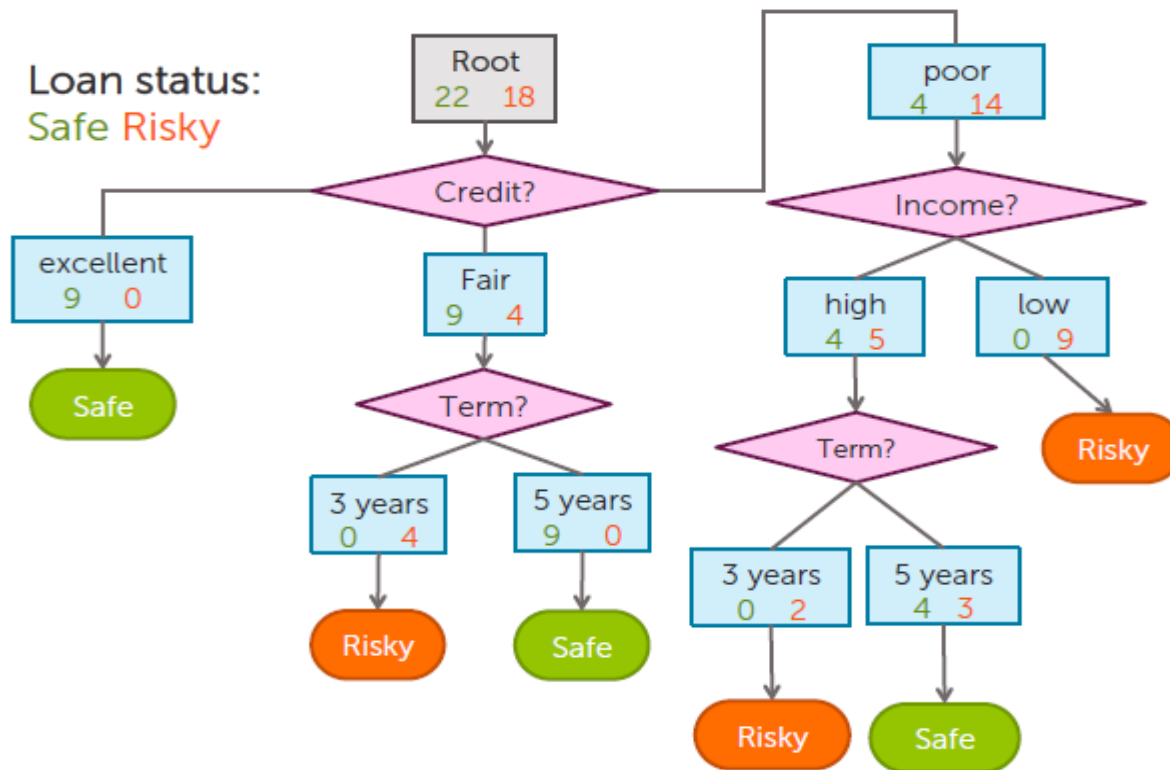
Second level



Recursive stump learning

171

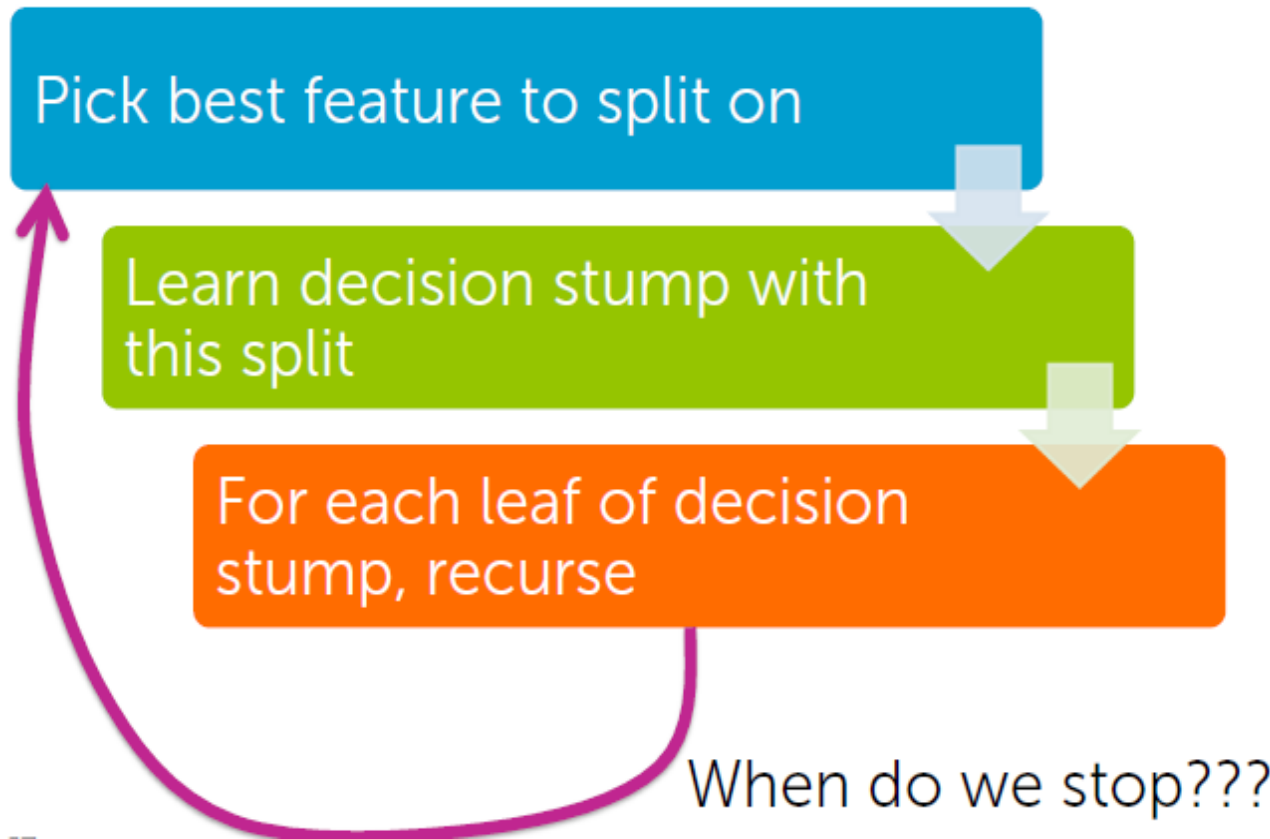
Final decision tree



Simple greedy decision tree learning

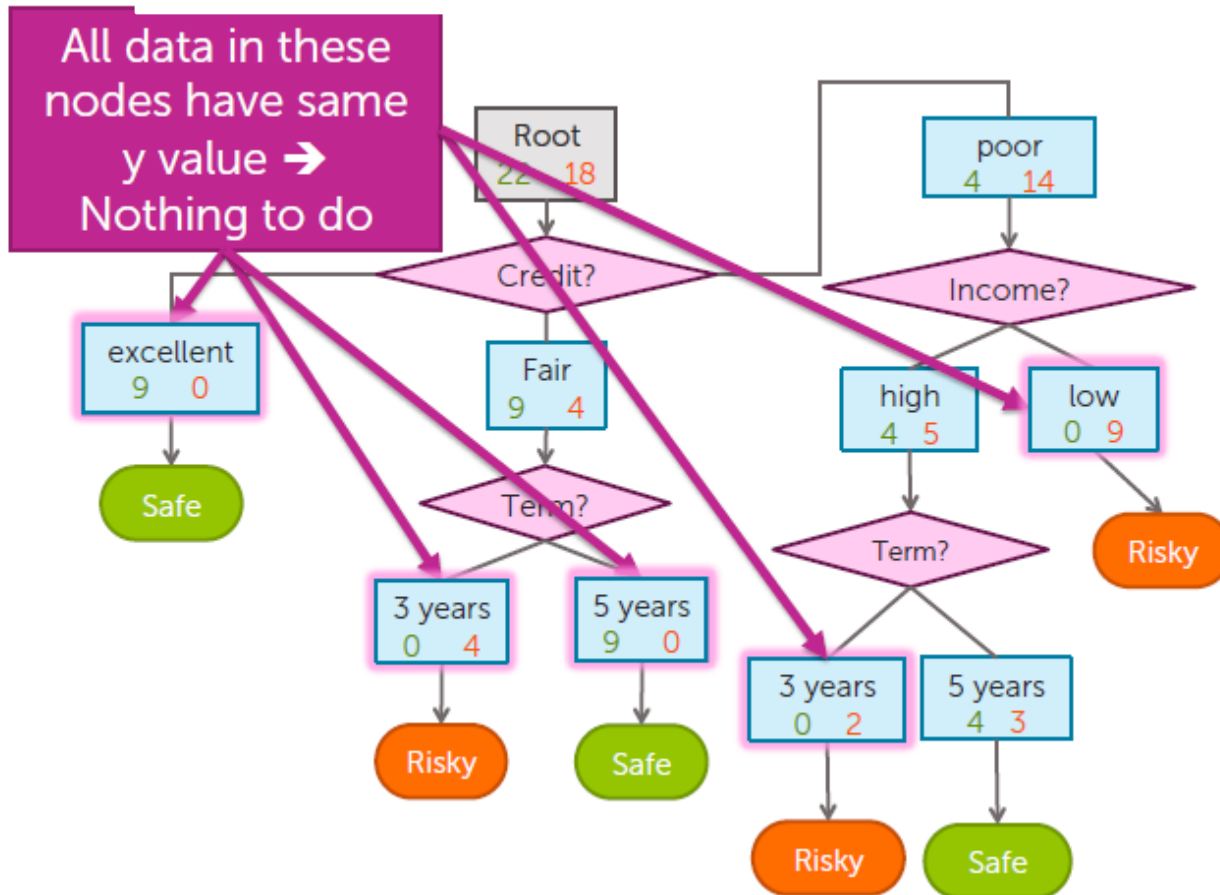
172

Recursive algorithm



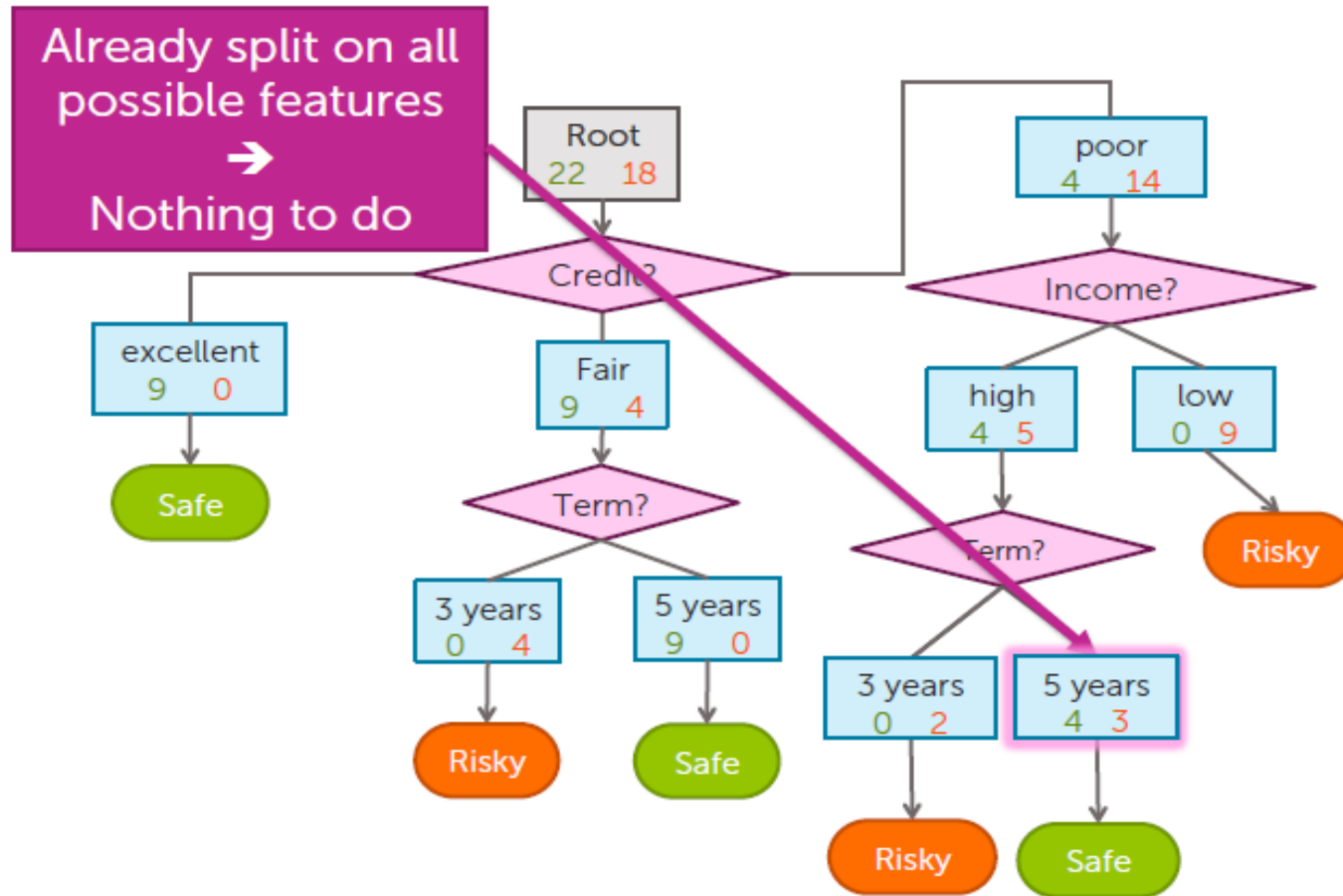
Stopping condition 1

173



Stopping condition 2

174



Greedy decision tree algorithm

175

- **Step 1:** Start with an empty tree

- **Step 2:** Select a feature to split data

- For each split of the tree:

- **Step 3:** If nothing more to, make predictions

- **Step 4:** Otherwise, go to **Step 2** & continue (recurse) on this split

Pick feature split leading to lowest classification error

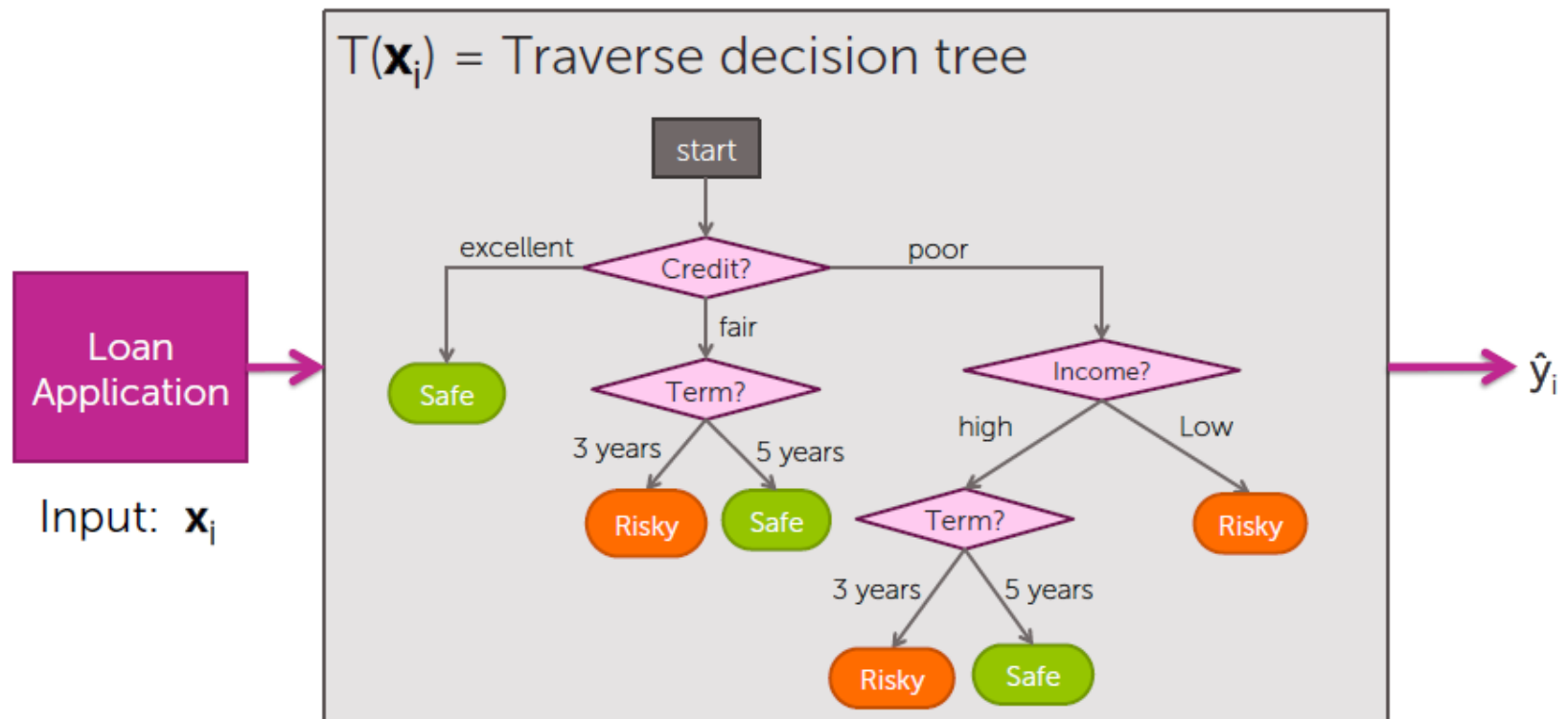
Stopping conditions 1 & 2

Recursion

Predictions with decision trees

176

Decision tree model

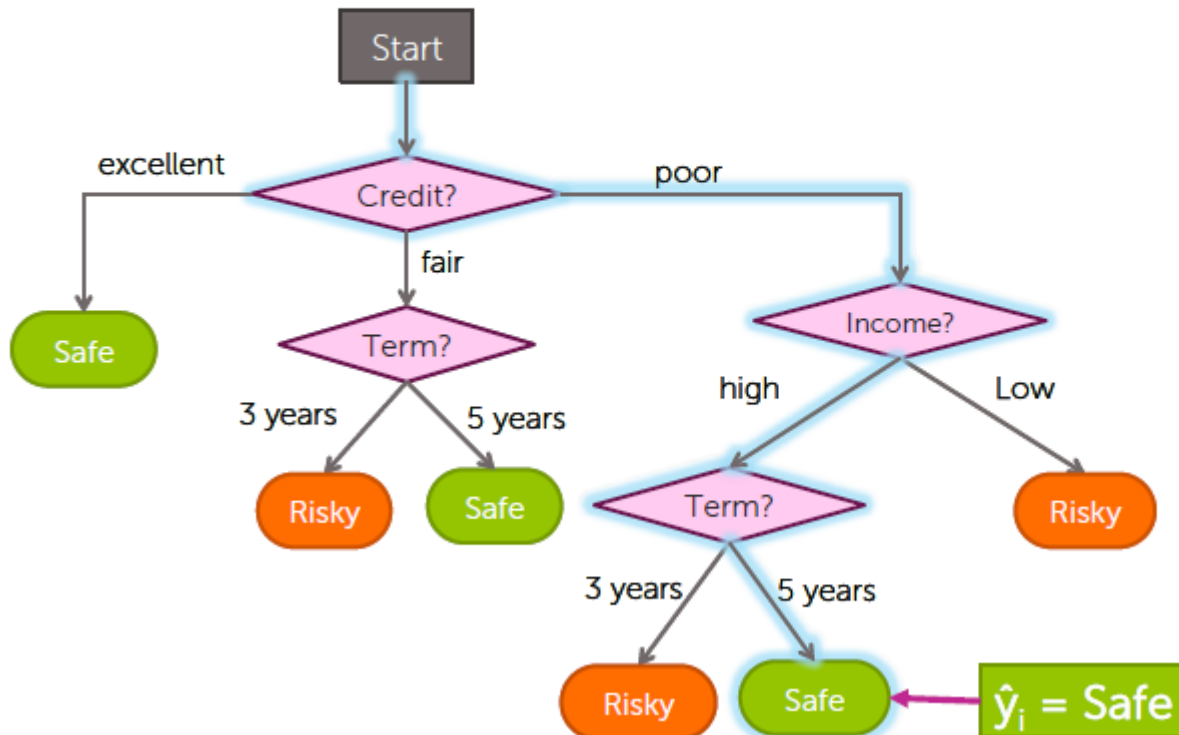


Predictions with decision trees

177

Traversing a decision tree

$x_i = (\text{Credit} = \text{poor}, \text{Income} = \text{high}, \text{Term} = 5 \text{ years})$



Predictions with decision tree

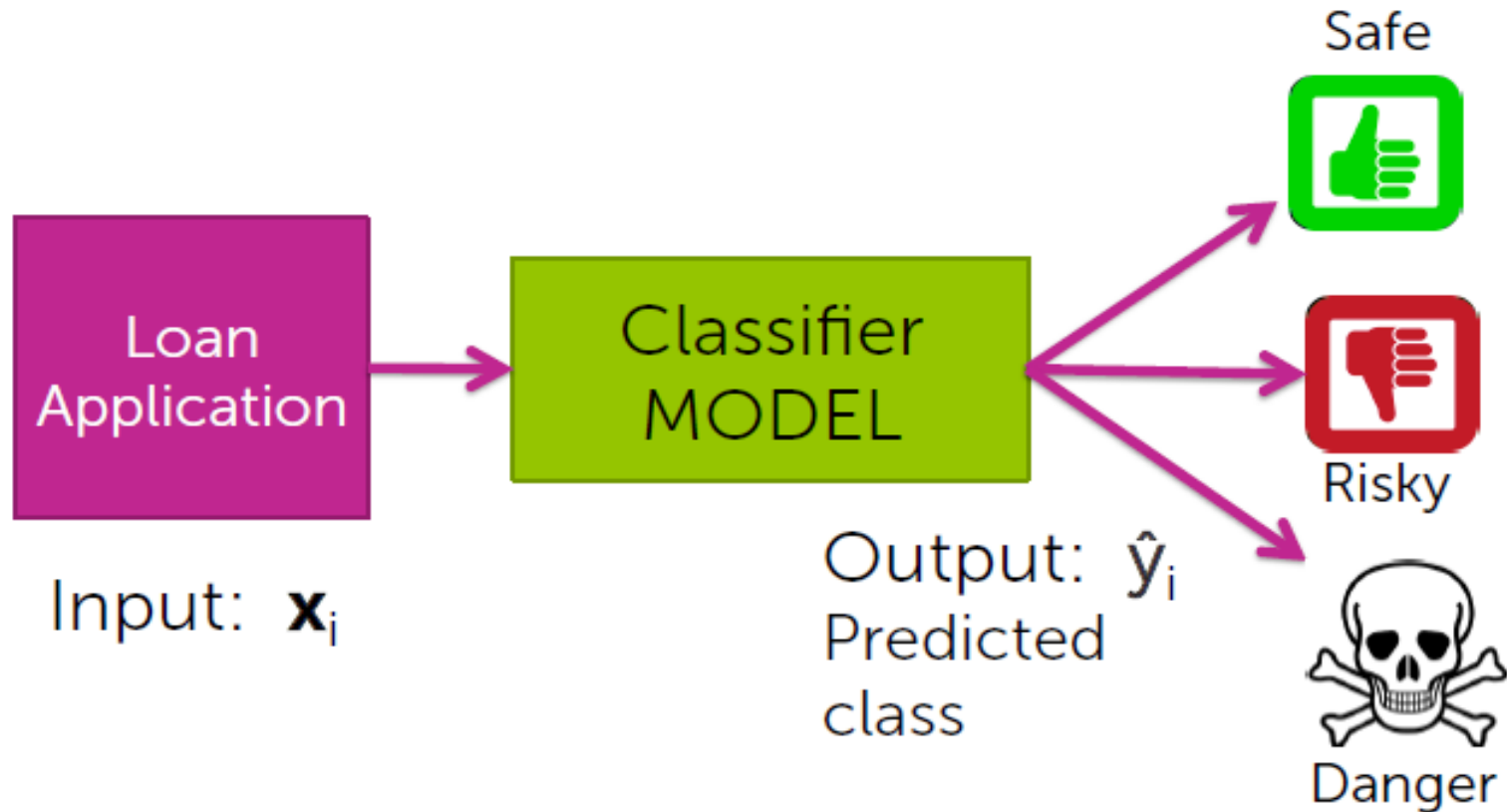
178

`predict(tree_node, input)`

- If current `tree_node` is a leaf:
 - `return` majority class of data points in leaf
- else:
 - `next_note` = child node of `tree_node` whose feature value agrees with input
 - `return predict(next_note, input)`

Multiclass prediction

179



Multiclass decision stump

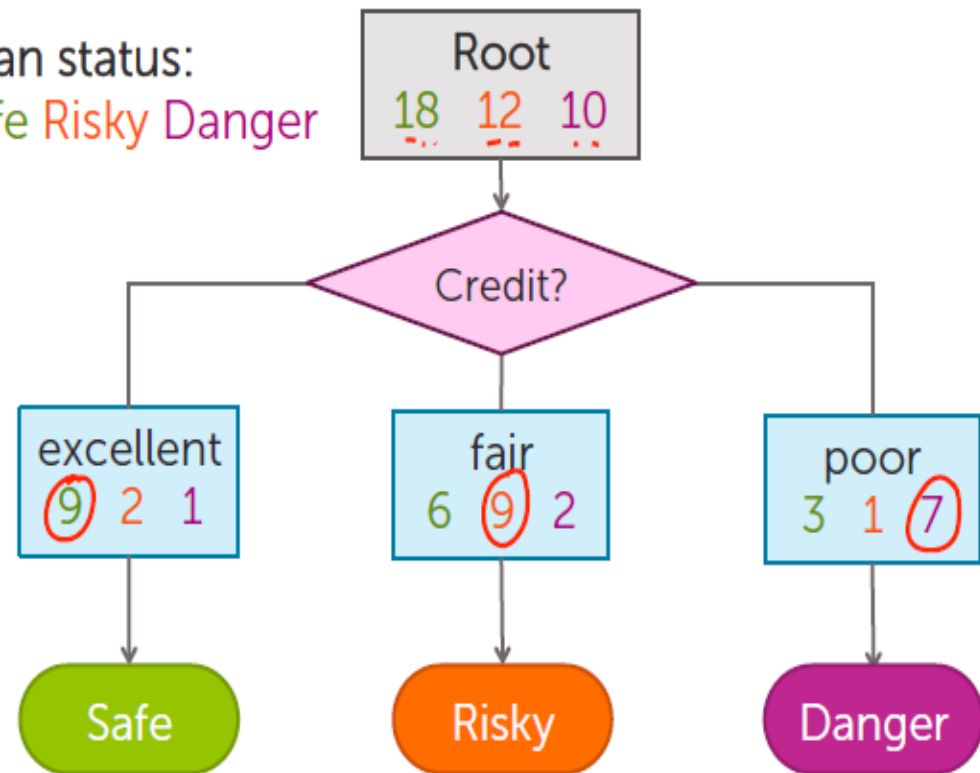
180

$N = 40$,
1 feature,
3 classes

Credit	y
excellent	<u>safe</u>
fair	<u>risky</u>
fair	safe
poor	<u>danger</u>
excellent	risky
fair	safe
poor	danger
poor	safe
fair	safe
...	...

