Classification

Motivations

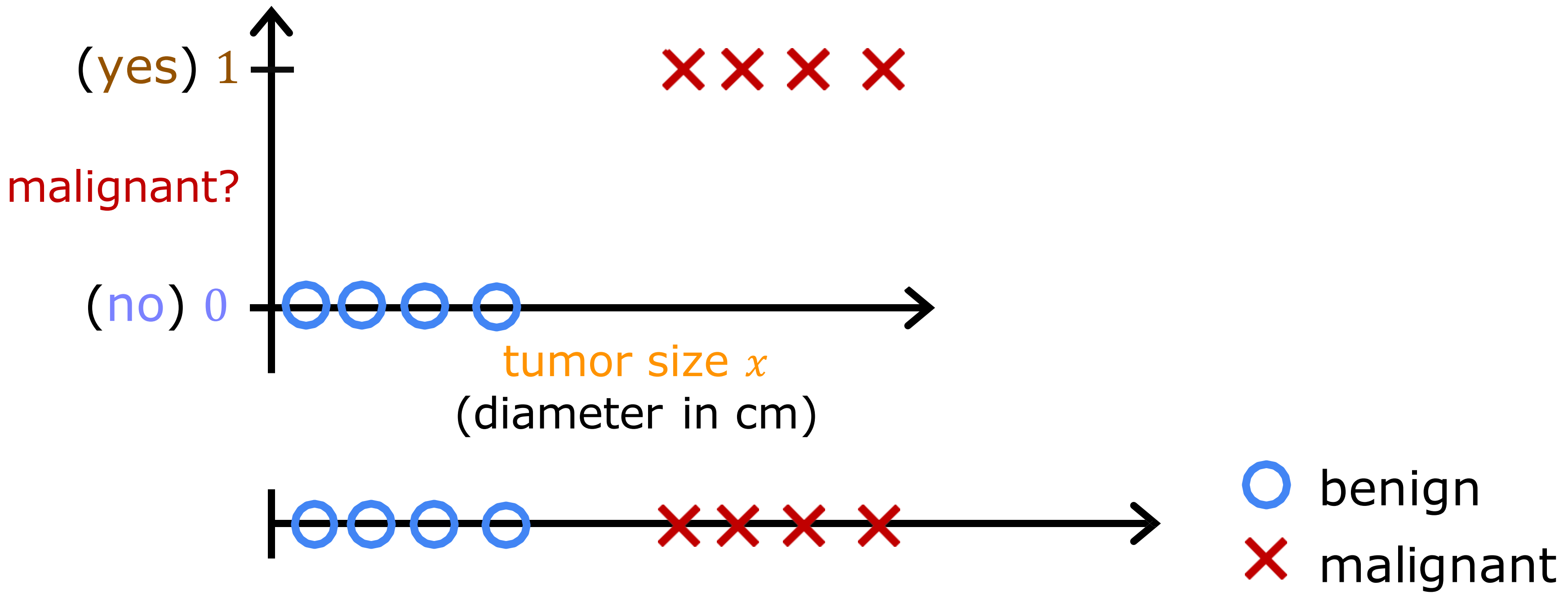
Classification

Question	Answer "y"	
Is this email spam?	no	yes
Is the transaction fraudulent?	no	yes
Is the tumor malignant?	no	yes
	false	true
	0	1

y can only be one of two values

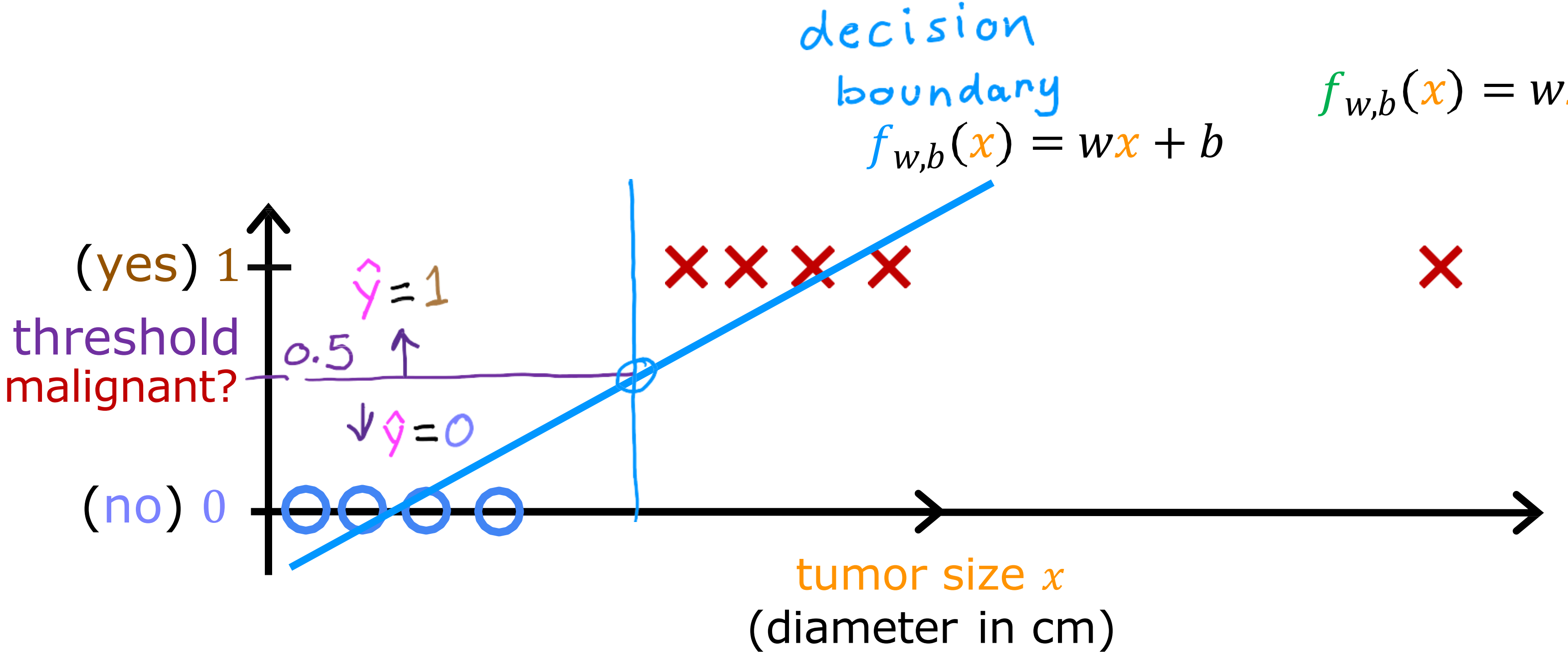
"binary classification"

