CLASSIFICATION AND LOGISTIC REGRESSION

DR. FARHAD RAZAVI

OUTLINE

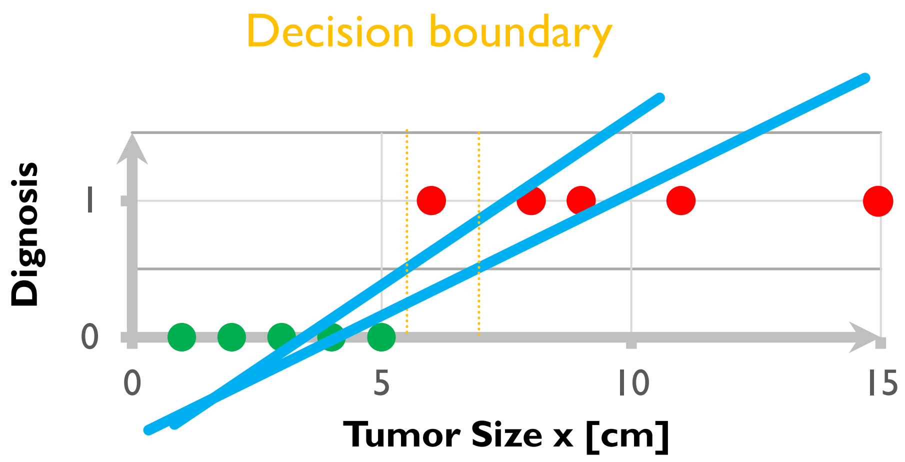
- Logistic regression
- Logistic cost function
- Gradient descent on Logistic regression
- Problem of overfitting

LOGISTIC REGRESSION

- Supervised Learning
- Classification or Categorization
- Binary Classifier
  - Online advertisement: Click or no click
  - Medical: Benign or Malignant
  - Political: Vote for candidate or not vote for a candidate
- Binary classifier has only two outcome. True or False! 0 or 1!

