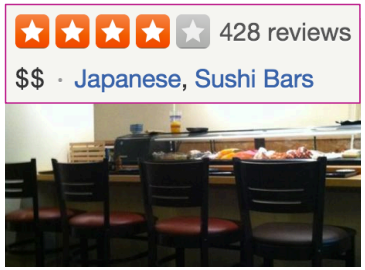
Linear classifiers:

Logistic regression

Predicting sentiment by topic:
An intelligent restaurant review system

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



Positive reviews not positive about everything

Sample review:

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

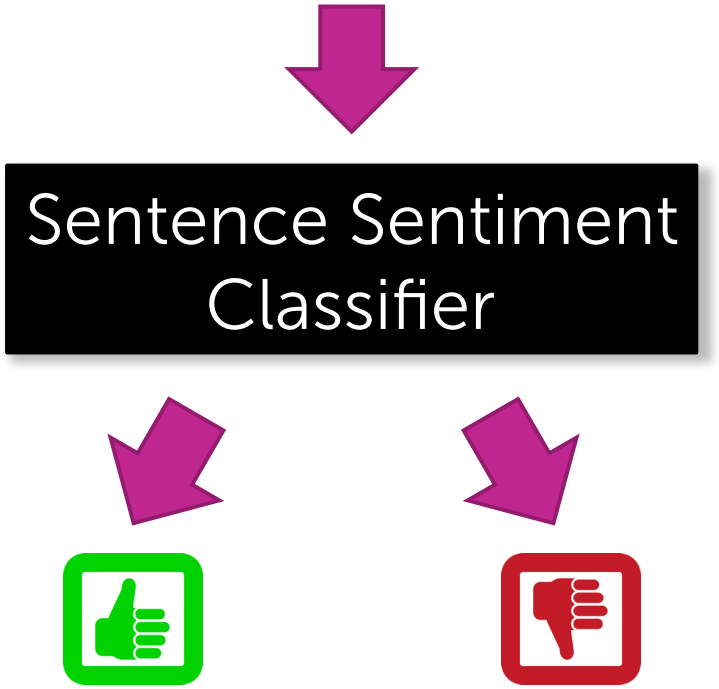
My wife tried their ramen and it was pretty forgettable.

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

Experience

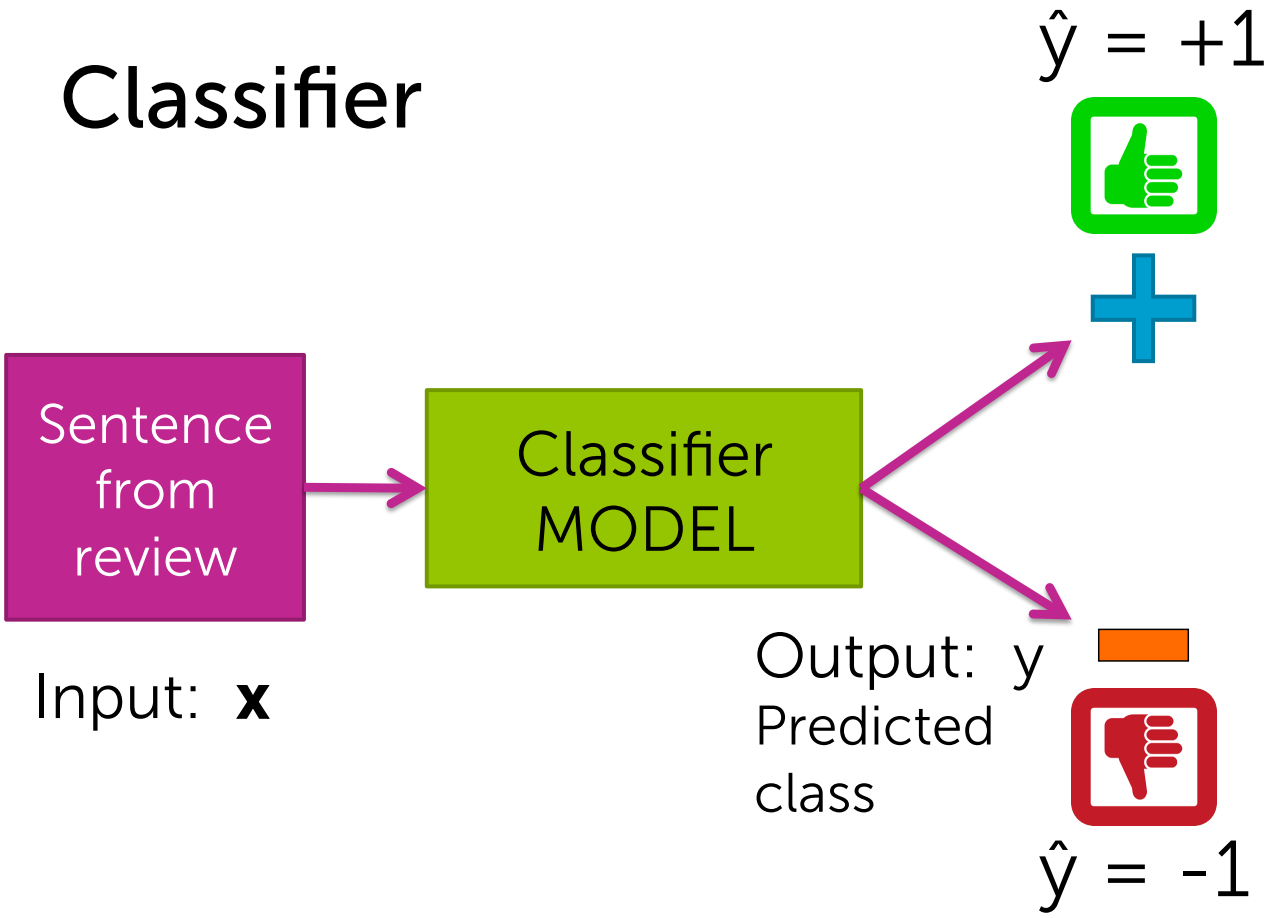
Classifying sentiment of review

Easily best sushi in city.



Linear classifier: Intuition

Classifier



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

A (linear) classifier

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

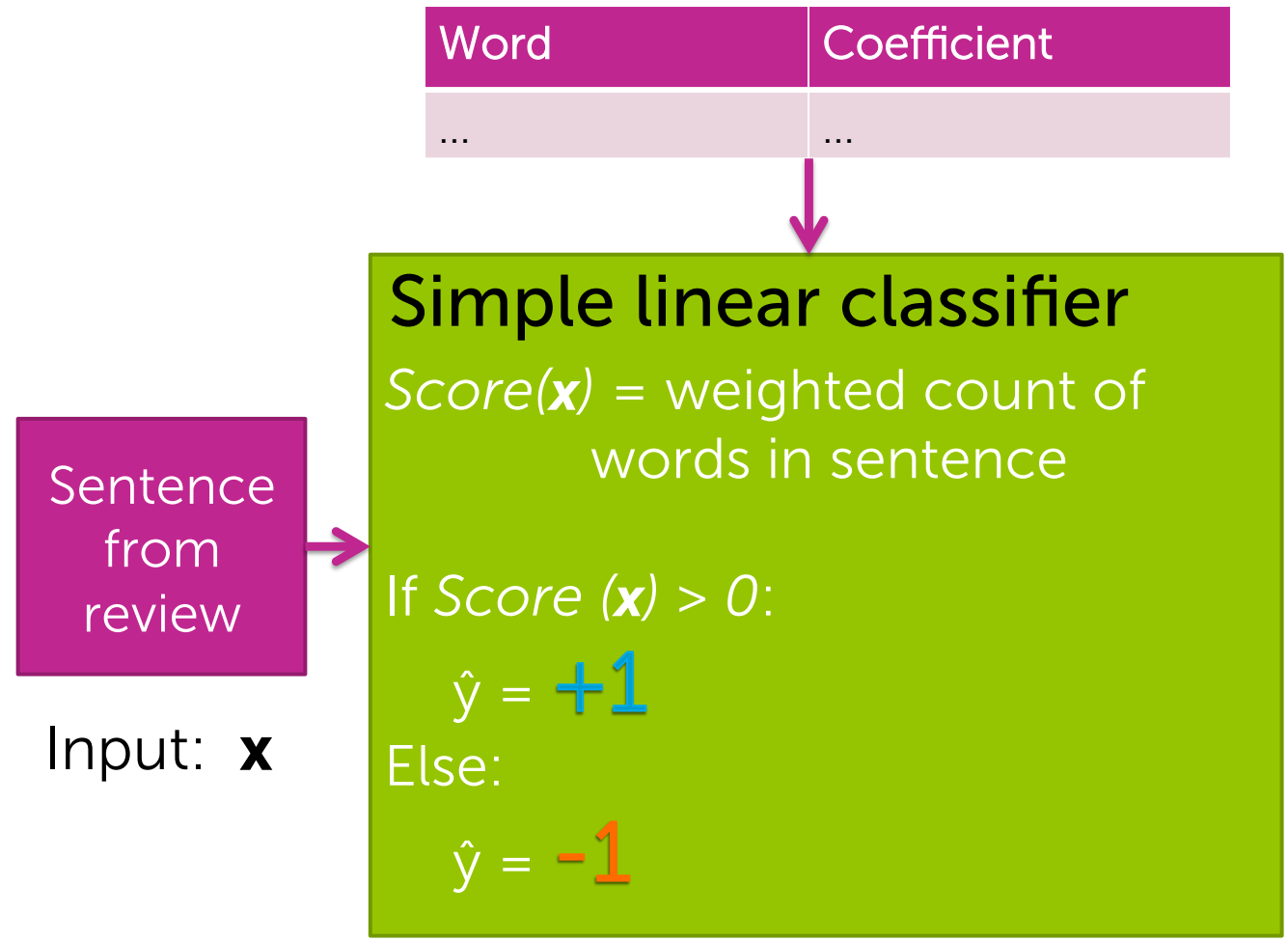
Word	Coefficient
good	1.0
great	1.5
awesome	2.7
bad	-1.0
terrible	-2.1
awful	-3.3
restaurant, the, we, where, …	0.0
…	…

Scoring a sentence

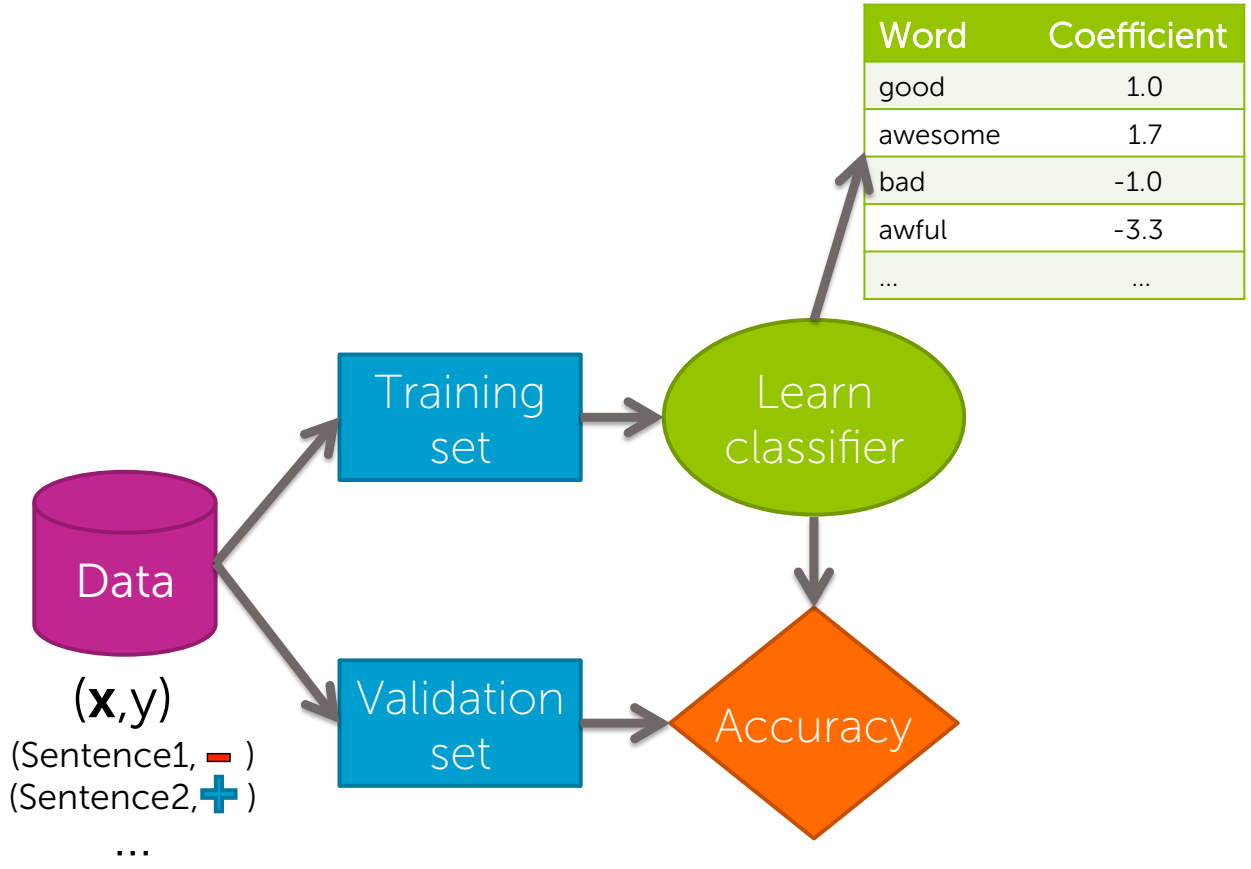
Word	Coefficient
good	1.0
great	1.2
awesome	1.7
bad	-1.0
terrible	-2.1
awful	-3.3
restaurant, the, we, where, …	0.0
…	…

Input $\mathbf{x}_i$:
Sushi was great,
the food was awesome,
but the service was terrible.

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


Word
Coefficient
...
...
Simple linear classifier
Score(x) = weighted count of words in sentence
If Score (x) > 0:
ŷ = +1
Else:
ŷ = -1
Sentence from review
Input: x

Training a classifier = Learning the coefficients

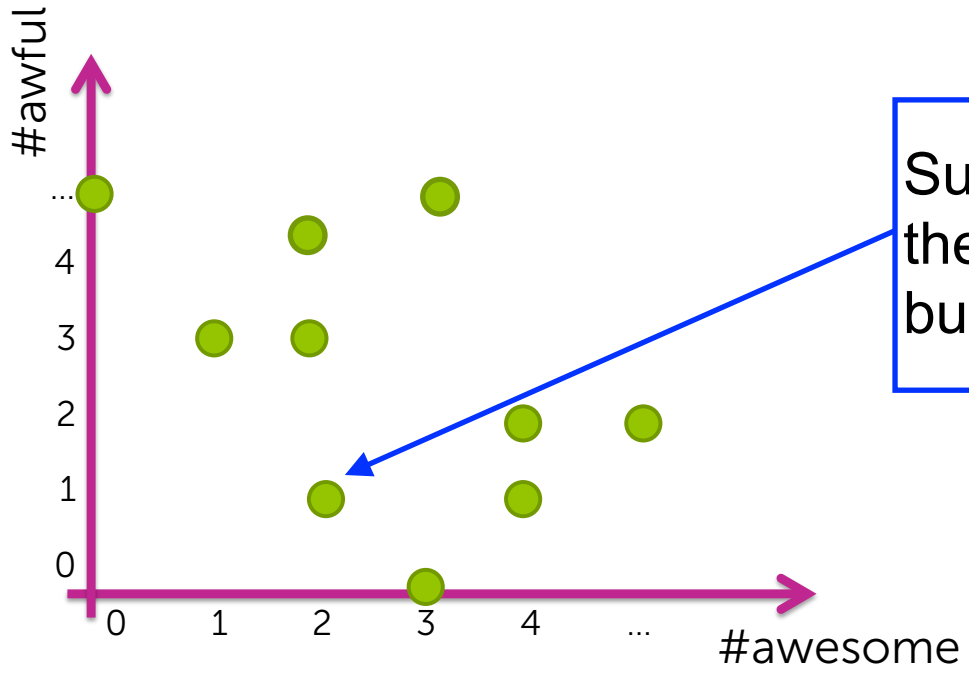


Decision boundaries

Suppose only two words had non-zero coefficient

Word	Coefficient
#awesome	1.0
#awful	-1.5

Score(x) = 1.0 #awesome – 1.5 #awful

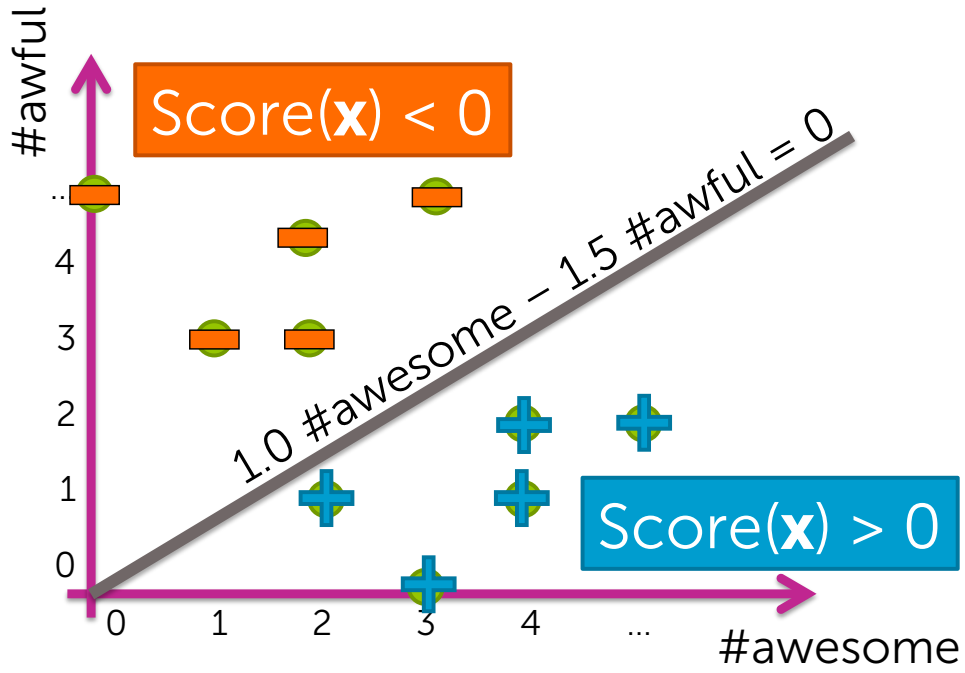


Sushi was awesome
the food was awesome
but the service was awful.