class = category



(yes) 1
malignant?
(no) 0
tumor size x
(diameter in cm)
benign
malignant

$f_{w,b}(x) = wx + b$

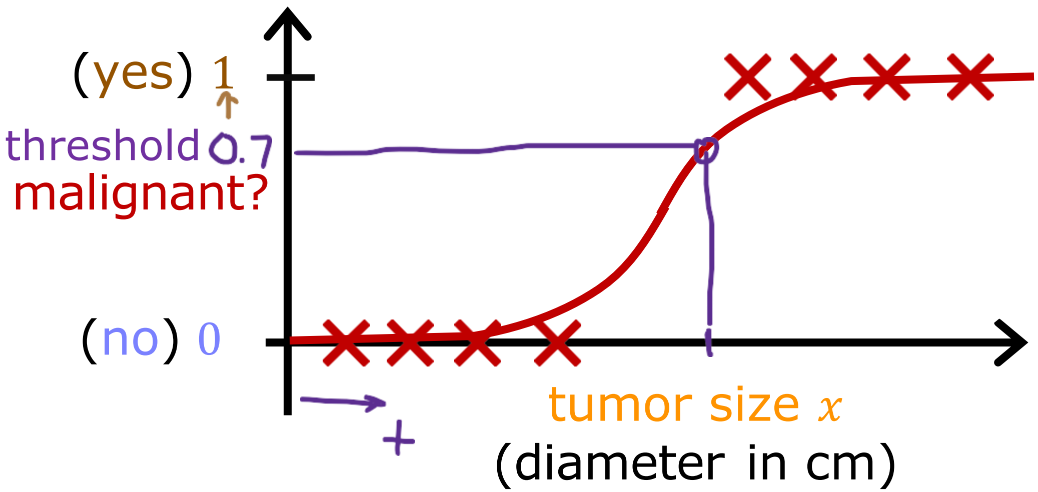


if $f_{w,b}(\ )$ $x < 0.5 \rightarrow y = 0$

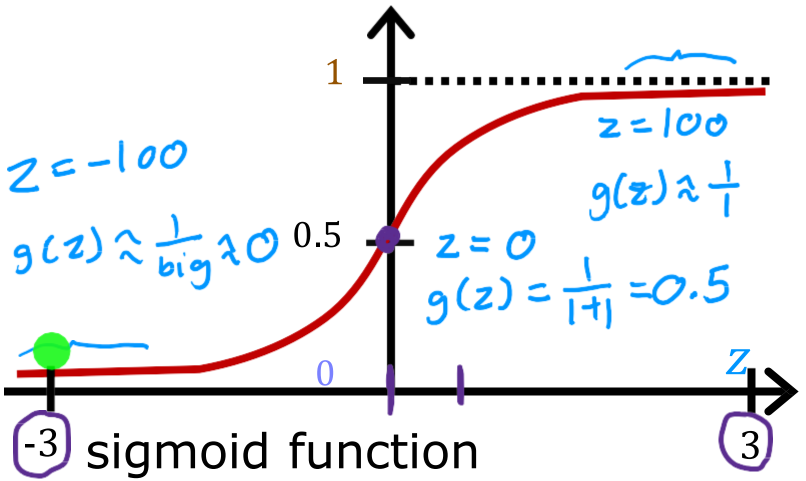
if $f_{w,b}(\ )$ $x \geq 0.5 \rightarrow y = 1$

Classification

Logistic Regression



Want outputs between 0 and 1



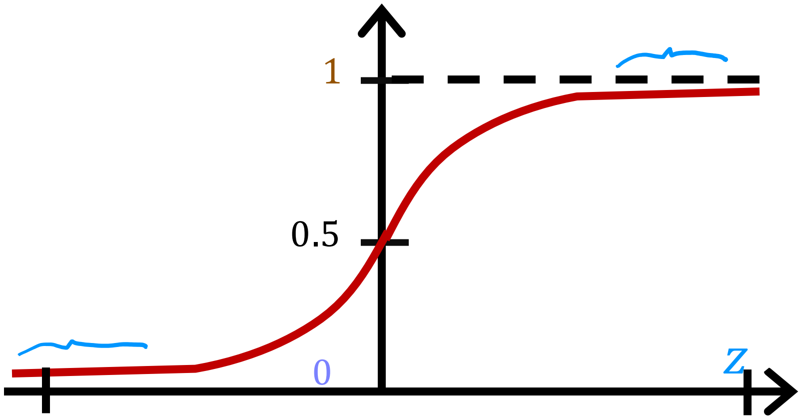
sigmoid function

logistic function

outputs between 0 and 1

$$g(z) = \frac{1}{1+e^{-z}} \qquad 0 < g(z) < 1$$

Want outputs between 0 and 1



sigmoid function

logistic function

outputs between 0 and 1

$g(z) = \frac{1}{1+e^{-z}}$ $\qquad 0 < g(z) < 1$

$f_{\vec{w},b}(\vec{x})$

$$\boxed{\vec{w}\cdot\vec{x}+b}$$

$\downarrow$

z

$\downarrow$

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

$$f_{\vec{w},b}(\vec{x}) = g(\underbrace{\vec{w}\cdot\vec{x}+b}_{z}) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$$

"logistic regression"

$e \approx 2.7$

Interpretation of logistic regression output

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$$

"probability" that class is 1

Example:

x is "tumor size"

y is 0 (not malignant)

or 1 (malignant)

$$f_{\vec{w},b}(\vec{x}) = 0.7$$

70% chance that y is 1

$P(1)$

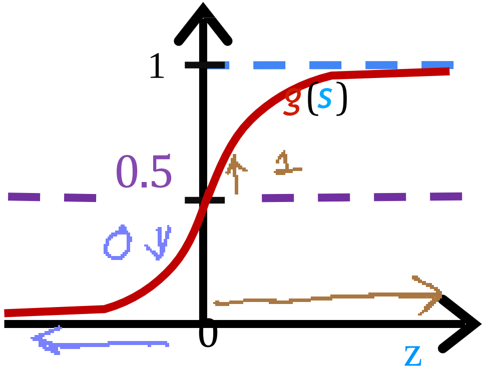
$$f_{\vec{w},b}(\vec{x}) = P(y = 1|\vec{x};\vec{w},b)$$

Probability that y is 1,
given input $\vec{x}$, parameters $\vec{w}$,b

$$P(y=0) + P(y=1) = 1$$

Classification

Decision Boundary



$f_{\vec{w},b}(\vec{x})$

$$\boxed{s = \vec{w} \cdot \vec{x} + b}$$

$$\downarrow$$

$$z$$

$$\downarrow$$

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

$$f_{\vec{w},b}(\vec{x}) = g(\underbrace{\vec{w} \cdot \vec{x} + b}_{z}) = \frac{1}{1+e^{-(\vec{w} \cdot \vec{x} + b)}}$$

$$= P(y = 1 \mid x; \vec{w}, b)$$

0.7 0.3

0 or 1? threshold

Is $f_{\vec{w},b}(\vec{x}) \geq 0.5$?