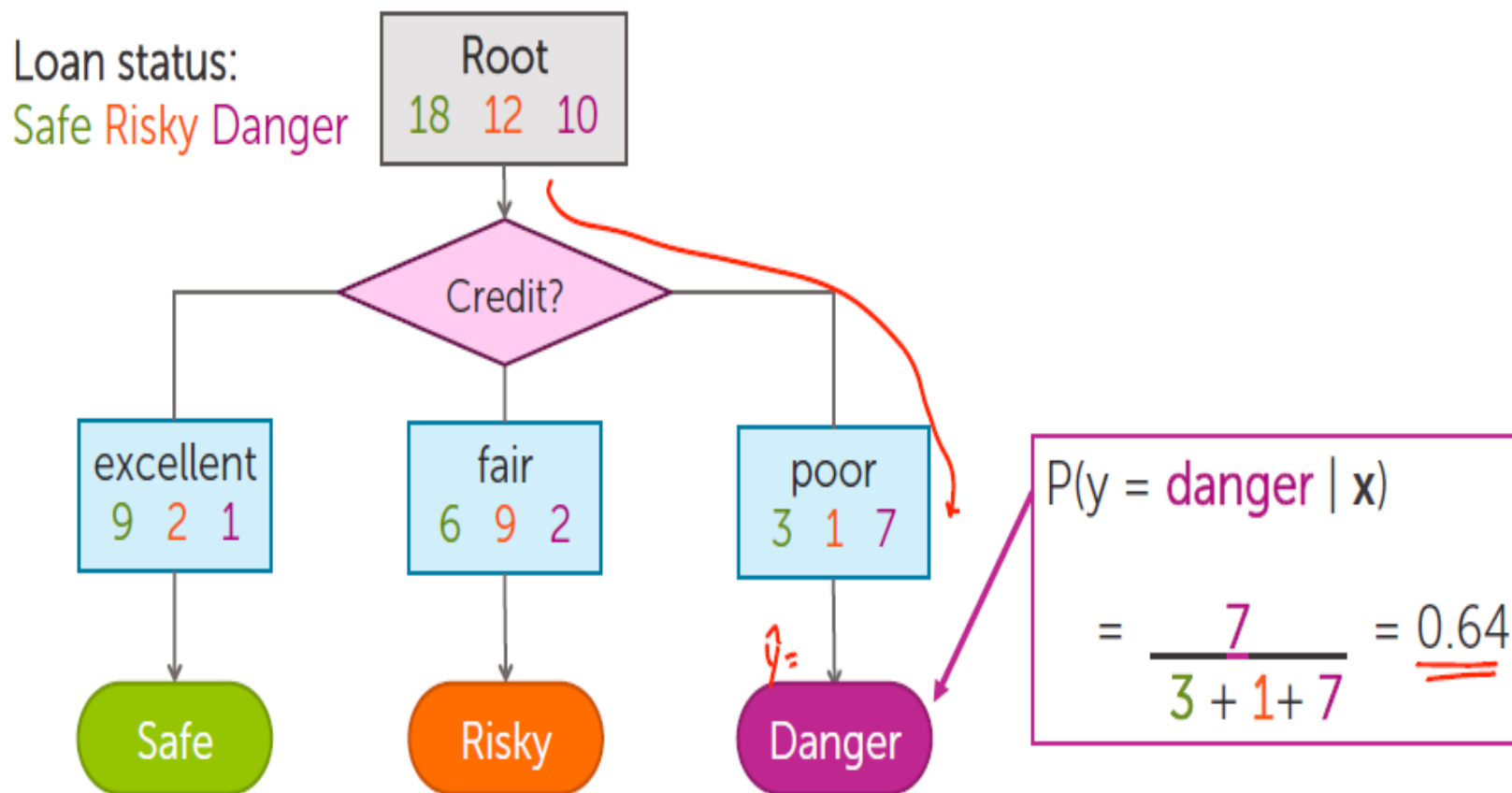


Loan status:
Safe Risky Danger



Predicting probabilities with decision trees

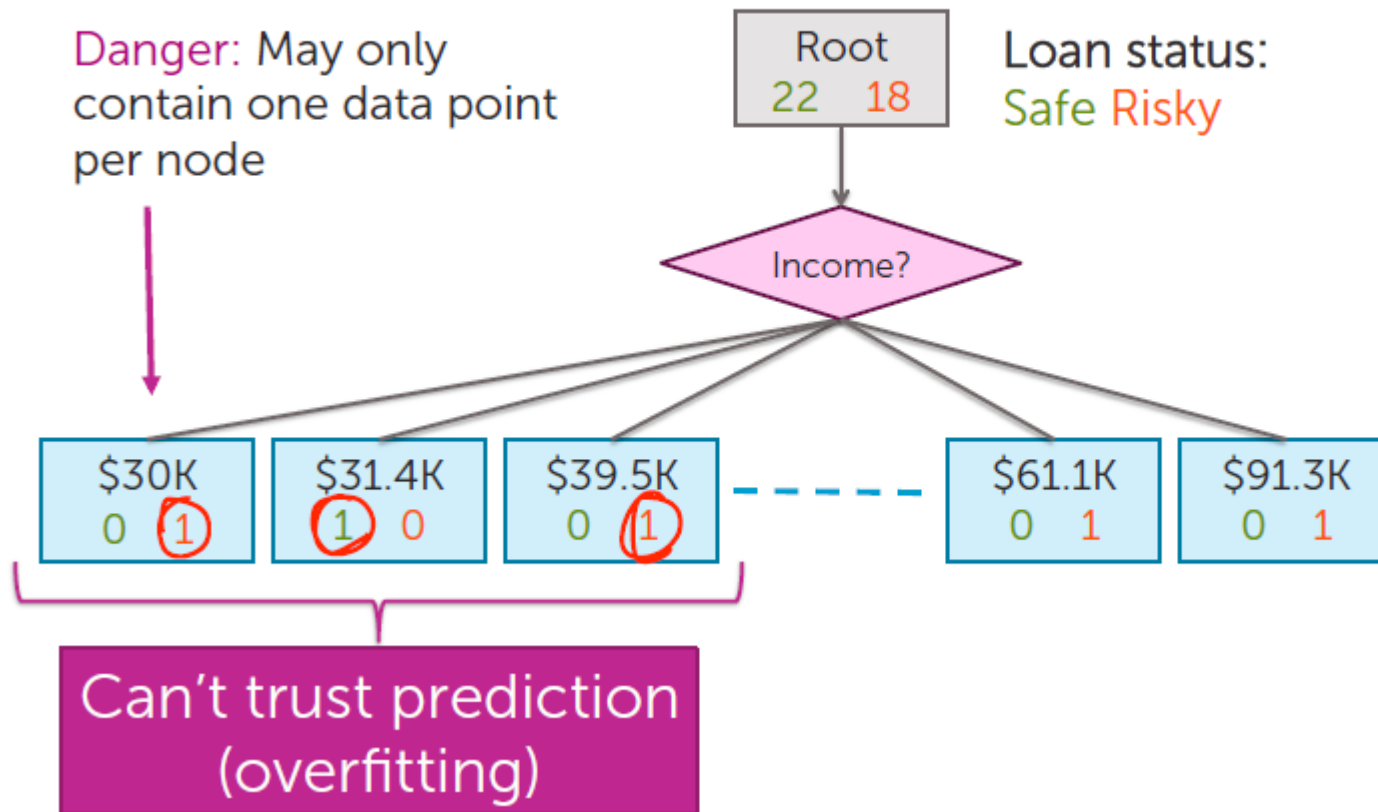
181



How to use real values inputs

182

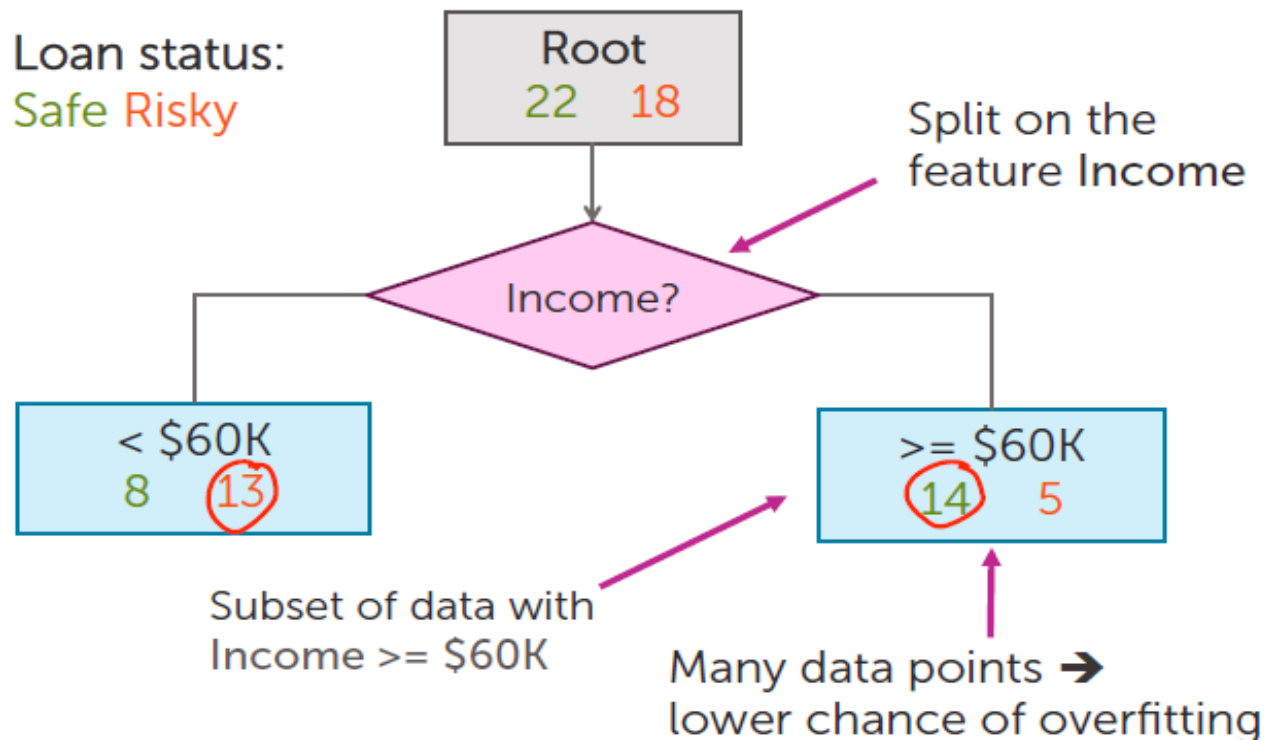
Split on each numeric value?



How to use real values inputs

183

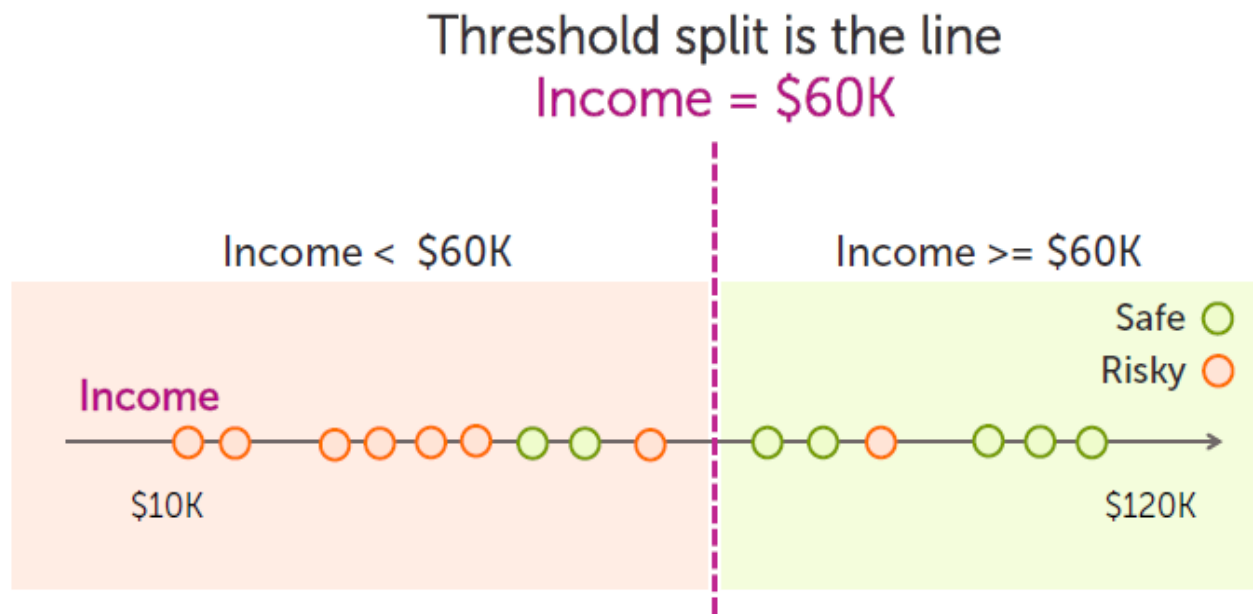
Alternative: Threshold split



Visualizing the threshold split

184

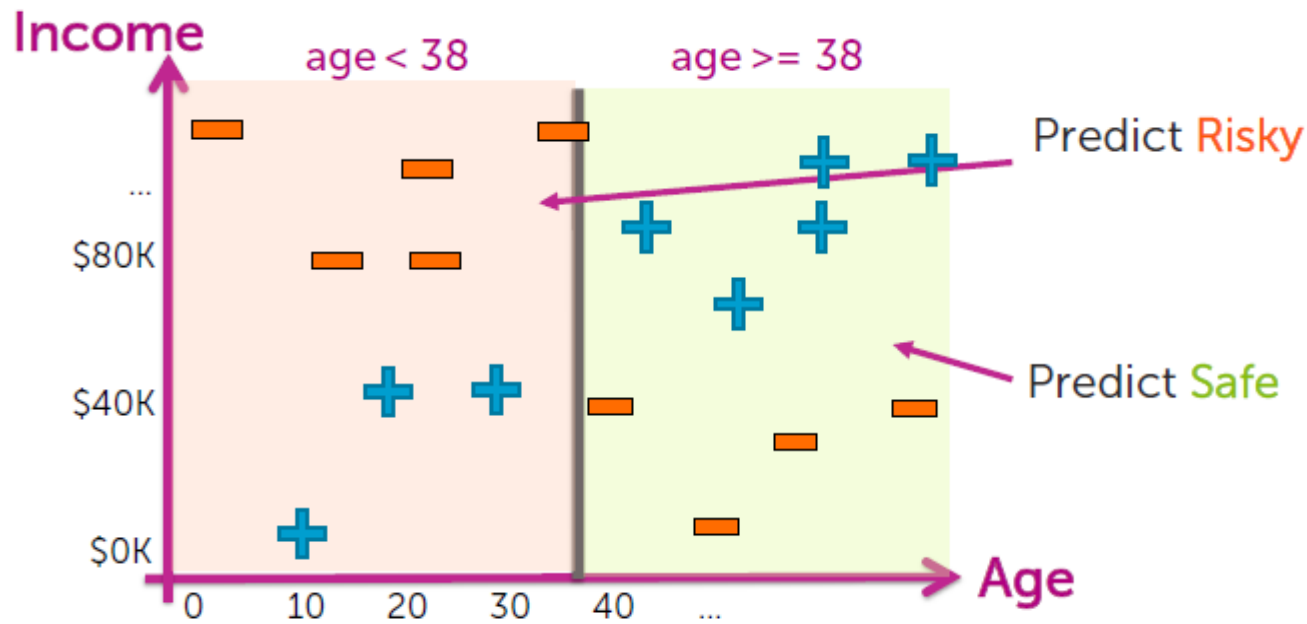
Threshold splits in 1-D



Visualizing the threshold split

185

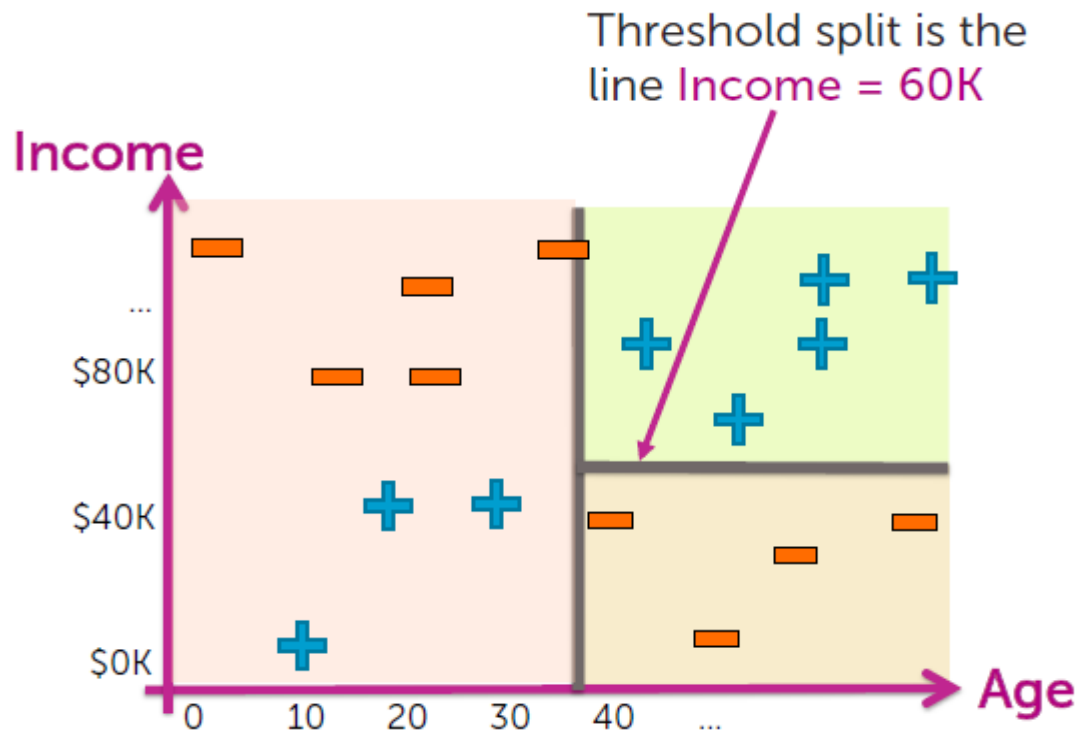
Split on Age ≥ 38



Visualizing the threshold split

186

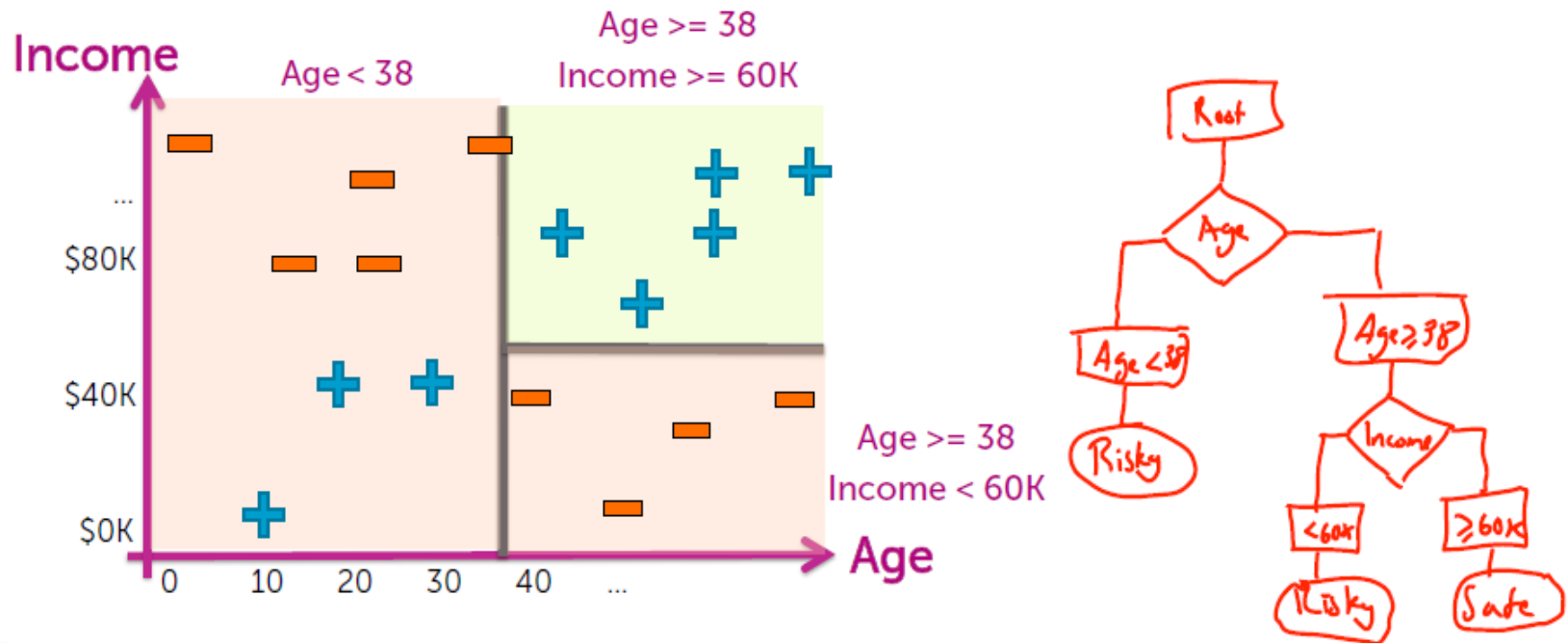
Depth 2: Split on $\text{Income} \geq \$60\text{K}$



Visualizing the threshold split

187

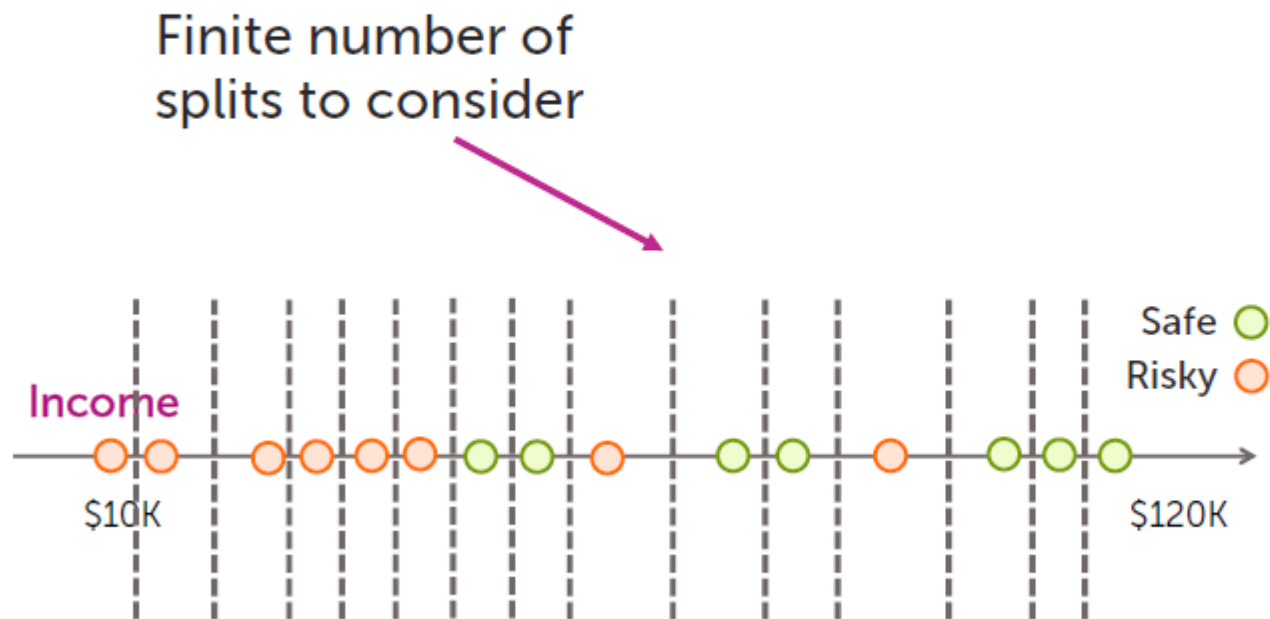
Each split partitions the 2-D space



Finding the best threshold split

188

Only need to consider mid-points



Finding the best threshold split

189

Threshold split selection algorithm

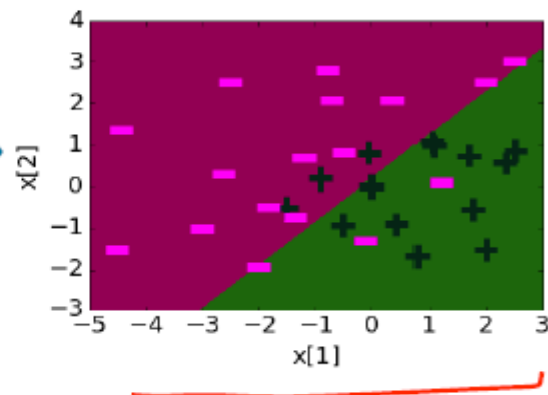
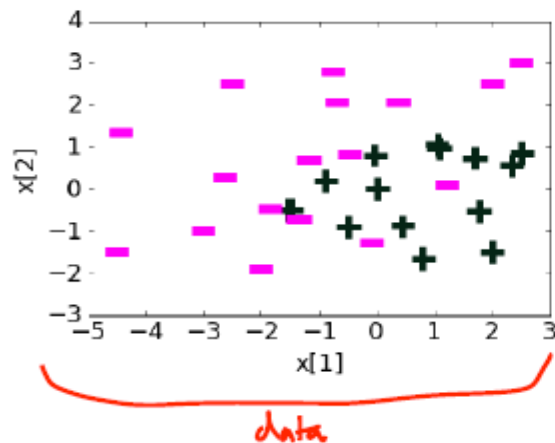
- **Step 1:** Sort the values of a feature $h_j(\mathbf{x})$:
Let $\{v_1, v_2, v_3, \dots, v_N\}$ denote sorted values
- **Step 2:**
 - For $i = 1 \dots N-1$
 - Consider split $t_i = (v_i + v_{i+1}) / 2$
 - Compute classification error for threshold split $h_j(\mathbf{x}) \geq t_i$
 - Chose the t^* with the lowest classification error