USING LINEAR REGRESSION

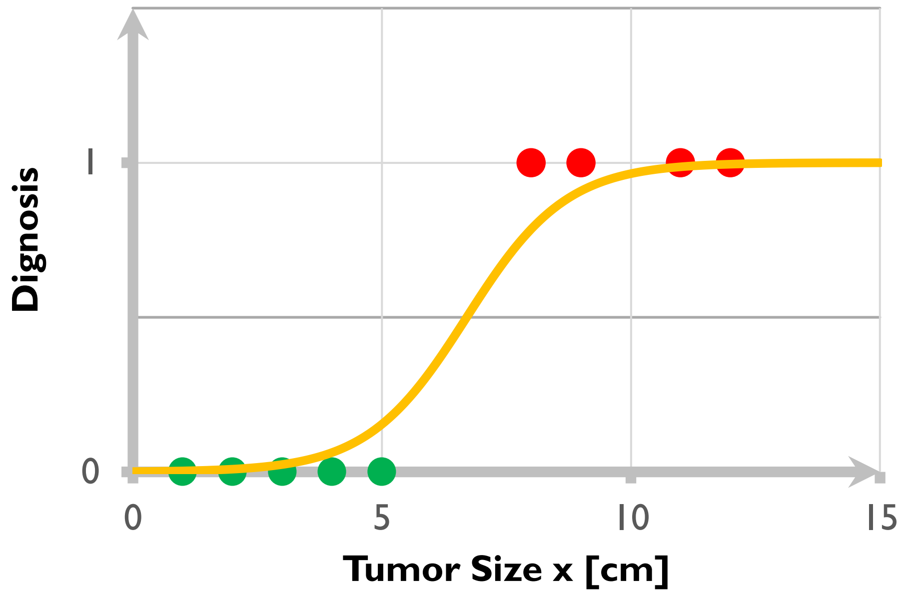
- $f_{w,b}(x) = wx + b$

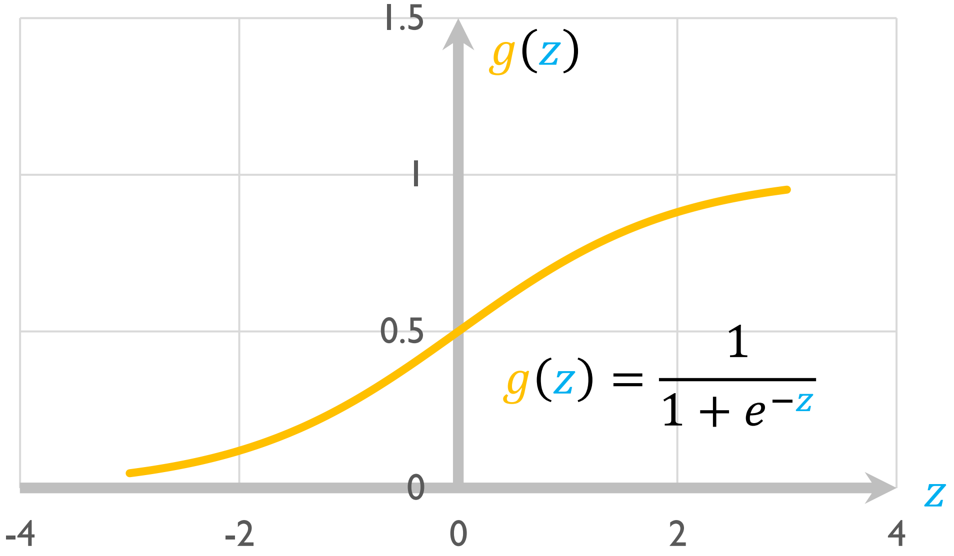


- The model has a continuous output prediction and does not predict 0 or 1.
- What if we assign a threshold?!

$$\hat{y} = \begin{cases} 1 & if \quad wx + b > 0.5 \\ 0 & if \quad wx + b \leq 0.5 \end{cases}$$

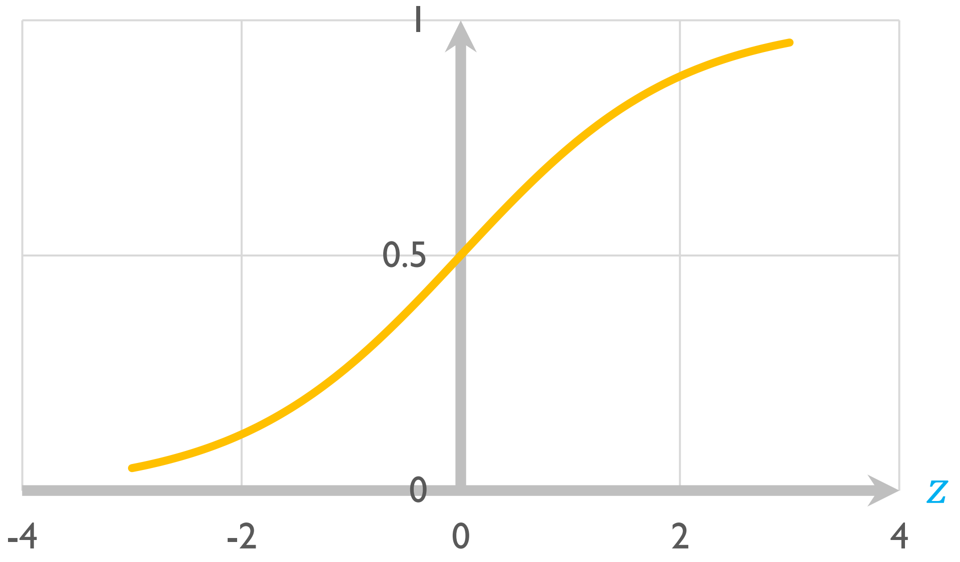
LOGISTIC REGRESSION





- Sigmoid function (logistic function).
- It outputs a value between 0, 1
- Now the value can be interpreted as the probability of outcome being 1.