Yes: $\hat{y} = 1$

No: $\hat{y} = 0$

When is $f_{\vec{w},b}(\vec{x}) \geq 0.5$?

$g(s) \geq 0.5$

$s \geq 0$

$\vec{w} \cdot \vec{x} + b \geq 0$

$\hat{y} = 1$

$\vec{w} \cdot \vec{x} + b < 0$

$\hat{y} = 0$

Decision boundary

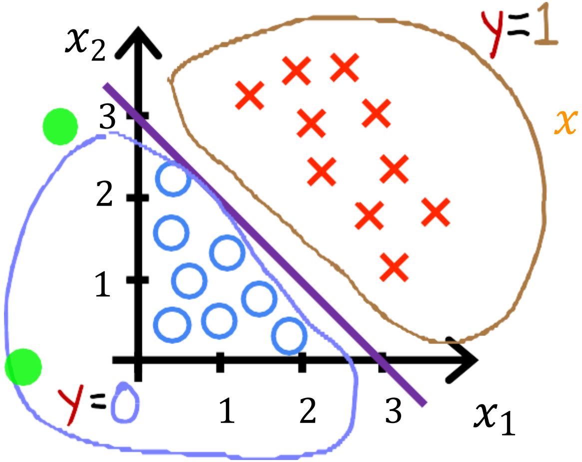
$$f_{\vec{w},b}(\vec{x}) = g(z) = g(\underbrace{\overset{1}{w_1}x_1 + \overset{1}{w_2}x_2 + \overset{-3}{b}}_{})$$

Decision boundary $z = \vec{w} \cdot \vec{x} + b = 0$

$$z = x_1 + x_2 - 3 = 0$$

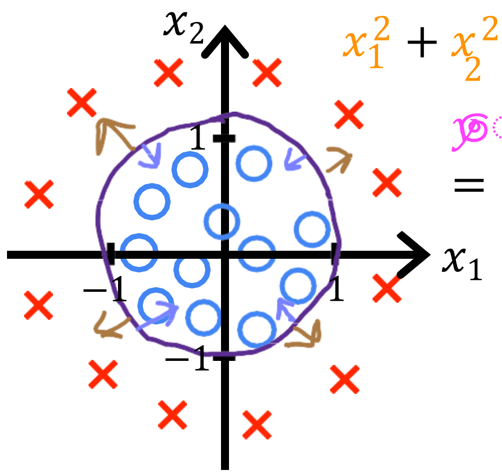
$$x_1 + x_2 = 3$$

$x_1 + x_2 < 3$



$x_1 + x_2 \geq 3$

Non-linear decision boundaries



$x_1^2 + x_2^2 \geq 1$

$y = 1$

$x_1^2 + x_2^2 < 1$

$y = 0$

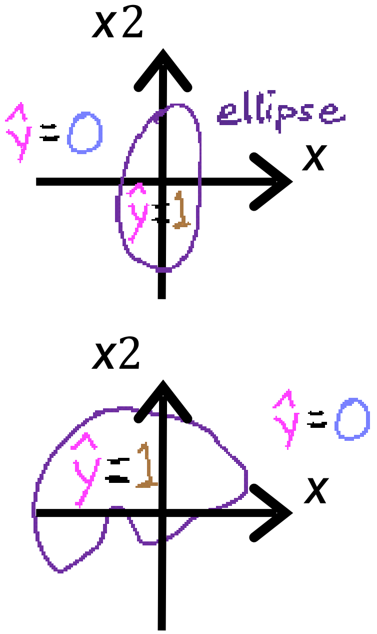
decision boundary

$$\overbrace{\underset{1}{w_1} x_1^2 + \underset{1}{w_2} x_2^2 + \underset{-1}{b}}^{z}$$

$$z = x_1^2 + x_2^2 - 1 = 0$$

$$x_1^2 + x_2^2 = 1$$

Non-linear decision boundaries



$$f_{\vec{w},b}(\vec{x}) = g(s) = g(r_1 x_1 + r_2 x_2 + r_3 x_1^2 + r_4 x_1 x_2 + r_5 x_2^2 + r_6 x_1^3 + \cdots + b)$$

Cost Function

Cost Function for Logistic Regression

Training set

	tumor size (cm) x_1	...	patient's age x_n	malignant? y
$i=1$	10		52	1
⋮	2		73	0
	5		55	0
	12		49	1
$i=m$	...		...	...

$i = 1, \ldots, m$ ← training examples

$j = 1, \ldots, n$ ← features

$\boxed{\text{target } y \text{ is } 0 \text{ or } 1}$

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w}\cdot\vec{x}+b)}}$$

How to choose $\vec{w} = [w_1 \quad w_2 \quad \cdots \quad w_n]$ and b?

Squared error cost

cost

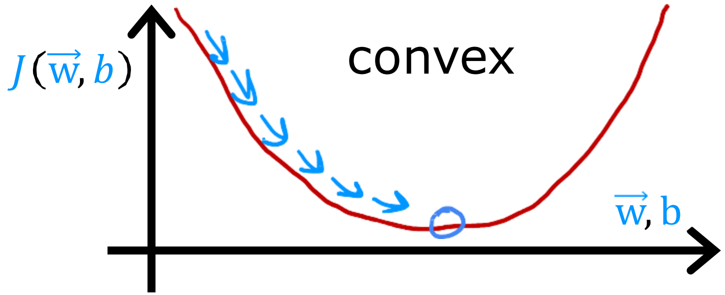
$$J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)^2$$