Decision trees vs logistic regression

190

Logistic regression

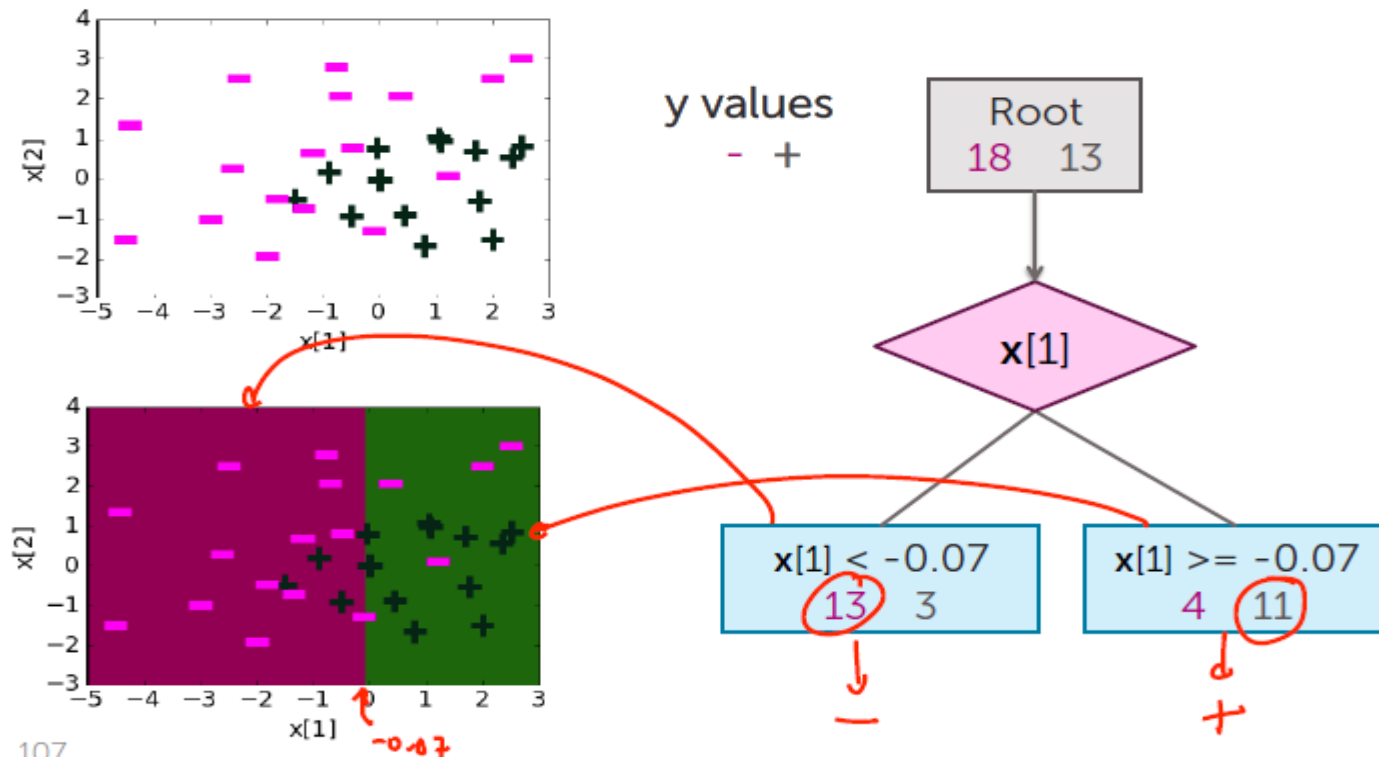
Feature	Value	Weight Learned
$h_0(x)$	1	0.22
$h_1(x)$	$x[1]$	1.12
$h_2(x)$	$x[2]$	-1.07



Decision trees vs logistic regression

191

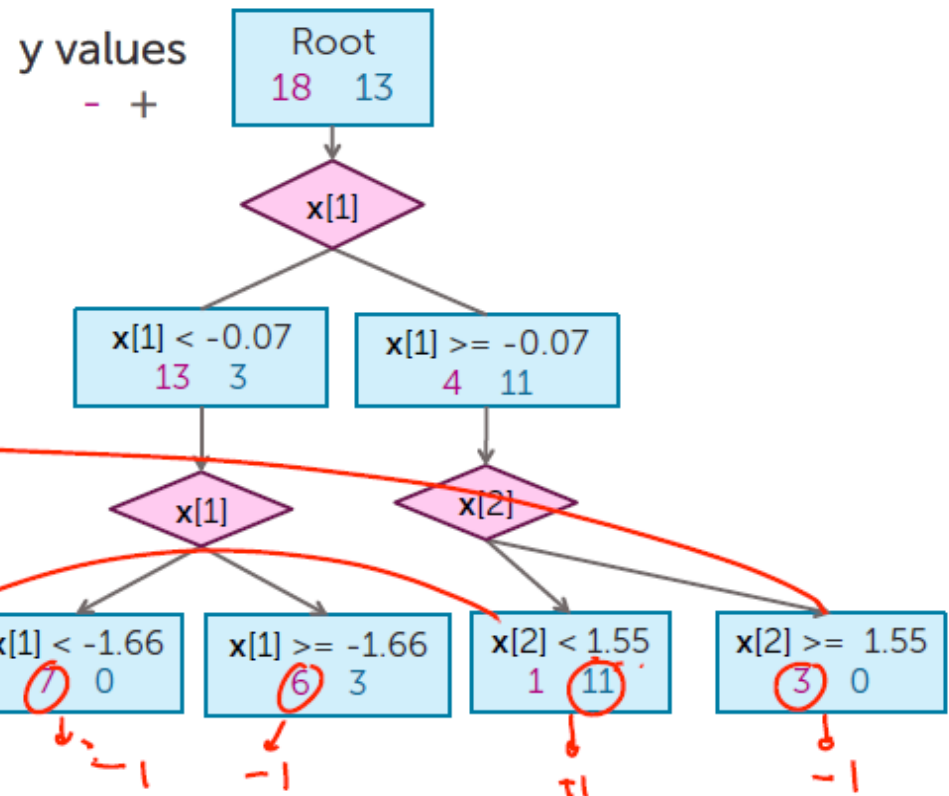
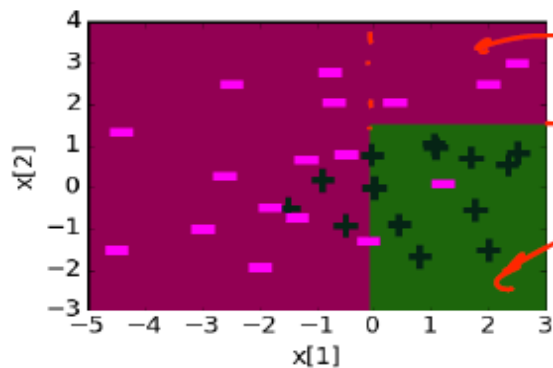
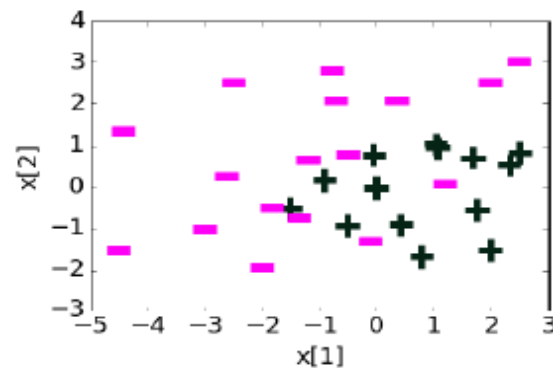
Depth 1: Split on $x[1]$



Decision trees vs logistic regression

192

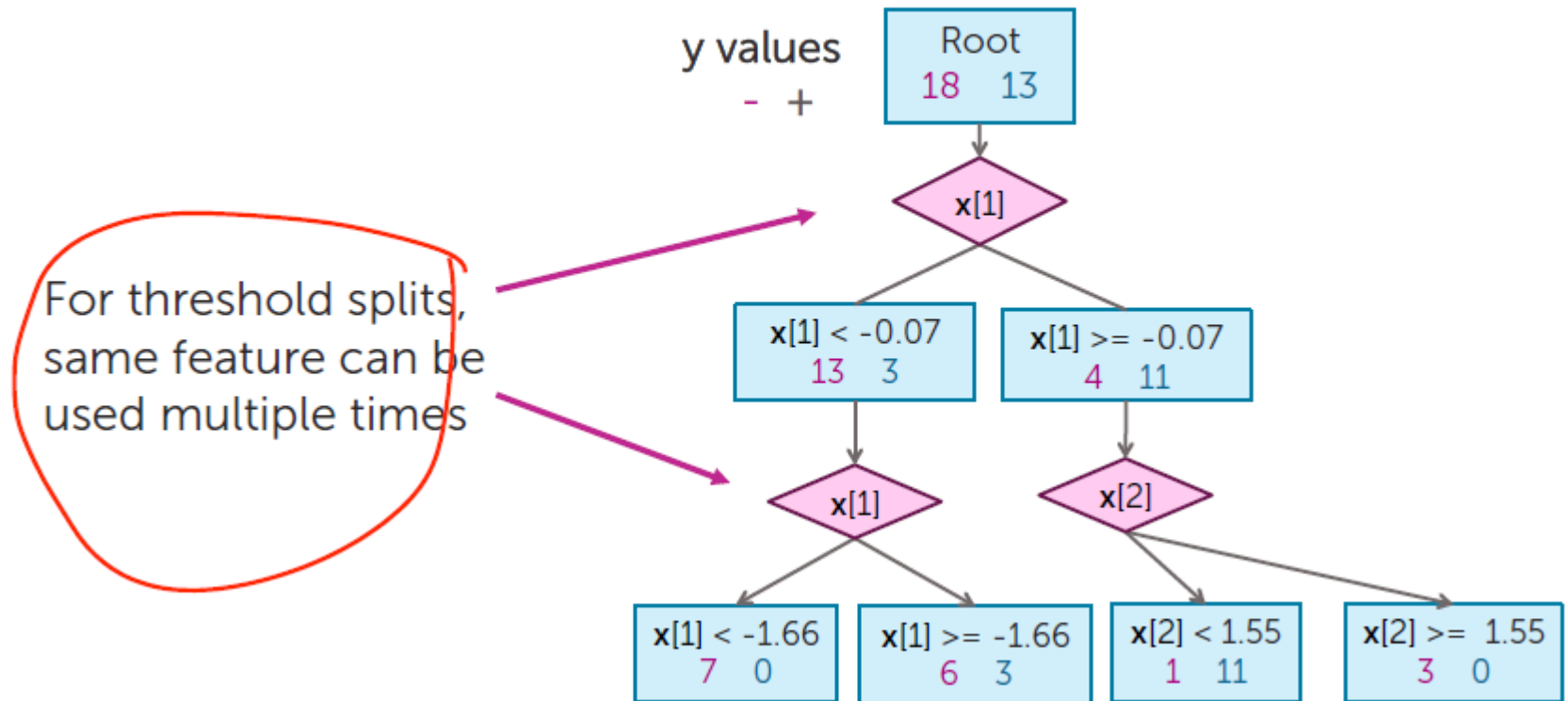
Depth 2



Decision tree vs logistic regression

193

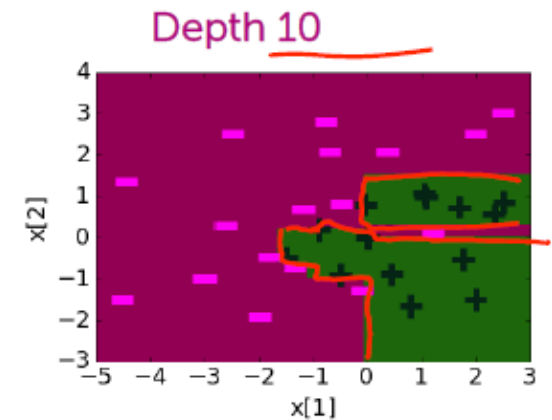
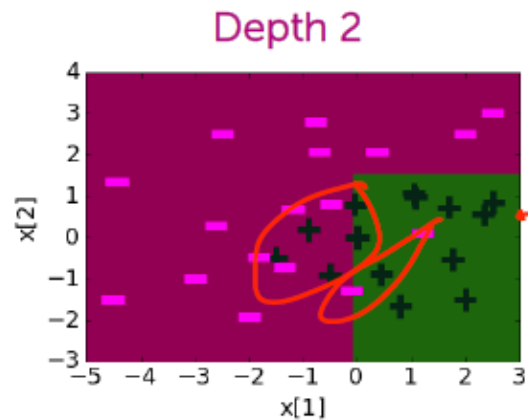
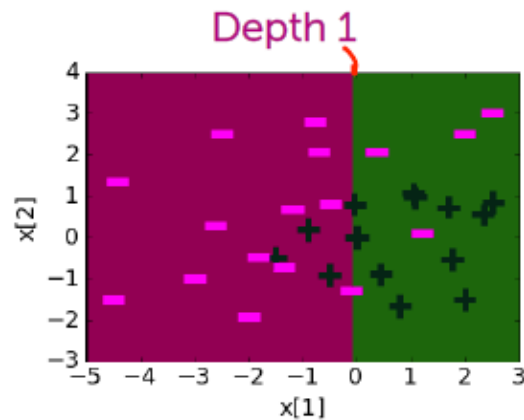
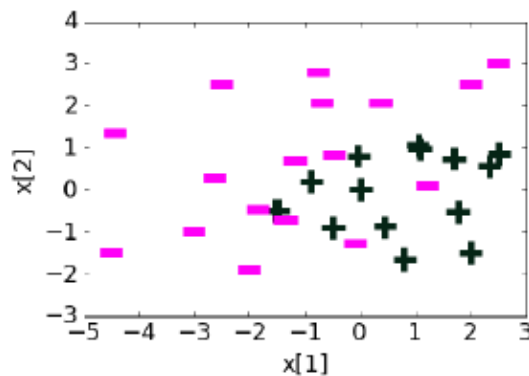
Threshold split caveat



Decision tree vs logistic regression

194

Decision boundaries



195

What you can do now

196

- Define a decision tree classifier
- Interpret the output of a decision trees
- Learn a decision tree classifier using greedy algorithm
- Traverse a decision tree to make predictions
 - Majority class predictions
 - Probability predictions
 - Multiclass classification

Overfitting in decision trees

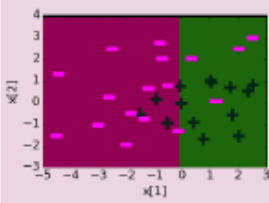
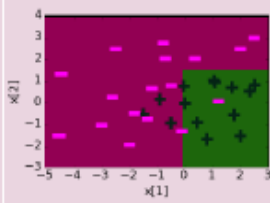
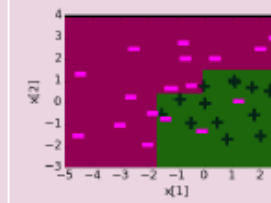
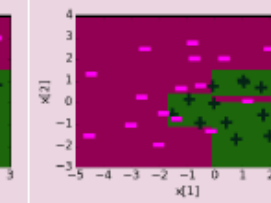
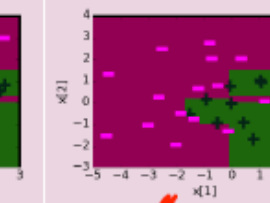
Overfitting in decision tree

198

What happens when we increase depth?

Training error reduces with depth

Big warning!!

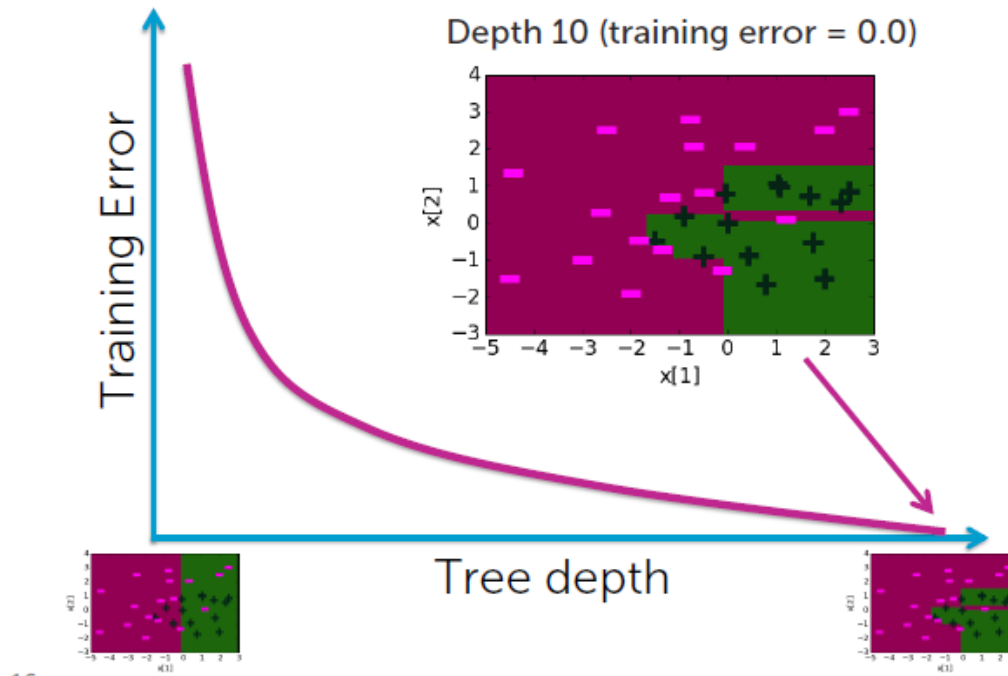
Tree depth	depth = 1	depth = 2	depth = 3	depth = 5	depth = 10
<u>Training error</u>	<u>0.22</u>	<u>0.13</u>	<u>0.10</u>	0.03	<u>0.00</u>
Decision boundary					

complexity of decision boundary →

Overfitting in decision tree

199

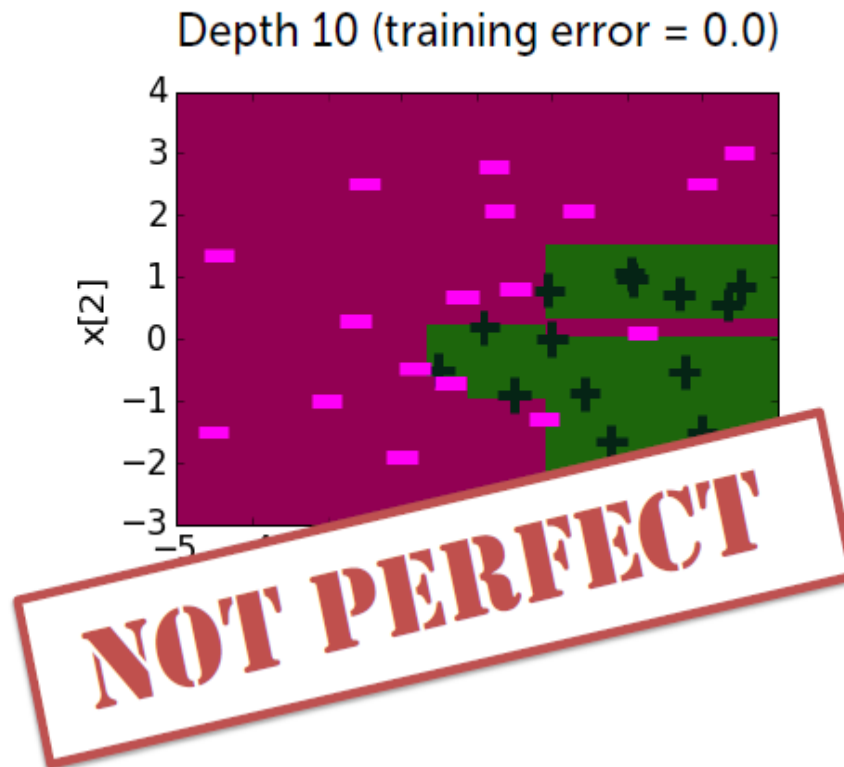
Deeper trees → lower training error



Overfitting in decision tree

200

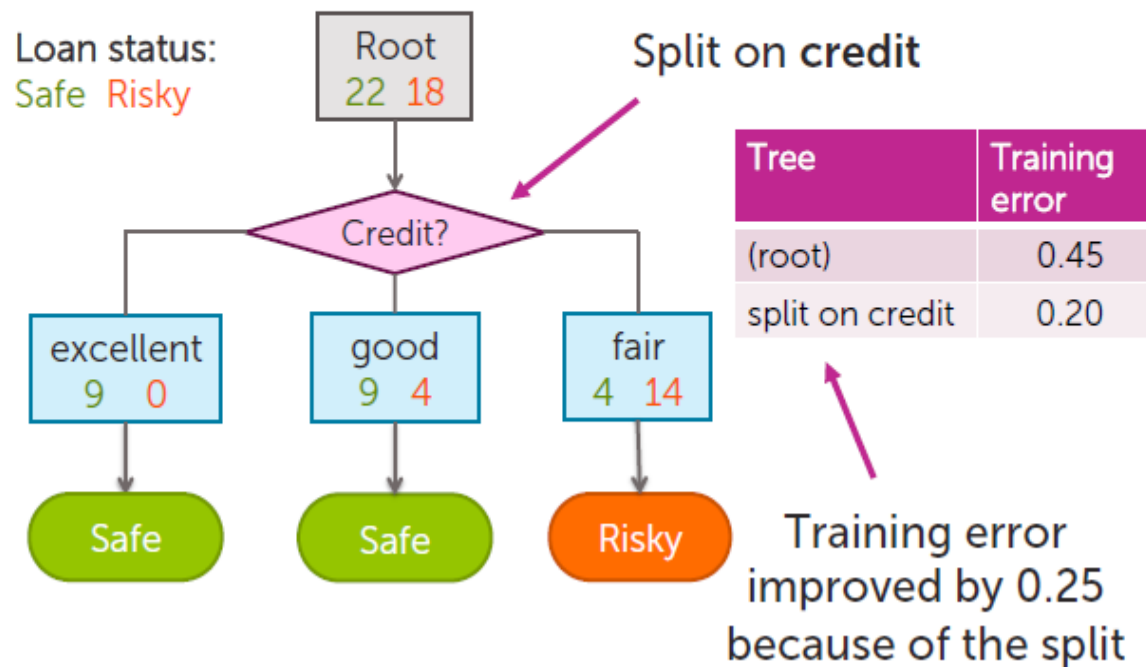
Training error = 0: Is this model perfect?



Overfitting in decision tree

201

Why training error reduces with depth?




Overfitting in decision tree

202

Feature split selection algorithm

- Given a subset of data M (a node in a tree)
- For each feature $h_i(x)$:
 1. Split data of M according to feature $h_i(x)$
 2. Compute classification error split
- Chose feature $h^*(x)$ with lowest classification error

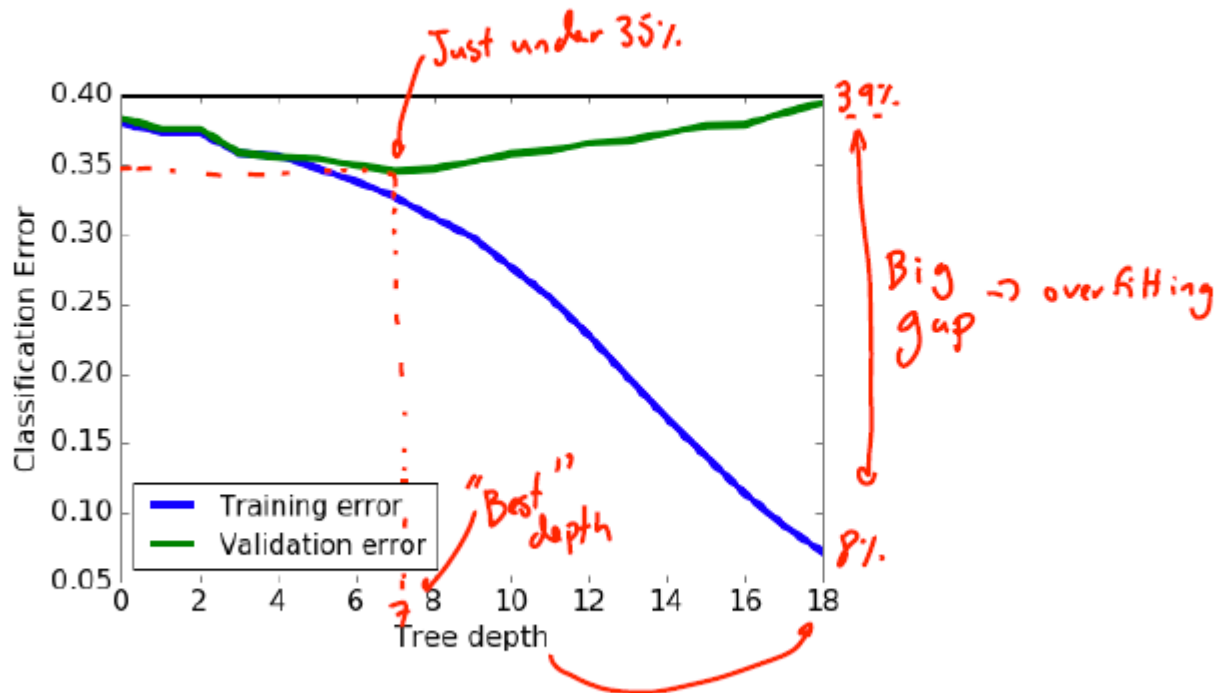


By design, each split
reduces training error

Overfitting in decision tree

203

Decision trees overfitting on loan data



Simplest tree is better

204

Principle of *Occam's Razor*



*"Among competing hypotheses, the one with fewest assumptions should be selected",
William of Occam, 13th Century*

Symptoms: S_1 and S_2

Diagnosis 1: 2 diseases

Two diseases D_1 and D_2 where
 D_1 explains S_1 , D_2 explains S_2

OR

SIMPLER

Diagnosis 2: 1 disease

Disease D_3 explains both
symptoms S_1 and S_2

Simplest tree is better

205

Occam's Razor for decision trees

When two trees have similar classification error on the validation set, pick the simpler one

Complexity	Train error	Validation error
Simple	0.23	0.24
Moderate	0.12	0.15
Complex	0.07	0.15
Super complex	0	0.18

Same validation error

pick

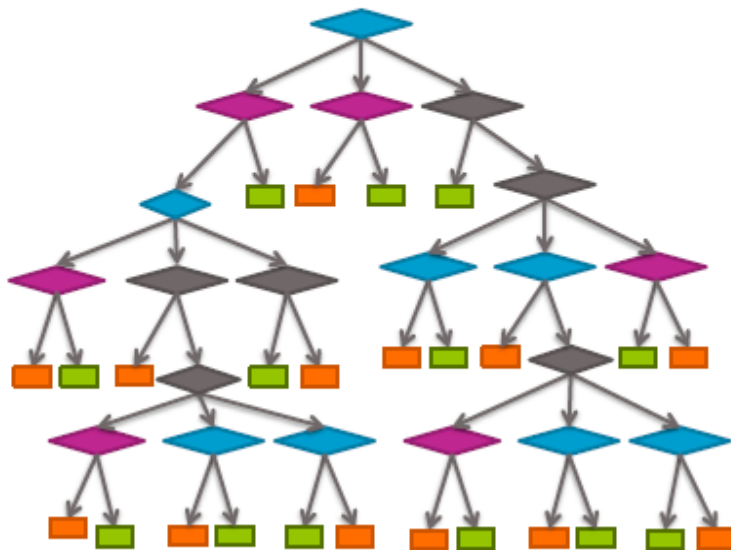
bad!

Overfit

Simplest tree is better

206

Which tree is simpler?



OR



SIMPLER

Simplest tree is better

207

How do we pick simpler trees?