MULTI PARAMETER LOGISTIC REGRESSION



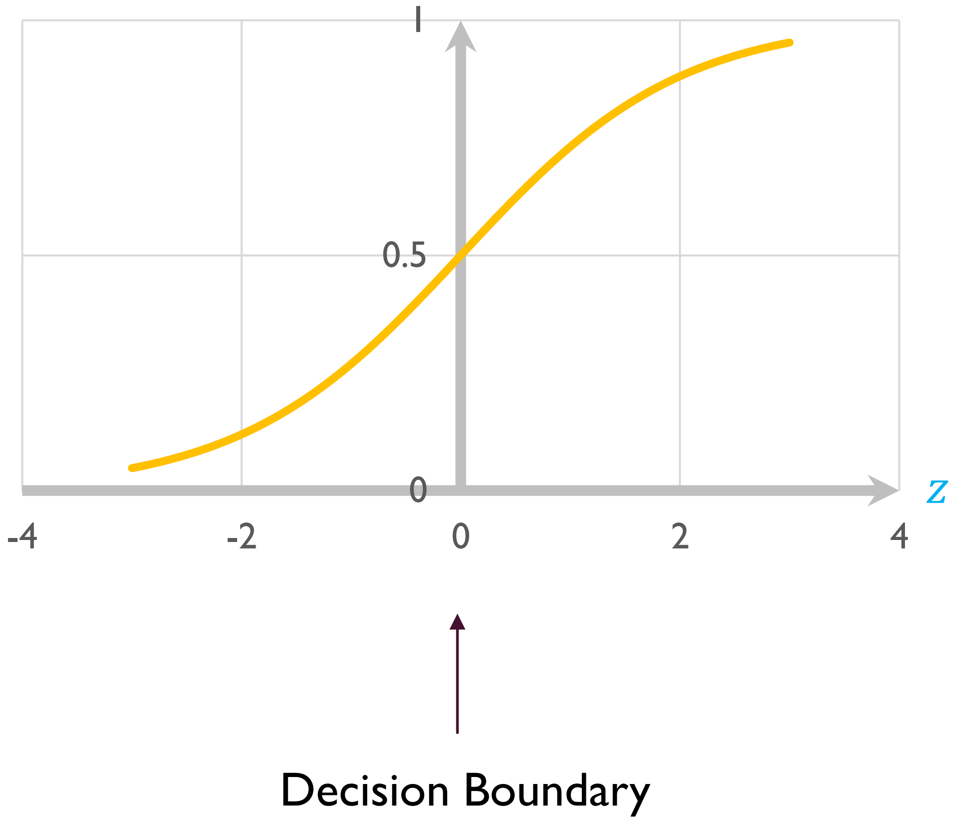
$$f_{\vec{w},b}(\vec{x})$$

$$\begin{cases} z = \vec{w}.\vec{x} + b \\ g(z) = \dfrac{1}{1+e^{-z}} \end{cases}$$

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

- $0 < g(z) < 1$
- Probability that the class under the test is 1
- $f_{\vec{w},b}(\vec{x})$ = 0.7 means that there is 70% chance that the tumor is malignant or y = 1.
- $P(y=0)+ P(y=1) = 1.$