average of training set

loss $\boxed{L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right)}$

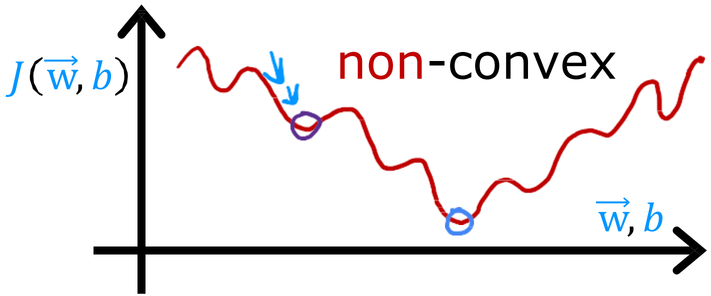
linear regression

$$f_{\vec{w},b}(\vec{x}) = \vec{w}\cdot\vec{x} + b$$



logistic regression

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w}\cdot\vec{x}+b)}}$$

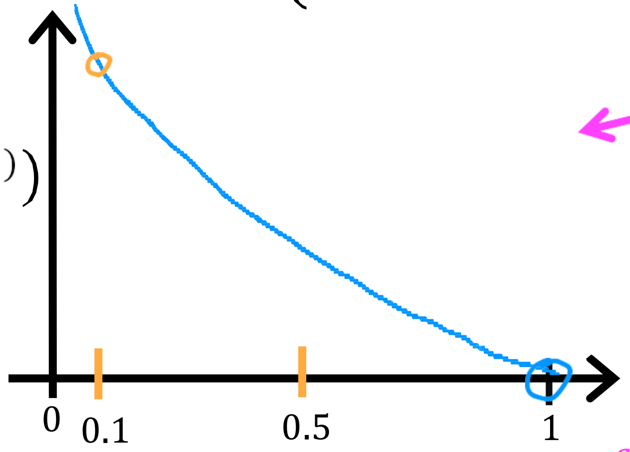


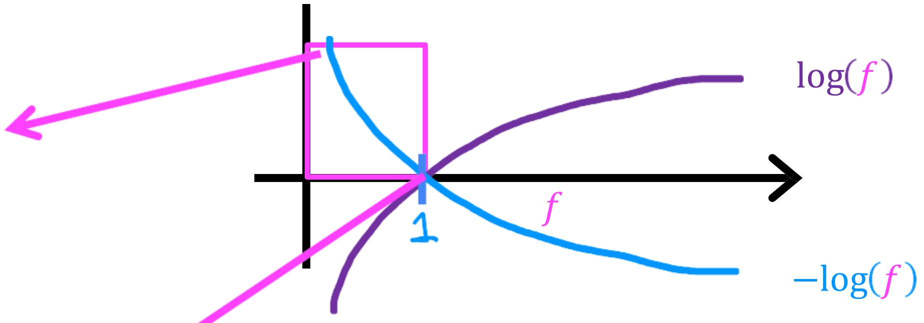
Logistic loss function

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 0 \end{cases}$$

$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right)$

if $y^{(i)} = 1$





$f_{\vec{w},b}\left(\vec{x}^{(i)}\right)$

As $f_{\vec{w},b}\left(\vec{x}^{(i)}\right) \rightarrow 1$ then loss $\rightarrow 0$

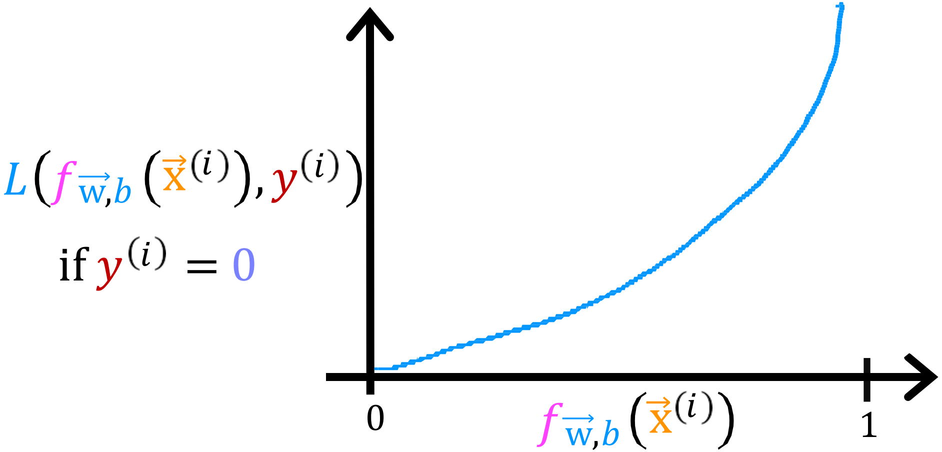
As $f_{\vec{w},b}\left(\vec{x}^{(i)}\right) \rightarrow 0$ then loss $\rightarrow \infty$

Loss is lowest when $f_{\vec{w},b}\left(\vec{x}^{(i)}\right)$ predicts close to true label $y^{(i)}$.

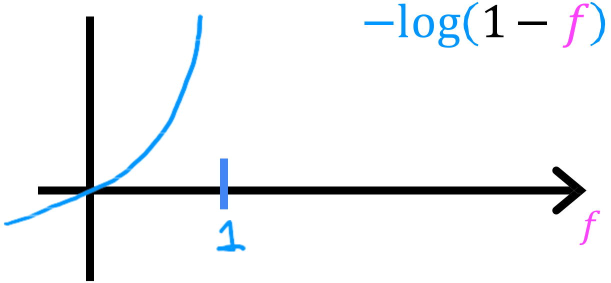
Logistic loss function

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right),y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 0 \end{cases}$$

As $f_{\vec{w},b}\left(\vec{x}^{(i)}\right) \to 0$ then loss $\to 0$!!



As $f_{\vec{w},b}\left(\vec{x}^{(i)}\right) \to 1$ then loss $\to \infty$!!



The further prediction $f_{\vec{w},b}\left(\vec{x}^{(i)}\right)$ is from target $y^{(i)}$, the higher the loss.