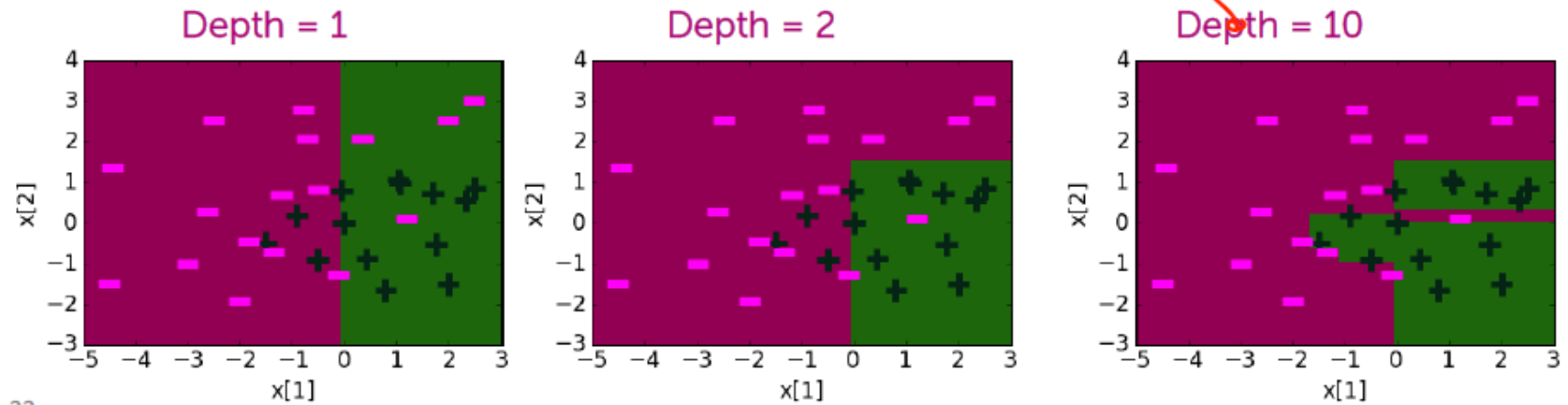
1. **Early Stopping:** Stop learning algorithm **before** tree become too complex
2. **Pruning:** Simplify tree **after** learning algorithm terminates

Early stopping for learning decision trees

208

Deeper trees →
Increasing complexity

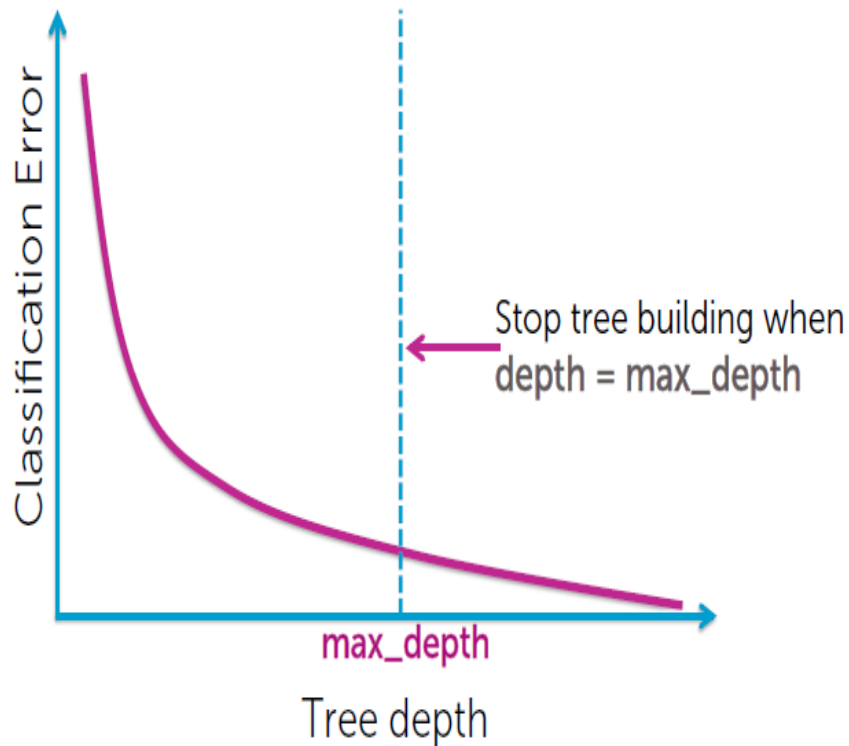
Model complexity increases with depth



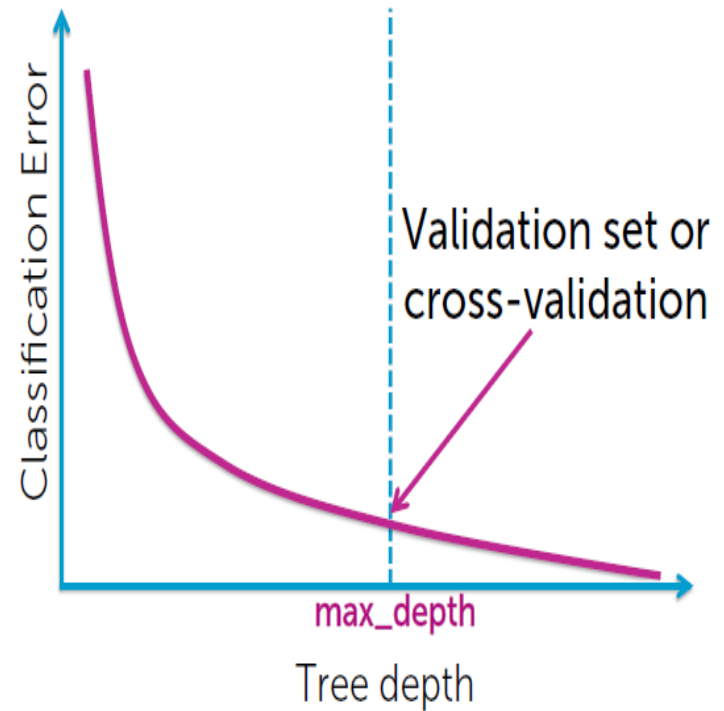
Early stopping condition 1

209

Limit depth of tree



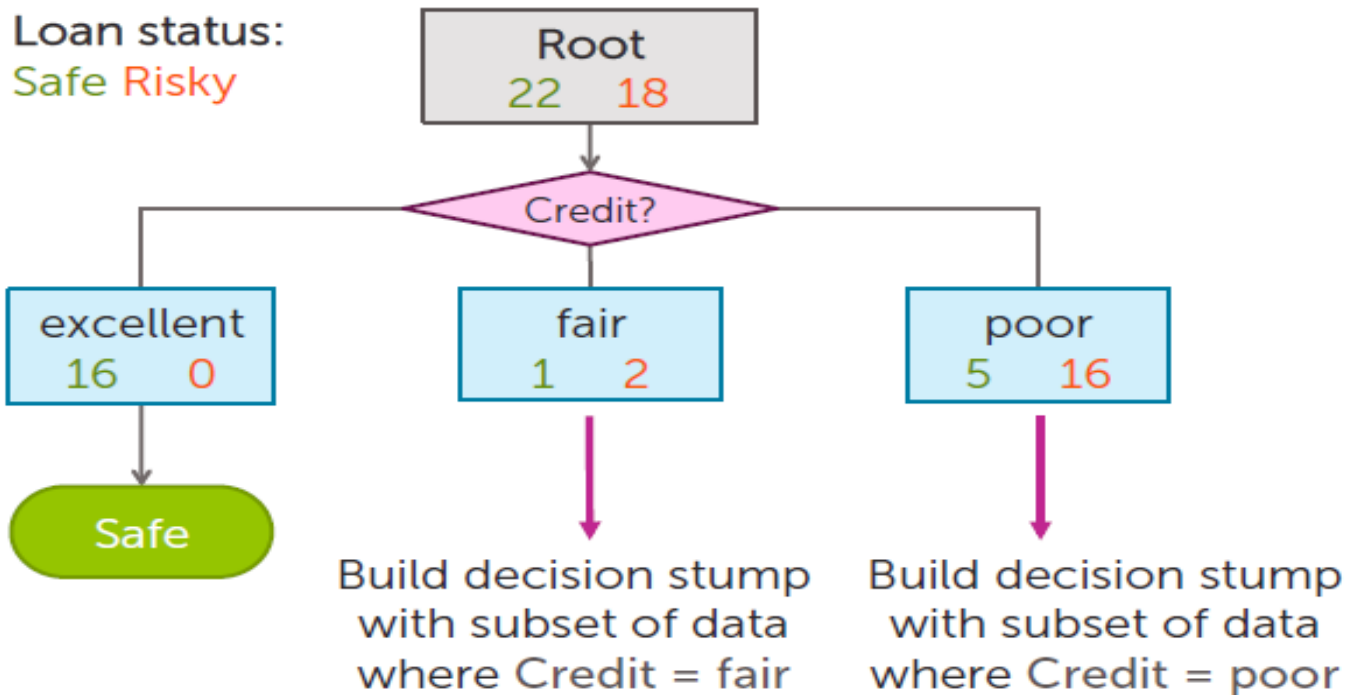
Picking value for **max_depth**???



Early stopping condition 2

210

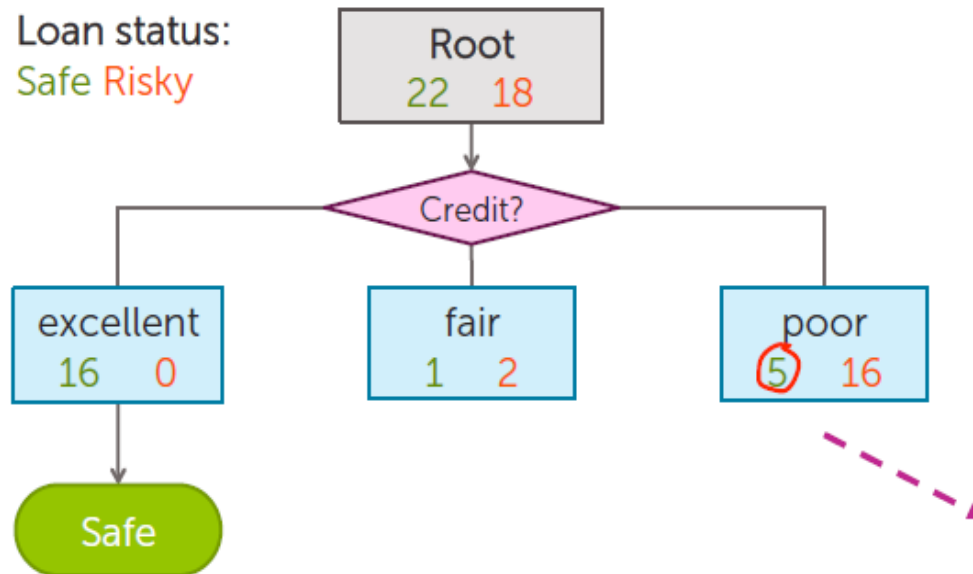
Decision tree recursion review



Early stopping condition 2

211

Split selection for credit=poor



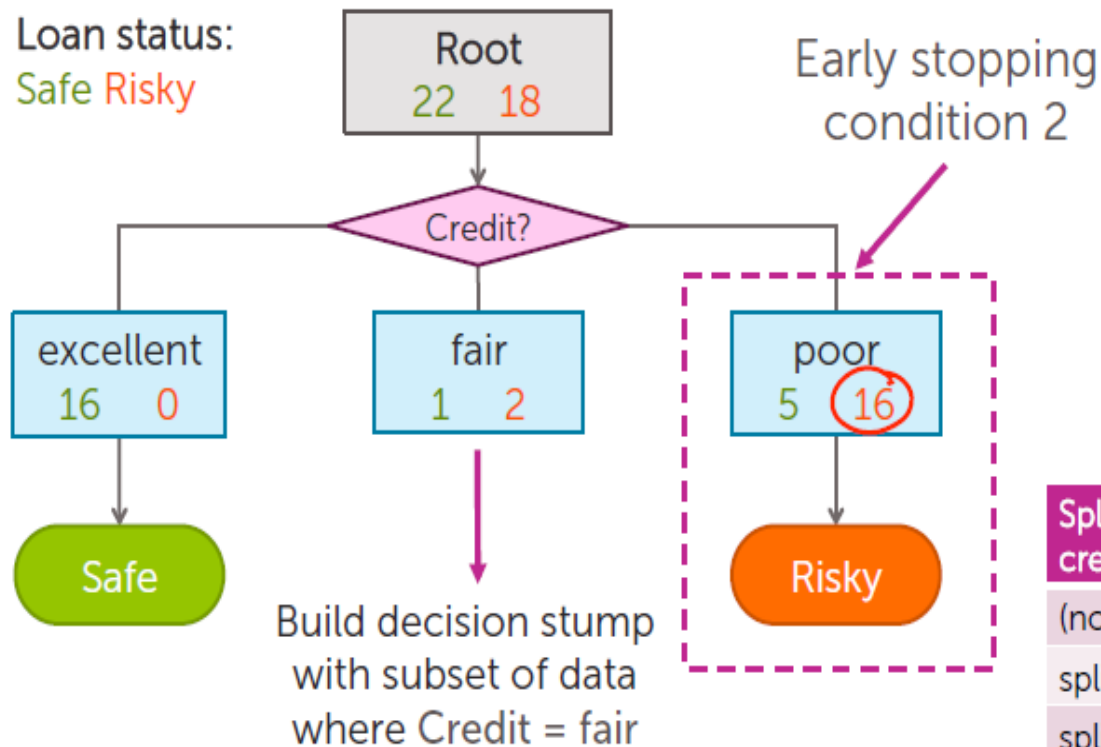
No split improves
classification error
→ Stop!

Splits for credit=poor	Classification error
(no split)	<u>0.24</u>
split on term	<u>0.24</u>
split on income	<u>0.24</u>

Early stopping condition 2

212

No split improves classification error



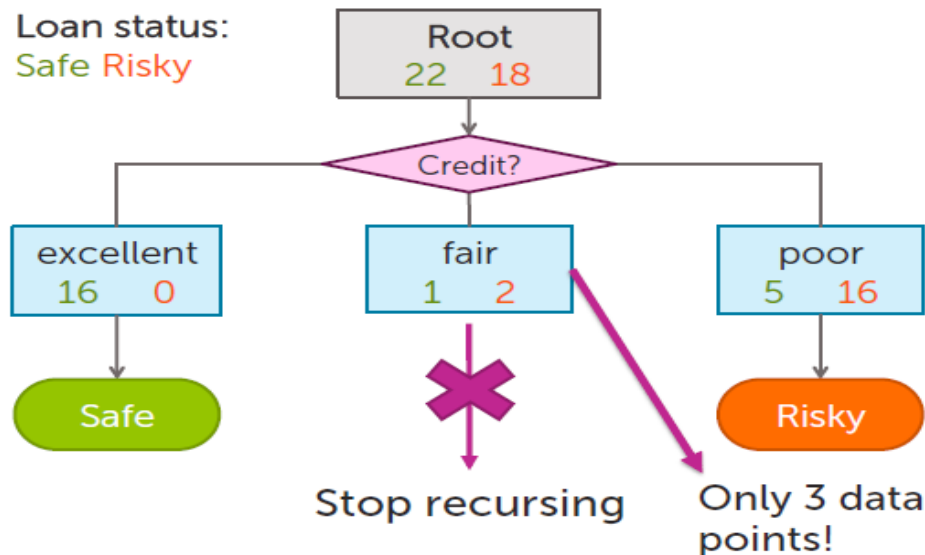
Splits for credit=poor	Classification error
(no split)	0.24
split on term	0.24
split on income	0.24

Early stopping condition 3

213

Stop if number of data points contained in a node is too small

Can we trust nodes with very few points?



Early stopping: Summary

214

1. **Limit tree depth:** Stop splitting after a certain depth
2. **Classification error:** Do not consider any split that does not cause a sufficient decrease in classification error
3. **Minimum node "size":** Do not split an intermediate node which contains too few data points

Greedy decision tree learning

215

- **Step 1:** Start with an empty tree
- **Step 2:** Select a feature to split data
- For each split of the tree:

- **Step 3:** If nothing more to, make predictions *← Majority*

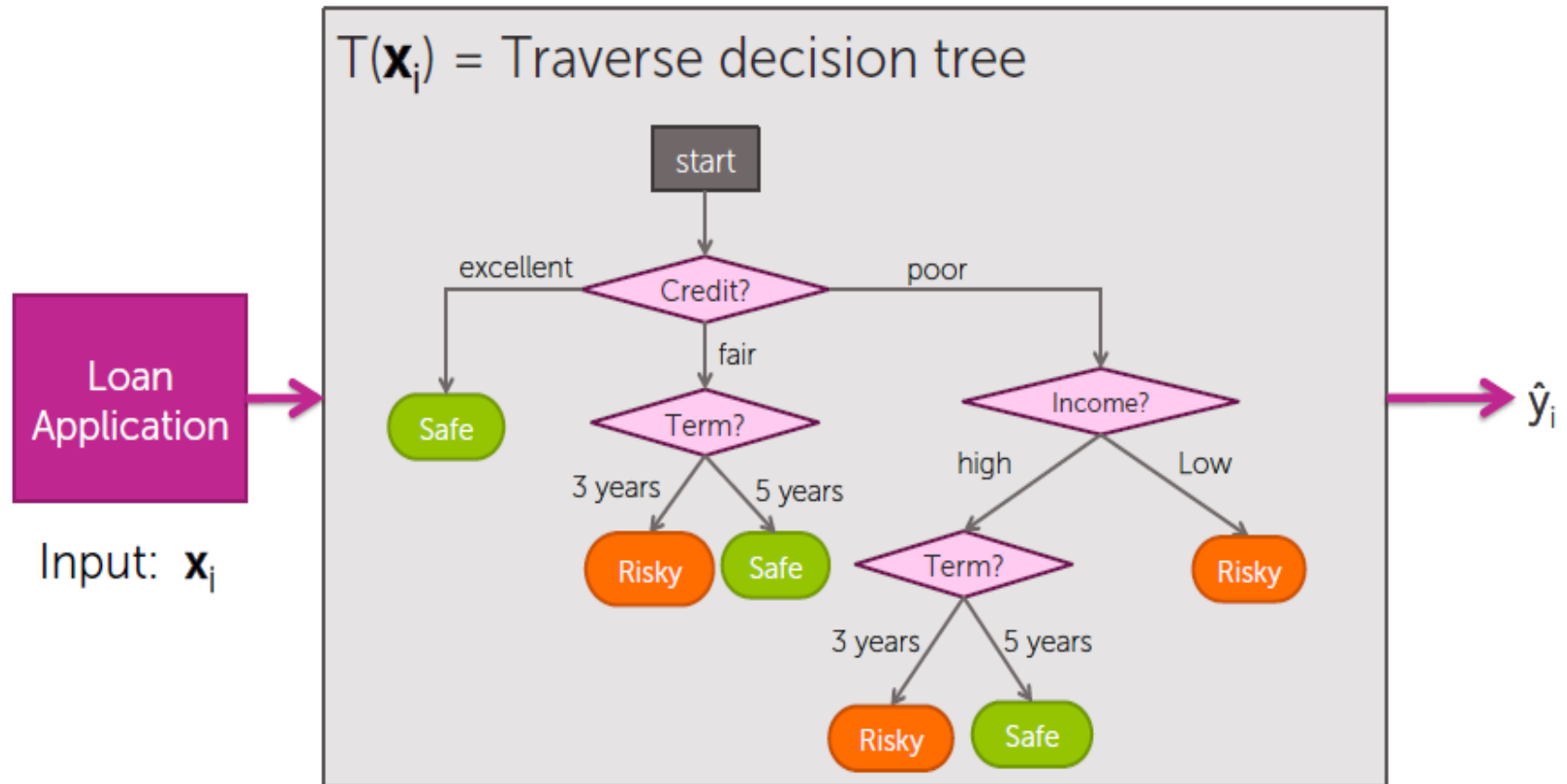
- **Step 4:** Otherwise, go to **Step 2** & continue (recurse) on this split

Stopping conditions 1 & 2
or
Early stopping conditions 1, 2 & 3
Recursion

Strategies for handling missing data

Decision tree review

217



Missing data

218

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	?	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	high	safe
poor	?	high	risky
poor	5 yrs	low	safe
fair	?	high	safe

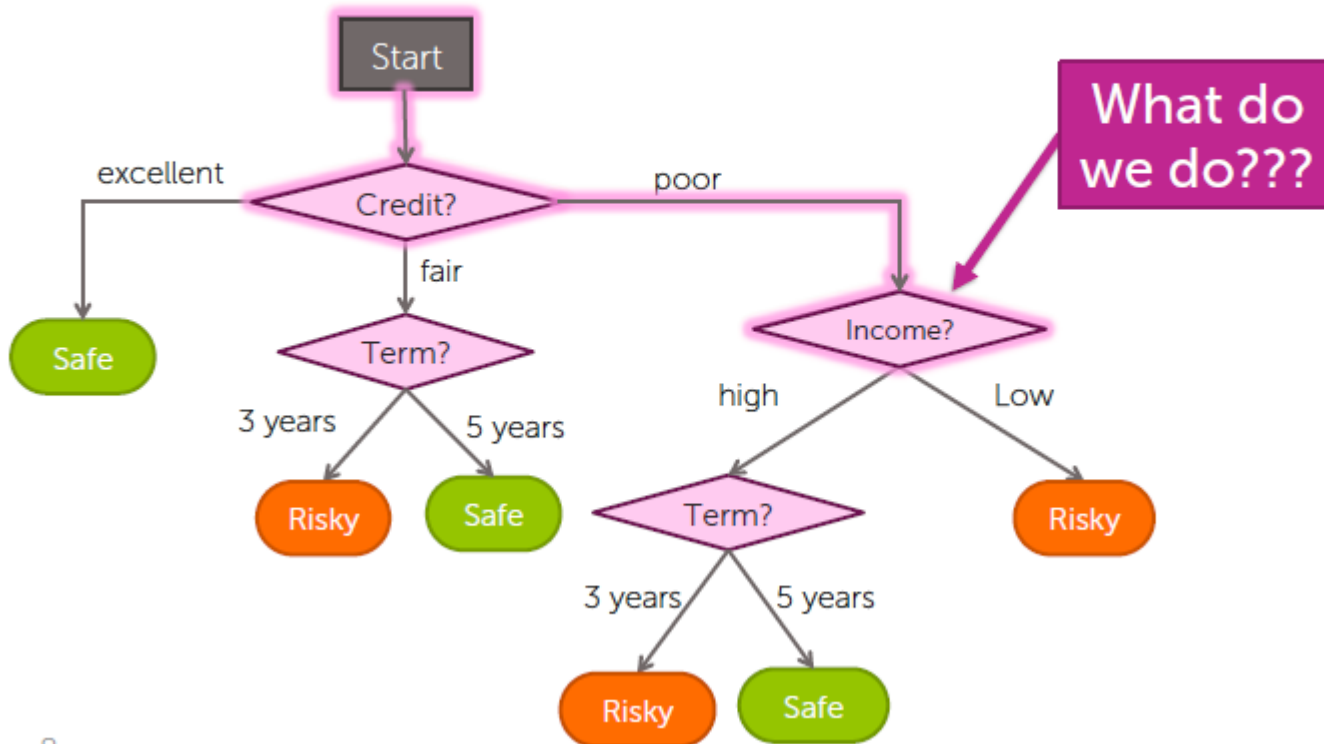
Loan application
may be
3 or 5 years

1. **Training data:** Contains "unknown" values
2. **Predictions:** Input at prediction time contains "unknown" values

Missing values during predictions

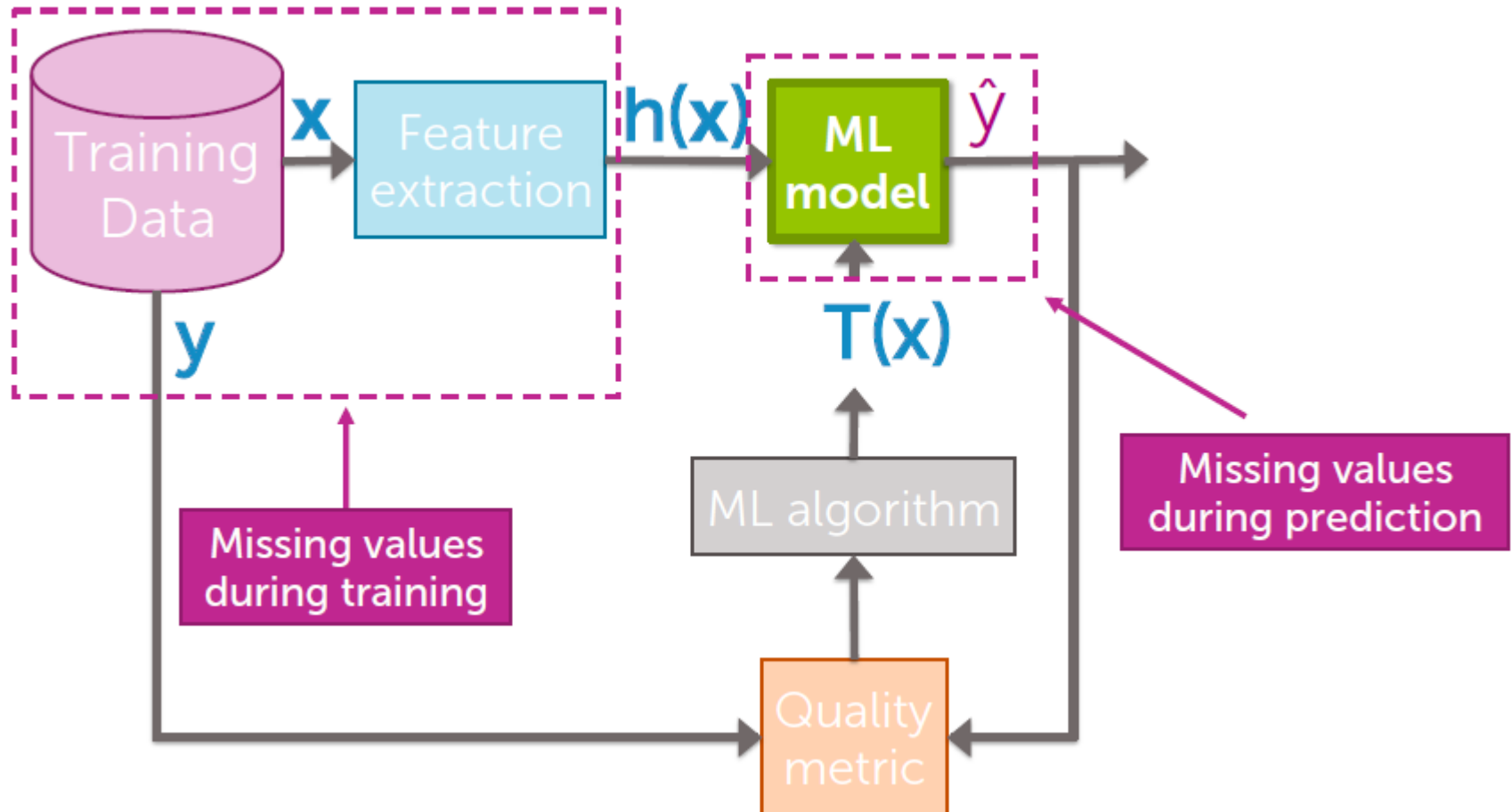
219

$x_i = (\text{Credit} = \text{poor}, \text{Income} = ?, \text{Term} = 5 \text{ years})$