Decision boundary example

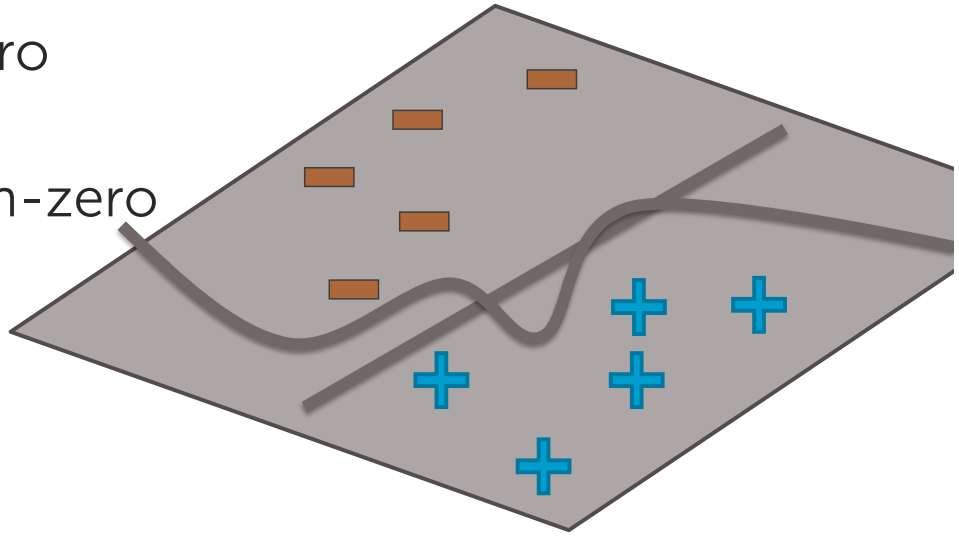
Word	Coefficient
#awesome	1.0
#awful	-1.5

Score(x) = 1.0 #awesome – 1.5 #awful

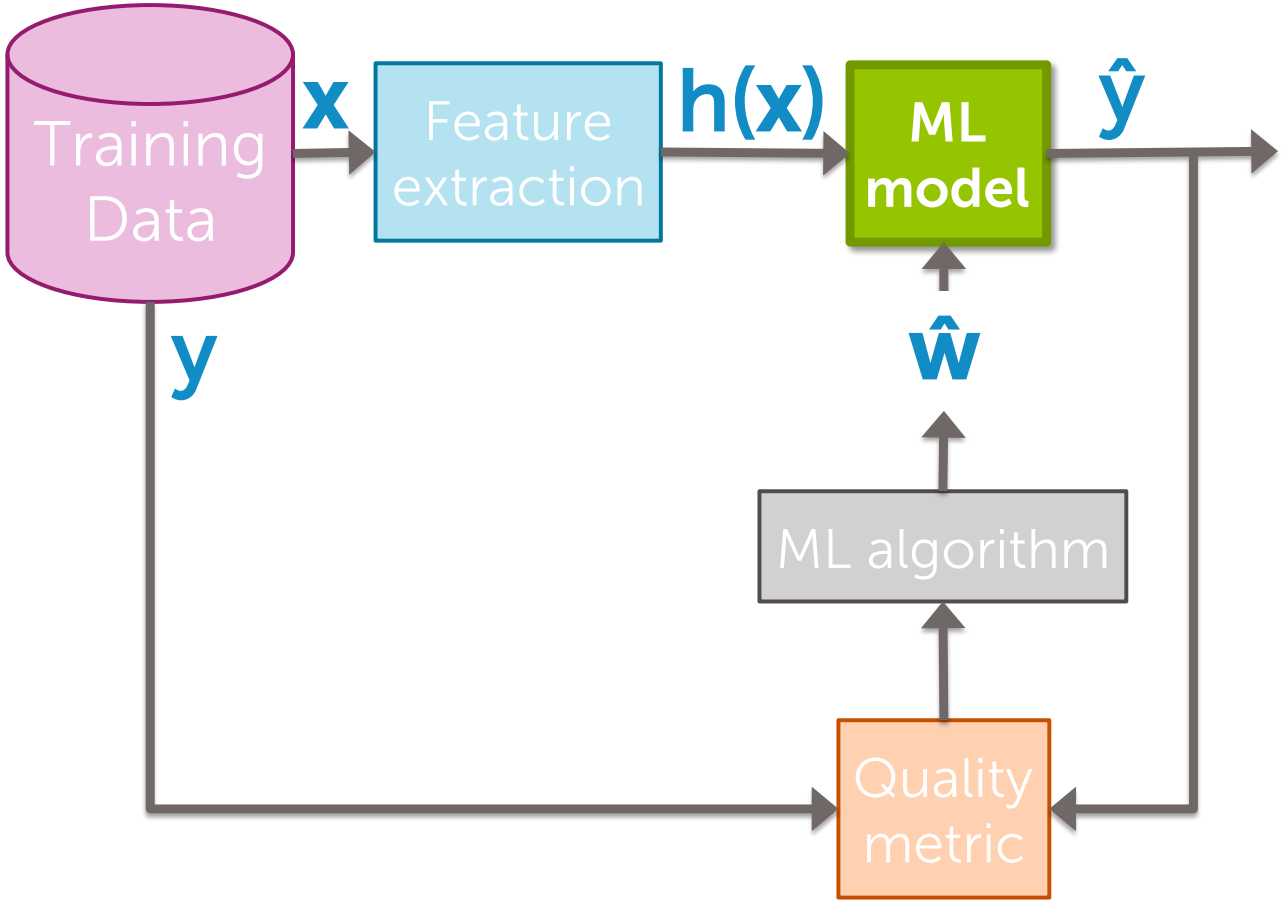


Decision boundary separates positive & negative predictions

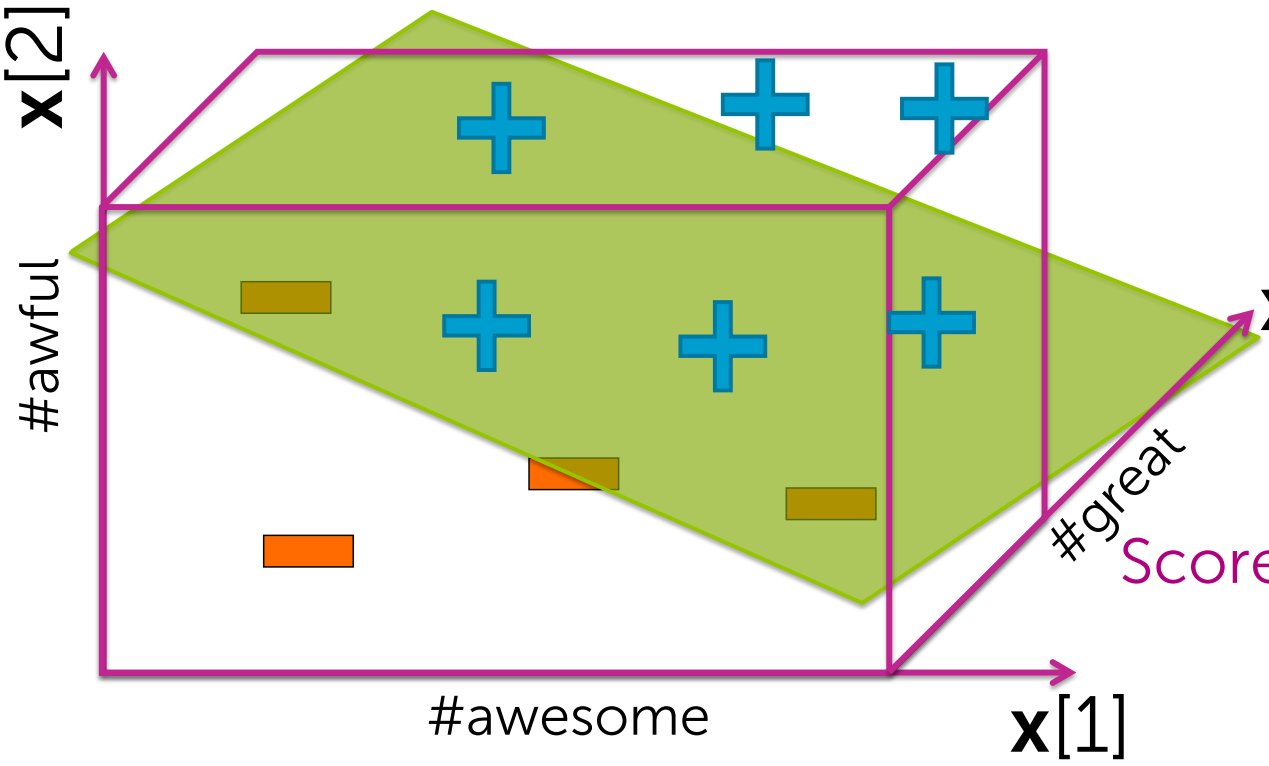
- For linear classifiers:
  - When 2 coefficients are non-zero
    ➔ line
  - When 3 coefficients are non-zero
    ➔ plane
  - When many coefficients are non-zero
    ➔ hyperplane
- For more general classifiers
  ➔ more complicated shapes



Linear classifier: Model


Training Data
x
Feature extraction
h(x)
ML model
ŷ
y
ŵ
ML algorithm
Quality metric

Coefficients of classifier



Score(**x**) = w_0
+ w_1 #awesome
+ w_2 #awful
+ w_3 #great

General notation

Output: y ← {-1,+1}

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

← d-dim vector

Notational conventions:

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

Simple hyperplane

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

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

feature 1 = 1

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

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

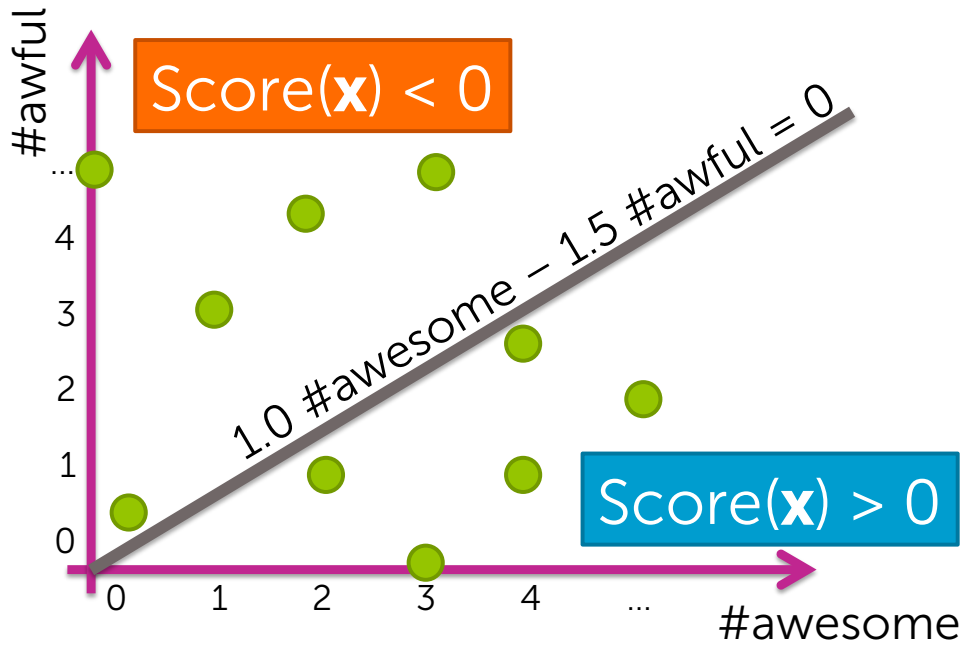
…

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

Decision boundary: effect of changing coefficients

Input	Coefficient	Value
	w_0	0.0
#awesome	w_1	1.0
#awful	w_2	-1.5

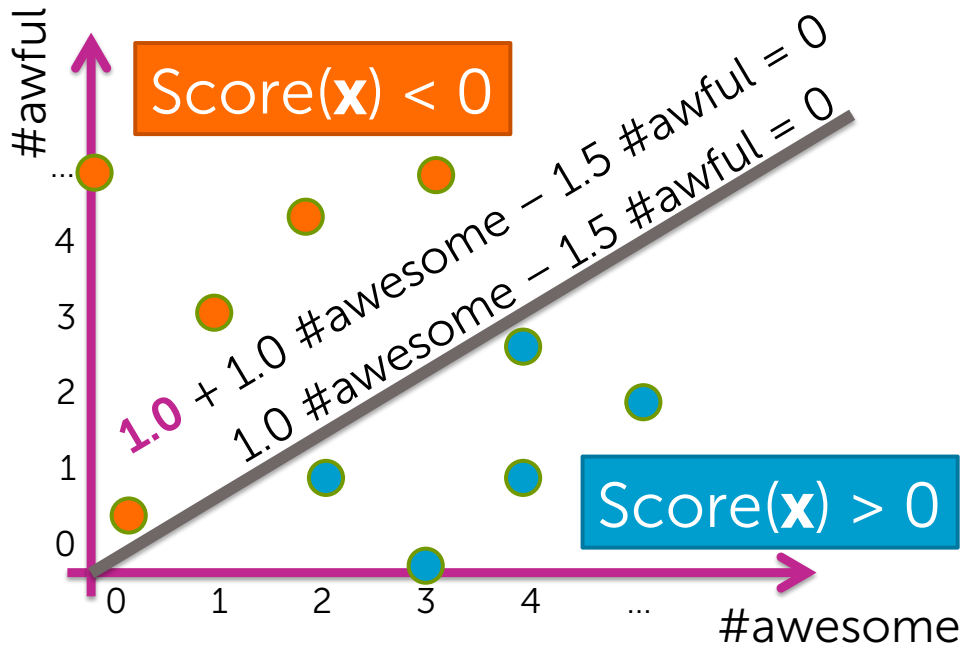
Score(x) = 1.0 #awesome – 1.5 #awful



Decision boundary: effect of changing coefficients

Input	Coefficient	Value
	w_0	**1.0**
#awesome	w_1	1.0
#awful	w_2	-1.5

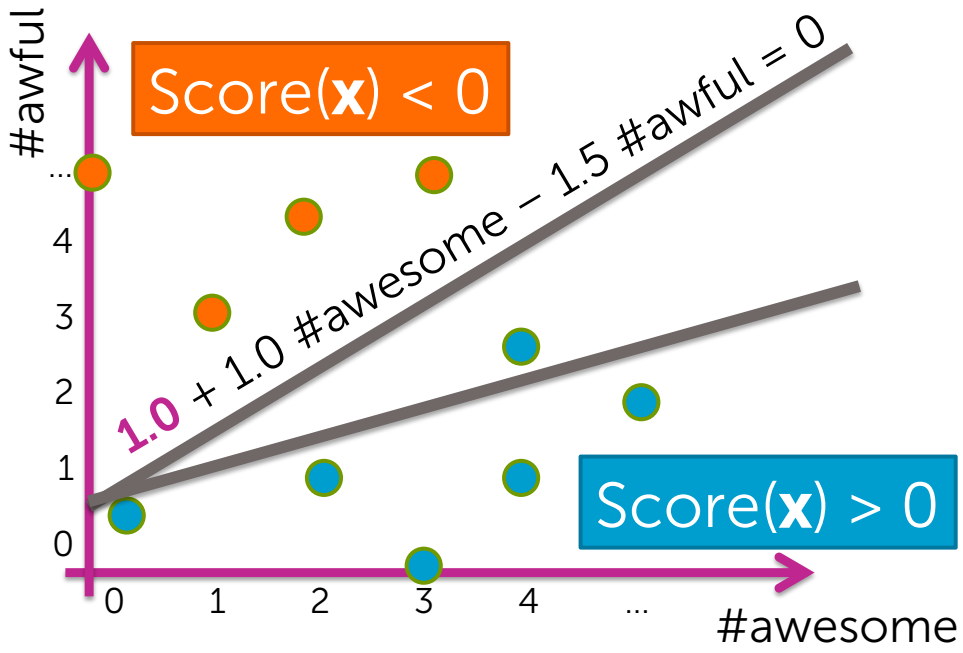
Score(x) = **1.0** #awesome – 1.5 #awful



Decision boundary: effect of changing coefficients

Input	Coefficient	Value
	w_0	**1.0**
#awesome	w_1	1.0
#awful	w_2	**–3.0**

Score(x) = **1.0** + 1.0 #awesome **– 3.0** #awful



More generic features… D-dimensional hyperplane

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

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

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

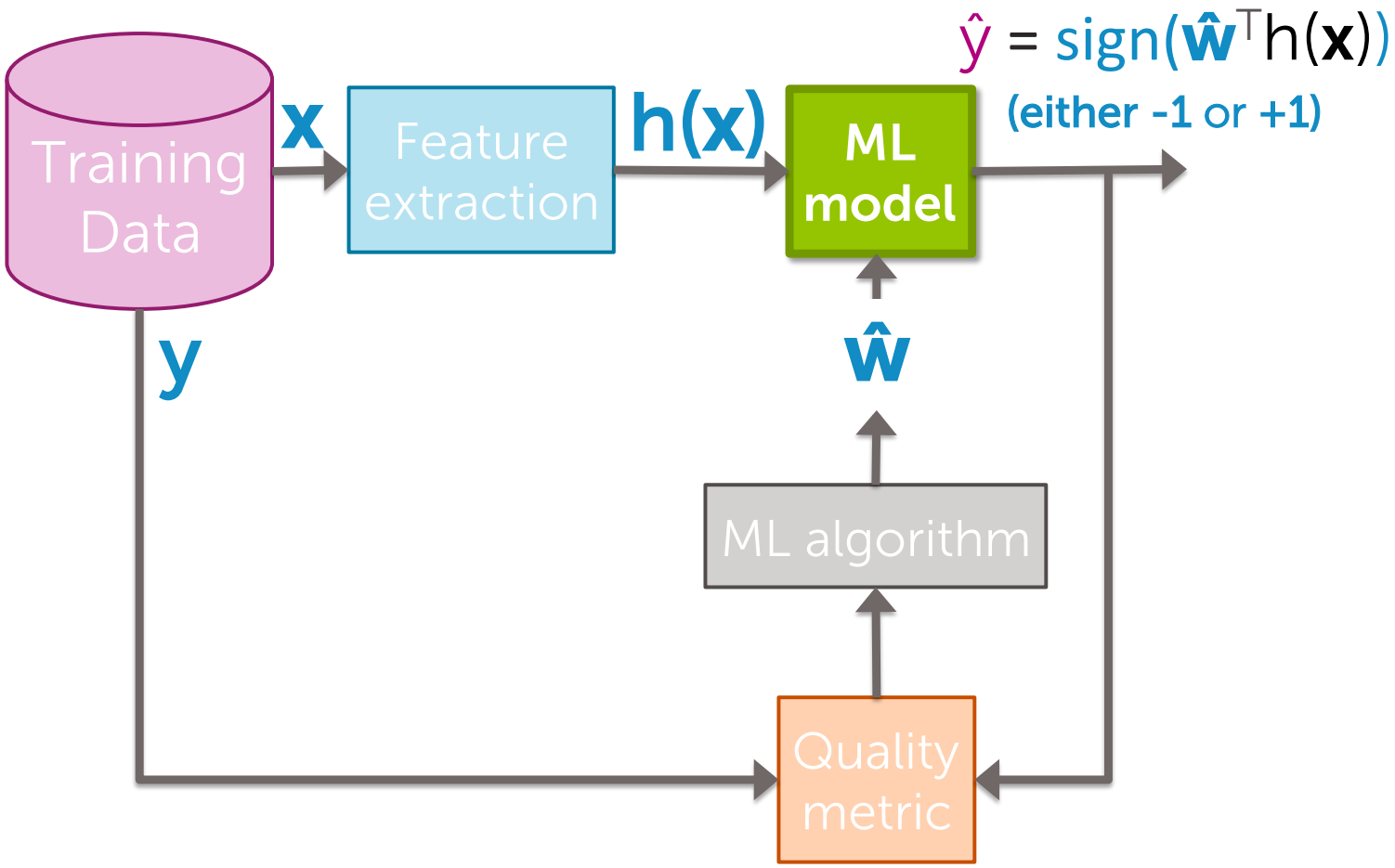
feature 1 = $h_0(\mathbf{x})$ … e.g., 1

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

feature 3 = $h_2(\mathbf{x})$ … e.g., $\mathbf{x}$[2] = #awful
or, log($\mathbf{x}$[7]) $\mathbf{x}$[2] = log(#bad) x #awful
or, tf-idf("awful")

…

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

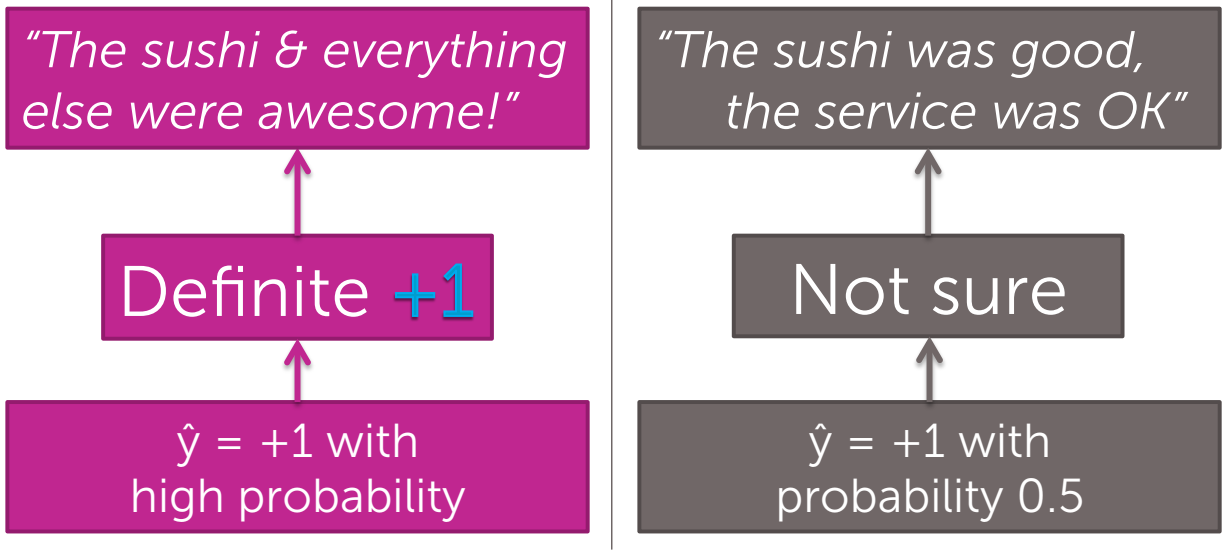

$\hat{y} = \text{sign}(\hat{\mathbf{w}}^T h(\mathbf{x}))$
(either -1 or +1)
Training Data
x
Feature extraction
h(x)
ML model
y
$\hat{\mathbf{w}}$
ML algorithm
Quality metric

Are you sure about the prediction?

Class probability

How confident is your prediction?