Cost

cost

$$J(\vec{w}, b) = \frac{1}{m} \sum_{i=1}^{m} L\underbrace{\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right)}_{\text{loss}}$$

$$= \begin{cases} -\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 0 \end{cases}$$

convex

↳ can reach a global minimum

find w, b that minimize cost J

Cost Function

Simplified Cost Function for Logistic Regression

Simplified loss function

$$L\left(f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right), y^{(i)}\right) = \begin{cases} \boxed{-\log\left(f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right)\right)} & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right)\right) & \text{if } y^{(i)} = 0 \end{cases}$$

$$\boxed{L\left(f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right), y^{(i)}\right) = -\,y^{(i)}\log\left(f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\log\left(1 - f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right)\right)}$$

$1 \qquad \underbrace{(1-1)}_{0}$

if $y^{(i)} = 1$:

$$L\left(f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right), y^{(i)}\right) = \underbrace{-1\log\left(f(\vec{x})\right)}$$

Simplified loss function

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 0 \end{cases}$$

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = -\,y^{(i)}\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right)$$

if $y^{(i)} = 1$:

$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = -1 \log\left(f(\vec{x})\right)$

if $y^{(i)} = 0$:

$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) =$

Simplified cost function

loss

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = -\underbrace{y^{(i)}\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right)}$$

convex

(single global minimum)

cost

$$J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}\left[\underbrace{L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right)}\right]$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right)\right]$$

maximum likelihood

Gradient Descent

Gradient Descent Implementation

Training logistic regression

Find $\vec{\mathrm{w}}, b$

Given new $\vec{\mathrm{x}}$, output $f_{\vec{\mathrm{w}},b}(\vec{\mathrm{x}}) = \dfrac{1}{1+e^{-(\vec{\mathrm{w}}\cdot\vec{\mathrm{x}}+b)}}$

$$P(y = 1|\vec{\mathrm{x}}; \vec{\mathrm{w}}, b)$$

Gradient descent

$$J(\vec{\mathrm{w}},b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{\mathrm{w}},b}(\vec{\mathrm{x}}^{(i)})\right) + (1-y^{(i)})\log\left(1-f_{\vec{\mathrm{w}},b}(\vec{\mathrm{x}}^{(i)})\right)\right]$$

repeat {

$$w_j = w_j - \alpha\frac{\partial}{\partial w}J\ \mathrm{w},b$$

$$b = b - \alpha\frac{\partial}{\partial b}J\ \mathrm{w},b$$

} simultaneous updates

$$\frac{\partial}{\partial w_j}J(\vec{\mathrm{w}},b) = \frac{1}{m}\sum_{i=}^{m}(f_{\vec{\mathrm{w}},b}(\vec{\mathrm{x}}^{(i)}) - y^{(i)})x^{(i)}$$

$$\frac{\partial}{\partial b}J(\vec{\mathrm{w}},b) = \frac{1}{m}\sum_{i=1}(f_{\vec{\mathrm{w}},b}(\vec{\mathrm{x}}^{(i)}) - y^{(i)})$$

Gradient descent for logistic regression

repeat {

looks like linear regression!

$$w_j = w_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} \right]$$

$$b = b - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right) \right]$$

} simultaneous updates

Same concepts:

- Monitor gradient descent (learning curve)
- Vectorized implementation
- Feature scaling

Linear regression $\quad f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$

Logistic regression $\quad f_{\vec{w},b}(\vec{x}) = \dfrac{1}{1 + e^{(-\vec{w} \cdot \vec{x} + b)}}$

Logistic regression hypothesis:

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

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

Cost function in logistic regression is:

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

Vectorized implementation:

$h = g(X\theta)$

$J(\theta) = \frac{1}{m}(-y^T log(h) - (1 - y)^T log(1 - h))$

The gradient of the cost is a vector of the same length as θ where j^{th} element (for $j = 0, 1, \ldots, n$) is defined as follows:

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

Newton's Method for parameter optimization:

Advantages :

Fast Convergence: Newton's method converges faster than gradient descent, especially for convex and well-behaved functions.

It typically achieves **quadratic convergence** near the optimum, meaning the error shrinks exponentially with each iteration.

Adaptability to Curvature: By using the Hessian, Newton's method adjusts the step size based on the curvature of the function, preventing overshooting or undershooting the minimum.

Precise Steps: The inclusion of second-order information allows for more precise updates, especially in regions where the function's curvature changes rapidly.

Newton's Method for Parameter Optimization:

- **Hessian Calculation**: Computing the Hessian matrix can be expensive, especially for functions with many parameters (large-scale optimization problems).
  The Hessian is an $n \times n$ matrix, where n is the number of parameters.

- **Hessian Inversion**: Newton's method requires inverting the Hessian matrix, which can be computationally expensive and numerically unstable if the Hessian is not well-conditioned (i.e., close to singular, i.e. eigenvalues are very small, or the determinant is close to zero).

- **Local Minimum**: Like gradient descent, Newton's method can get stuck in local minima if the function is not convex

Modified or Truncated Newton Methods

- Due to the computational cost of calculating and inverting the Hessian matrix, several modified versions of Newton's method are commonly used in practice.
- These methods make Newton's method more efficient for large-scale problems.

Quasi-Newton Methods:

•Instead of calculating the Hessian matrix explicitly, quasi-Newton methods approximate the Hessian using only first-order information (gradients).

•**BFGS (Broyden-Fletcher-Goldfarb-Shanno)** and **L-BFGS (Limited-memory BFGS)** are popular quasi-Newton methods.

• They approximate the Hessian based on gradient evaluations, making them much more efficient for large problems.

Truncated Newton (TNC):

•TNC uses a conjugate gradient approach to solve the optimization problem. It only computes an approximation of the Hessian and "truncates" the step when necessary to avoid going too far or using too much computational effort.

•This makes it more scalable for large problems while maintaining the benefits of second-order methods.

- For machine learning applications, Newton's methods are often used in training models like **logistic regression**, where the optimization problem involves minimizing a log-likelihood function.
- In these cases, algorithms like **BFGS** or **TNC** are used for efficient optimization especially in large-scale optimization.

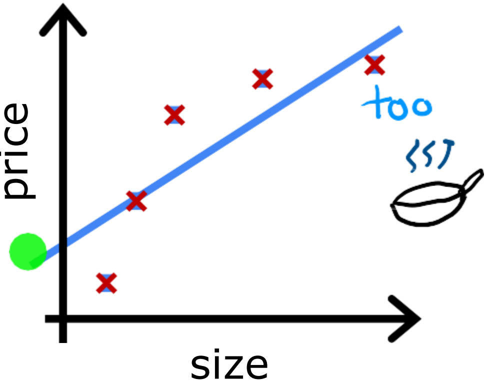
Summary of Solvers in Scikit-learn Logistic Regression

Solver	Type	Key Characteristics	Use Case
`lbfgs`	Quasi-Newton	Efficient for large datasets, default solver	Binary and multinomial classification
`newton-cg`	Newton-based	High precision, better for multinomial problems	Multinomial logistic regression
`sag`	Gradient descent	Efficient for large datasets, supports L2 regularization	Large datasets with L2 regularization
`saga`	Gradient descent	Supports L1, L2, and elastic net regularization	Large datasets, sparse data, high-dimensional problems
`liblinear`	Coordinate descent	Good for small datasets, L1/L2 regularization	Binary classification, L1 regularization