Missing values

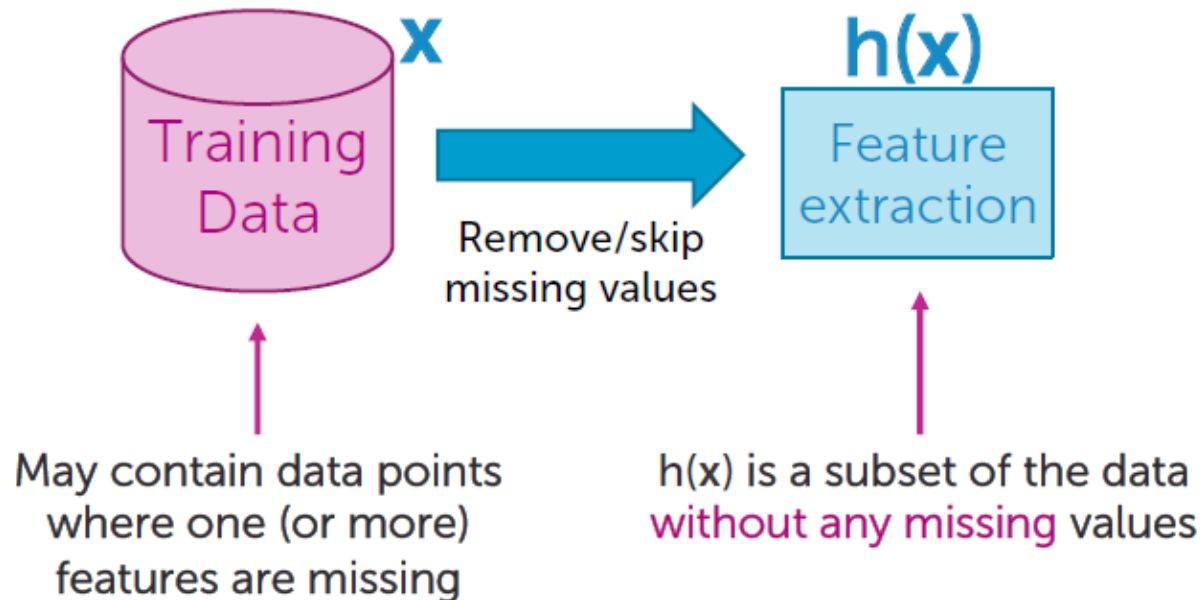
220



Handling missing data

221

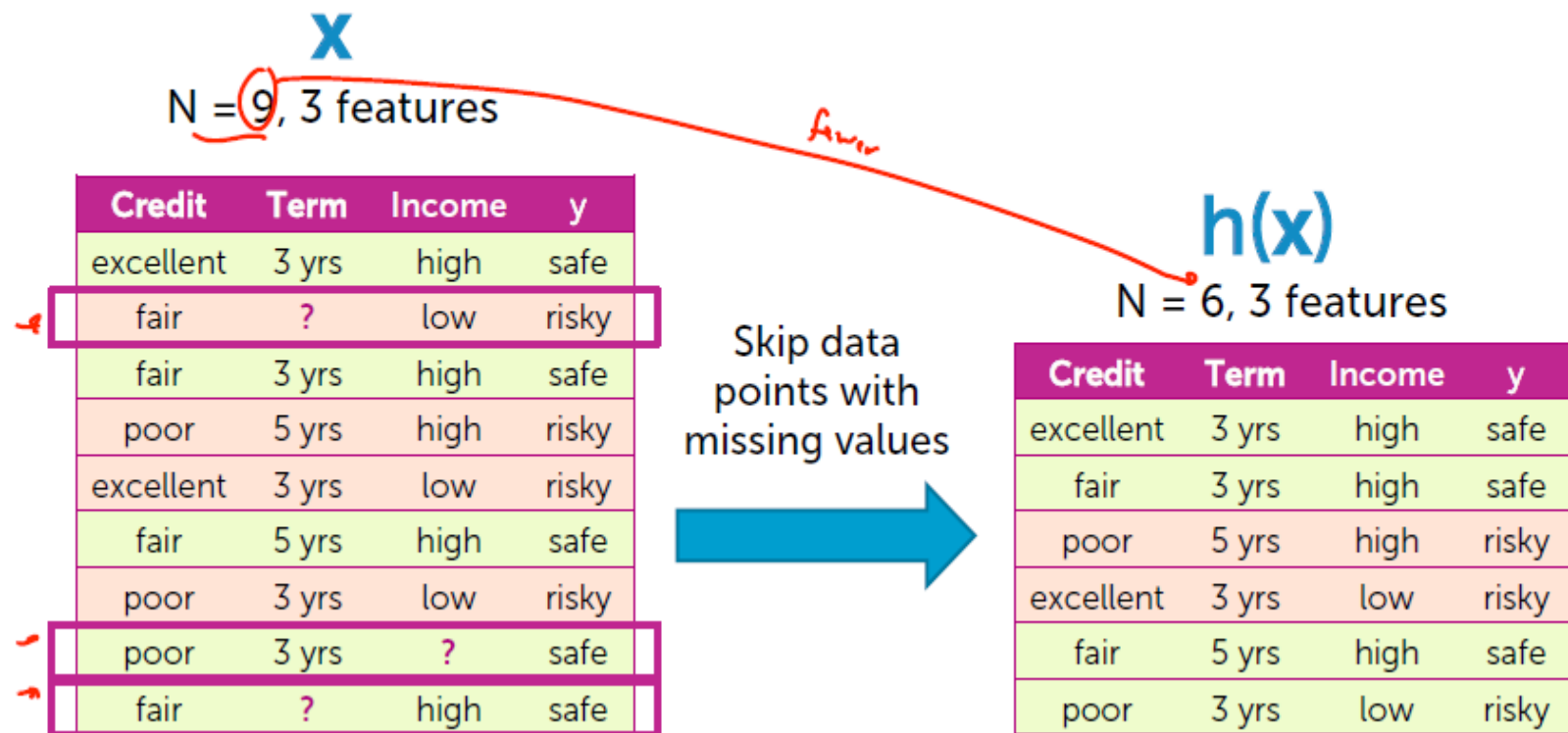
Idea 1: Purification by skipping/removing



Handling missing data

222

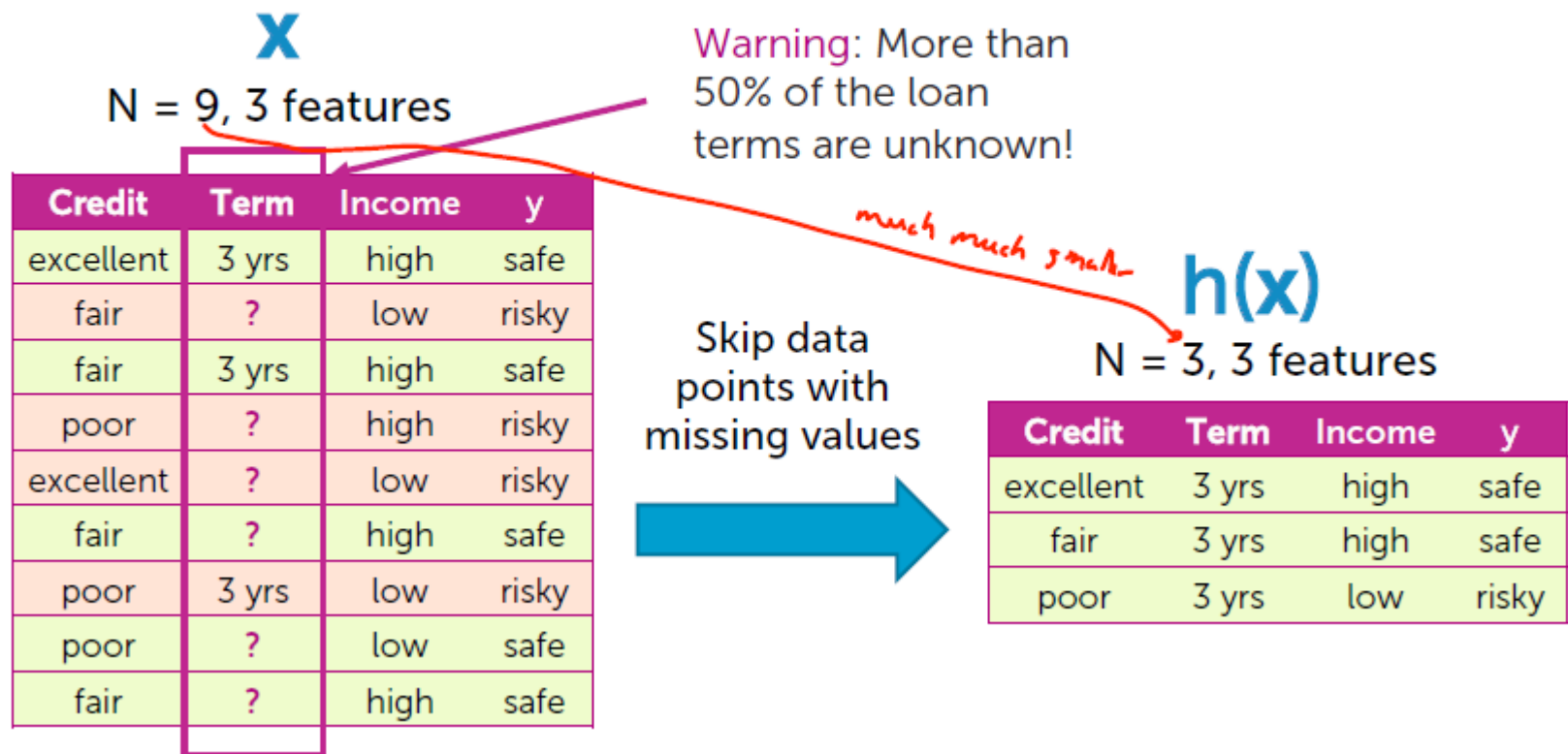
Idea 1: Skip data points with missing values



Handling missing data

223

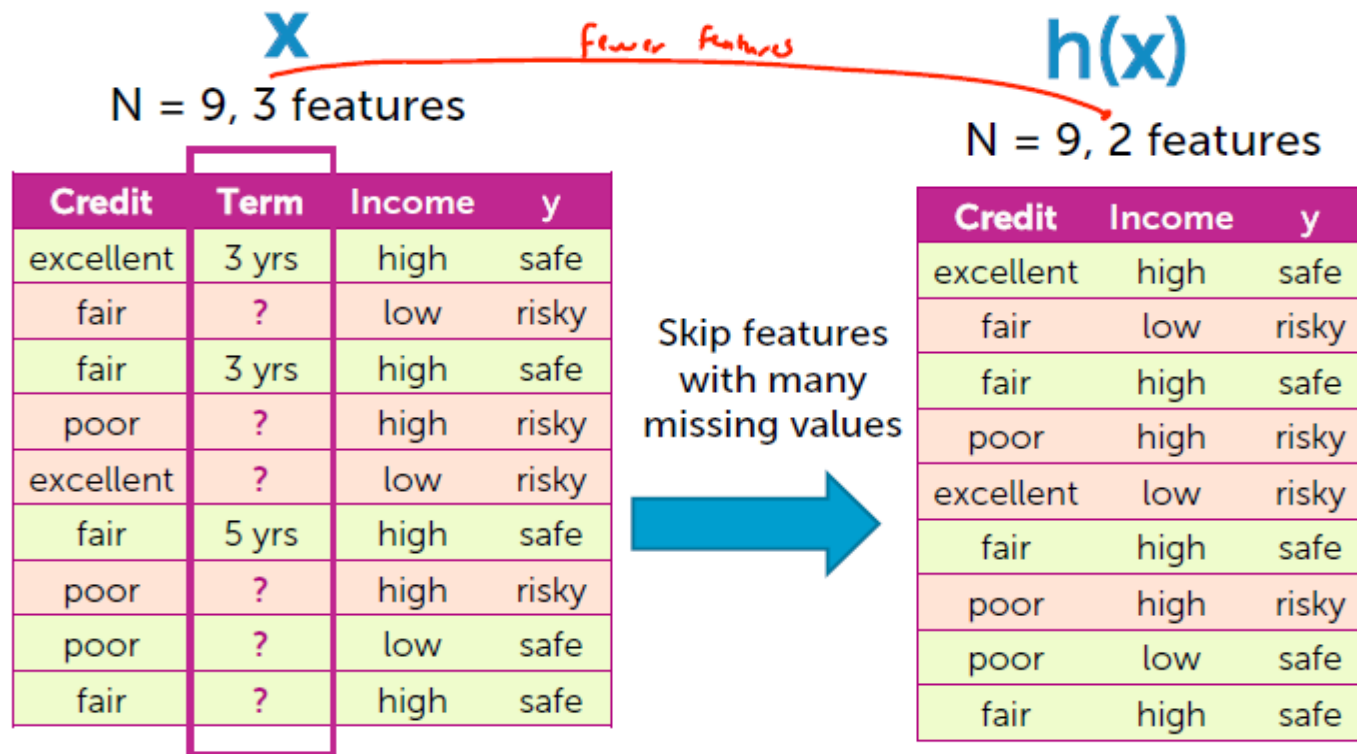
The challenge with Idea 1



Missing data

224

Idea 2: Skip features with missing values



Handling missing data

225

Missing value skipping: Ideas 1 & 2

Idea 1: Skip data points where any feature contains a missing value

- Make sure only a few data points are skipped

Idea 2: Skip an entire feature if it's missing for many data points

- Make sure only a few features are skipped

Handling missing data

226

Missing value skipping: Pros and Cons

Pros

- Easy to understand and implement
- Can be applied to any model
(decision trees, logistic regression, linear regression,...)

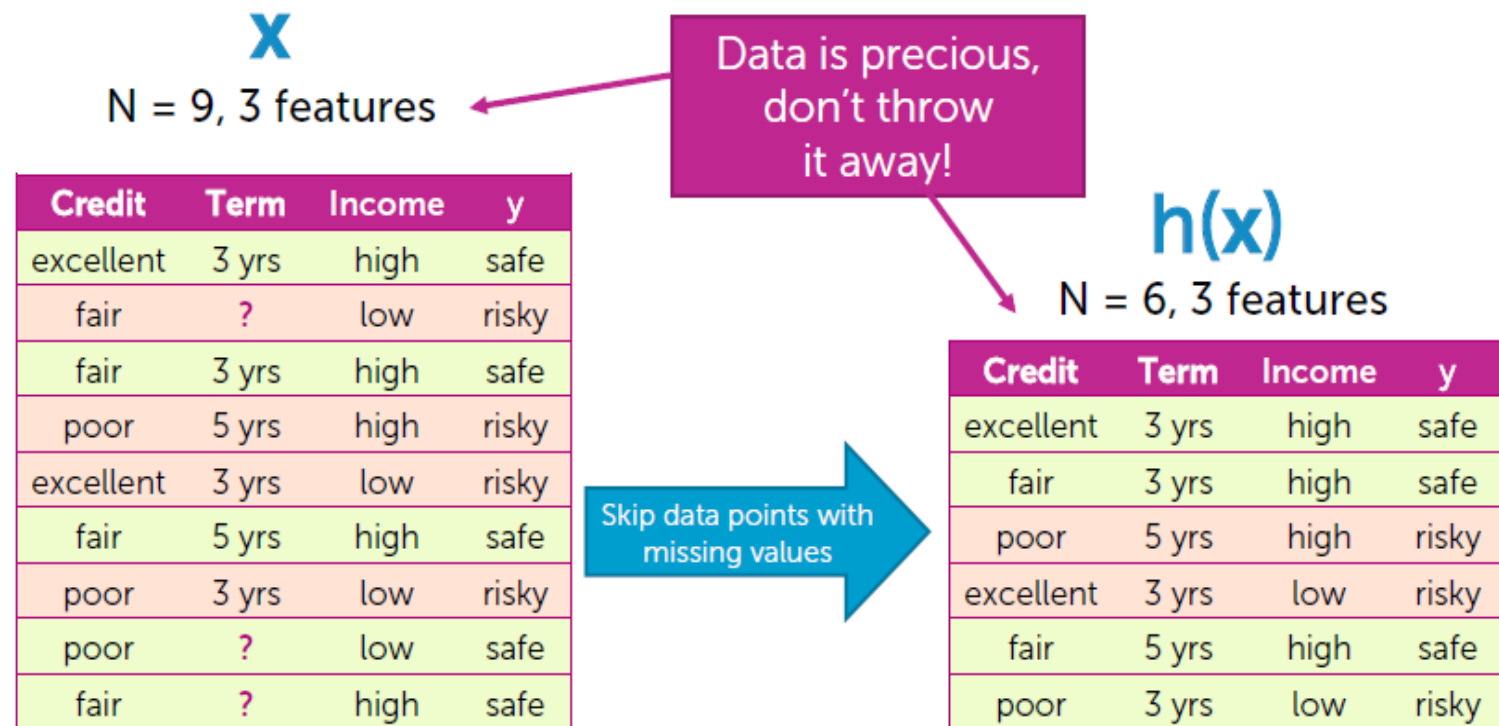
Cons

- Removing data points and features may remove important information from data
- Unclear when it's better to remove data points versus features
- Doesn't help if data is missing at prediction time

Data is precious

227

Main drawback of skipping strategy



Data is precious

228

Can we keep all the data?

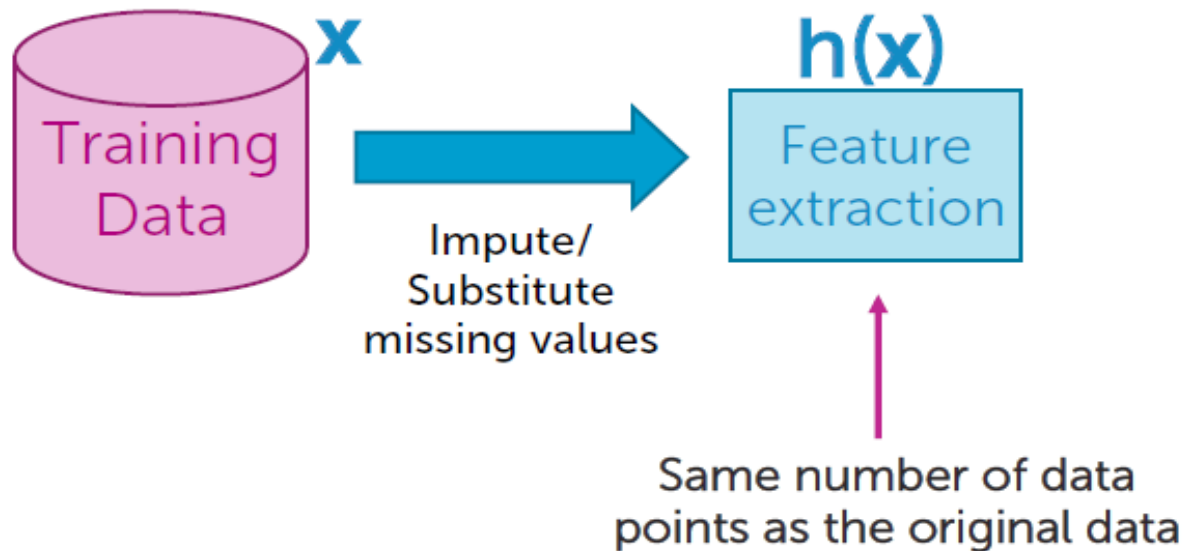
credit	term	income	y
excellent	3 yrs	high	safe
fair	?	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	high	safe
poor	3 yrs	high	risky
poor	?	low	safe
fair	?	high	safe

Use other data points
in x to "guess" the "?"

Handling missing data

229

Idea 2: Purification by imputing



Handling missing data

230

Idea 2: Imputation/Substitution

N = 9, 3 features

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	?	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	high	safe
poor	3 yrs	high	risky
poor	?	low	safe
fair	?	high	safe

Fill in each missing value with a calculated guess



N = 9, 3 features

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	3 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	high	safe
poor	3 yrs	high	risky
poor	3 yrs	low	safe
fair	3 yrs	high	safe

28

©2015-2016 Fasil Fayyaz, Codes Creative

Machine Learning Creative

29/10, 5/11 2019

Example

231

Example: Replace ? with most common value

3 year loans: 4
5 year loans: 2

simple
Best guess

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	?	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	high	safe
poor	3 yrs	high	risky
poor	?	low	safe
fair	?	high	safe

Purification by imputing

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	3 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	high	safe
poor	3 yrs	high	risky
poor	3 yrs	low	safe
fair	3 yrs	high	safe

29

©2015-2016 Emily Fox & Carlos Guestrin

Machine Learning Specialization

29/10, 5/11 2019

Handling missing data

232

Common (simple) rules for purification by imputation

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	?	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	high	safe
poor	3 yrs	high	risky
poor	?	low	safe
fair	?	high	safe

Impute each feature with missing values:

1. Categorical features use mode: Most popular value (mode) of non-missing x_i
2. Numerical features use average or median: Average or median value of non-missing x_i

Many advanced methods exist, e.g., expectation-maximization (EM) algorithm

Handling missing data

233

Missing value imputation: Pros and Cons

Pros

- Easy to understand and implement
- Can be applied to any model
(decision trees, logistic regression, linear regression,...)
- Can be used at prediction time: use same imputation rules

Cons

- May result in systematic errors

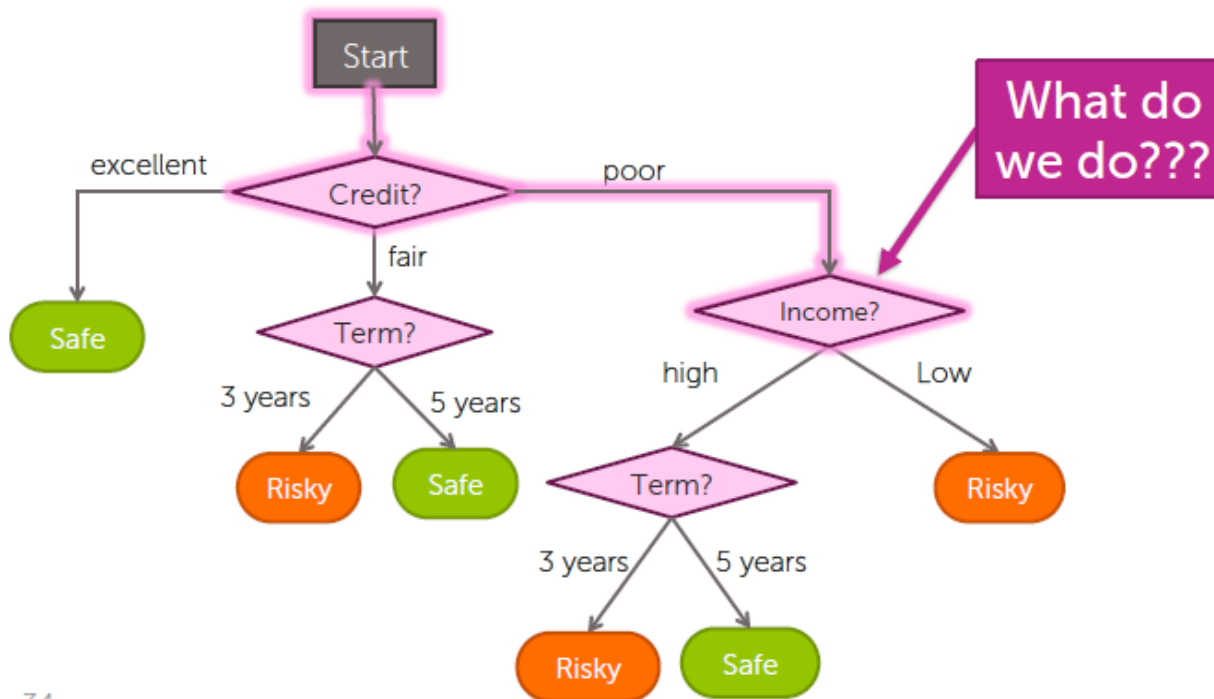
Example: Feature "age" missing in all banks in Washington by state law

Strategy 3: adapt algorithm

234

Missing values during prediction: *revisited*

$x_i = (\text{Credit} = \text{poor}, \text{Income} = ?, \text{Term} = 5 \text{ years})$

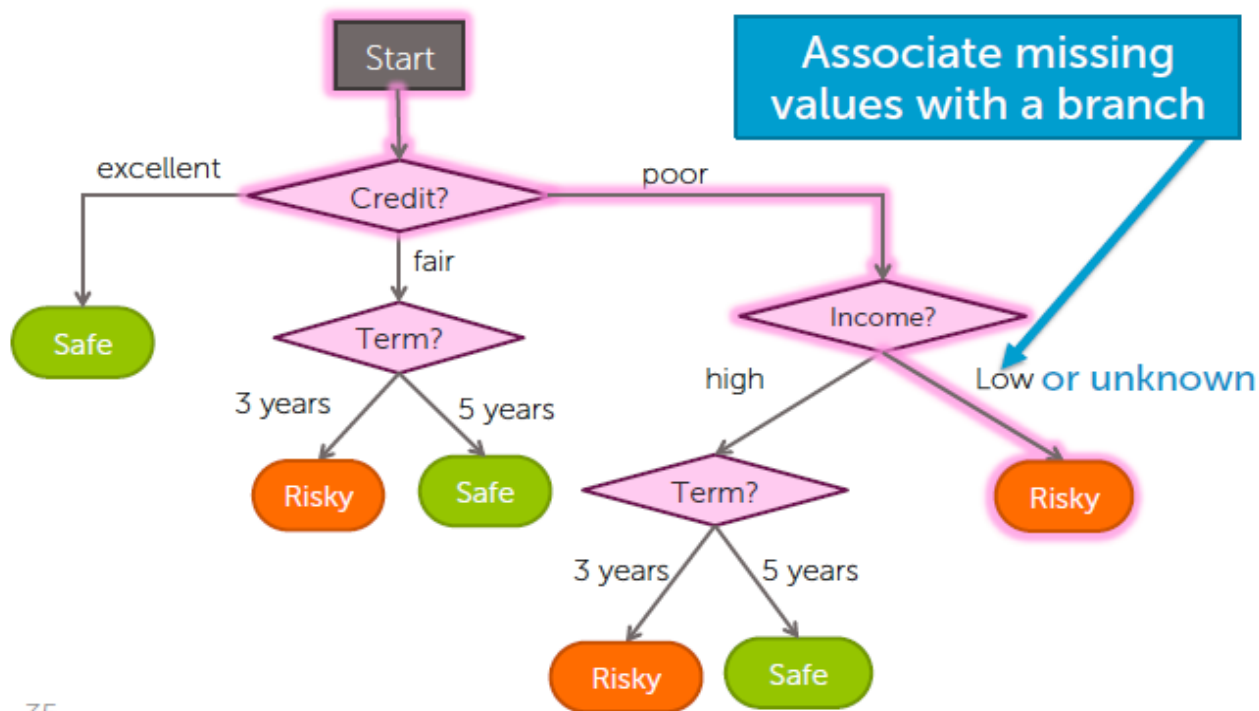


Strategy 3: adapt algorithm

235

Add missing values to the tree definition

$x_i = (\text{Credit} = \text{poor}, \text{Income} = ?, \text{Term} = 5 \text{ years})$