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



Basics of probabilities – quick review

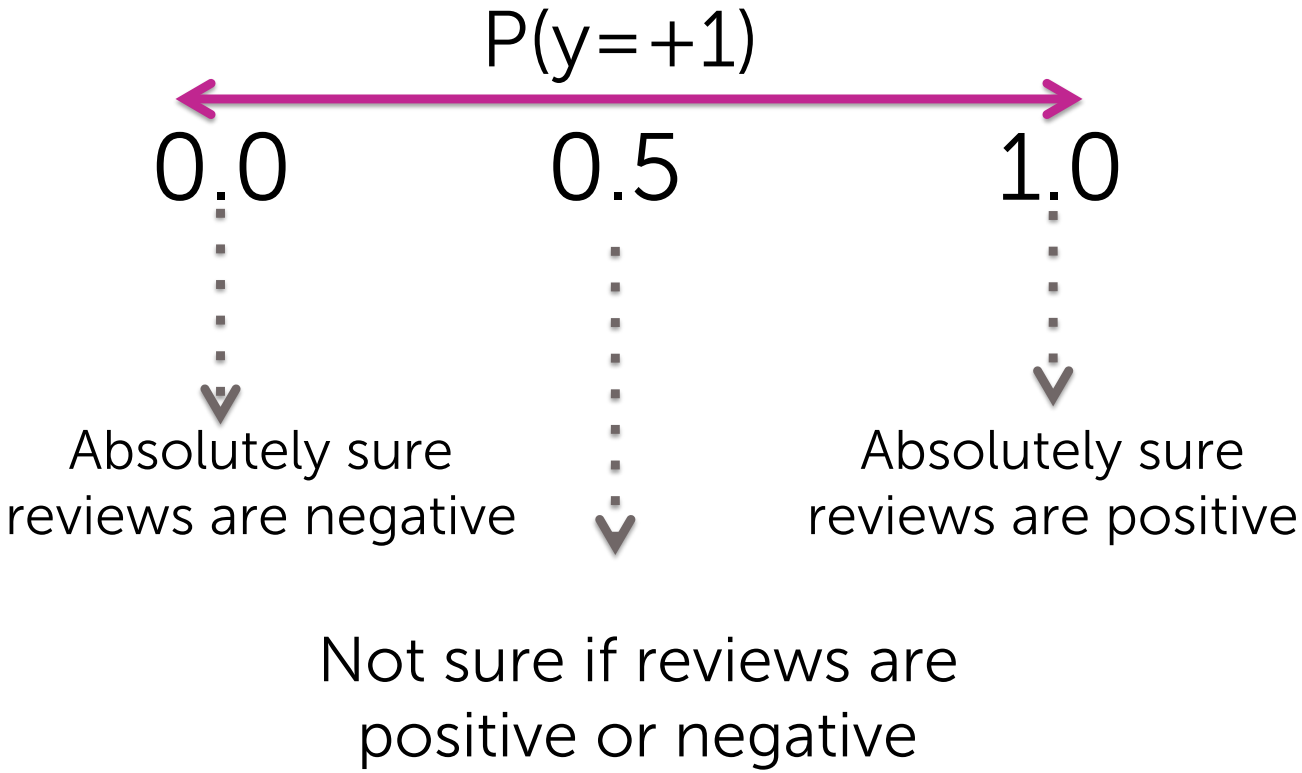
Basic probability

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



Key properties of probabilities

Property	Two class (e.g., y is +1 or -1)	Multiple classes (e.g., y is dog, cat or bird)
Probabilities always between 0 & 1		
Probabilities sum up to 1		

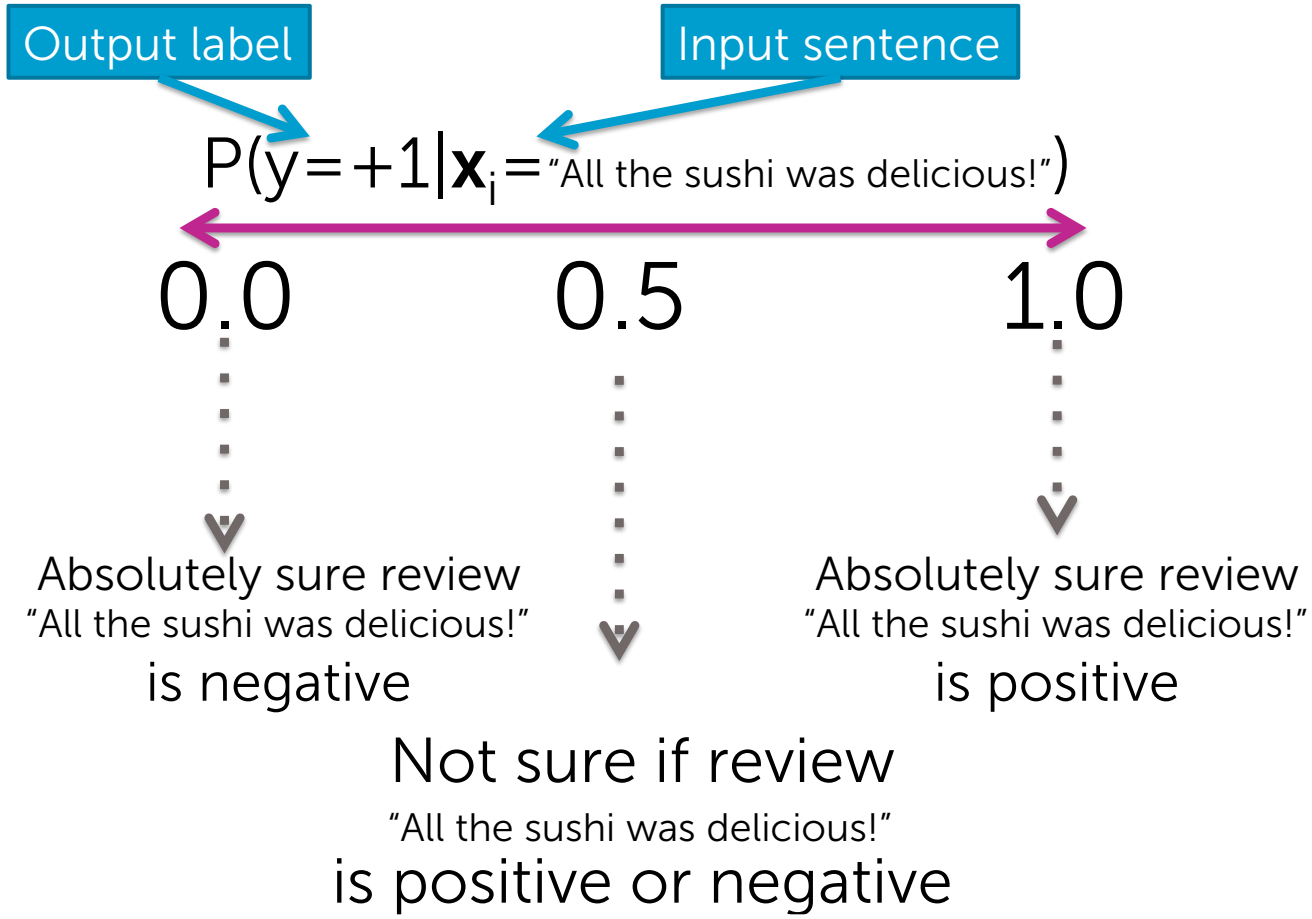
Conditional probability

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

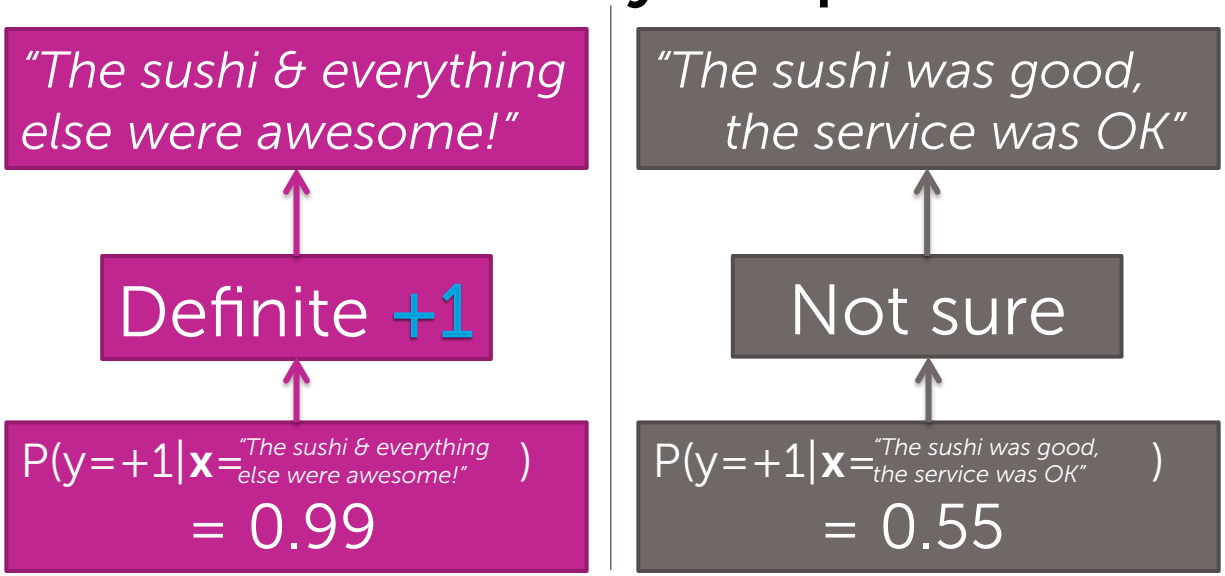


Key properties of conditional probabilities

Property	Two class (e.g., y is +1 or -1, $\mathbf{x}_i$ is review text)	Multiple classes (e.g., y is dog, cat or bird, $\mathbf{x}_i$ is image)
Conditional probabilities always between 0 & 1		
Conditional probabilities sum up to 1 over y, but not over **x**		

Using probabilities in classification

How confident is your prediction?



Many classifiers provide a degree of certainty:

Output label → $P(y|\mathbf{x})$ ← Input sentence

Extremely useful in practice

Goal: Learn conditional probabilities from data

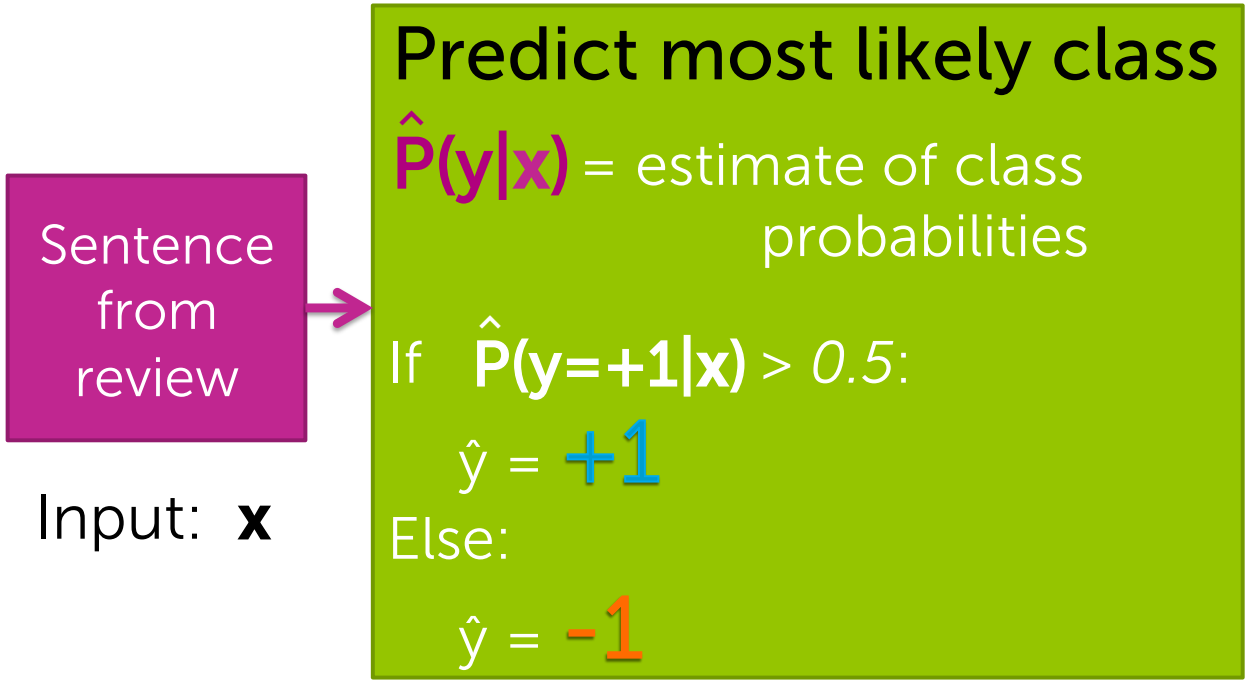
Training data: N observations $(\mathbf{x}_i, y_i)$

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

Optimize **quality metric** on training data

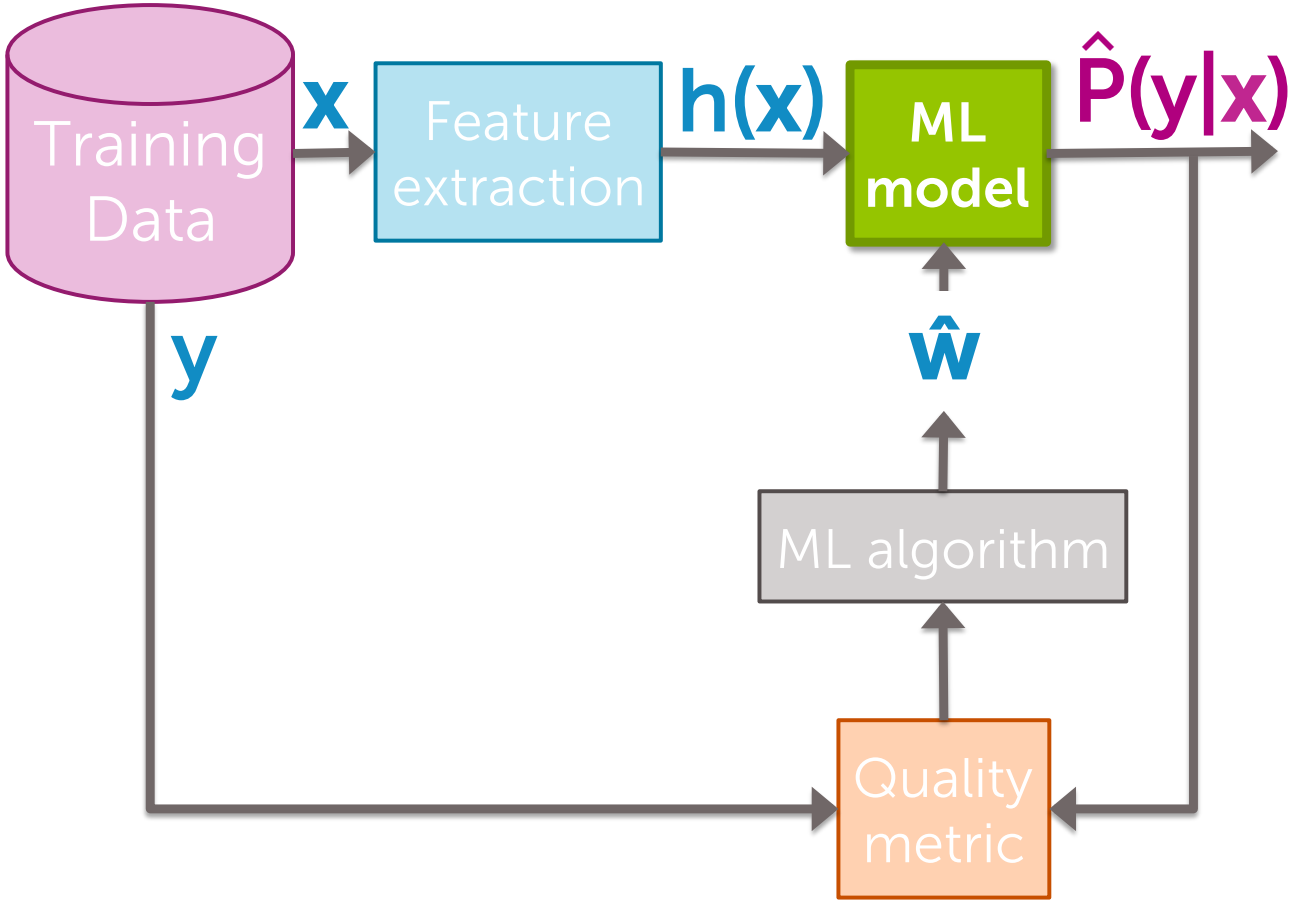
Find best model $\hat{P}$ by finding best $\hat{\mathbf{w}}$

Useful for predicting $\hat{y}$



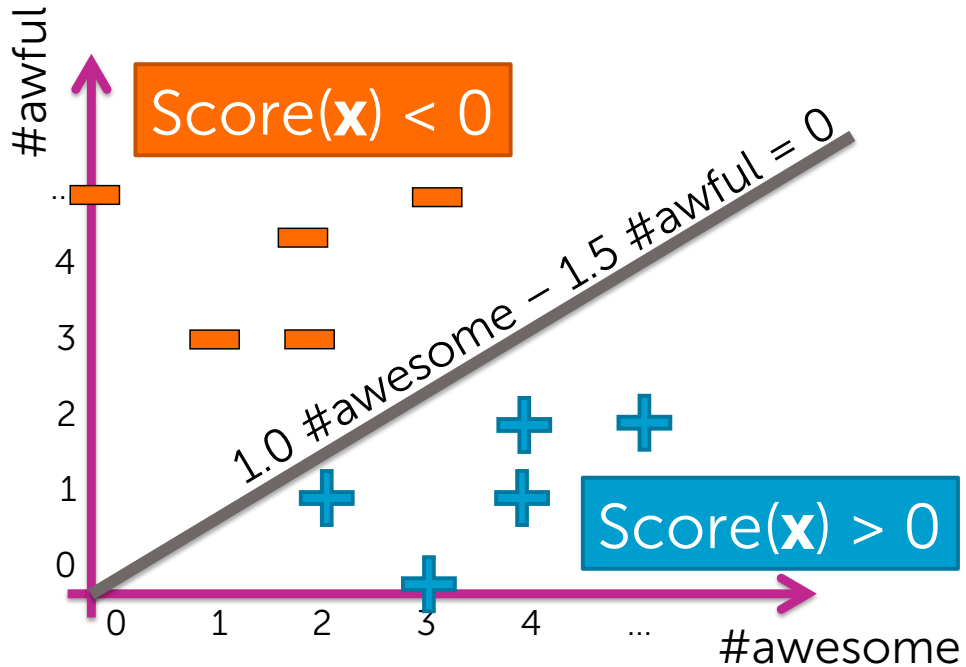
- Estimating $\hat{P}(y|x)$ improves **interpretability**:
  - Predict ŷ = +1 **and** tell me how sure you are

Predicting class probabilities with generalized linear models


Training Data
x
Feature extraction
h(x)
ML model
$\hat{P}(y|x)$
y
$\hat{w}$
ML algorithm
Quality metric

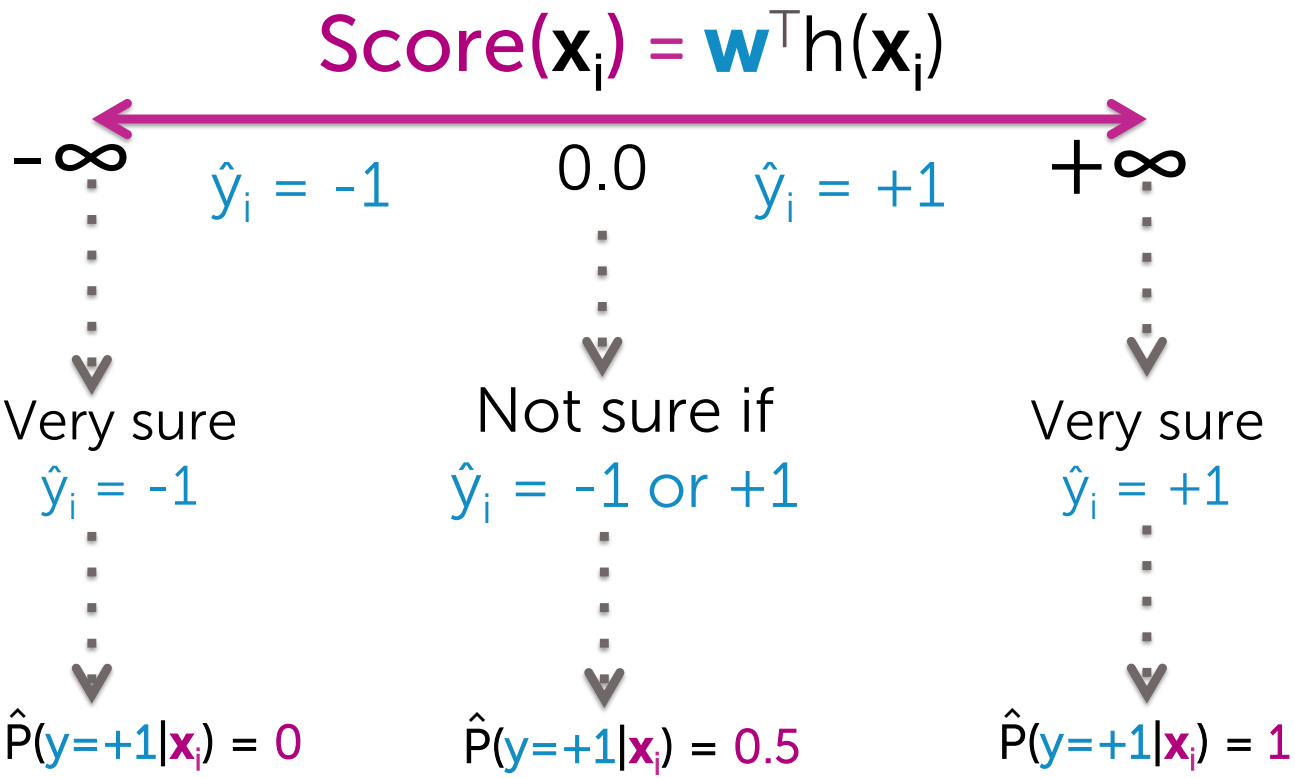
Thus far, we focused on decision boundaries

$$\text{Score}(\mathbf{x}_i) = w_0\, h_0(\mathbf{x}_i) + w_1\, h_1(\mathbf{x}_i) + \ldots + w_D\, h_D(\mathbf{x}_i)$$
$$= \mathbf{w}^T h(\mathbf{x}_i)$$