Regularization to Reduce Overfitting

The Problem of Overfitting

Regression example

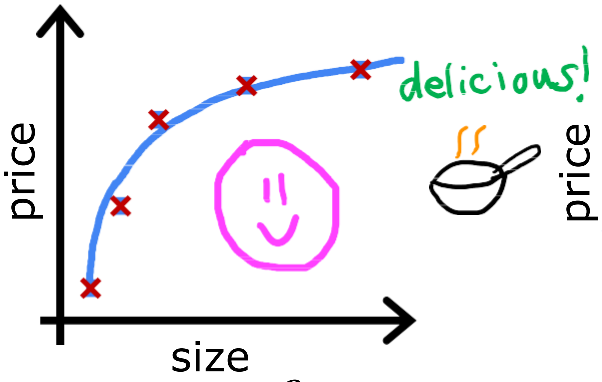


$w_1x + b$

Under fit

- Does not fit the training set well

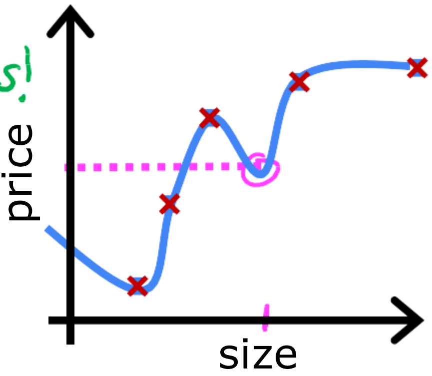
high bias



$w_1x + w_2x^2 + b$

- Fits training set pretty well

generalization



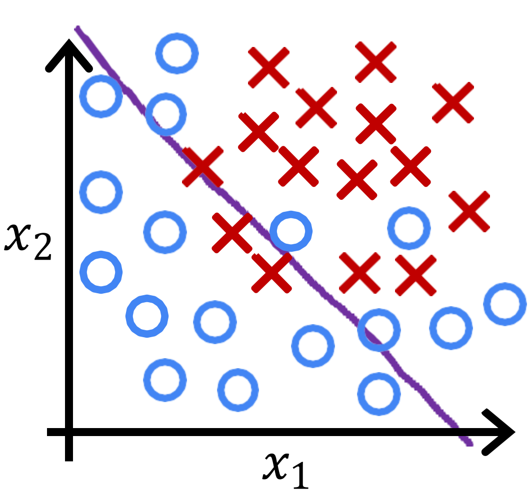
$w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b$

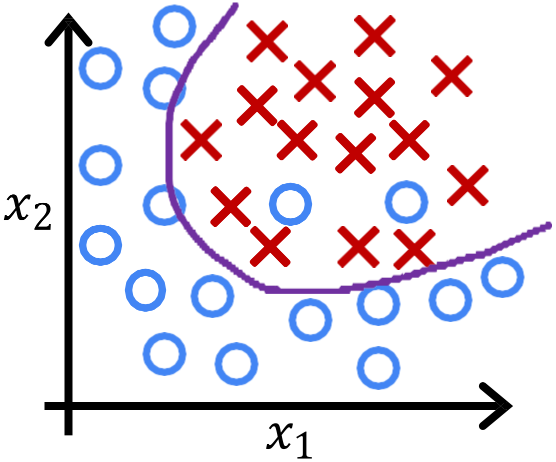
overfit

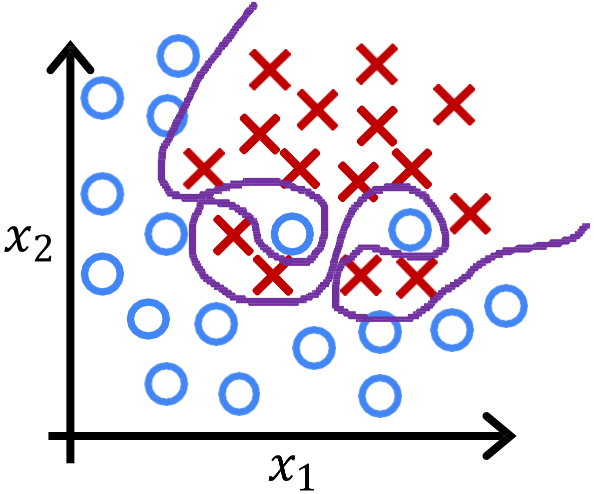
- Fits the training set extremely well

High variance

Classification







$z = w_1x_1 + w_2x_2 + b$

$f_{\vec{w},b}(\vec{x}) = g(z)$

g is the sigmoid function

underfit high bias

$z = w_1x_1 + w_2x_2$
$+w_3x_1^2 + w_4x_2^2$
$+w_5x_1x_2 + b$

just right

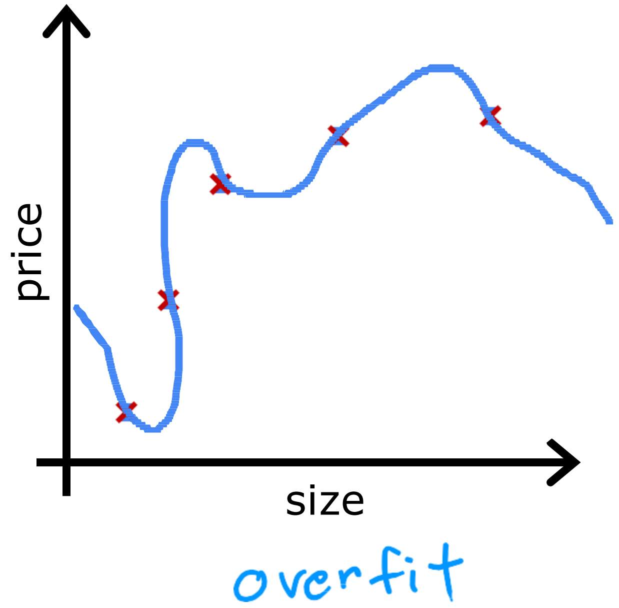
$z = w_1x_1 + w_2x_2$
$+w_3x_1^2x_2 + w_4x_1^2x_2^2$
$+w_5x_1^2x_2^3 + w_6x_1^3x_2$
$+ \cdots + b$