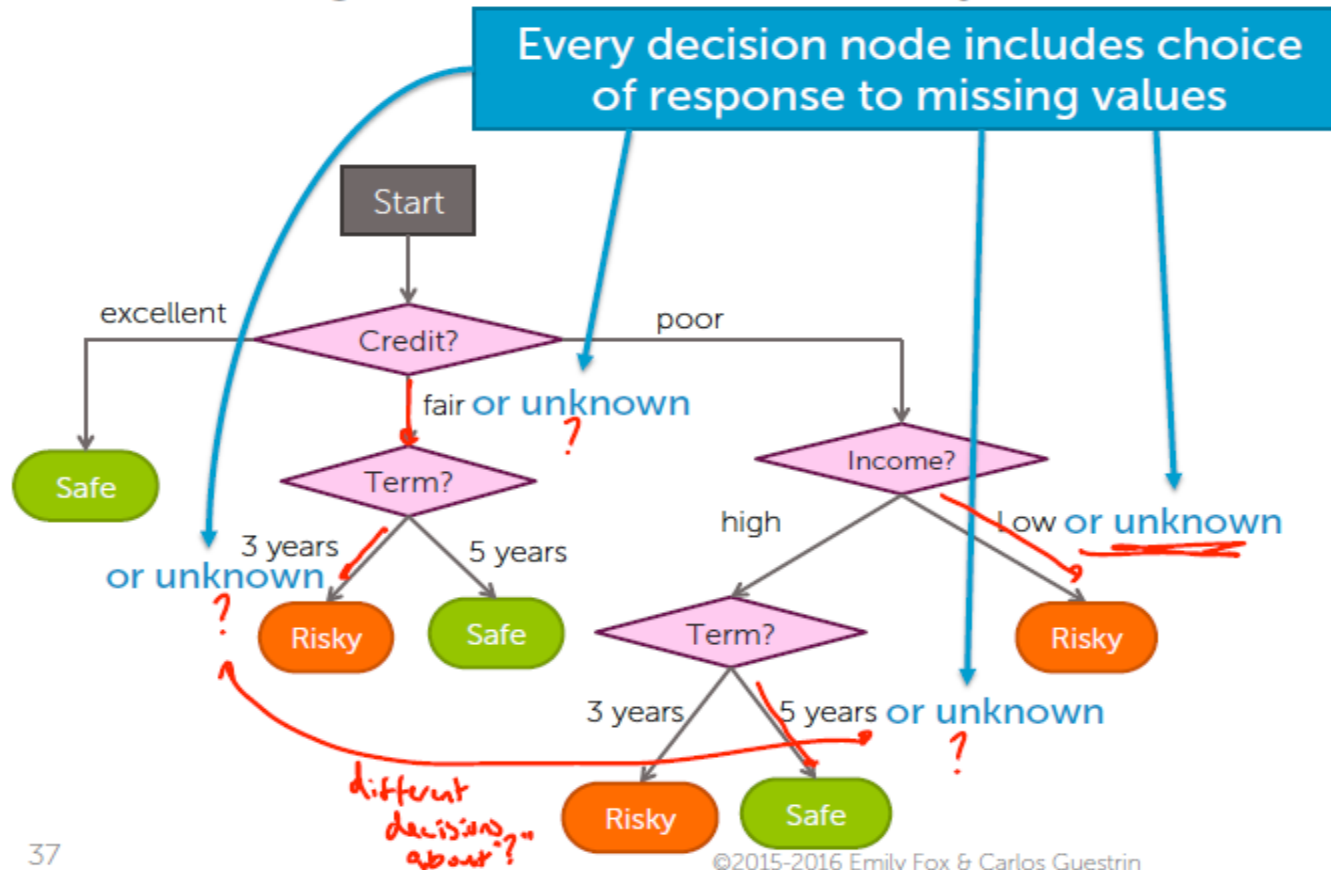


35

Strategy 3: adapt algorithm

236

Add missing value choice to every decision node



37

©2015-2016 Emily Fox & Carlos Guestrin

Machine Learning

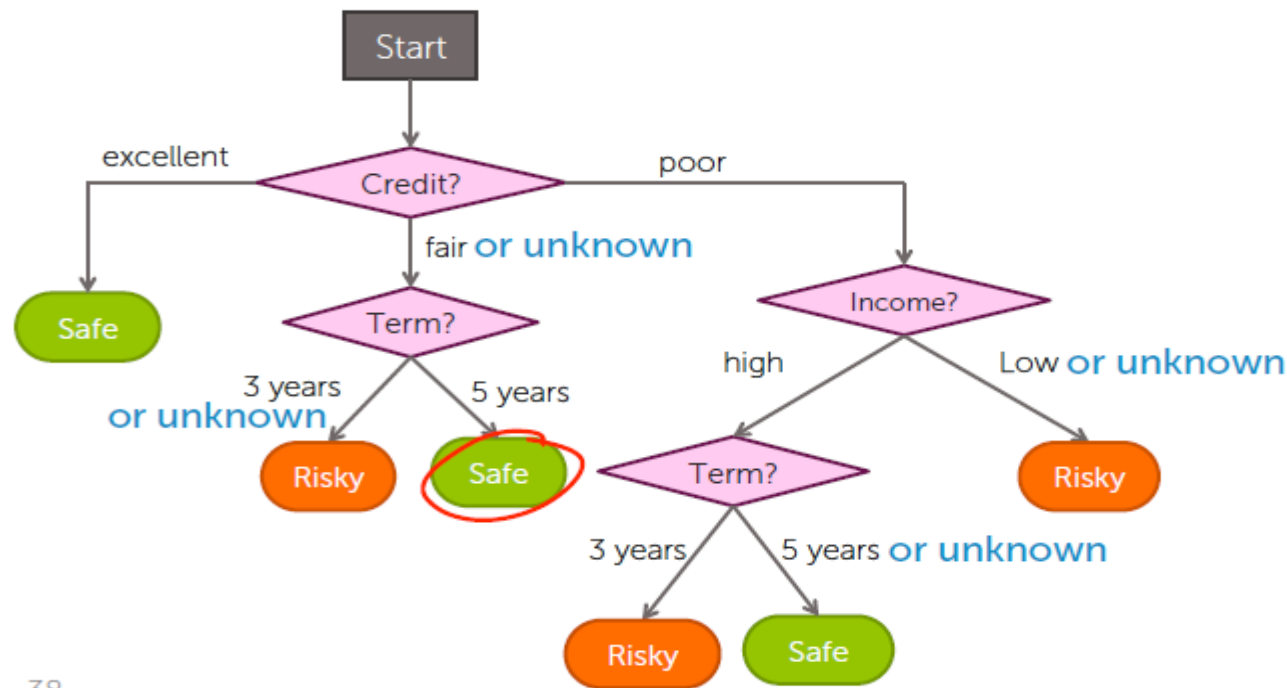
29/10, 5/11 2019

Strategy 3: adapt algorithm

237

Prediction with missing values becomes simple

$x_i = (\text{Credit} = ?, \text{Income} = \text{high}, \text{Term} = 5 \text{ years})$



38

©2015-2016 Emily Fox & Carlos Guestrin

Machine

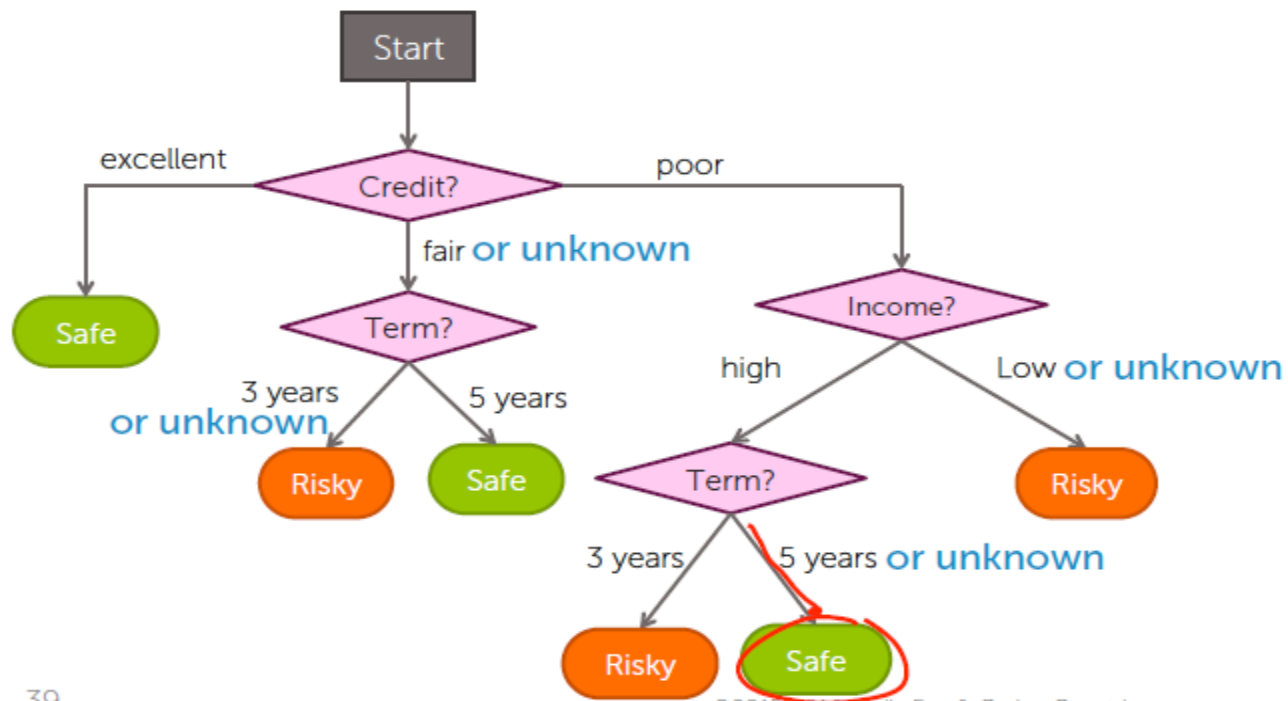
29/10, 5/11 2019

Strategy 3: addapt algorithm

238

Prediction with missing values becomes simple

$x_i = (\text{Credit} = \text{poor}, \text{Income} = \text{high}, \text{Term} = ?)$



39

Strategy 3: adapt algorithm

239

Explicitly handling missing data by learning algorithm: Pros and Cons

Pros

- Addresses training and prediction time
- More accurate predictions

Cons


- Requires modification of learning algorithm
 - Very simple for decision trees

Feature split selection with missing data

240

Greedy decision tree learning

- Step 1: Start with an empty tree
- Step 2: Select a feature to split data
- For each split of the tree:
 - Step 3: If nothing more to, make predictions
 - Step 4: Otherwise, go to Step 2 & continue (recurse) on this split



Pick feature split
leading to lowest
classification error

Must select feature &
branch for missing values!

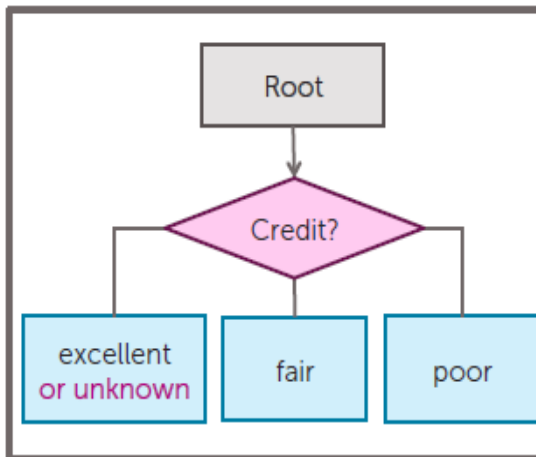
Feature split selection with missing data

241

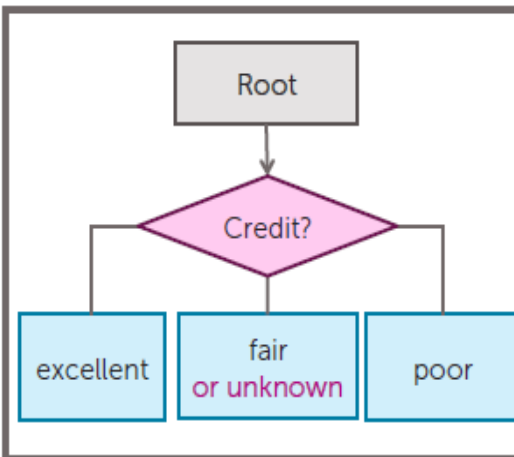
Should **missing** go left, right, or middle?

Choose branch that leads to lowest classification error!

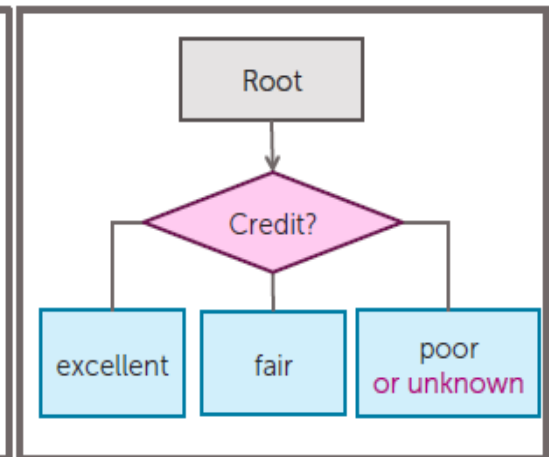
Choice 1: Missing values go with Credit=excellent



Choice 2: Missing values go with Credit=fair



Choice 3: Missing values go with Credit=poor



Feature split selection with missing data

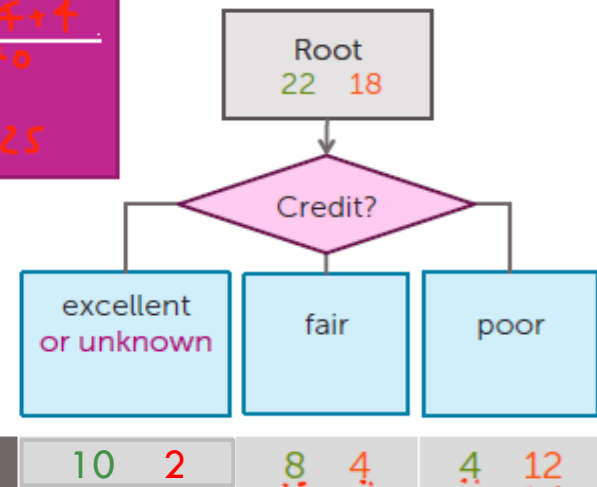
242

Computing classification error of decision stump with missing data

N = 40, 3 features

Credit	Term	Income	y
excellent	3 yrs	high	safe
?	5 yrs	low	<u>risky</u>
fair	3 yrs	high	safe
poor	5 yrs	high	risky
?	3 yrs	low	<u>risky</u>
?	5 yrs	low	<u>safe</u>
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe
...

$$\text{Error} = \frac{2+4+4}{40} = 0.25$$



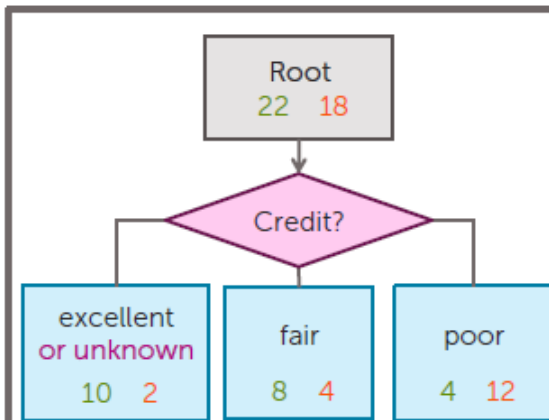
Feature split selection with missing data

243

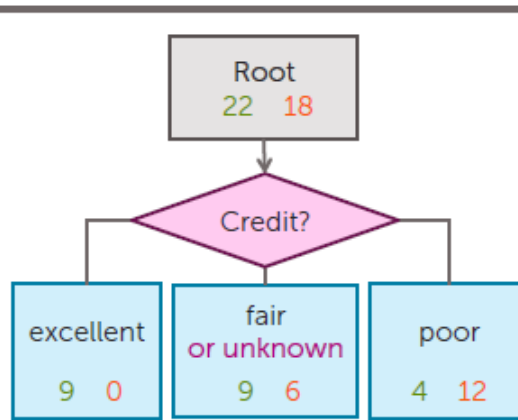
Use classification error to decide

Best choice \rightarrow assign "unknown" to Credit = poor

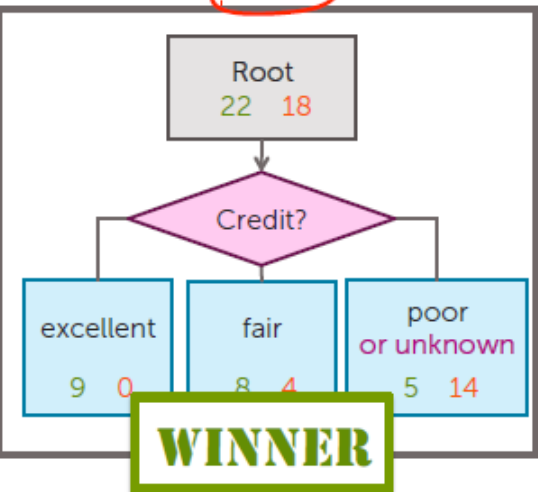
Choice 1: error = 0.25



Choice 2: error = 0.25



Choice 3: error = 0.225



Feature split selection with missing data

244

- Given a subset of data M (a node in a tree)
- For each feature $h_i(x)$:
 1. Split data points of M where $h_i(x)$ is *not* "unknown" according to feature $h_i(x)$
 2. Consider assigning data points with "unknown" value for $h_i(x)$ to each branch
 - A. Compute classification error split & branch assignment of "unknown" values
- Chose feature $h^*(x)$ & branch assignment of "unknown" with lowest classification error

What can you do now

245

Describe common ways to handling missing data:

1. Skip all rows with any missing values
2. Skip features with many missing values
3. Impute missing values using other data points

Modify learning algorithm (**decision trees**) to handle missing data:

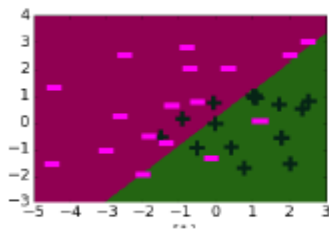
1. Missing values get added to one branch of split
2. Use classification error to determine where missing values go

Ensemble classifiers and boosting

Simple classifiers

247

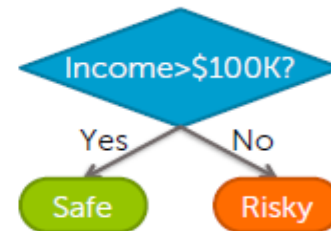
Simple (weak) classifiers are good!



Logistic
regression
w. simple
features



Shallow
decision trees



Decision
stumps

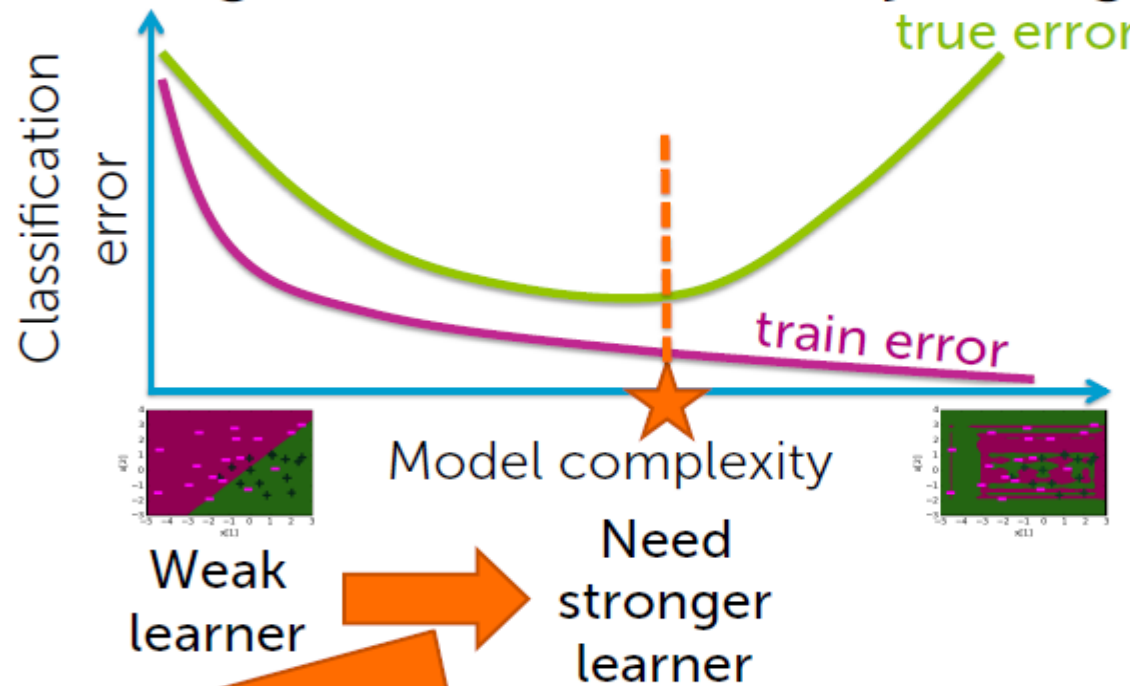
Low variance. Learning is fast!

But high bias...

Simple classifiers

248

Finding a classifier that's just right




Option 1: add more features or depth
Option 2: ?????

Can they be combined?

249

Boosting question

"Can a set of weak learners be combined to create a stronger learner?" *Kearns and Valiant (1988)*



Yes! *Schapire (1990)*



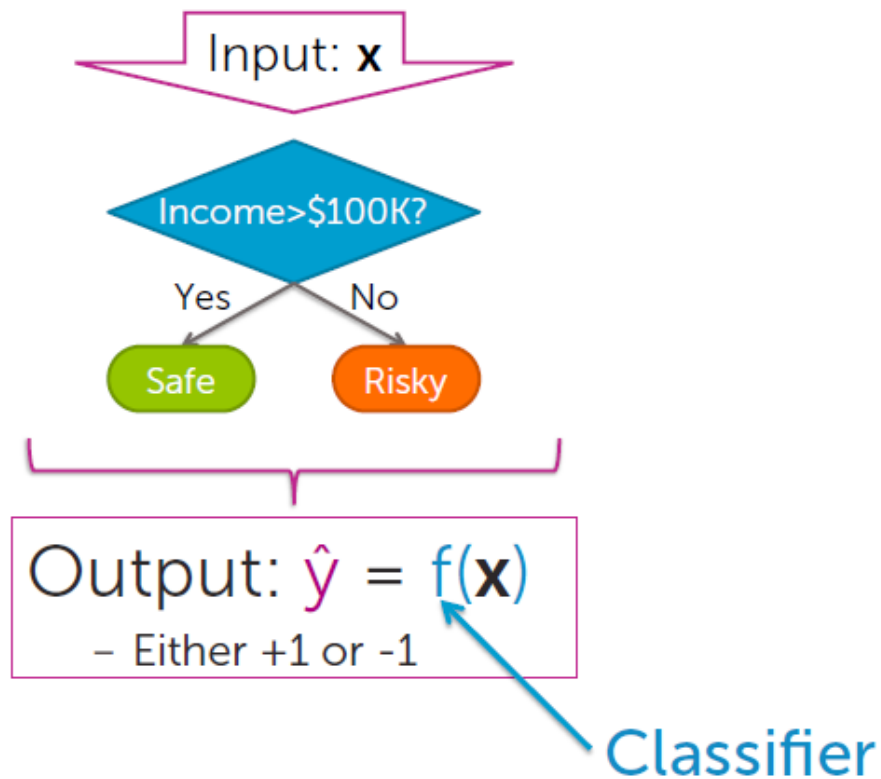
Boosting



Amazing impact: • simple approach • widely used in industry • wins most Kaggle competitions

A single classifier

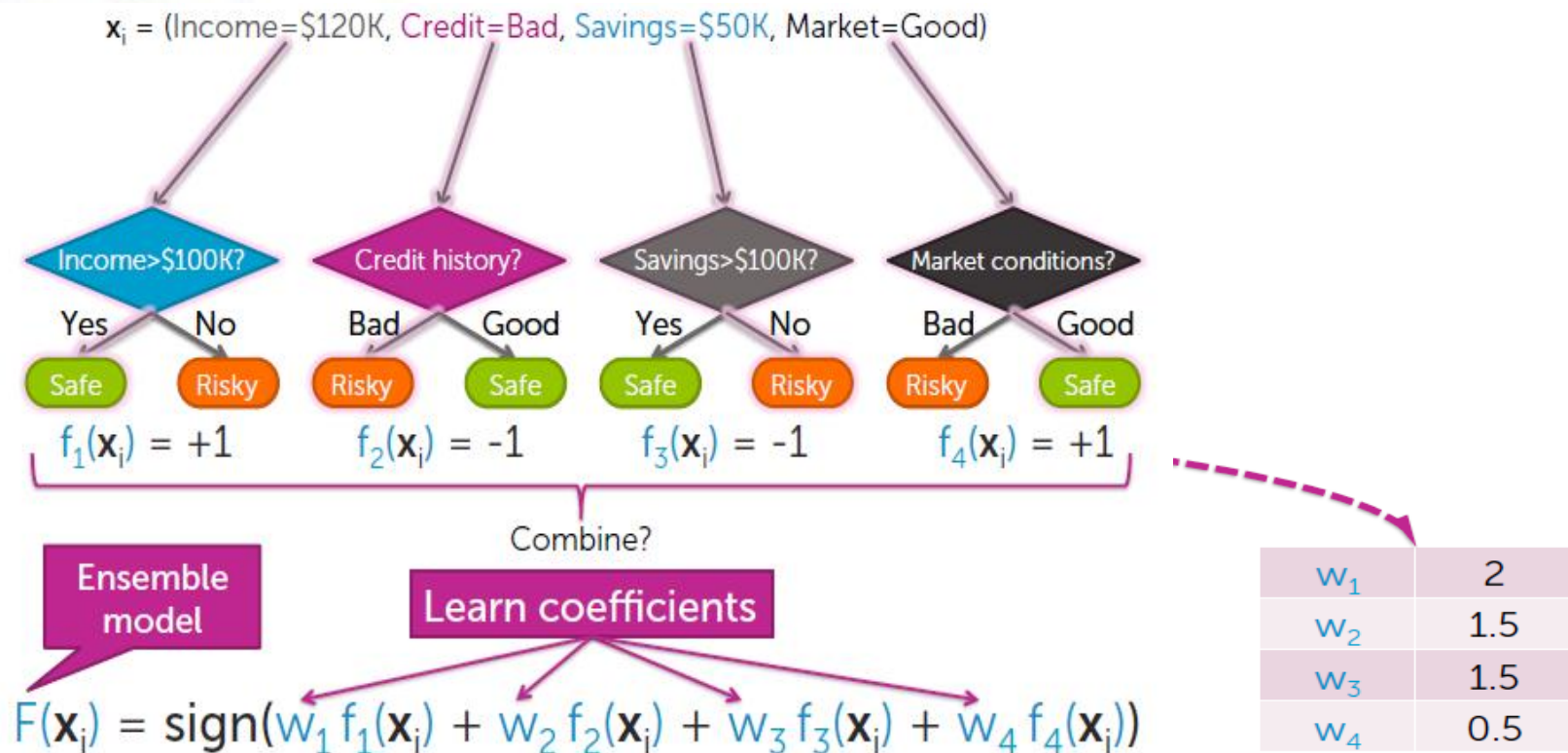
250



Ensemble methods

251

Each classifier "votes" on prediction



Ensemble classifier

252

- Goal:
 - Predict output y
 - Either +1 or -1
 - From input \mathbf{x}
- Learn ensemble model:
 - Classifiers: $f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_T(\mathbf{x})$
 - Coefficients: $\hat{w}_1, \hat{w}_2, \dots, \hat{w}_T$
- Prediction:

$$\hat{y} = \text{sign} \left(\sum_{t=1}^T \hat{w}_t f_t(\mathbf{x}) \right)$$

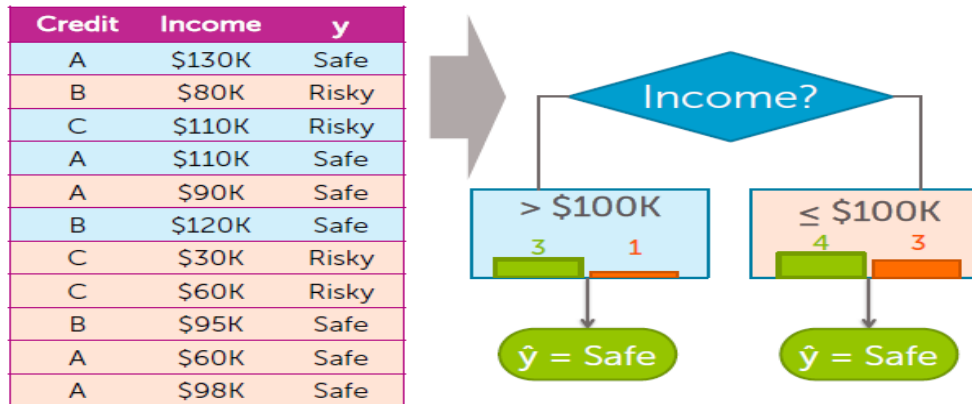
Boosting

253

Training a classifier



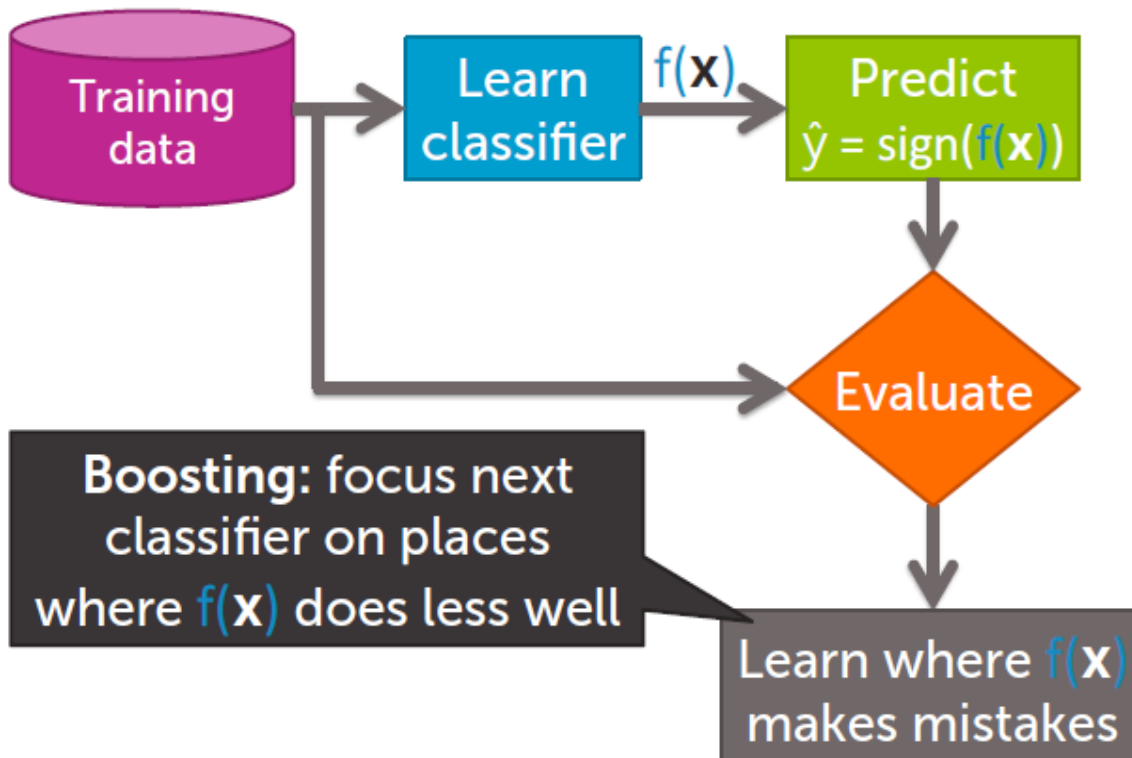
Learning decision stump



Boosting

254

Boosting = Focus learning on “hard” points



Weighted data

255

Learning on weighted data:

More weight on “hard” or more important points

- Weighted dataset:
 - Each \mathbf{x}_i, y_i weighted by α_i
 - More important point = higher weight α_i
- Learning:
 - Data point j counts as α_j data points
 - E.g., $\alpha_j = 2 \rightarrow$ count point twice

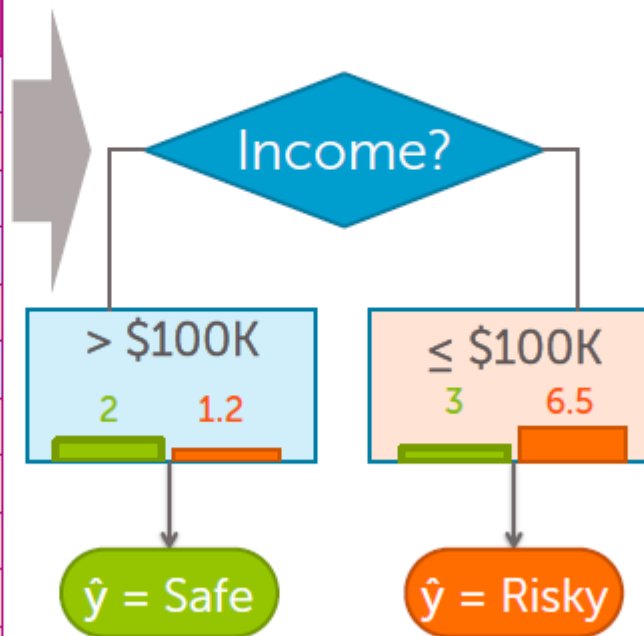
Weighted data

256

Learning a decision stump on weighted data

Increase weight α of harder/
misclassified points

Credit	Income	y	Weight α
A	\$130K	Safe	0.5
B	\$80K	Risky	1.5
C	\$110K	Risky	1.2
A	\$110K	Safe	0.8
A	\$90K	Safe	0.6
B	\$120K	Safe	0.7
C	\$30K	Risky	3
C	\$60K	Risky	2
B	\$95K	Safe	0.8
A	\$60K	Safe	0.7
A	\$98K	Safe	0.9



*Use sum over
weights of the
data points*

Weighted data

257

Learning from weighted data in general

- Usually, learning from weighted data
 - Data point i counts as α_i data points
- E.g., gradient ascent for logistic regression:

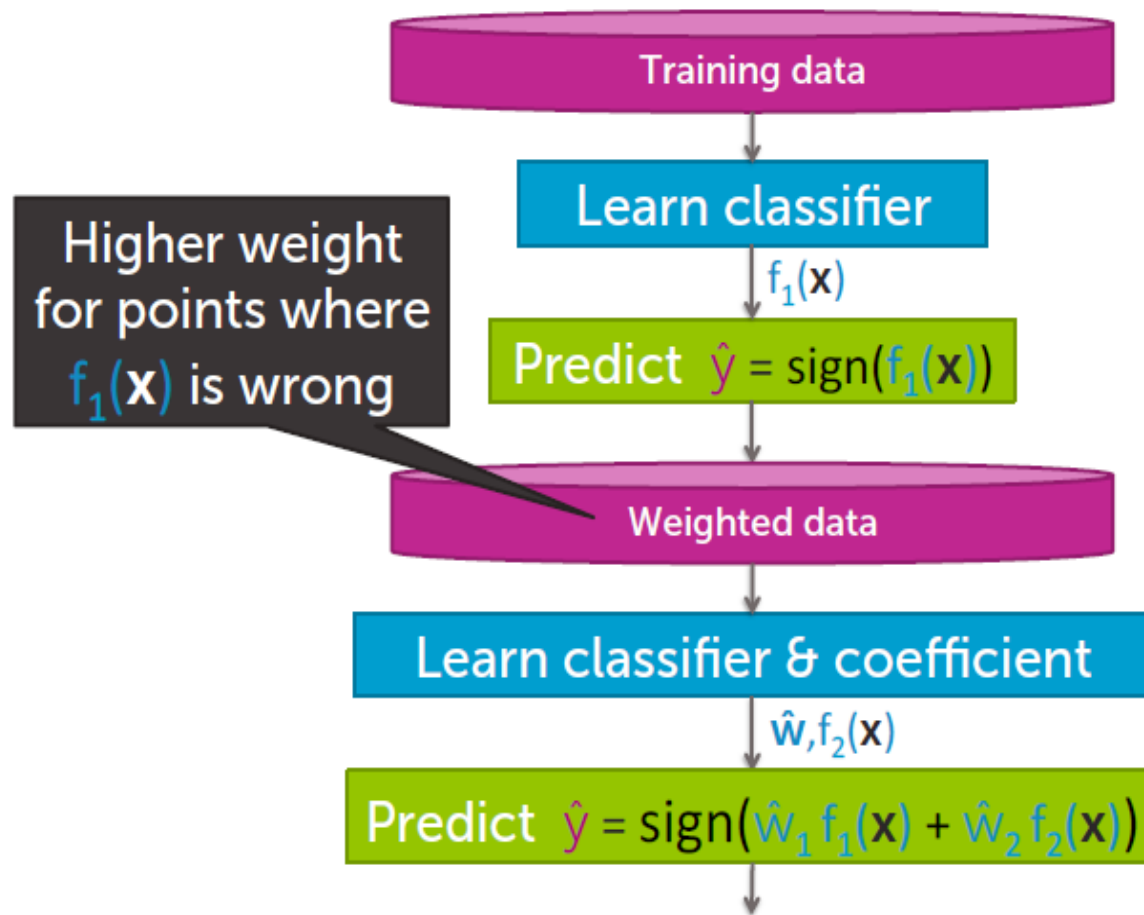
$$\mathbf{w}_j^{(t+1)} \leftarrow \mathbf{w}_j^{(t)} + \eta \sum_{i=1}^N \alpha_i (\mathbf{x}_i) \left(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w}^{(t)}) \right)$$

Sum over data points

Weigh each point by α_i

Boosting = greedy learning ensembles from data

258



AdaBoost: learning ensemble

[Freund & Schapire 1999]

259

- Start same weight for all points: $\alpha_i = 1/N$
 - For $t = 1, \dots, T$
 - Learn $f_t(\mathbf{x})$ with data weights α_i
 - Compute coefficient \hat{w}_t
 - Recompute weights α_i
- Problem 1: How much do I trust f_t ?
- Problem 2: weigh mistakes more?

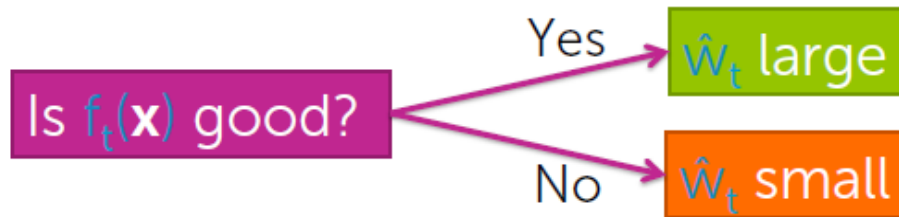
- Final model predicts by:

$$\hat{y} = \text{sign} \left(\sum_{t=1}^T \hat{w}_t f_t(\mathbf{x}) \right)$$

(Note: \hat{w}_t is labeled as coefficient in the original image)

AdaBoost: Computing coefficients w_t

260



- $f_t(\mathbf{x})$ is good $\rightarrow f_t$ has low training error
- Measuring error in weighted data?
 - Just weighted # of misclassified points

Weighted classification error

261

- Total weight of mistakes:

$$= \sum_{i=1}^N \alpha_i \underbrace{\mathbb{1}(\hat{y}_i \neq y_i)}_{\text{mistake?}}$$

- Total weight of all points:

$$= \sum_{i=1}^N \alpha_i$$

- Weighted error measures fraction of weight of mistakes:

$$\text{weighted_error} = \frac{\text{Total weight of mistakes}}{\text{Total weight of all data points}}$$


- Best possible value is 0.0 \rightarrow worst 1.0 \rightarrow Random classifier = 0.5

AdaBoost formula

262

AdaBoost: Formula for computing coefficient \hat{w}_t of classifier $f_t(x)$

$$\hat{w}_t = \frac{1}{2} \ln \left(\frac{1 - \text{weighted_error}(f_t)}{\text{weighted_error}(f_t)} \right)$$



	weighted_error(f_t) on training data	$\frac{1 - \text{weighted_error}(f_t)}{\text{weighted_error}(f_t)}$	\hat{w}_t
Yes	0.01	$\frac{1 - 0.01}{0.01} = 99$	$\frac{1}{2} \ln 99 = 2.3$
No	0.5	$\frac{1 - 0.5}{0.5} = 1$	0
	0.99	$\frac{1 - 0.99}{0.99} = 0.01$	-2.3

Terrible classifier, but $1 - f_t$ is awesome !!

AdaBoost: learning ensemble

263

- Start same weight for all points: $\alpha_i = 1/N$

- For $t = 1, \dots, T$

- Learn $f_t(\mathbf{x})$ with data weights α_i

- Compute coefficient \hat{w}_t

$$\hat{w}_t = \frac{1}{2} \ln \left(\frac{1 - \text{weighted_error}(f_t)}{\text{weighted_error}(f_t)} \right)$$

- Recompute weights α_i

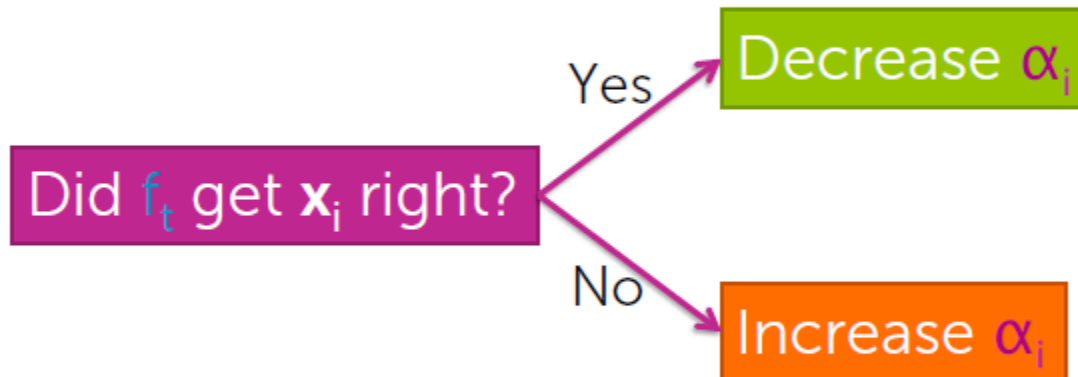
- Final model predicts by:

$$\hat{y} = \text{sign} \left(\sum_{t=1}^T \hat{w}_t f_t(\mathbf{x}) \right)$$

AdaBoost: updating weights α_i

264

Updating weights α_i based on where classifier $f_t(x)$ makes mistakes



AdaBoost: updating weights α_i

265

AdaBoost: Formula for updating weights α_i

$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{W}_t}, & \text{if } f_t(\mathbf{x}_i) = y_i \leftarrow \text{Correct} \\ \alpha_i e^{\hat{W}_t}, & \text{if } f_t(\mathbf{x}_i) \neq y_i \leftarrow \text{Mistake} \end{cases}$$

		$f_t(\mathbf{x}_i) = y_i ?$	\hat{W}_t	Multiply α_i by	Implication
Did f_t get \mathbf{x}_i right?	Yes	Correct	2.3	$e^{-2.3} = 0.1$	Decrease importance of \mathbf{x}_i, y_i
		Correct	0	$e^0 = 1$	Keep importance the same
	No	Mistake	2.3	$e^{2.3} = 9.98$	Increasing importance of \mathbf{x}_i, y_i
		Mistake	0	$e^0 = 1$	Keep importance the same

AdaBoost: learning ensemble

266

- Start same weight for all points: $\alpha_i = 1/N$

- For $t = 1, \dots, T$

- Learn $f_t(\mathbf{x})$ with data weights α_i

- Compute coefficient \hat{w}_t

$$\hat{w}_t = \frac{1}{2} \ln \left(\frac{1 - \text{weighted_error}(f_t)}{\text{weighted_error}(f_t)} \right)$$

- Recompute weights α_i

- Final model predicts by:

$$\hat{y} = \text{sign} \left(\sum_{t=1}^T \hat{w}_t f_t(\mathbf{x}) \right)$$

$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i) = y_i \\ \alpha_i e^{\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i) \neq y_i \end{cases}$$

AdaBoost: normalizing weights α_i

267

x_i

If x_i often mistake,
weight α_i gets very
large

If x_i often correct,
weight α_i gets very
small

Can cause numerical instability
after many iterations

Normalize weights to
add up to 1 after every iteration

$$\alpha_i \leftarrow \frac{\alpha_i}{\sum_{j=1}^N \alpha_j}$$

AdaBoost: learning ensemble

268

- Start same weight for all points: $\alpha_i = 1/N$

$$\hat{w}_t = \frac{1}{2} \ln \left(\frac{1 - \text{weighted_error}(f_t)}{\text{weighted_error}(f_t)} \right)$$

- For $t = 1, \dots, T$

- Learn $f_t(\mathbf{x})$ with data weights α_i

- Compute coefficient \hat{w}_t

- Recompute weights α_i

- Normalize weights α_i

$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i) = y_i \\ \alpha_i e^{\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i) \neq y_i \end{cases}$$

- Final model predicts by:

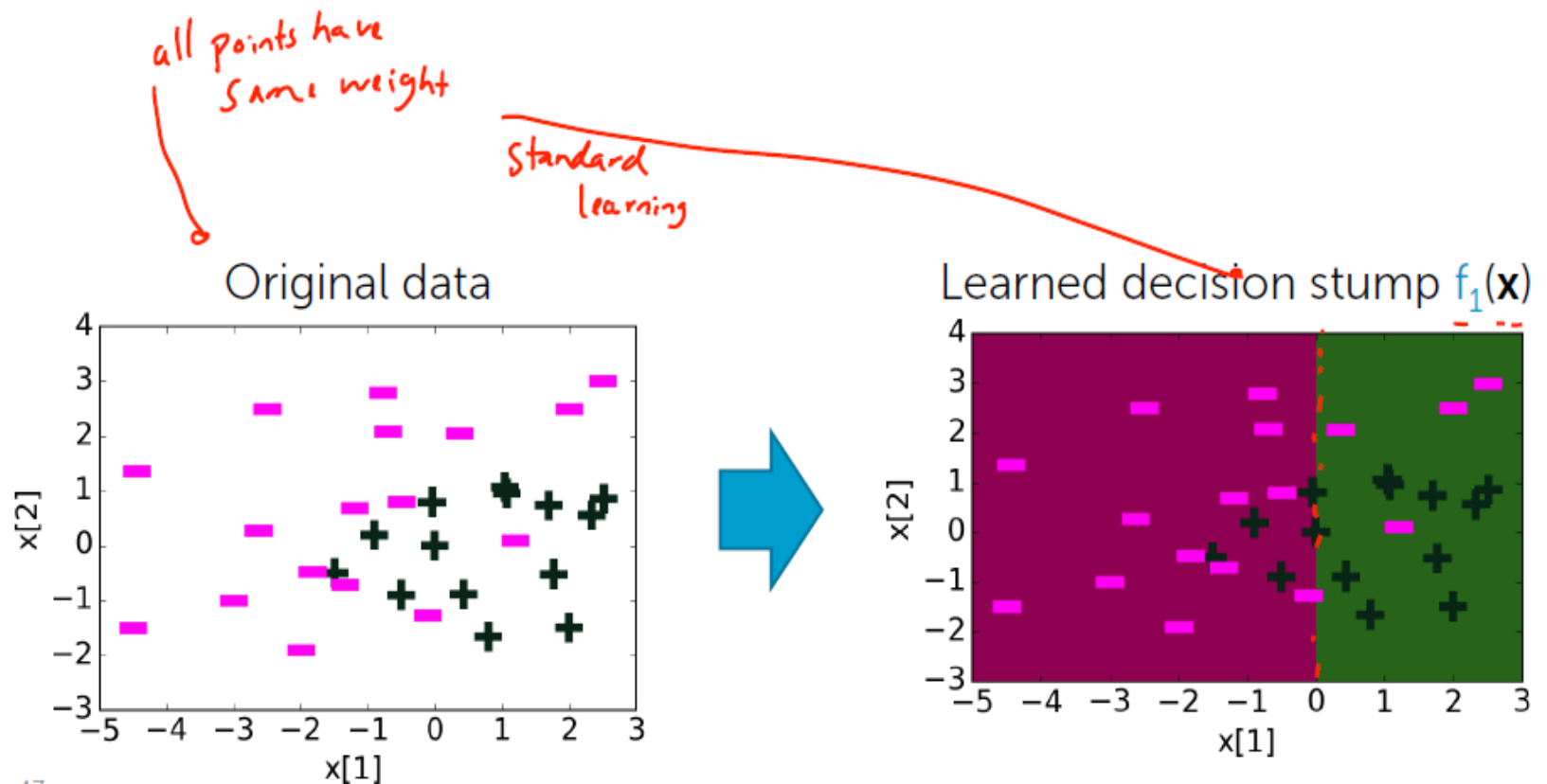
$$\hat{y} = \text{sign} \left(\sum_{t=1}^T \hat{w}_t f_t(\mathbf{x}) \right)$$

$$\alpha_i \leftarrow \frac{\alpha_i}{\sum_{j=1}^N \alpha_j}$$

AdaBoost: example

269

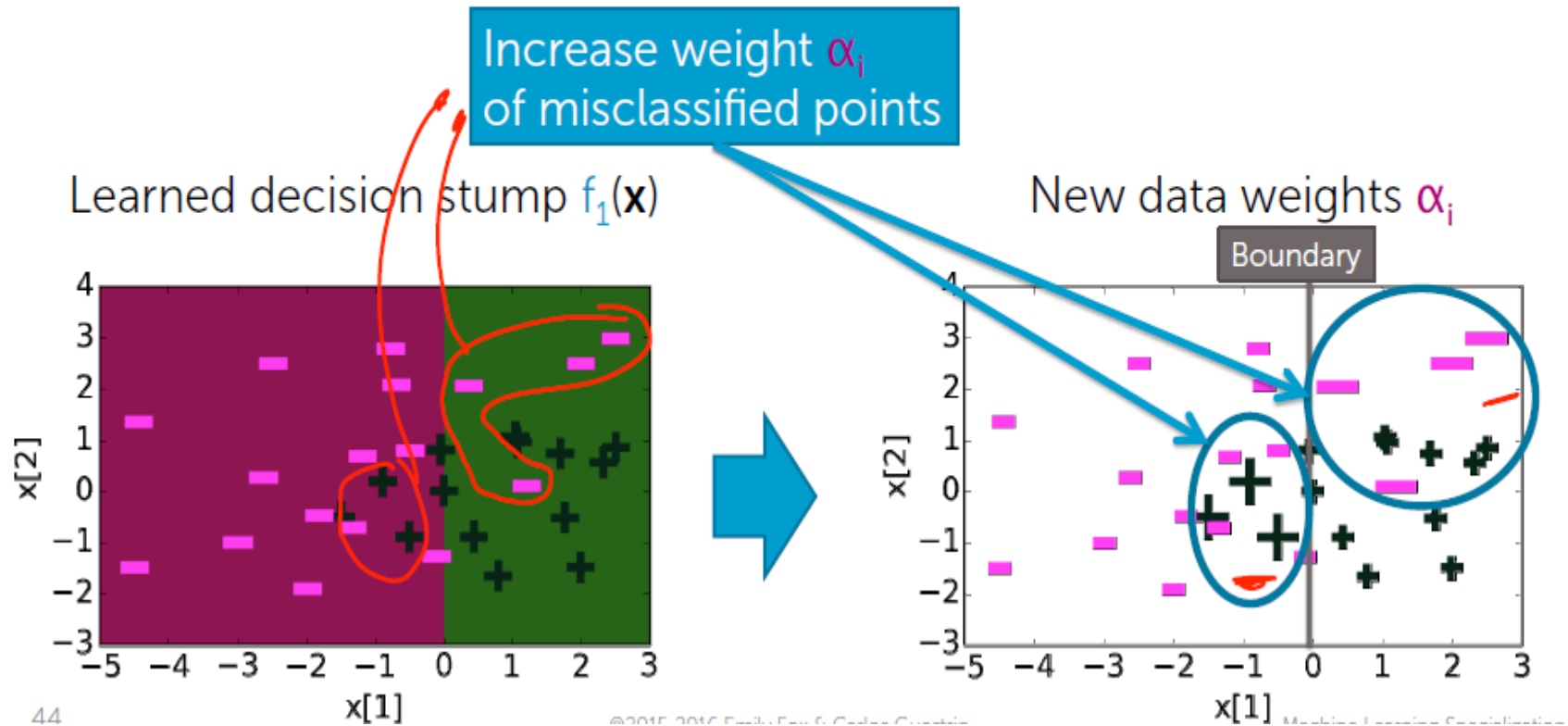
$t=1$: Just learn a classifier on original data