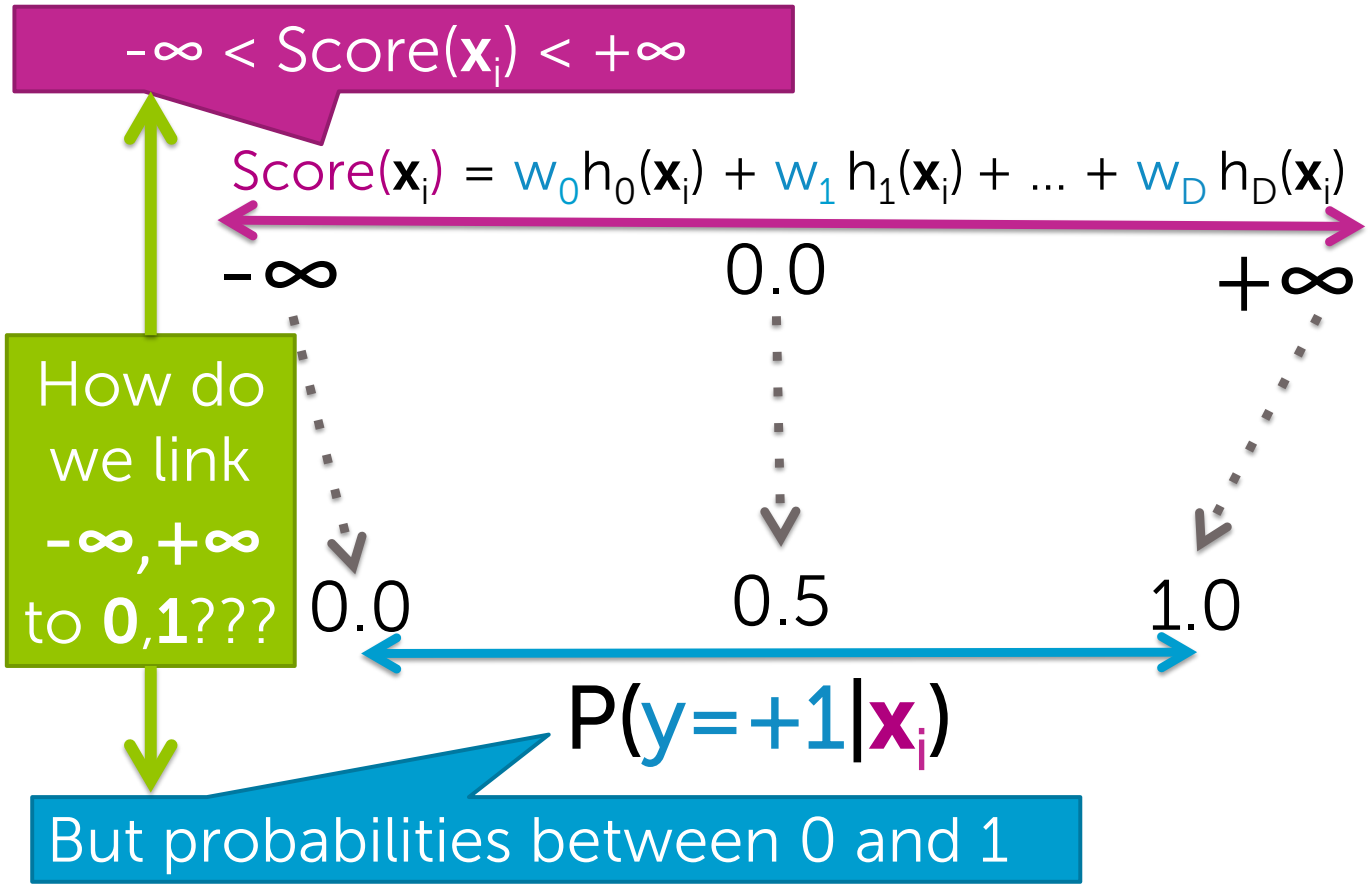


Relate $\text{Score}(\mathbf{x}_i)$ to $\hat{P}(y=+1|\mathbf{x},\hat{\mathbf{w}})$?

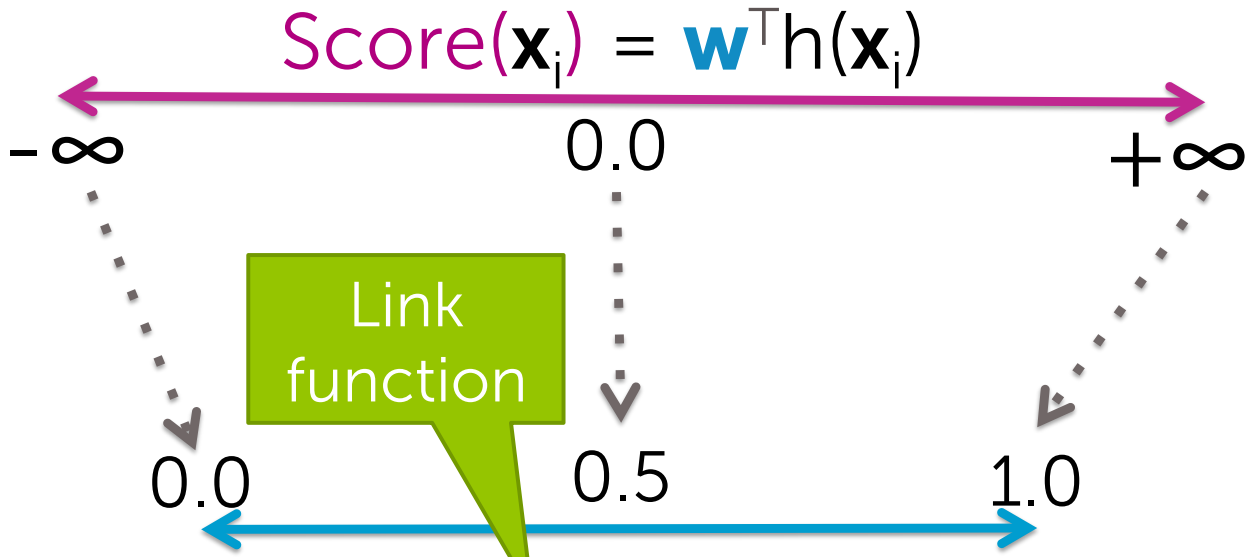
Interpreting Score($\mathbf{x}_i$)



Why not just use regression to build classifier?



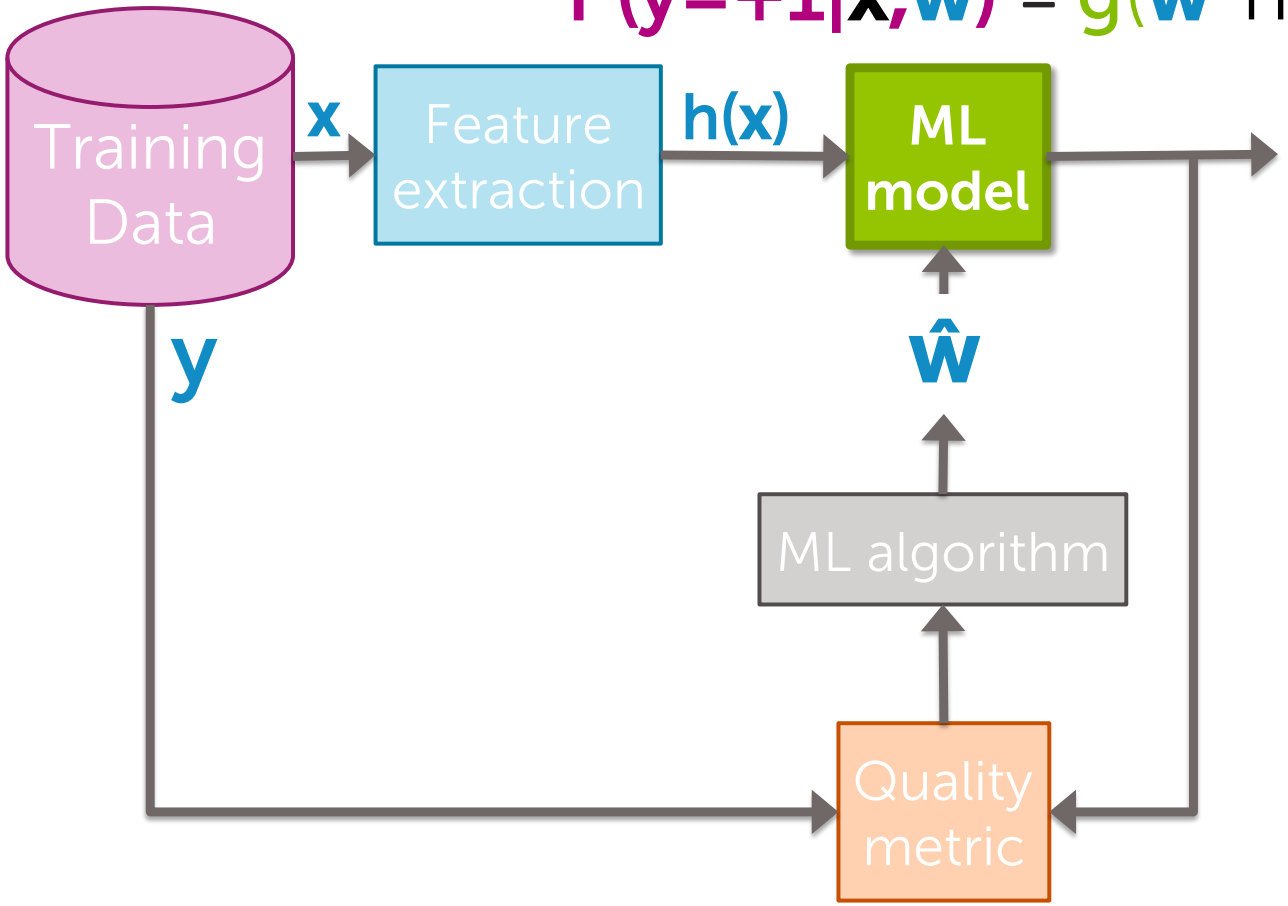
Link function: squeeze real line into [0,1]



$\hat{P}(y=+1|\mathbf{x}_i) = g(\mathbf{w}^T h(\mathbf{x}_i))$

Generalized linear model

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



Logistic regression classifier:

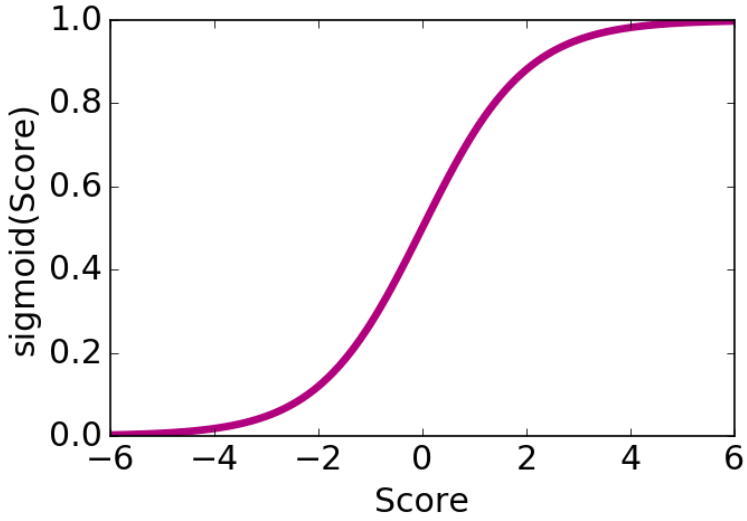
linear score with

logistic link function

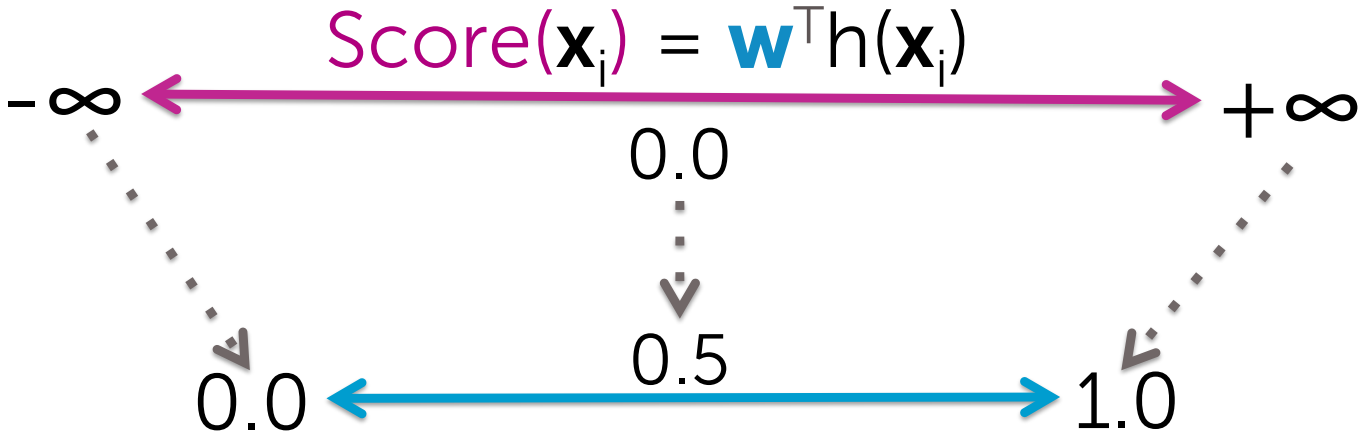
Logistic function (sigmoid, logit)

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

Score	$-\infty$	-2	0.0	+2	$+\infty$
sigmoid(Score)					



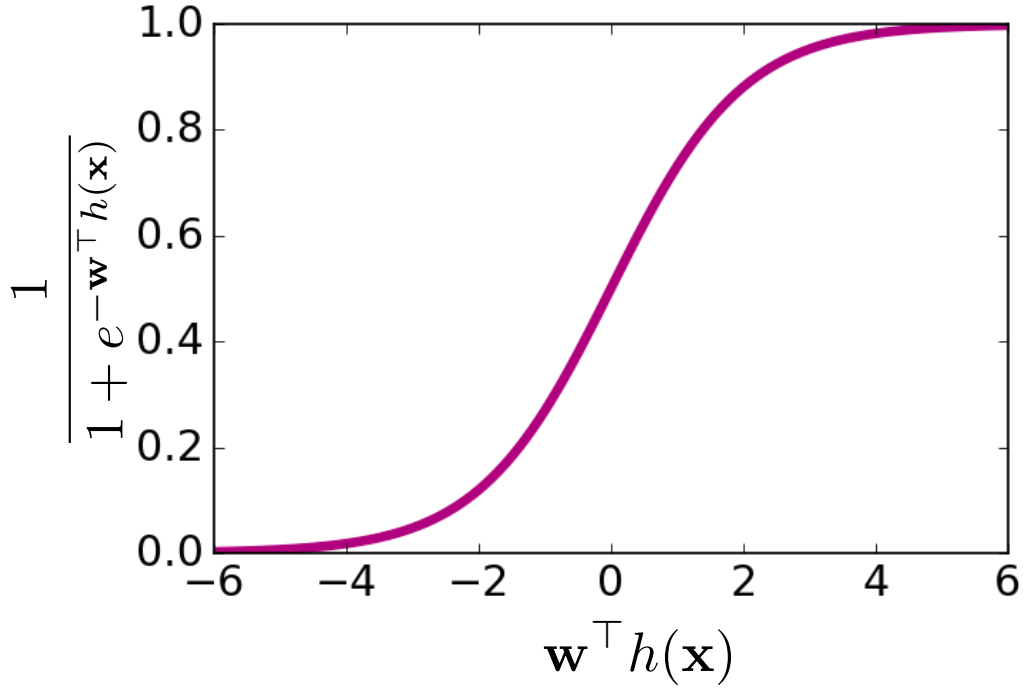
Logistic regression model



$P(y=+1|\mathbf{x}_i,\mathbf{w}) = \text{sigmoid}(\text{Score}(\mathbf{x}_i))$

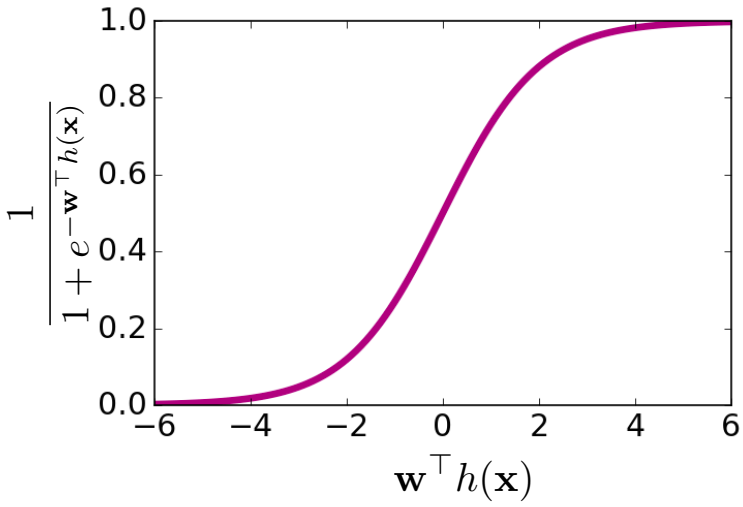
Understanding the logistic regression model

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



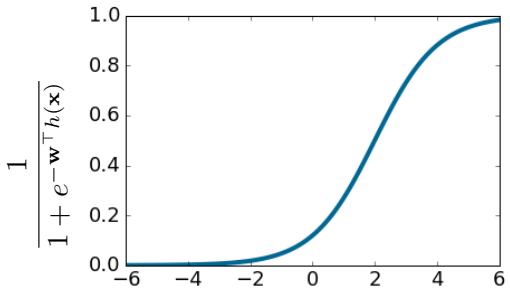
Score($\mathbf{x}_i$)	P(y=+1|$\mathbf{x}_i$,$\mathbf{w}$)

Logistic regression ➔
Linear decision boundary



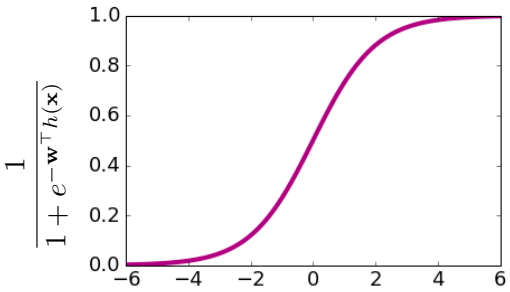
Effect of coefficients on logistic regression model