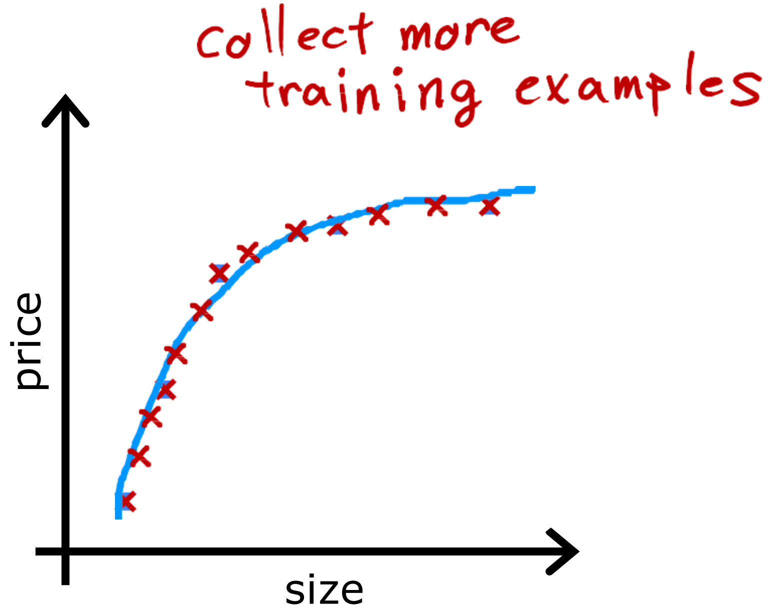
overfit

Regularization to Reduce Overfitting

Addressing Overfitting

Collect more training examples





Select features to include/exclude

size	bedrooms	floors	age	avg income	...	distance to coffee shop	price
x_1	x_2	x_3	x_4	x_5		x_{100}	y

all features

+

insufficient data

↓

overfit

selected features

size
bedrooms
age
just right

feature selection

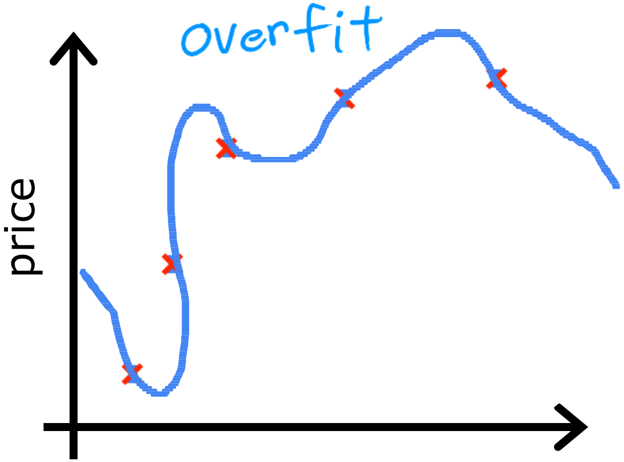
disadvantage

↓

useful features could be lost

Regularization

Reduce the size of parameters w_j

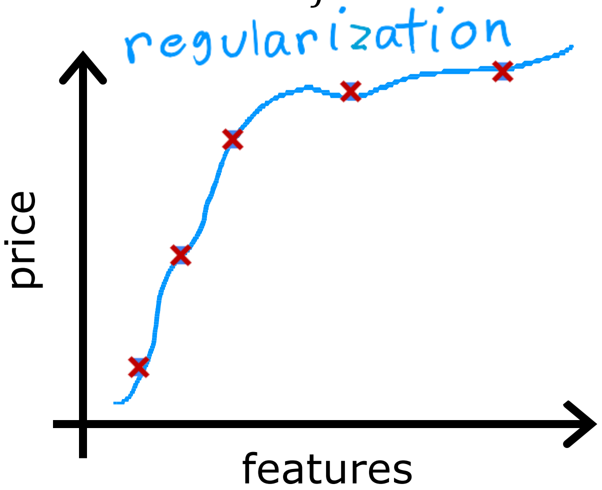


features

$$f\ x\ = 28x - 385x^2 + 39x^3 - 174x^4 + 100$$

eliminate feature

large values w_j



$$f(x) = 13x - 0.23x^2 + \underbrace{0.000014x^3}_{} - \underbrace{0.0001}_{}\ x^4 + 10$$

small values for w_j

Addressing overfitting

Options

1. Collect more data

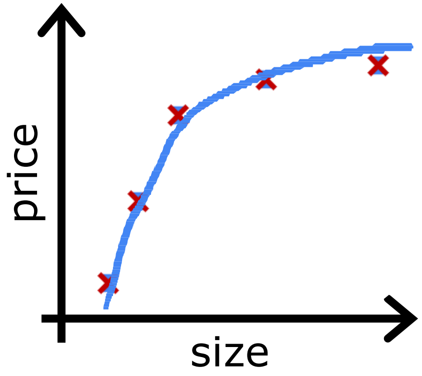
2. Select features
   – Feature selection

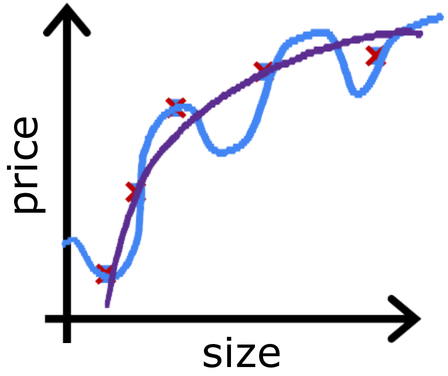
3. Reduce size of parameters
   – "Regularization"

Regularization to Reduce Overfitting

Cost Function with Regularization

Intuition





$w_1x + w_2x^2 + b$

$w_1x + w_2x^2 + \cancel{\underbrace{w_3}_{\approx 0}x^3} + \cancel{\underbrace{w_4}_{\approx 0}x^4} + b$

make w_3, w_4 really small (≈ 0)

$$\min_{\vec{w},b} \frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)^2 + \cancel{1000\underbrace{w_3^2}_{0.001}} + \cancel{1000\underbrace{w_4^2}_{0.002}}$$