AdaBoost: example

270

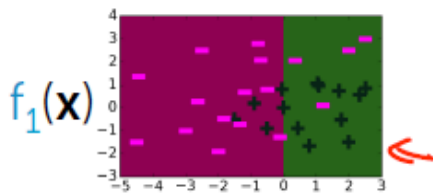
Updating weights α_i



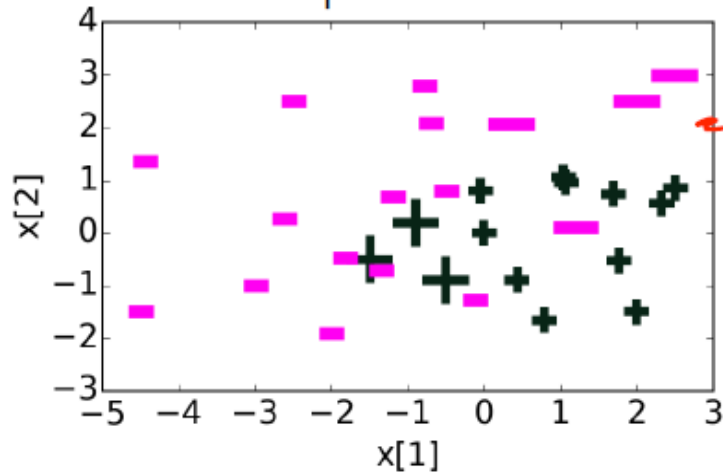
AdaBoost: example

271

$t=2$: Learn classifier on weighted data

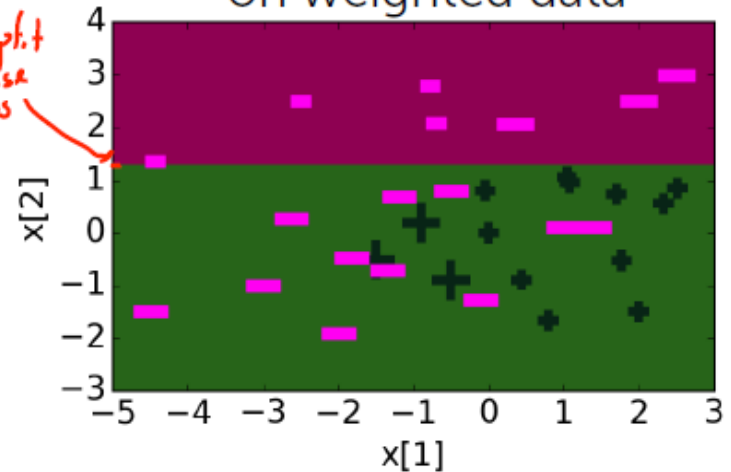


Weighted data: using α_i
chosen in previous iteration



better split
for these
weights

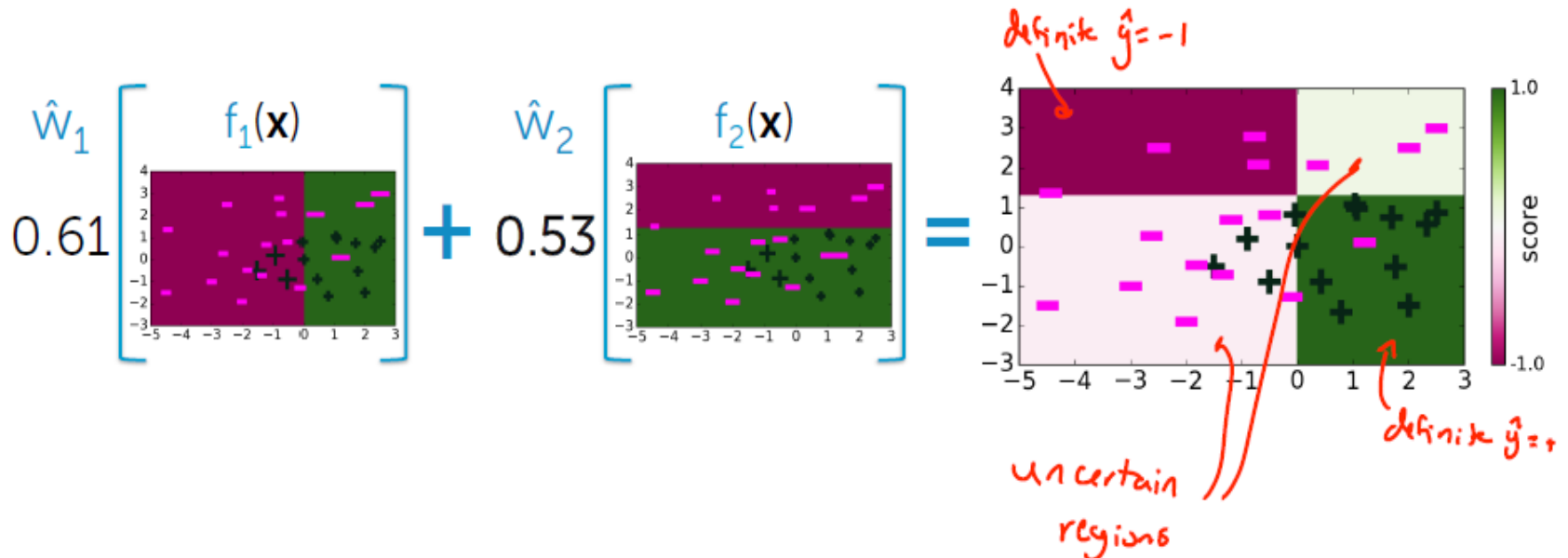
Learned decision stump $f_2(x)$
on weighted data



AdaBoost: example

272

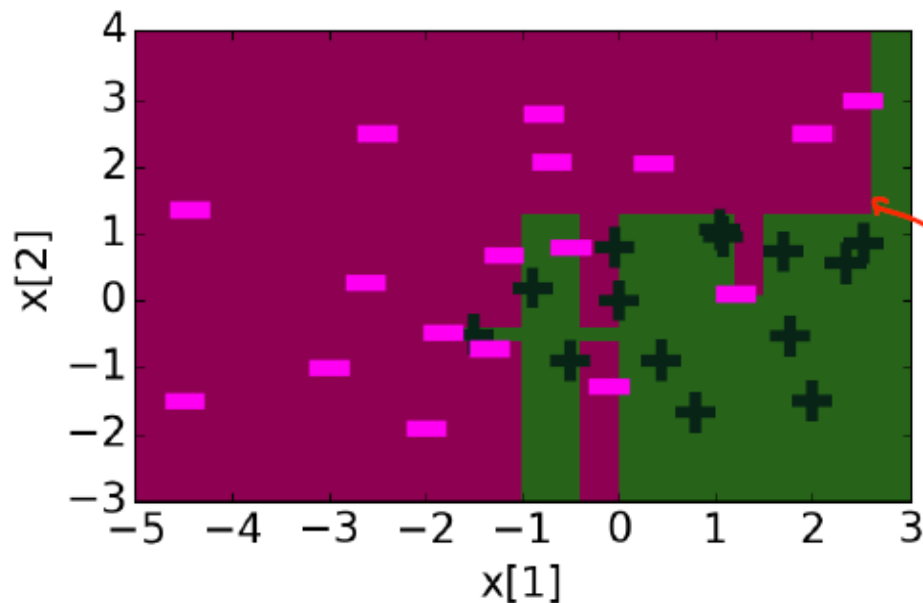
Ensemble becomes weighted
sum of learned classifiers



AdaBoost: example

273

Decision boundary of ensemble classifier
after 30 iterations



training_error = 0

Decision boundary is
crazy!!

probably
overfitting

AdaBoost: learning ensemble

274

- Start same weight for all points: $\alpha_i = 1/N$

$$\hat{w}_t = \frac{1}{2} \ln \left(\frac{1 - \text{weighted_error}(f_t)}{\text{weighted_error}(f_t)} \right)$$

- For $t = 1, \dots, T$

- Learn $f_t(\mathbf{x})$ with data weights α_i

- Compute coefficient \hat{w}_t

- Recompute weights α_i

- Normalize weights α_i

$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i) = y_i \\ \alpha_i e^{\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i) \neq y_i \end{cases}$$

- Final model predicts by:

$$\hat{y} = \text{sign} \left(\sum_{t=1}^T \hat{w}_t f_t(\mathbf{x}) \right)$$

$$\alpha_i \leftarrow \frac{\alpha_i}{\sum_{j=1}^N \alpha_j}$$

Boosted decision stumps

275

- Start same weight for all points: $\alpha_i = 1/N$
- For $t = 1, \dots, T$
 - Learn $f_t(\mathbf{x})$: pick decision stump with lowest weighted training error according to α_i
 - Compute coefficient \hat{w}_t
 - Recompute weights α_i
 - Normalize weights α_i

- Final model predicts by:

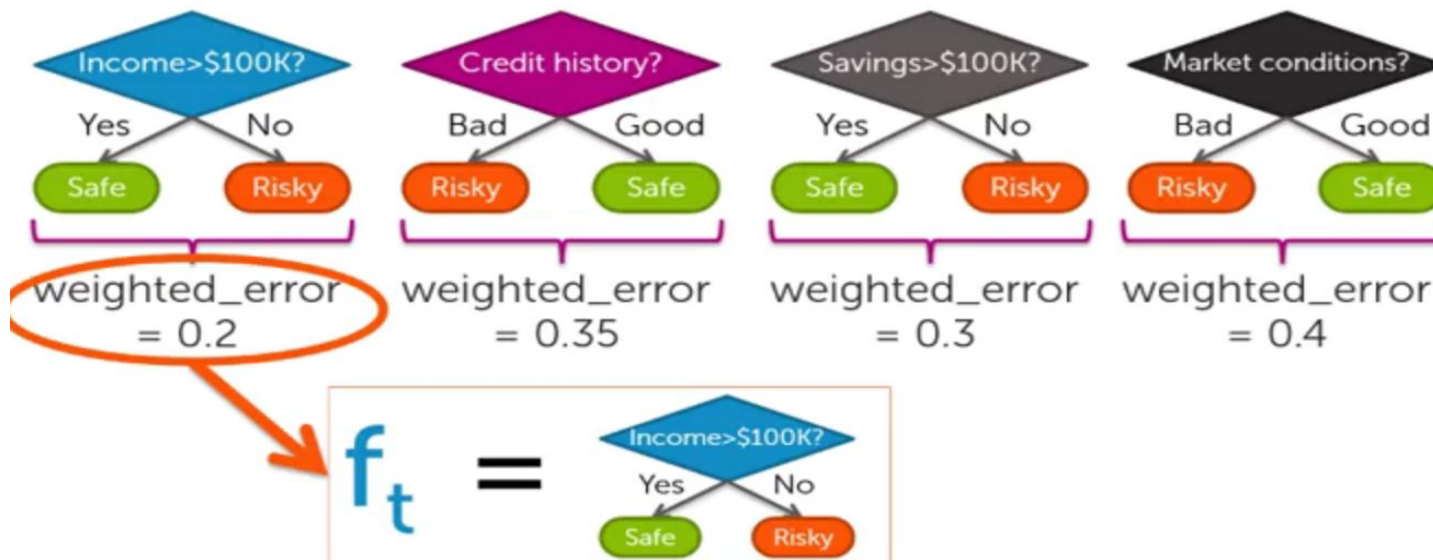
$$\hat{y} = \text{sign} \left(\sum_{t=1}^T \hat{w}_t f_t(\mathbf{x}) \right)$$

Boosted decision stumps

276

Finding best next decision stump $f_t(\mathbf{x})$

Consider splitting on each feature:



$$\hat{W}_t = \frac{1}{2} \ln \left(\frac{1 - \text{weighted_error}(f_t)}{\text{weighted_error}(f_t)} \right) = 0.69$$

Boosted decision stumps

277

- Start same weight for all points: $\alpha_i = 1/N$
- For $t = 1, \dots, T$
 - Learn $f_t(\mathbf{x})$: pick decision stump with lowest weighted training error according to α_i
 - Compute coefficient \hat{w}_t
 - Recompute weights α_i
 - Normalize weights α_i

- Final model predicts by:

$$\hat{y} = \text{sign} \left(\sum_{t=1}^T \hat{w}_t f_t(\mathbf{x}) \right)$$

Boosted decision stumps

278

Updating weights α_i



$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{v}_i} = \alpha_i e^{-0.69} = \alpha_i / 2, & \text{if } f_t(x_i) = y_i \\ \alpha_i e^{\hat{v}_i} = \alpha_i e^{0.69} = 2 \alpha_i, & \text{if } f_t(x_i) \neq y_i \end{cases}$$

Credit	Income	y	\hat{y}	Previous weight α	New weight α
A	\$130K	Safe	Safe	0.5	$0.5/2 = 0.25$
B	\$80K	Risky	Risky	1.5	0.75
C	\$110K	Risky	Safe	1.5	$2 * 1.5 = 3$
A	\$110K	Safe	Safe	2	1
A	\$90K	Safe	Risky	1	2
B	\$120K	Safe	Safe	2.5	1.25
C	\$30K	Risky	Risky	3	1.5
C	\$60K	Risky	Risky	2	1
B	\$95K	Safe	Risky	0.5	1
A	\$60K	Safe	Risky	1	2
A	\$98K	Safe	Risky	0.5	1

Boosting convergence & overfitting

279

Boosting question revisited

"Can a set of weak learners be combined to create a stronger learner?" *Kearns and Valiant (1988)*



Yes! *Schapire (1990)*

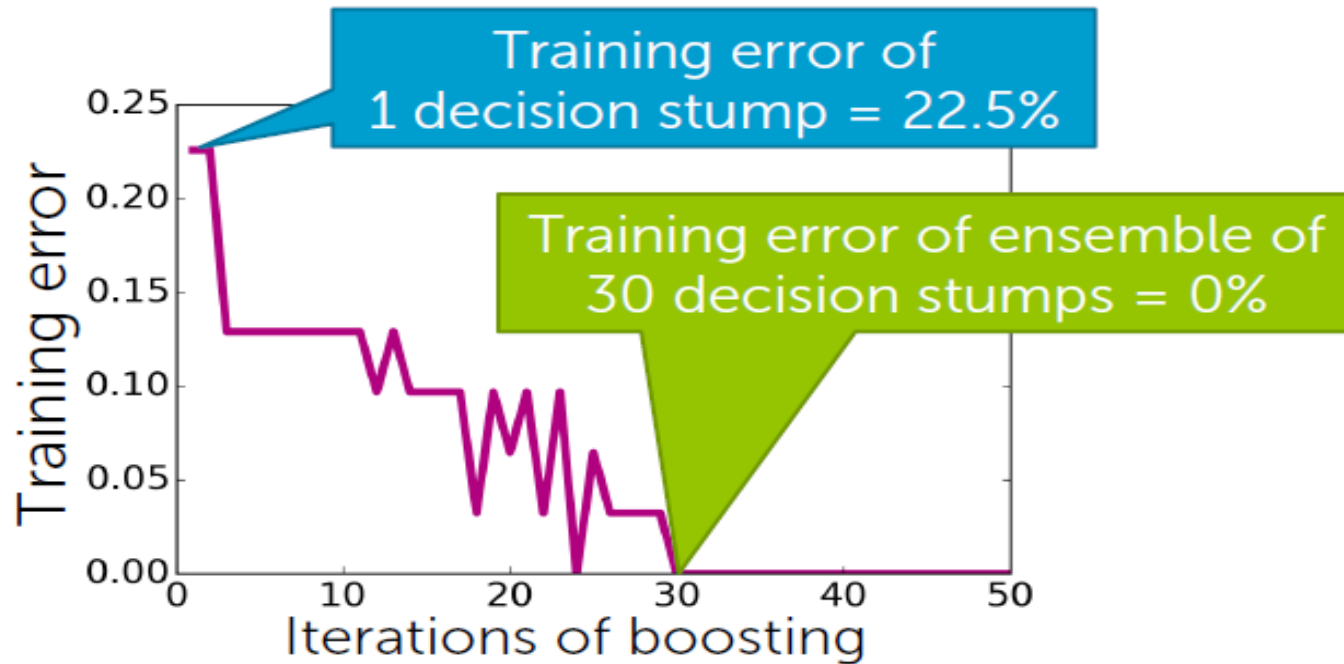


Boosting

Boosting convergence & overfitting

280

After some iterations,
training error of boosting goes to zero!!!



Boosted decision stumps on toy dataset

Boosting convergence & overfitting

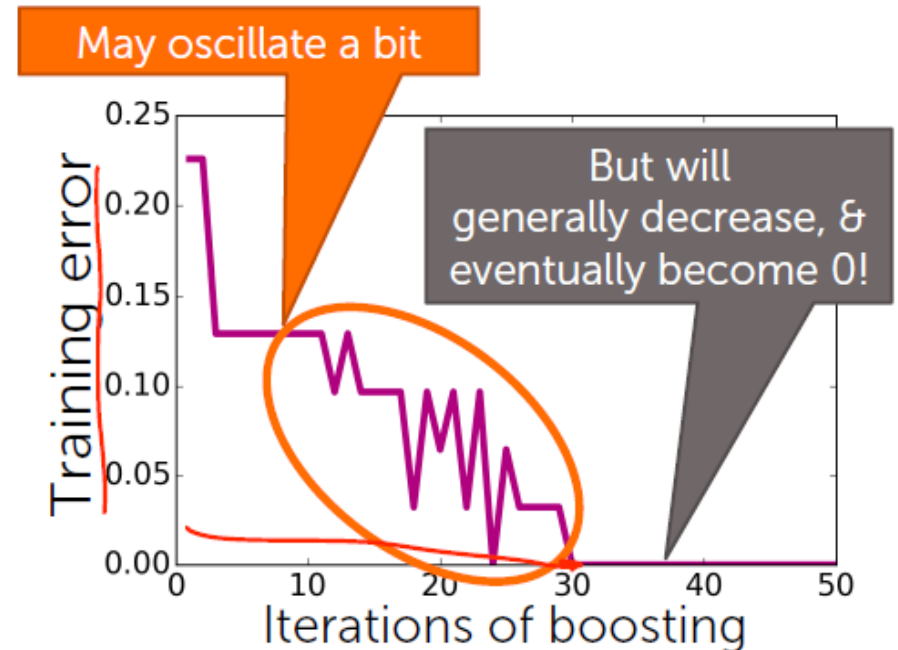
281

AdaBoost Theorem

Under some technical conditions...



Training error of
boosted classifier $\rightarrow 0$
as $T \rightarrow \infty$



Boosting convergence & overfitting

282

Condition of AdaBoost Theorem

Under some technical conditions...



Training error of
boosted classifier $\rightarrow 0$
as $T \rightarrow \infty$

Condition = At every t ,
can find a weak learner with
 $\text{weighted_error}(f_t) < 0.5$



Not always
possible



Nonetheless, boosting often
yields great training error

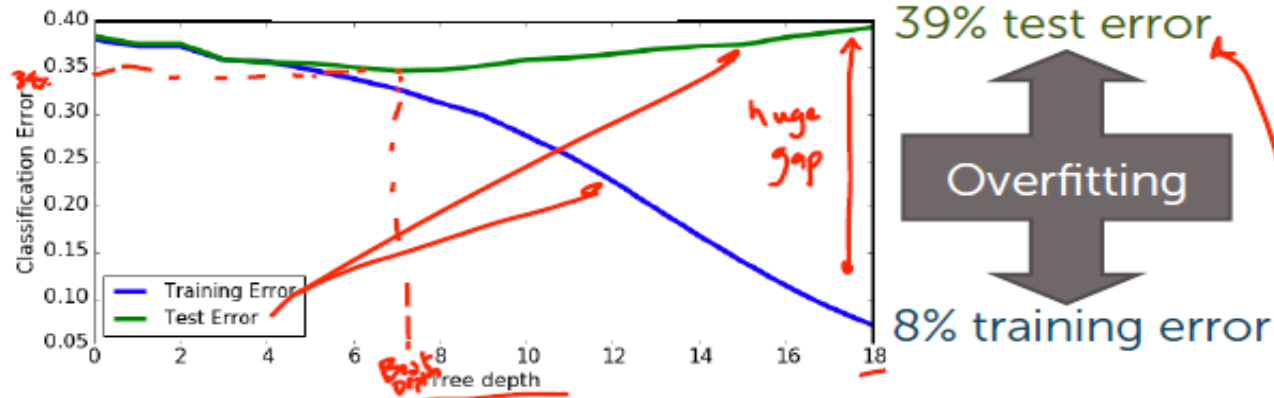
Extreme example:
No classifier can
separate a +1
on top of -1



Example

283

Decision trees on loan data



Boosted decision stumps on loan data

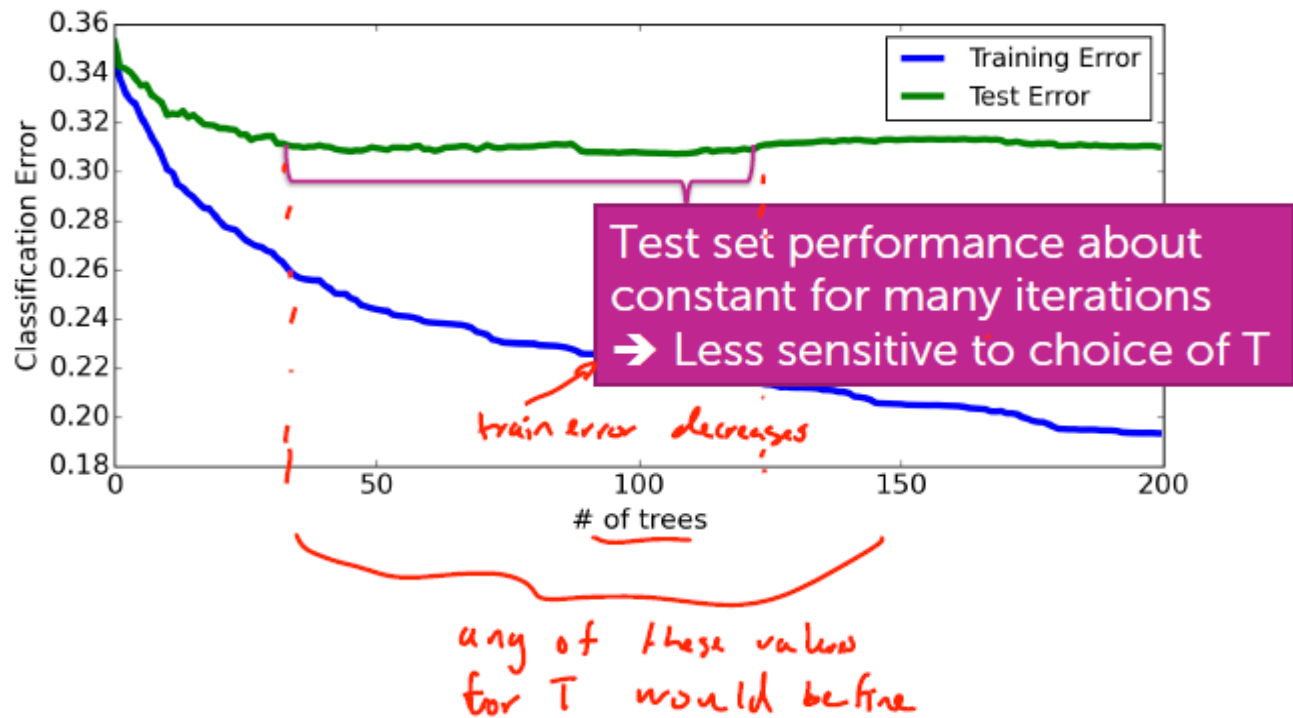


67

Example

284

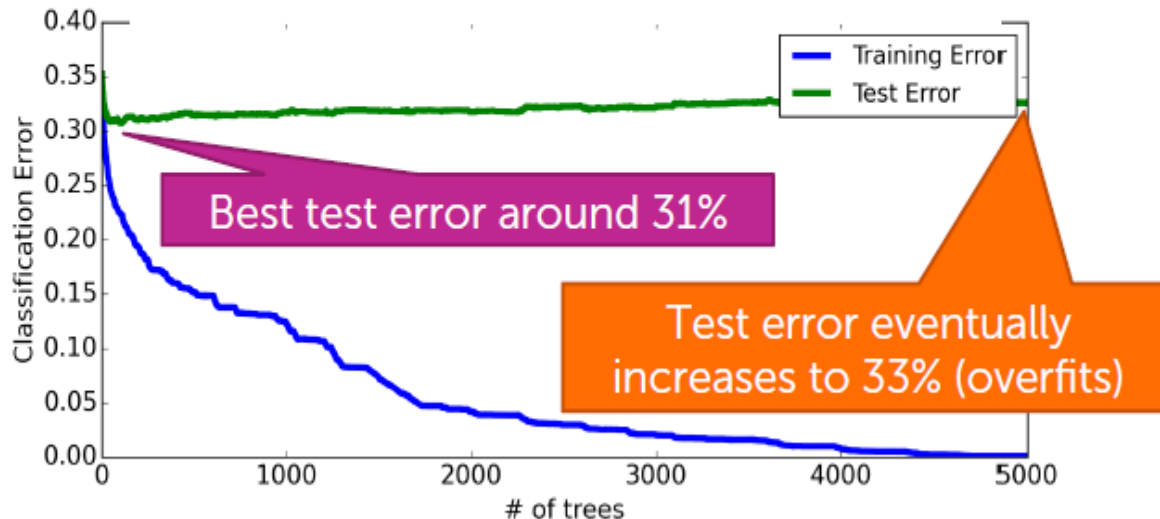
Boosting tends to be robust to overfitting



Example

285

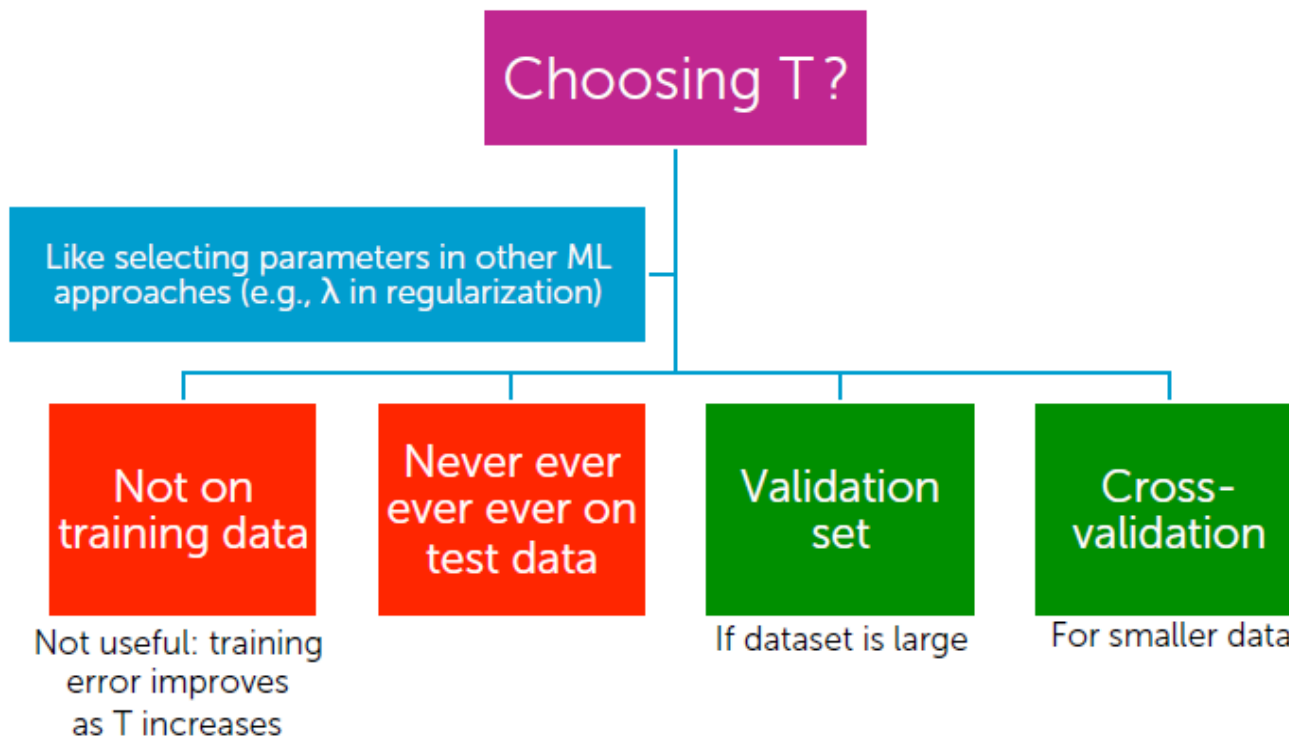
But boosting will eventually overfit,
so must choose max number of components T



Example

286

How do we decide when to stop boosting?



Boosting: summary

287

Variants of boosting and related algorithms

There are hundreds of variants of boosting, most important:

Gradient boosting

- Like AdaBoost, but useful beyond basic classification

Many other approaches to learn ensembles, most important:

Random forests

- Bagging: Pick random subsets of the data
 - Learn a tree in each subset
 - Average predictions
- Simpler than boosting & easier to parallelize
- Typically higher error than boosting for same number of trees (# iterations T)

Boosting: summary

288

Impact of boosting (spoiler alert... *HUGE IMPACT*)

Amongst most useful
ML methods ever created

Extremely useful in
computer vision

- Standard approach for face detection, for example

Used by **most winners** of
ML competitions
(Kaggle, KDD Cup,...)

- Malware classification, credit fraud detection, ads click through rate estimation, sales forecasting, ranking webpages for search, Higgs boson detection,...

Most deployed ML systems
use model ensembles

- Coefficients chosen manually, with boosting, with bagging, or others

What you can do now

289

- Identify notion ensemble classifiers
- Formalize ensembles as the weighted combination of simpler classifiers
- Outline the boosting framework – sequentially learn classifiers on weighted data
- Describe the AdaBoost algorithm
 - Learn each classifier on weighted data
 - Compute coefficient of classifier
 - Recompute data weights
 - Normalize weights
- Implement AdaBoost to create an ensemble of decision stumps
- Discuss convergence properties of AdaBoost & how to pick the maximum number of iterations T