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



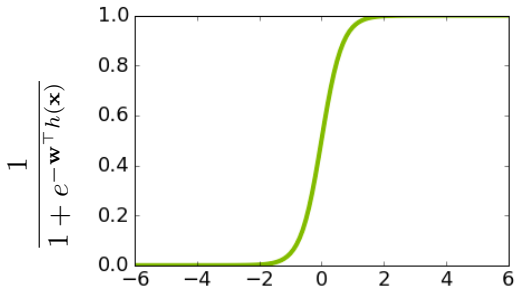
#awesome - #awful

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



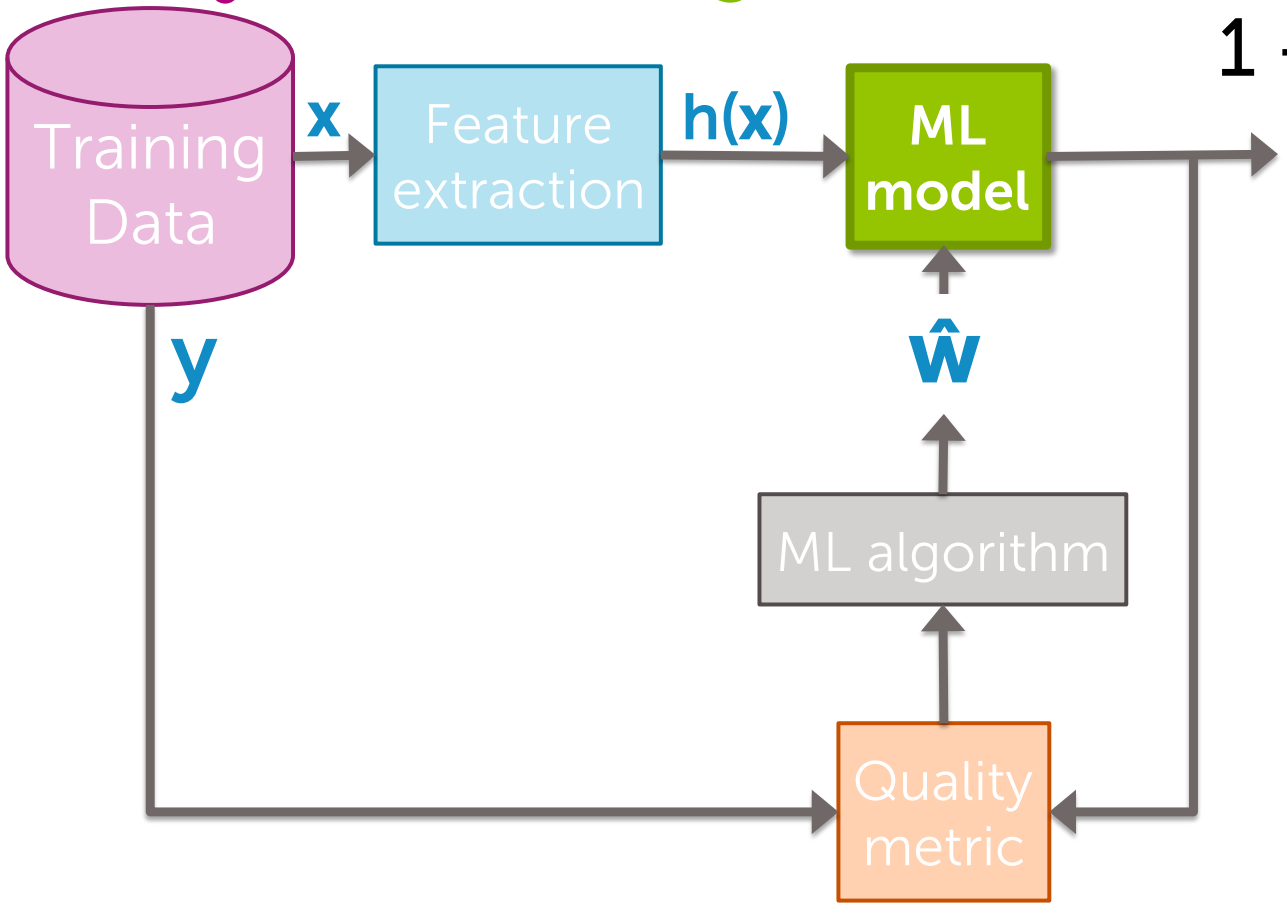
#awesome - #awful

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



#awesome - #awful

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

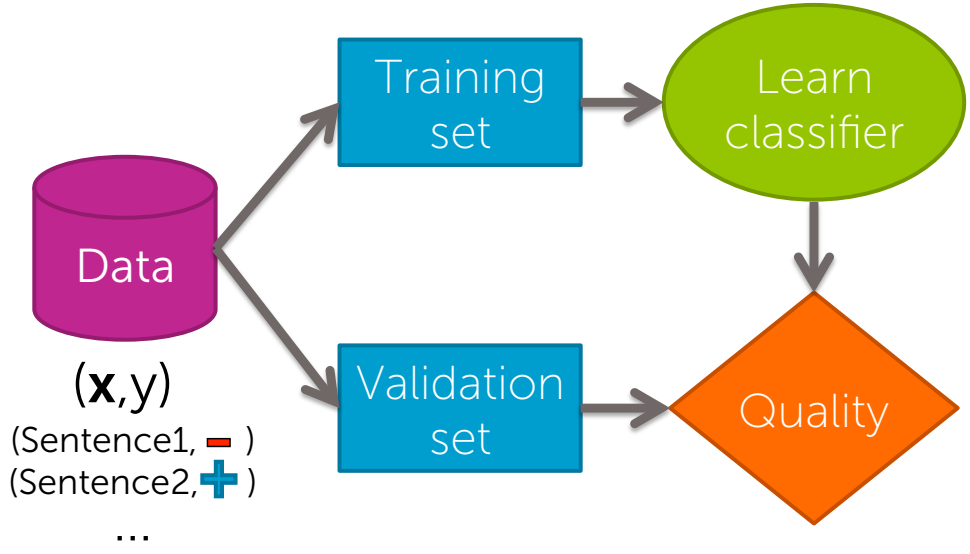


Overview of learning logistic regression model

Training a classifier = Learning the coefficients

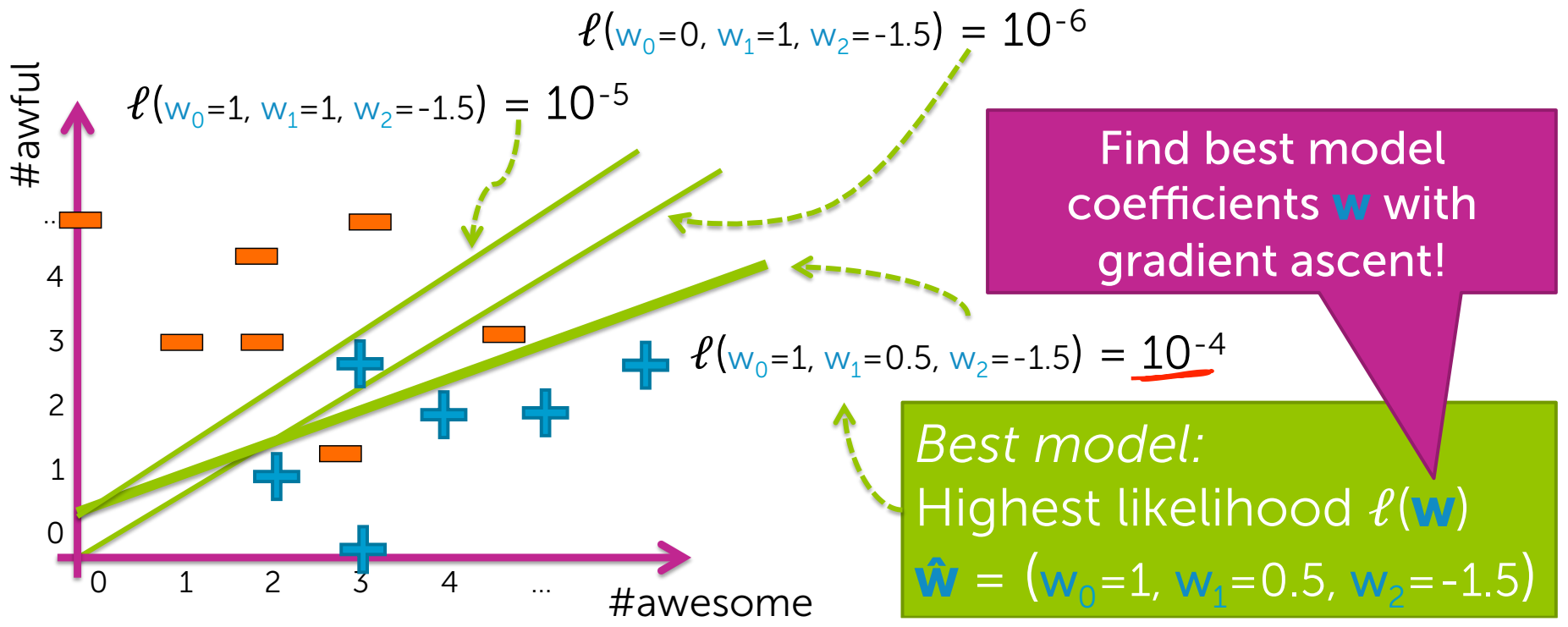
Word	Coefficient	Value
	$\hat{w}_0$	-2.0
good	$\hat{w}_1$	1.0
awesome	$\hat{w}_2$	1.7
bad	$\hat{w}_3$	-1.0
awful	$\hat{w}_4$	-3.3
...	...	...

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



Find "best" classifier =
Maximize quality metric over all possible w_0, w_1, w_2

Likelihood $\ell(\mathbf{w})$



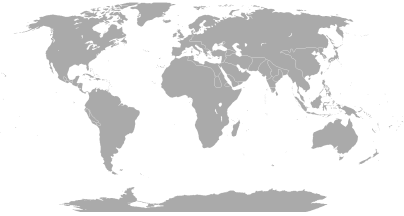
Encoding categorical inputs

Categorical inputs

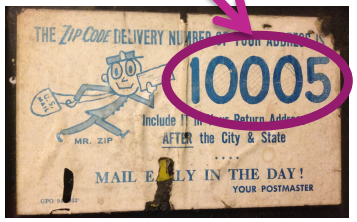
- Numeric inputs:
  - #awesome, age, salary,…
  - Intuitive when multiplied by coefficient
    - e.g., 1.5 #awesome
- Categorical inputs:

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

Gender
(Male, Female,…)



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

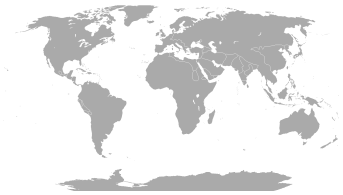


Zipcode
(10005, 98195,…)

How do we multiply category by coefficient???
Must convert categorical inputs into numeric features

Encoding categories as numeric features

x =



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

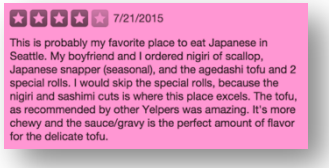
196 categories

1-hot encoding

x	$h_1(x)$ (Argentina)	$h_2(x)$ (Brazil)	...	$h_{195}(x)$	$h_{196}(x)$ (Zimbabwe)
Brazil	0	1	0	0	0
Zimbabwe	0	0	0	0	1

196 features

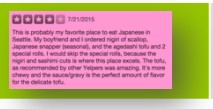
x =



Restaurant review
(Text data)

10,000 words in vocabulary

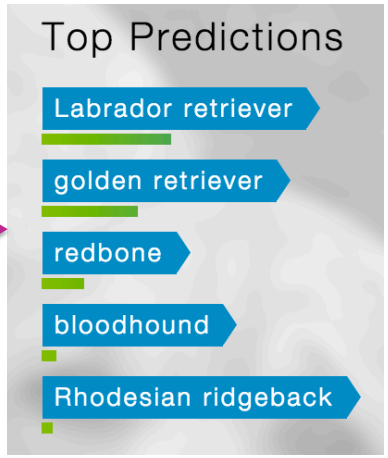
Bag of words

x	$h_1(x)$ (#awesome)	$h_2(x)$ (#awful)	...	$h_{9999}(x)$	$h_{10000}(x)$ (#sushi)
	2	0	0	0	3

10.000 features

Multiclass classification using 1 versus all

Multiclass classification



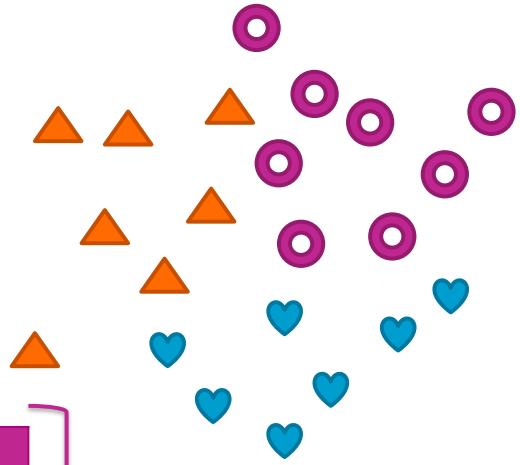
Input: x
Image pixels

Output: y
Object in image

Multiclass classification formulation

- 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:

$\hat{P}(y=▲|x)$

$\hat{P}(y=♥|x)$

$\hat{P}(y=◎|x)$

1 versus all:

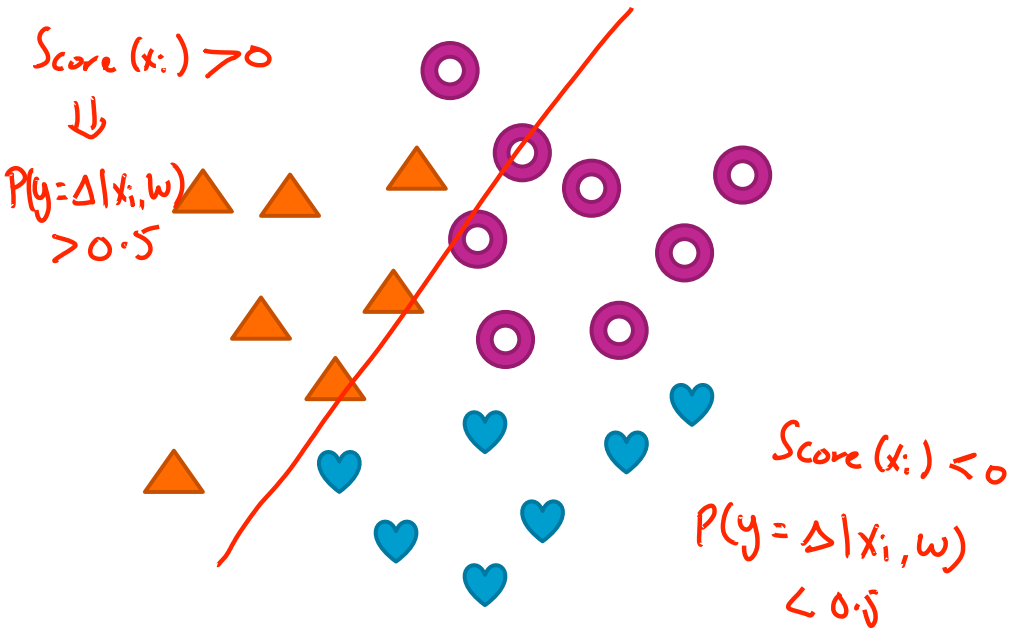
Estimate $\hat{P}(y=\blacktriangle|\mathbf{x})$ using 2-class model

+1 class: points with $y_i=$ ▲

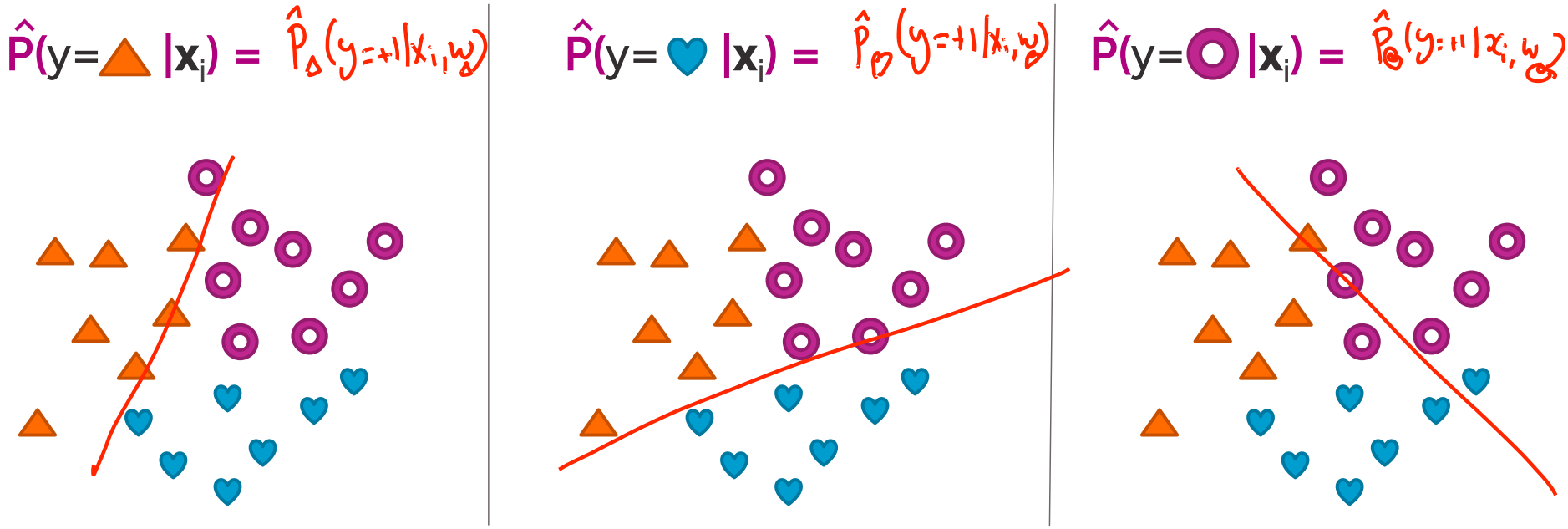
-1 class: points with $y_i=$ ♥ OR ◎

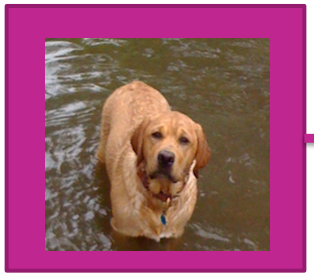
Train classifier: $\hat{P}(y=+1|\mathbf{x})$

Predict: $\hat{P}(y=\blacktriangle|\mathbf{x}_i)=\hat{P}(y=+1|\mathbf{x}_i)$



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





Input: $\mathbf{x}_i$

Multiclass training

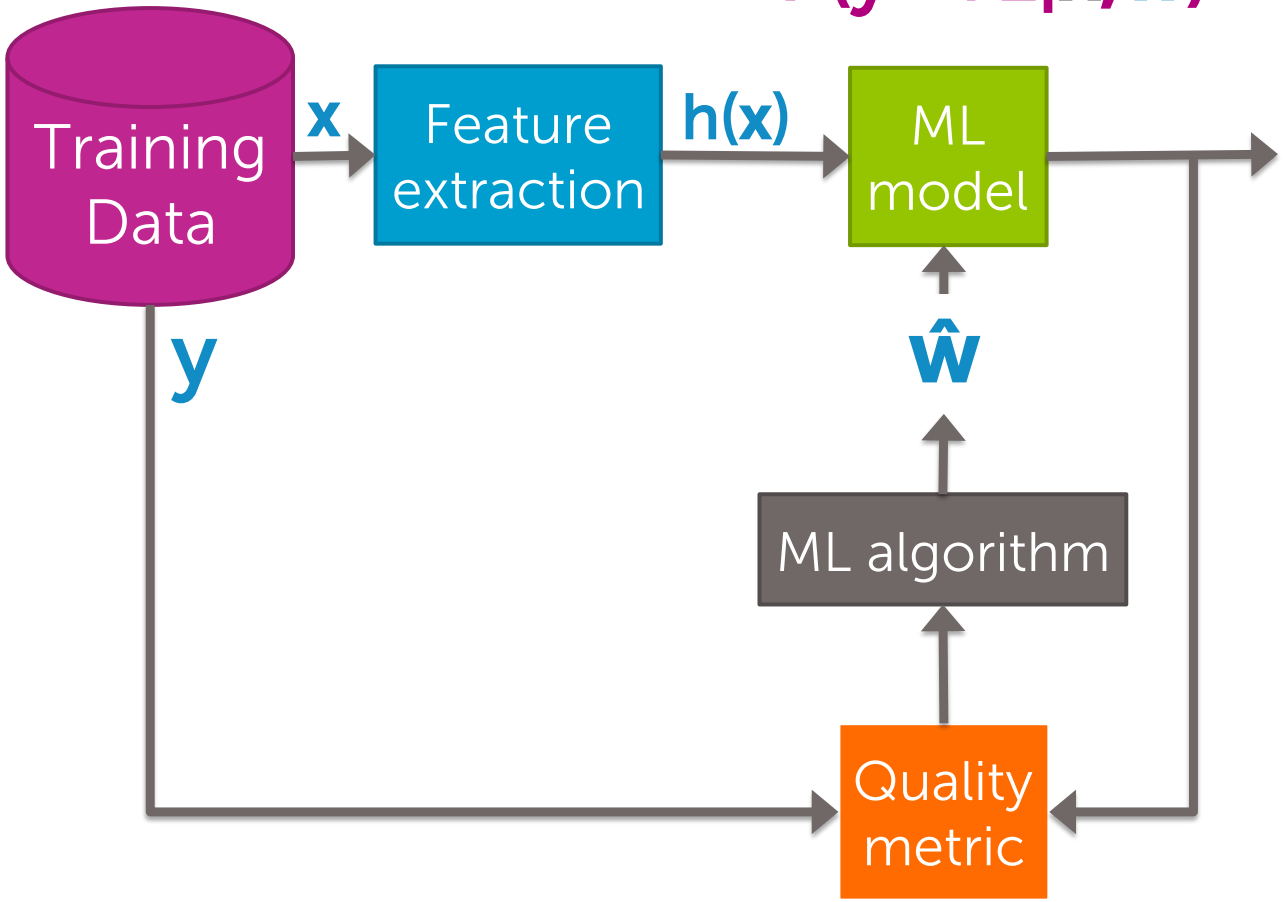
$\hat{P}_c(y=+1|\mathbf{x})$ = estimate of 1 vs all model for each class