Regularization

small values $w_1, w_2, \cdots, w_n, b$

simpler model $w_3 \approx 0$

less likely to overfit $w_4 \approx 0$

size	bedrooms	floors	age	avg income	...	distance to coffee shop	price
x_1	x_2	x_3	x_4	x_5		x_{100}	y

n features

$n = 100$

$w_1, w_1, w_2, \cdots, w_{100}, b$

$$J(\vec{w}, b) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)^2 + \overbrace{\frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2}^{\text{regularization term}} + \frac{\lambda}{2m} b^2\right]$$

can include or exclude b

"lambda"
regularization parameter

$\lambda > 0$

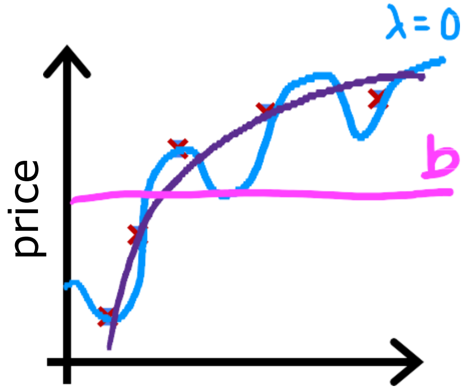
Regularization

$$\min_{\vec{w},b} J(\vec{w},b) = \min_{\vec{w},b}\left[\overbrace{\frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2}^{\text{mean squared error}} + \overbrace{\frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2}^{\text{regularization term}}\right]$$

fit data

Keep w_j small

λ balances both goals



choose $\lambda = 10^{10}$

$$f_{\vec{w},b}(\vec{x}) = \underbrace{\cancel{w_1 x}}_{\approx 0} + \underbrace{\cancel{w_2 x^2}}_{\approx 0} + \underbrace{\cancel{w_3 x^3}}_{\approx 0} + \underbrace{\cancel{w_4 x^4}}_{\approx 0} + b$$

$f(x) = b$

Choose λ

Regularization to Reduce Overfitting

Regularized Linear Regression

Regularized linear regression

$$\min_{\vec{w},b} J(\vec{w},b) = \min_{\vec{w},b}\left[\frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2\right]$$

Gradient descent

repeat {

$$w_j = w_j - \alpha \frac{\partial}{\partial w} J(\vec{w},b) \qquad = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{m} w_j$$

$j = 1, \ldots, n$

$$b = b - \alpha \frac{\partial}{\partial b} J(\vec{w},b) \qquad = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)$$

} simultaneous update

Implementing gradient descent

repeat {

$$w_j = w_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left[\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} \right] + \frac{\lambda}{m} w_j \right]$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right)$$

} simultaneous update $j = 1 \ldots n$

Implementing gradient descent

repeat {

$$w_j = w_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left[\left(f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right) - y^{(i)} \right) x_j^{(i)} \right] + \frac{\lambda}{m} w_j \right]$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{\mathrm{w}},b}\left(\vec{\mathrm{x}}^{(i)}\right) - y^{(i)} \right)$$

} simultaneous update $j = 1 \ldots n$

$$w_j = \underbrace{1 w_j - \alpha \frac{\lambda}{m} w_j}_{w_j \left(1 - \alpha \frac{\lambda}{m}\right)} \underbrace{- \alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_{w,b}\left(\vec{x}^{(i)}\right) - y^{(i)} \right) x_j^{(i)}}_{\text{usual update}}$$

$$w_j \underbrace{\left(1 - \alpha \frac{\lambda}{m}\right)}_{\text{shrink } w_j}$$

$$\alpha \frac{\lambda}{m}$$

$$0.01 \frac{1}{50} = 0.0002$$

$$w_j \left(\underbrace{1 - 0.0002}_{0.9998} \right)$$

How we get the derivative term (optional)

$$\frac{\partial}{\partial w} J(\vec{\mathrm{w}}, b) = \frac{d}{dw_j}\left[\frac{1}{2m}\sum_{i=1}^{m}\left(\underbrace{f(\vec{x}^{(i)})}_{\vec{w}\cdot\vec{x}^{(i)}+b} - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2\right]$$

$$= \frac{1}{\cancel{2}m}\sum_{i=1}^{m}\left[\left(\vec{w}\cdot\vec{x}^{(i)} + b - y^{(i)}\right)\cancel{2}x_j^{(i)}\right] + \frac{\lambda}{\cancel{2}m}\cancel{2}w_j \quad \text{No } \sum_{j=1}^{n}$$

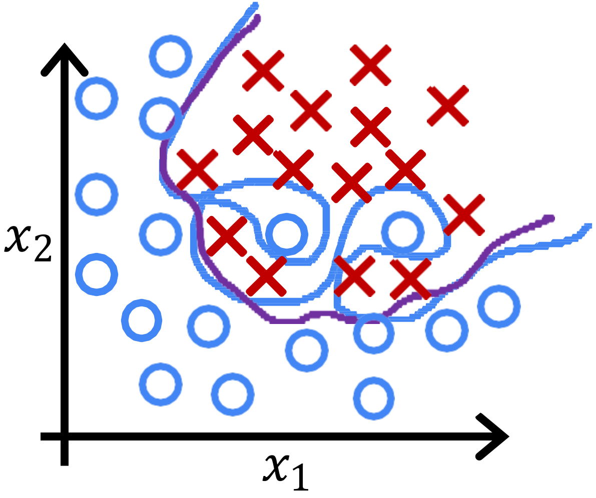
$$= \frac{1}{m}\sum_{i=1}^{m}\left[\left(\underbrace{\vec{w}\cdot\vec{x}^{(i)} + b}_{f(\vec{x})} - y^{(i)}\right)x_j^{(i)}\right] + \frac{\lambda}{m}w_j$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left[\left(f_{\vec{\mathrm{w}},b}(\vec{\mathrm{x}}^{(i)}) - y^{(i)}\right)x_j^{(i)}\right] + \frac{\lambda}{m}w_j$$

Regularization to Reduce Overfitting

Regularized Logistic Regression

Regularized logistic regression



$$z = w_1x_1 + w_2x_2 + w_3x_1^2x_2 + w_4x_1^2x_2^2 + w_5x_1^2x_2^3 + \cdots + b$$

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-z}}$$

Cost function

$$J(\vec{w},b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y^{(i)})\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

$$\min_{\vec{w},b} J(\vec{w},b) \rightarrow w_j \downarrow$$

Regularized logistic regression

$$J(\vec{w},b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y^{(i)})\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

$\min_{\vec{w},b}$

Gradient descent

repeat {

$$w_j = w_j - \alpha\frac{\partial}{\partial w_j}J(\vec{w},b)$$

$j=1..n$

$$b = b - \alpha\frac{\partial}{\partial b}J(\vec{w},b)$$

}

$$= \frac{1}{m}\sum_{i=1}^{m}\left[\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)}\right] + \frac{\lambda}{m}w_j$$

logistic regression

$$= \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)$$

don't have to regularize b