Predict most likely class

max_prob = 0; $\hat{y}$ = 0
For c = 1,…,C:
  If $\hat{P}_c(y=+1|\mathbf{x}_i)$ > max_prob:
    $\hat{y}$ = c
    max_prob = $\hat{P}_c(y=+1|\mathbf{x}_i)$

Summary of logistic regression classifier

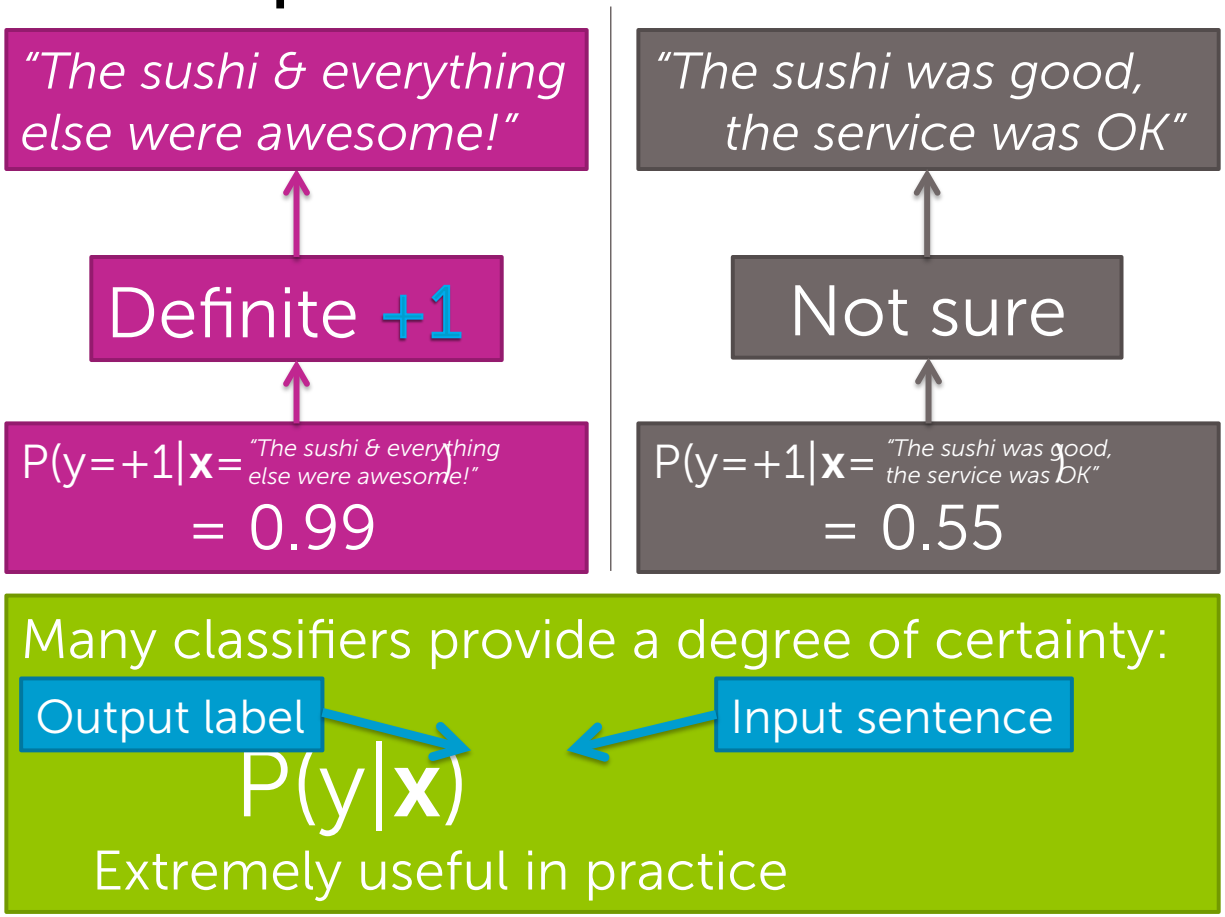
$$\hat{P}(y=+1|\mathbf{x},\hat{\mathbf{w}}) = \frac{1}{1 + e^{-\hat{\mathbf{w}}^T h(\mathbf{x})}}$$



Linear classifiers:

Parameter learning

Learn a probabilistic classification model

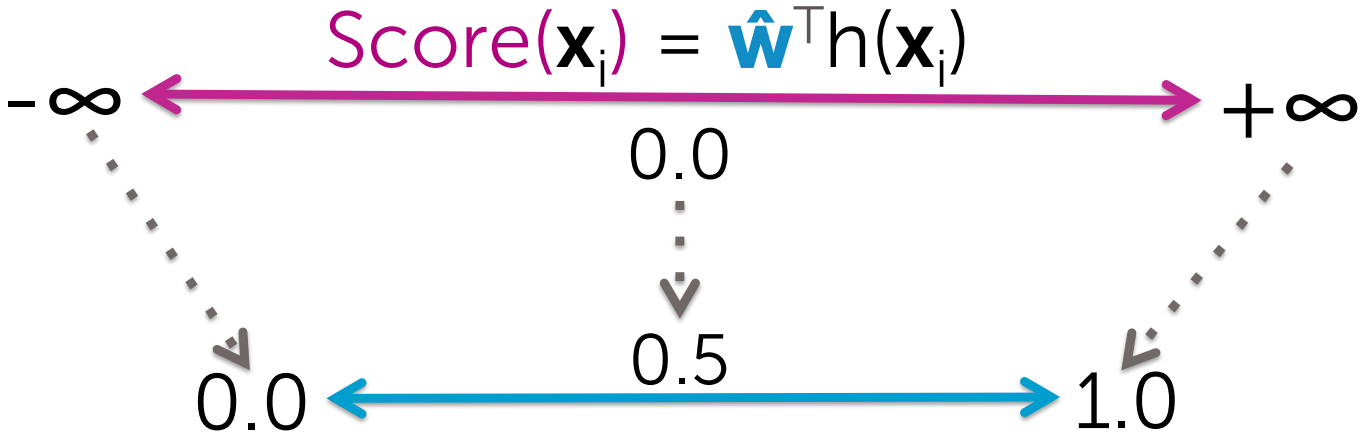


A (linear) classifier

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

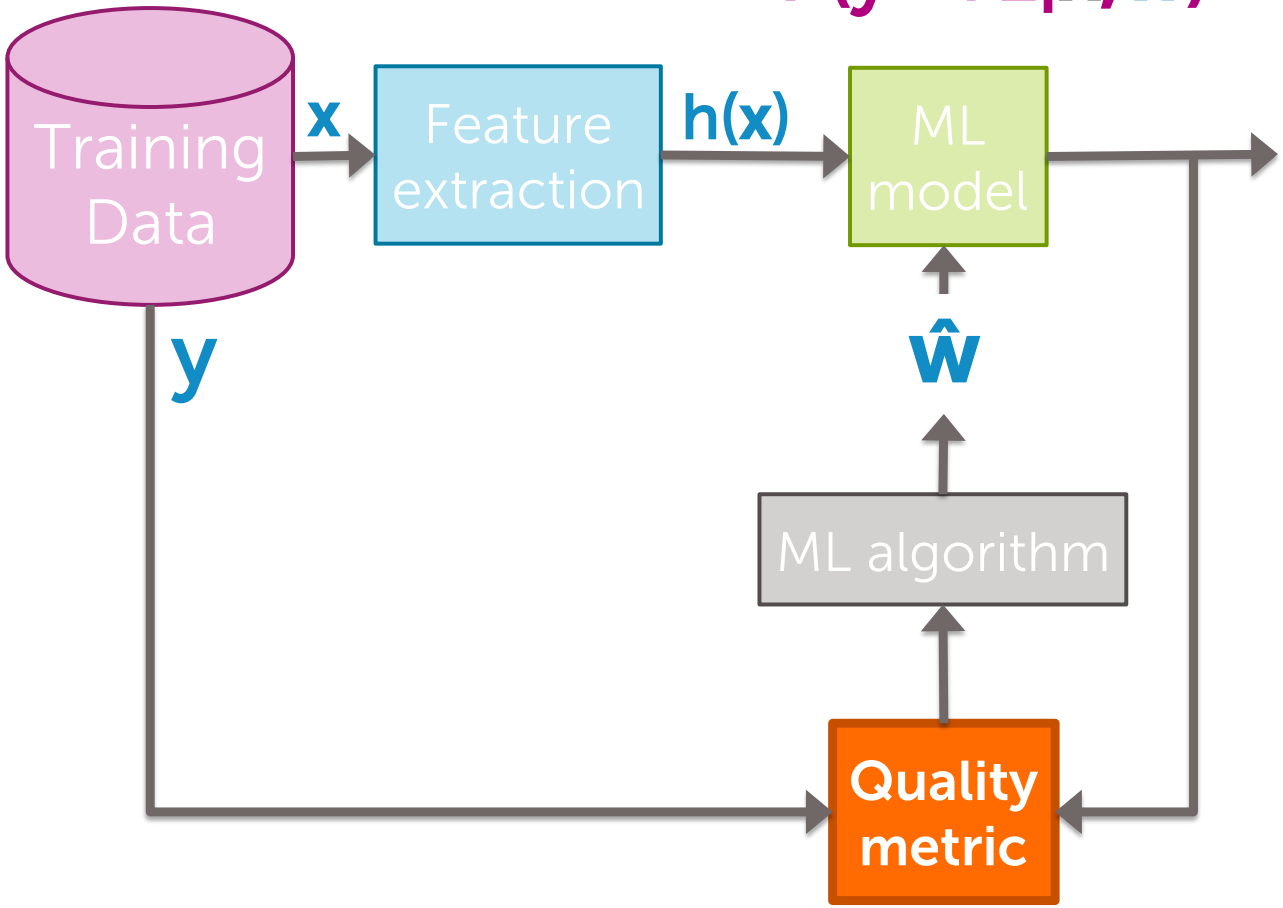
Word	Coefficient	Value
	$\hat{w}_0$	-2.0
good	$\hat{w}_1$	1.0
great	$\hat{w}_2$	1.5
awesome	$\hat{w}_3$	2.7
bad	$\hat{w}_4$	-1.0
terrible	$\hat{w}_5$	-2.1
awful	$\hat{w}_6$	-3.3
restaurant, the, we, …	$\hat{w}_7$, $\hat{w}_8$, $\hat{w}_{9,\ldots}$	0.0
…		…

Logistic regression model



Quality metric for logistic regression: Maximum likelihood estimation

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



Learning problem

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

x[1] = #awesome	x[2] = #awful	y = 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

Finding best coefficients

x[1] = #awesome	**x**[2] = #awful	y = 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

Finding best coefficients

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

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

Finding best coefficients

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

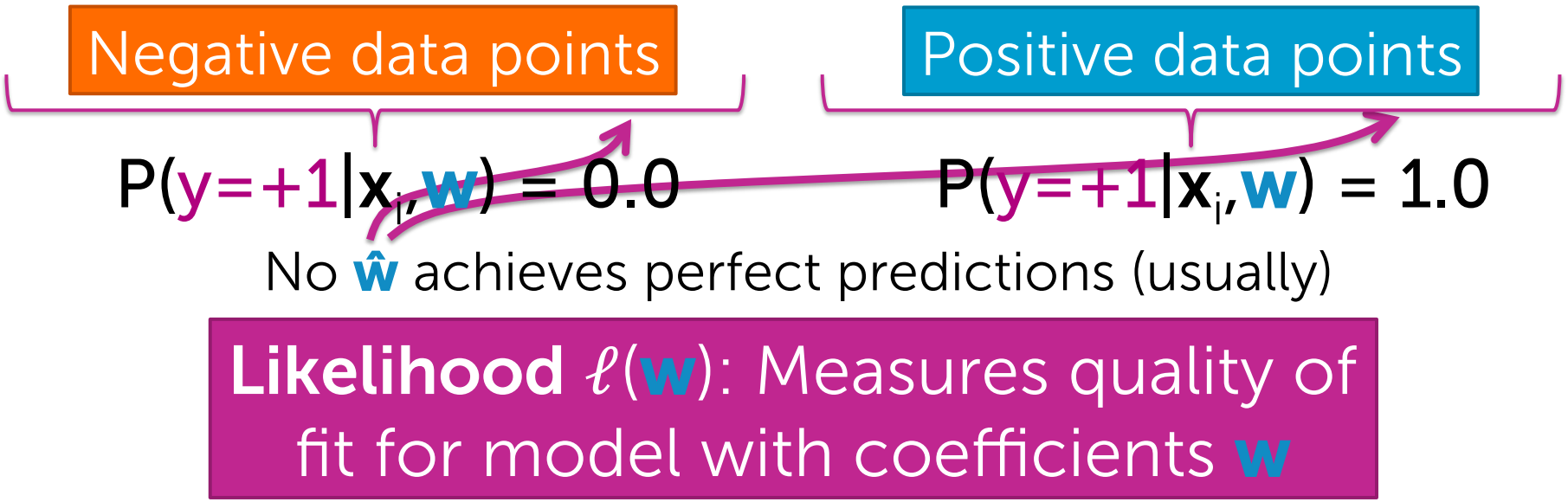
x[1] = #awesome	x[2] = #awful	y = sentiment
2	1	+1
4	1	+1
1	1	+1
2	1	+1

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

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

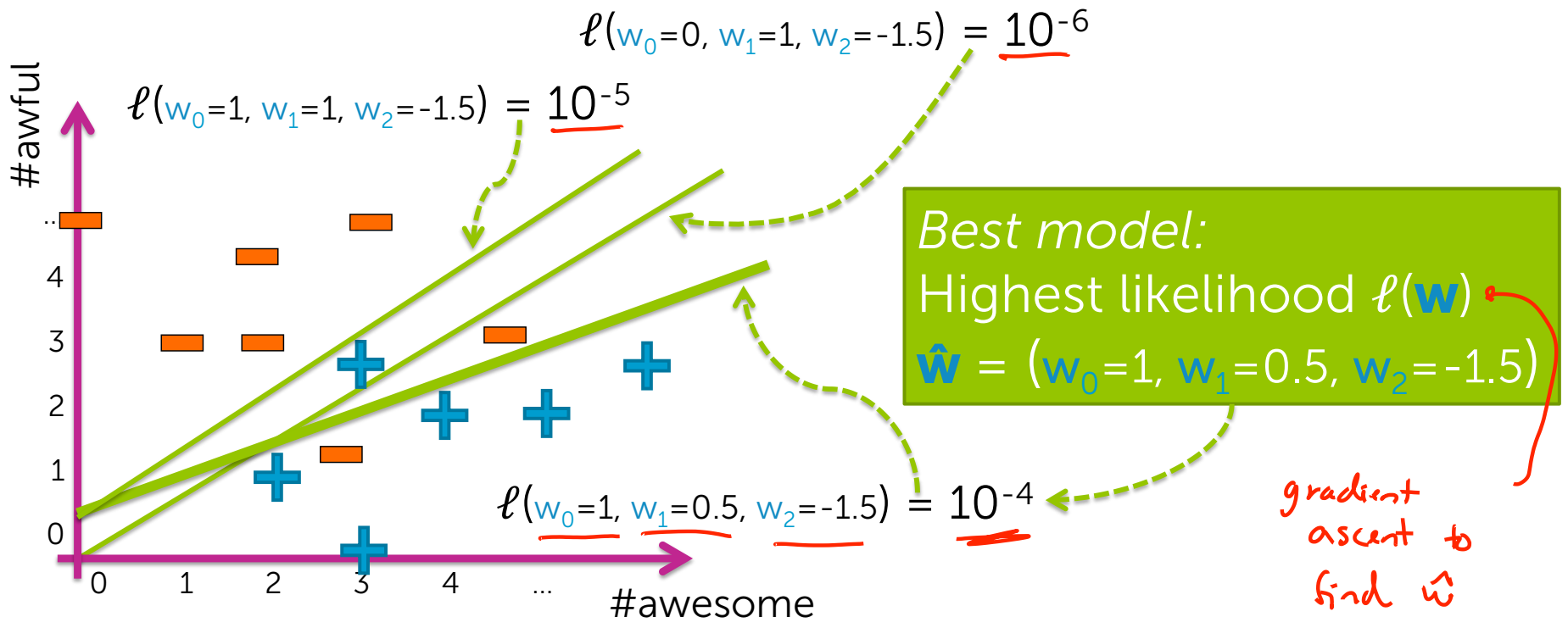
Pick $\hat{\mathbf{w}}$ that makes

Quality metric = Likelihood function



Find "best" classifier

Maximize likelihood over all possible w_0, w_1, w_2



Data likelihood

Quality metric: probability of data

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

x_2= y_2=

If model good, should predict:

$\hat{y}_2 = -1$

Pick **w** to maximize:

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

Maximizing likelihood (probability of data)

Data point	x[1]	x[2]	y	Choose w to maximize
$\mathbf{x}_1, y_1$	2	1	+1	$P(y=+1 \mid x_1, w) = P(y=+1 \mid x[1]=2, x[2]=1, w)$
$\mathbf{x}_2, y_2$	0	2	-1	$P(y=-1 \mid x_2, w)$
$\mathbf{x}_3, y_3$	3	3	-1	$P(y=-1 \mid x_3, w)$
$\mathbf{x}_4, y_4$	4	1	+1	$P(y=+1 \mid x_4, w)$
$\mathbf{x}_5, y_5$	1	1	+1	
$\mathbf{x}_6, y_6$	2	4	-1	
$\mathbf{x}_7, y_7$	0	3	-1	
$\mathbf{x}_8, y_8$	0	1	-1	
$\mathbf{x}_9, y_9$	2	1	+1	

Must combine into single measure of quality ?

Multiply probabilities

$P(y=+1 \mid x_1, w)\, P(y=-1 \mid x_2, w)\, P(y=-1 \mid x_3, w) \ldots$

Learn logistic regression model with maximum likelihood estimation (MLE)

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

$$\ell(\mathbf{w}) = P(y_1|\mathbf{x}_1,\mathbf{w})\; P(y_2|\mathbf{x}_2,\mathbf{w})\; P(y_3|\mathbf{x}_3,\mathbf{w})\; P(y_4|\mathbf{x}_4,\mathbf{w})$$

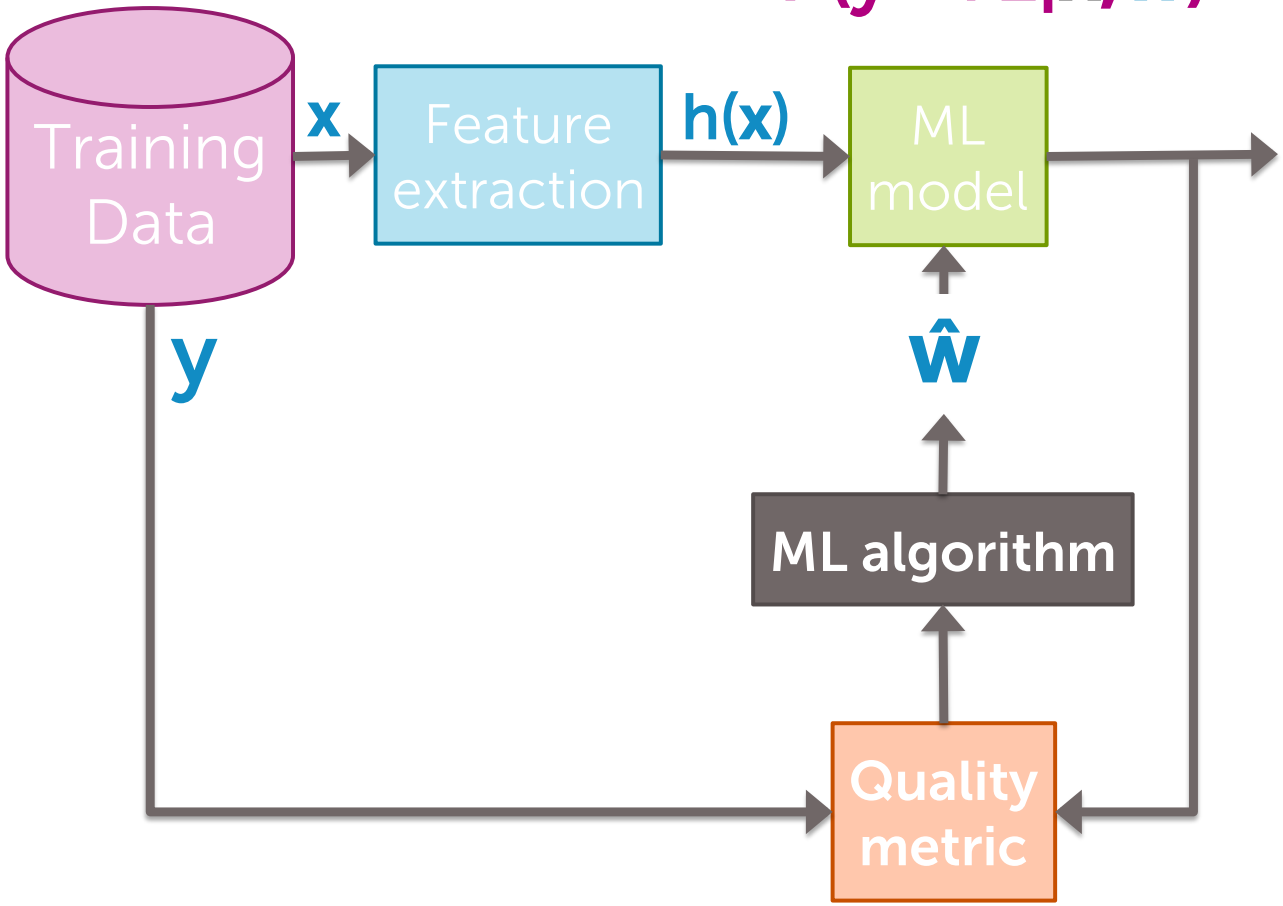
Num. of data points → N

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

← pick w to make this fn. as large as possible

Finding best linear classifier with gradient ascent

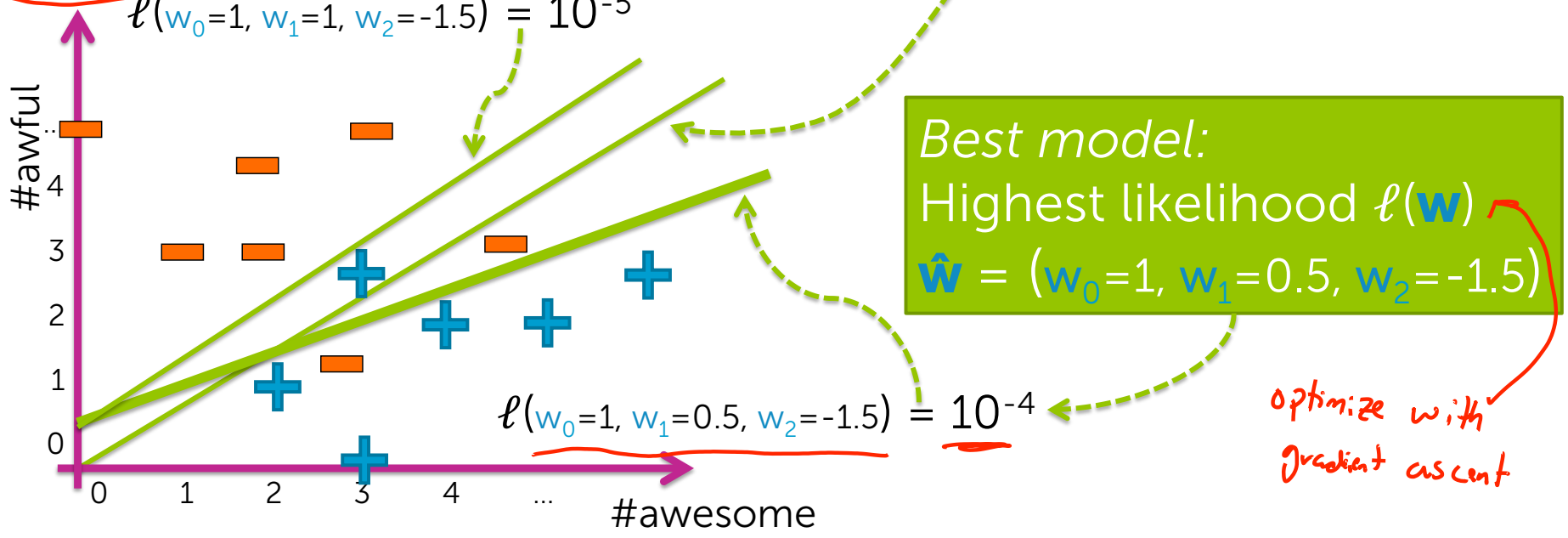
$$\hat{P}(y=+1|\mathbf{x},\hat{\mathbf{w}}) = \frac{1}{1 + e^{-\hat{\mathbf{w}}^T h(\mathbf{x})}}$$



Find "best" classifier

Maximize likelihood over all possible w_0, w_1, w_2

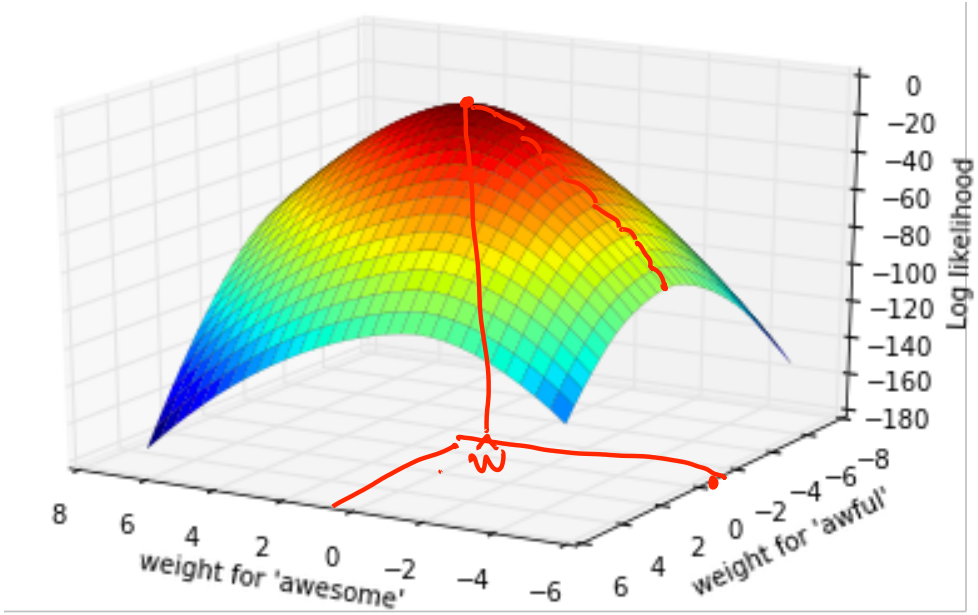
$$\ell(\mathbf{w}) = \prod_{i=1}^{N} P(y_i \mid \mathbf{x}_i, \mathbf{w})$$



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

Maximizing likelihood



Maximize function over all possible w_0, w_1, w_2

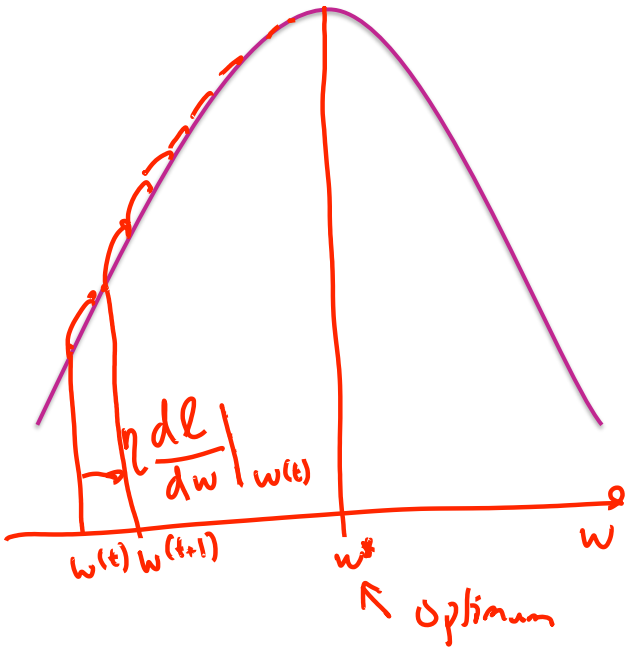
$$\max_{w_0,w_1,w_2} \prod_{i=1}^{N} P(y_i \mid \mathbf{x}_i, \mathbf{w})$$

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

No closed-form solution ➔ use gradient ascent

Review of gradient ascent

Finding the max via hill climbing



Algorithm:

while not converged

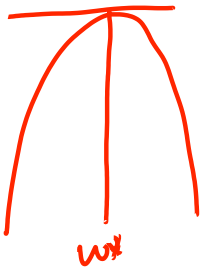
$$w^{(t+1)} \leftarrow w^{(t)} + \eta \left.\frac{d\ell}{dw}\right|_{w^{(t)}}$$

step size

Convergence criteria

For convex functions, optimum occurs when

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



In practice, stop when

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

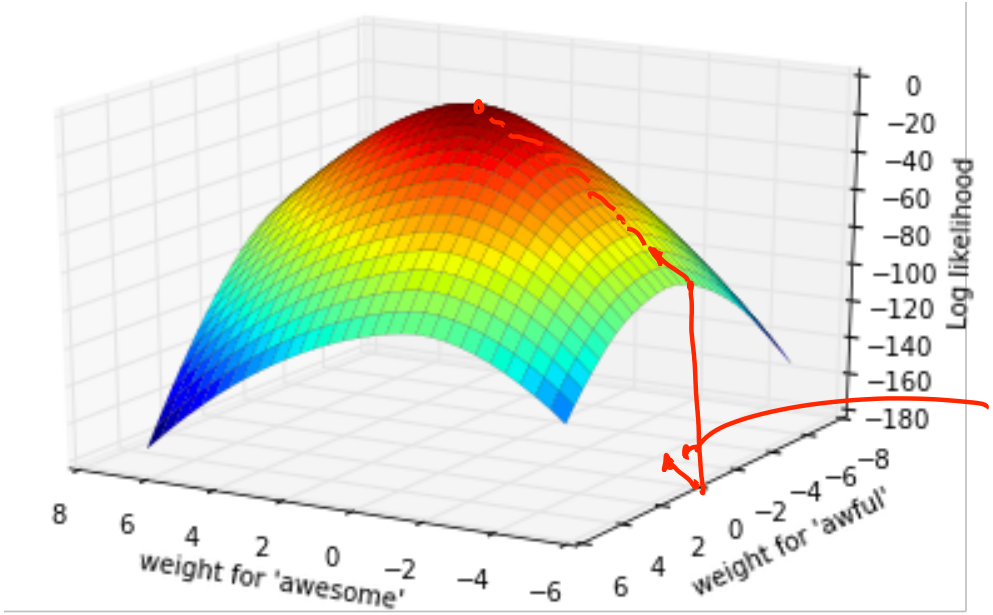
ε: tolerance

Algorithm:

while not converged

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

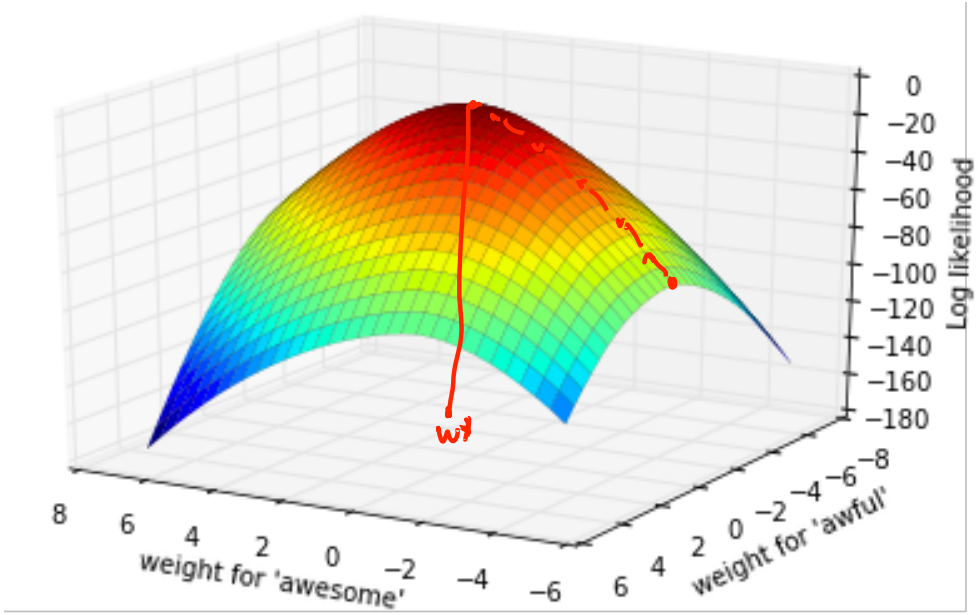
Moving to multiple dimensions: Gradients

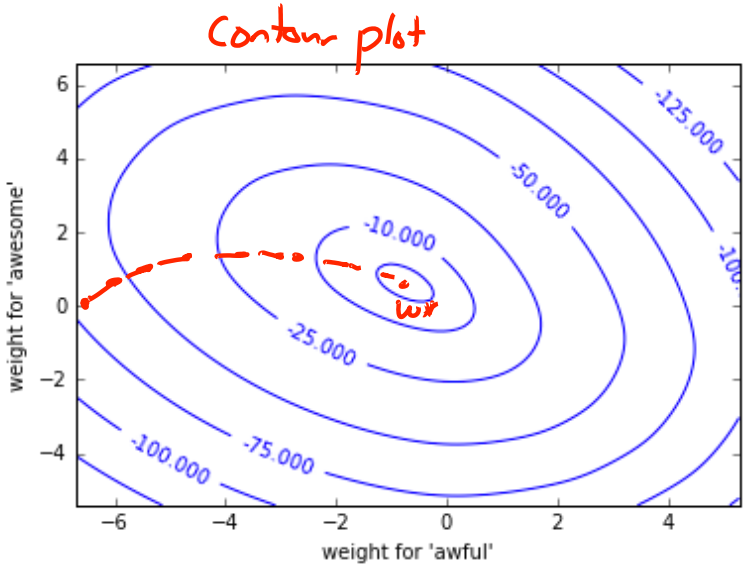


$$\nabla \ell(\mathbf{w}) = \begin{bmatrix} \frac{\partial \ell}{\partial w_0} \\ \frac{\partial \ell}{\partial w_1} \\ \vdots \\ \frac{\partial \ell}{\partial w_D} \end{bmatrix}$$

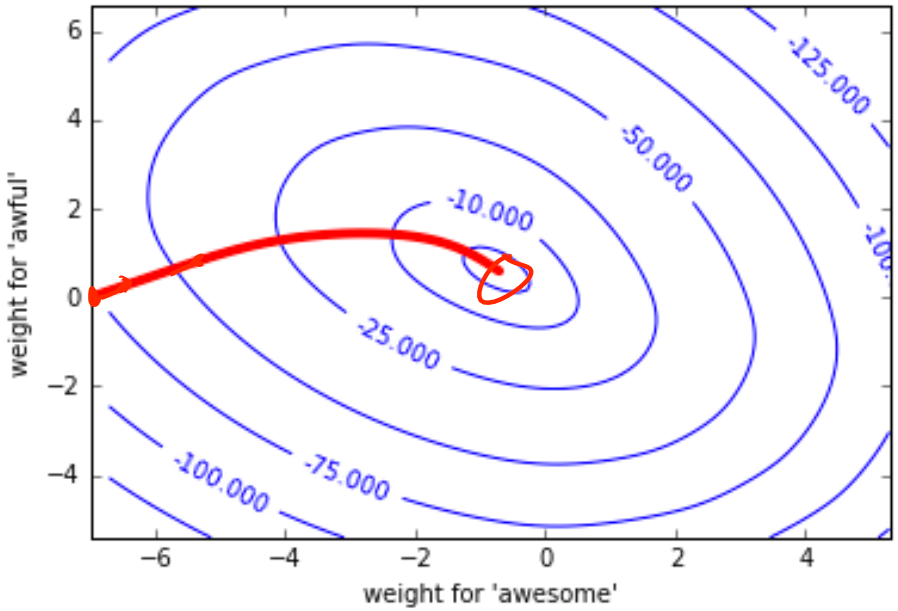
D+1 dim vector

Contour plots





Gradient ascent



Algorithm:

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

while not converged

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

step size

Learning algorithm for logistic regression

Derivative of (log-)likelihood

Sum over data points

Feature value

Difference between truth and prediction

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

predict y_i is positive

Indicator function:

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

Computing derivative

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

$w^{(t)}$:

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

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

$h_1(x)$ = #awesome

x[1]	x[2]	y	P(y=+1|x_i,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:

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

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

$= 1 + 0.1 \times 1.33 = 1.133$

Derivative of (log-)likelihood: Interpretation

Sum over data points

Feature value

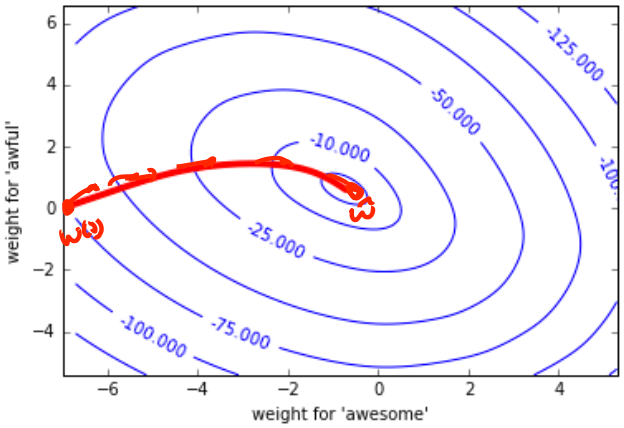
Difference between truth and prediction

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

Δ_i

If $h_j(\mathbf{x}_i)=1$:	$P(y=+1\|\mathbf{x}_i,\mathbf{w}) \approx 1$	$P(y=+1\|\mathbf{x}_i,\mathbf{w}) \approx 0$
$y_i=+1$	$\Delta_i = (1-1) \approx 0$ ↳ don't change anything!	$\Delta_i \approx 1 \Rightarrow$ increase w_j ⇒ score(x_i) becomes larger ⇒ $P(y=+1\|x_i,w)$ increases
$y_i=-1$	$\Delta_i = -1 \Rightarrow w_j$ to decrease ⇒ score(x_i) decreases ⇒ $P(y=+1\|x_i,w)$ decrease	$\Delta_i \approx 0$ ⇒ don't change anything

Summary of gradient ascent for logistic regression



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

while $||\nabla \ell(\mathbf{w}^{(t)})|| > \varepsilon$ ← tolerance

  for j=0,…,D ← coefficient

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

    $w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta$ partial[j]

  t ← t + 1

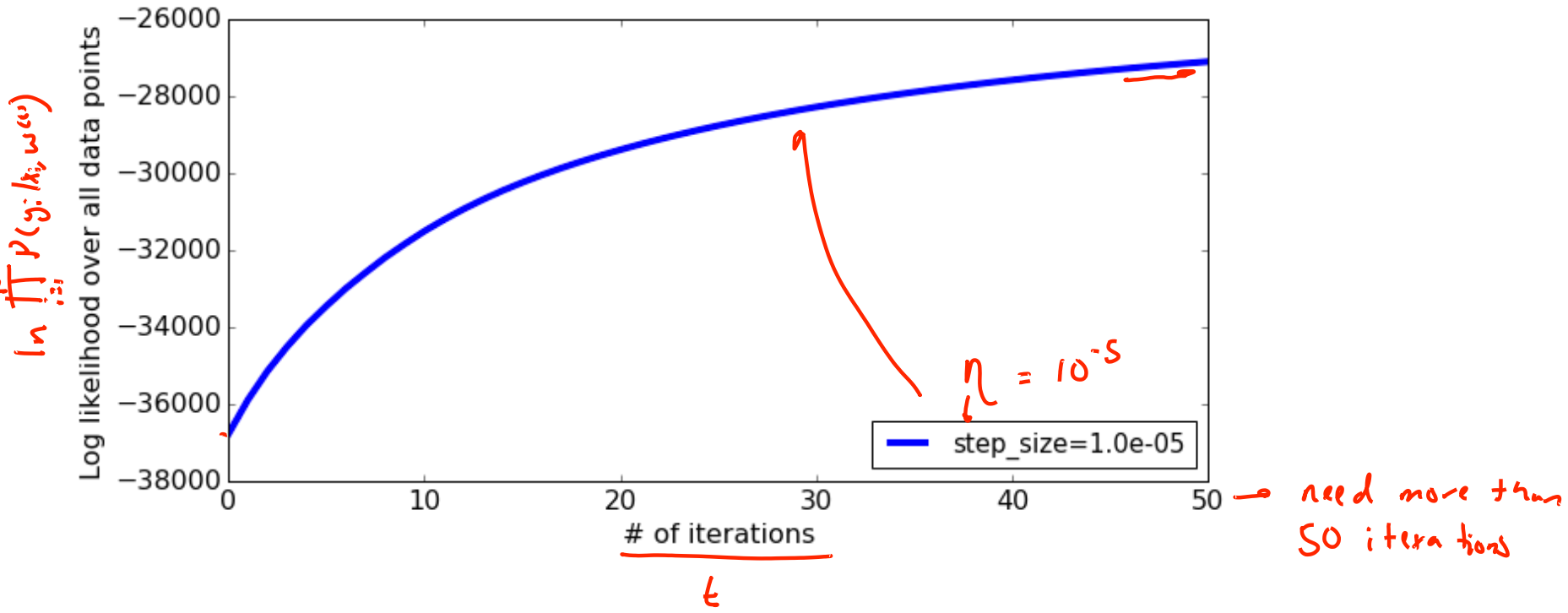
$P(y = +1 \mid \mathbf{x}_i, \mathbf{w}^{(t)}) = \frac{1}{1 + e^{-w^{(t)} \cdot h(x_i)}}$

η: step size

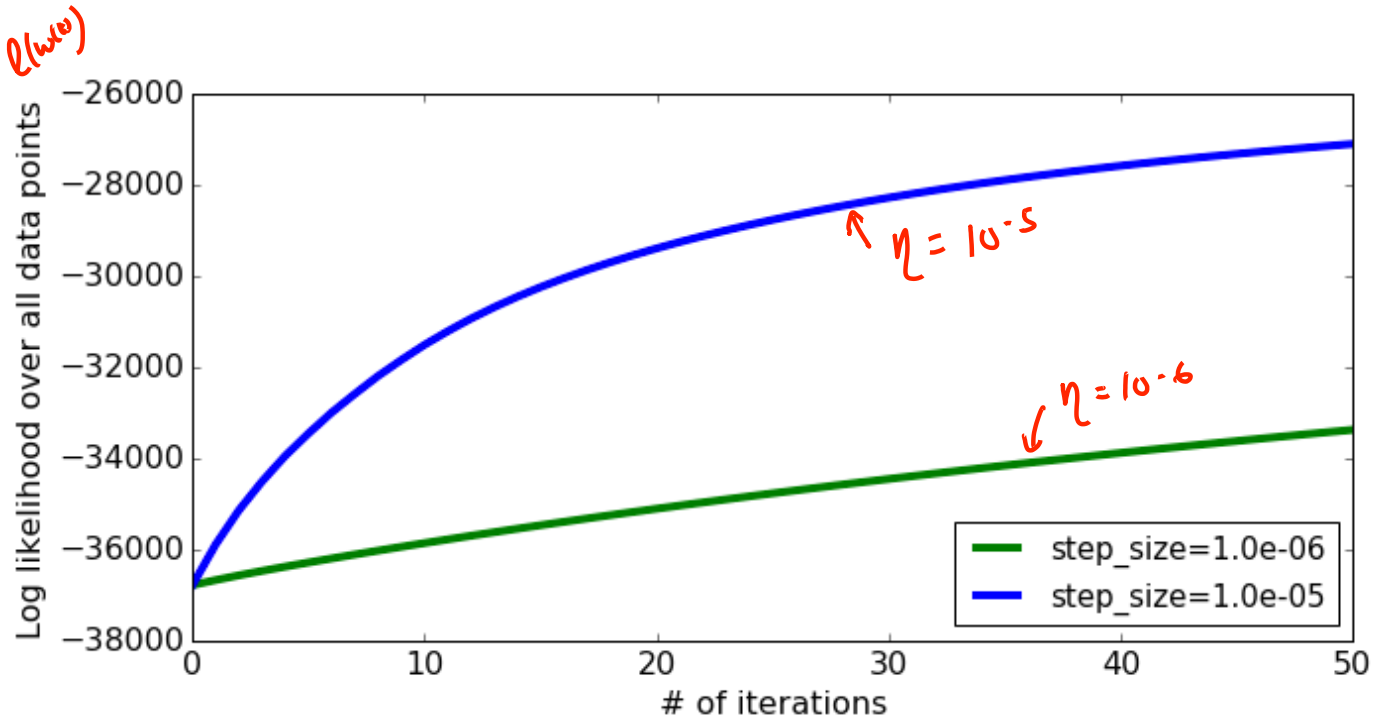
partial[j]: $\frac{\partial \ell(w^{(t)})}{\partial w_j}$

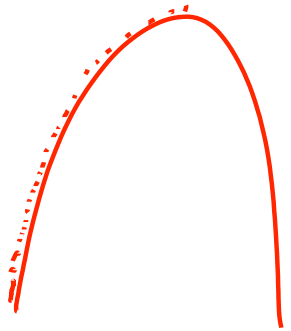
Choosing the step size η

Learning curve:
Plot quality (likelihood) over iterations

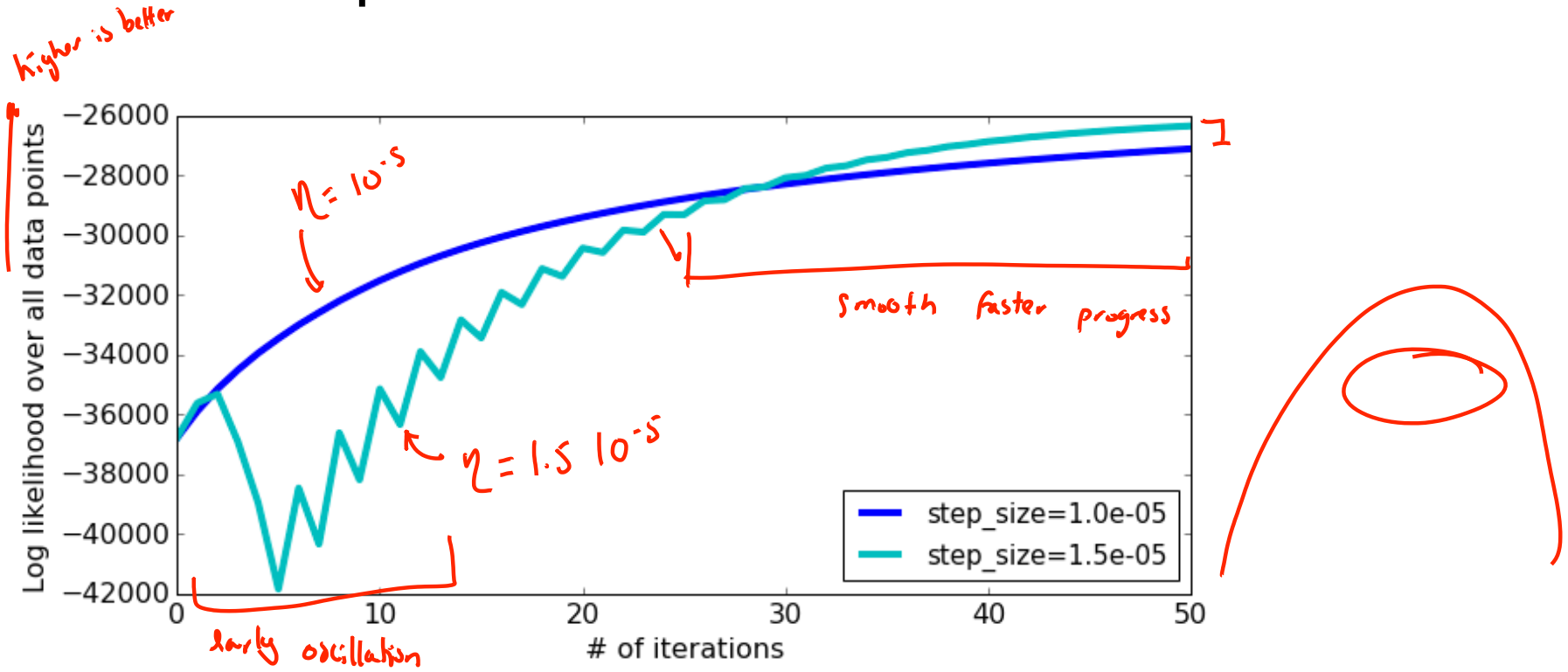


If step size is too small, can take a long time to converge

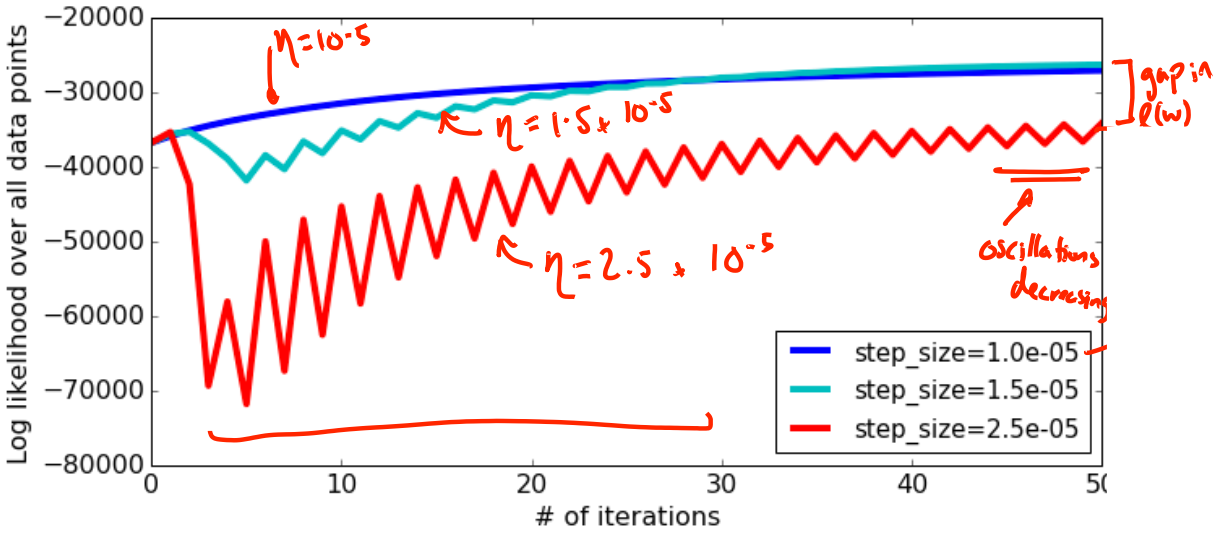




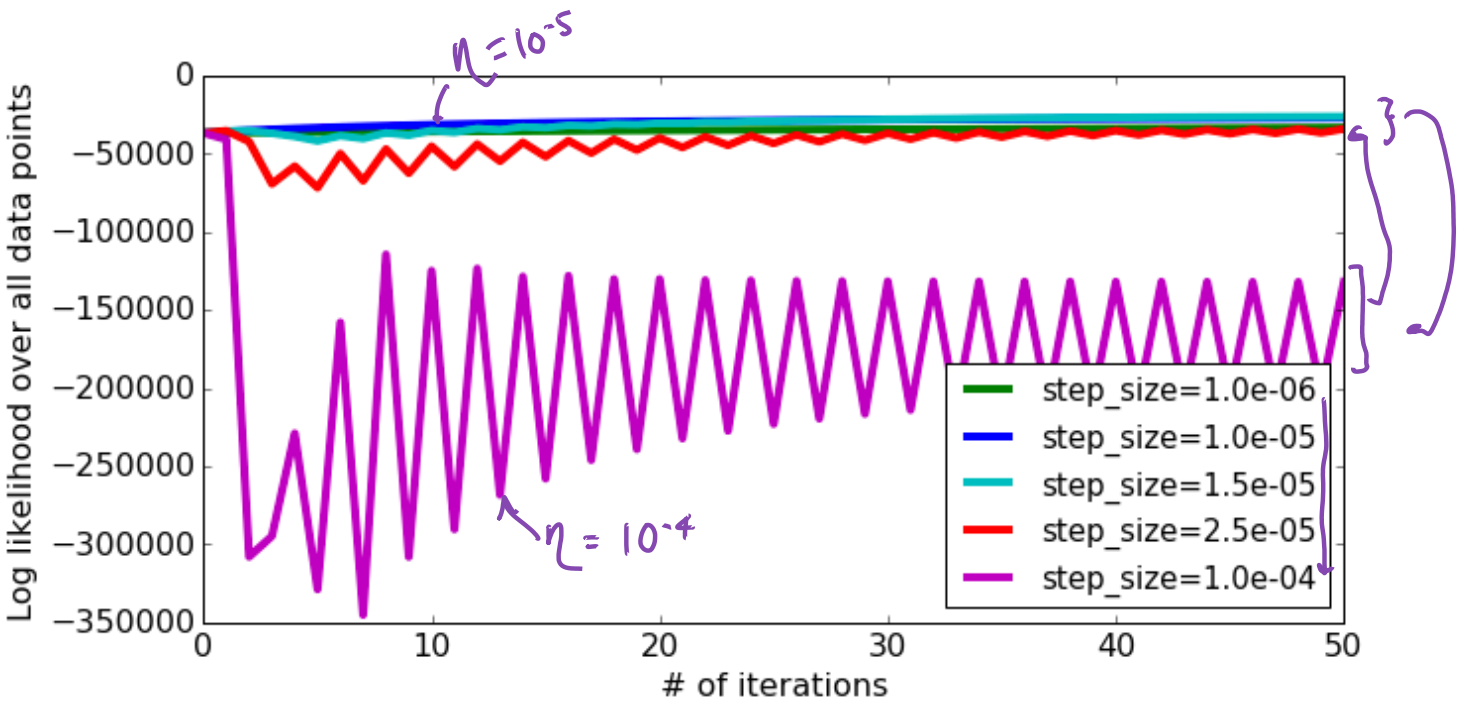
Compare converge with different step sizes



Careful with step sizes that are too large



Very large step sizes can even cause divergence or wild oscillations



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.,

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

Deriving the gradient for logistic regression

Log-likelihood function

- Goal: choose coefficients **w** maximizing likelihood:

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

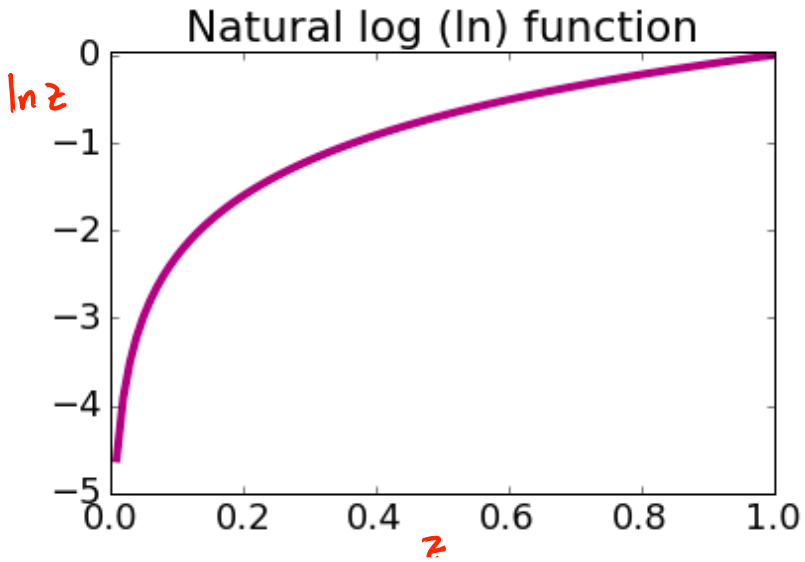
- Math simplified by using log-likelihood – taking (natural) log:

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

natural log

The log trick, often used in ML…

- 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($\mathbf{w}$):
    $\hat{w} = \arg\max_{w} f(w)$
    the w that makes f(w) largest
  - Then $\hat{\mathbf{w}}_{ln}$ maximizes ln(f($\mathbf{w}$)):
    $\hat{w}_{ln} = \arg\max_{w} \ln(f(w))$
    $\hat{w} = \hat{w}_{ln}$



Expressing the log-likelihood

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:

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

$$= \sum_{i=1}^{N} \ln P(y_i \mid x_i, w)$$

Rewriting log-likelihood

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

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

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

if $y_i = +1$

if $y_i = -1$

Deriving probability that y=-1 given **x**

Logistic regression model: P(y=-1|x,w)

- Probability model predicts y=+1:

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

- Probability model predicts y=-1:

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

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

Rewriting the log-likelihood

Plugging in logistic function for 1 data point

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

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

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

$$= -\mathbb{1}[y_i=+1] \ln(1+e^{-w^\top h(x_i)}) + (1-\mathbb{1}[y_i=+1])\left[-w^\top h(x_i) - \ln(1+e^{-w^\top h(x_i)})\right]$$

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

Simpler form

$\ln e^a = a$

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

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

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

$\ln e^{-w^\top h(x_i)} - \ln(1+e^{-w^\top h(x_i)})$

$-w^\top h(x_i)$

Deriving gradient of log-likelihood

Gradient for 1 data point

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

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

$$= -(1 - \mathbb{1}[y_i = +1])\, h_j(x_i) + h_j(x_i)\, P(y = -1 \mid x_i, w)$$

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

$$\frac{\frac{\partial}{\partial w_j} w^\top h(x_i)}{\frac{\partial}{\partial w_j} \ln\left(1 + e^{-w^\top h(x_i)}\right)}$$

$$= -h_j(x_i) \underbrace{\frac{e^{-w^\top h(x_i)}}{1 + e^{-w^\top h(x_i)}}}_{P(y=-1 \mid x_i, w)}$$

Finally, gradient for all data points

- Gradient for one data point:

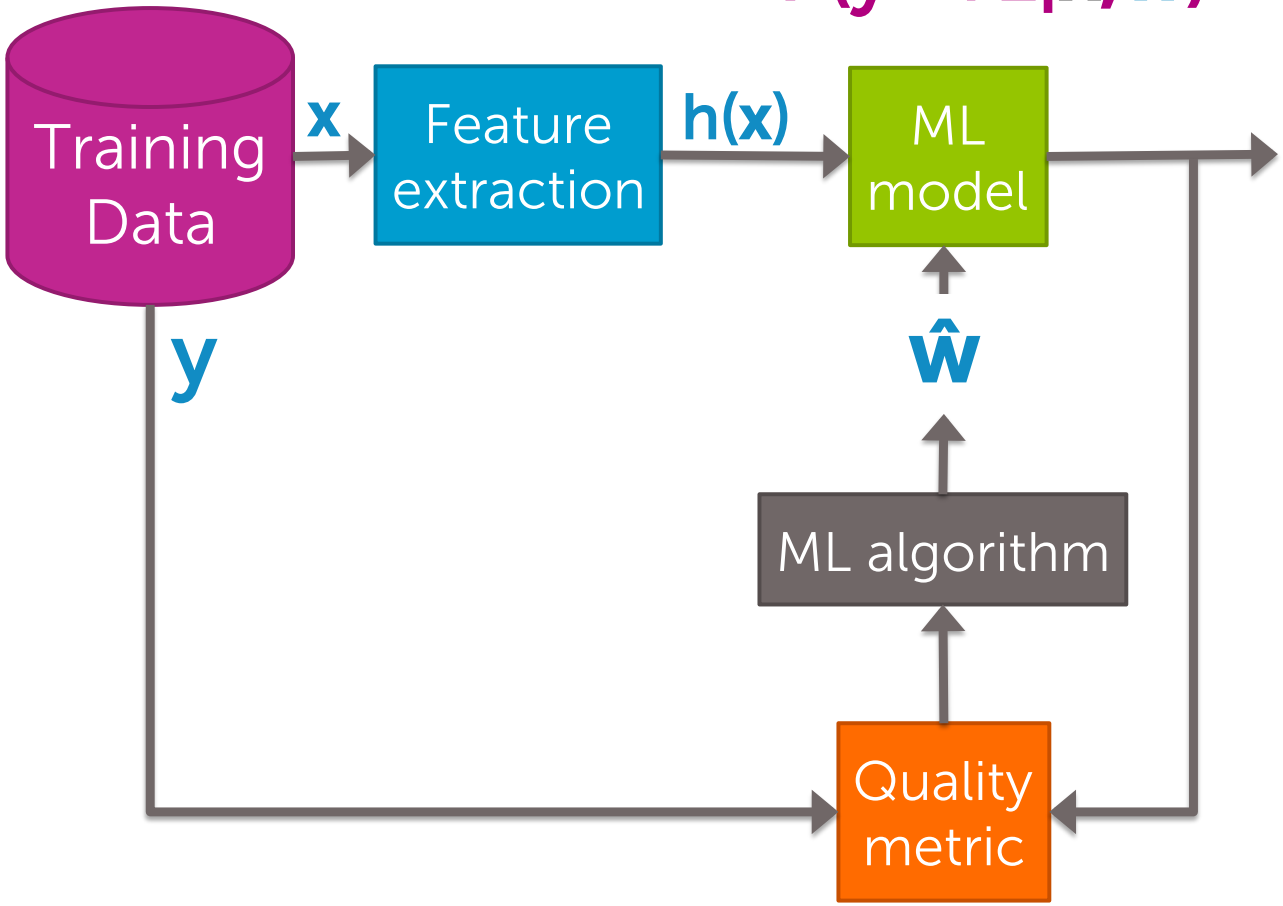
$$h_j(\mathbf{x}_i)\Big(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w})\Big)$$

- Adding over data points:

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

Summary of logistic regression classifier

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



Learn logistic regression model with maximum likelihood estimation (MLE)

- Choose coefficients **w** that maximize likelihood:

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

- No closed-form solution ➔ use gradient ascent

What you can do now...

- 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