DECISION BOUNDARY

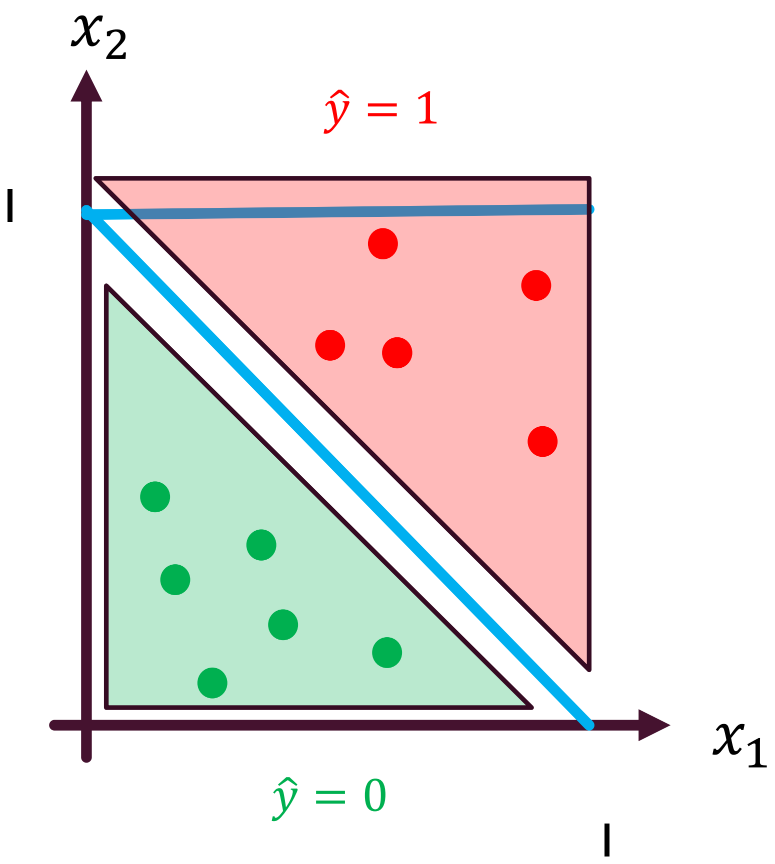


$$f_{w,b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w}.\vec{x}+b)}}$$

- $f_{w,b}(\vec{x}) = g(\vec{w}.\vec{x} + b)$

  $= P(y = 1|\vec{x};\ \vec{w}, b)$
- Our model now predicts probability of outcome is a certain class.
- We want to make decision whether $\hat{y} = 0$ or $\hat{y} = 1$
- A reasonable case is when $f_{w,b}(\vec{x}) = 0.5$
  - This point corresponds to $z = \vec{w}.\vec{x} + b = 0$

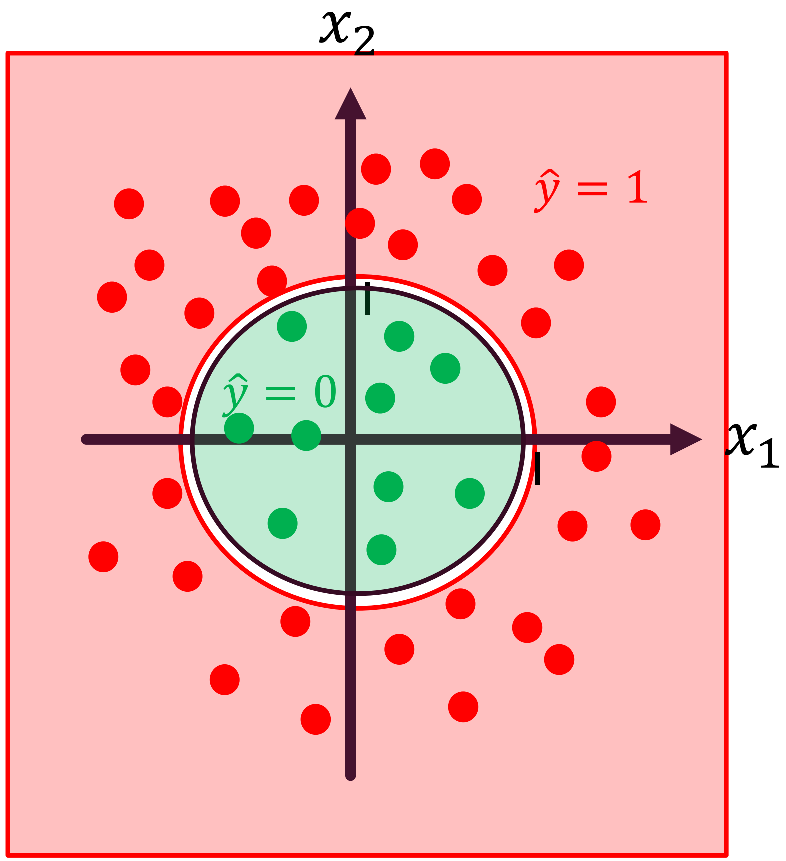
DECISION BOUNDARY FOR MULTIPLE VARIABLES



- $f_{w,b}(\vec{x}) = \text{g}(\text{z}) = \text{g}(w_1 x_1 + w_2 x_2 + b)$
- $z = w_1 x_1 + w_2 x_2 + b = 0$

- $w_1 = 0, w_2 = 1, b = -1$
  - $z = 0 \times x_1 + 1 \times x_2 - 1 = 0$
  - $x_2 = 1$ (Not a good decision boundary)
- $w_1 = 1, w_2 = 1, b = -1$
  - $z = 1 \times x_1 + 1 \times x_2 - 1 = 0$
  - $x_1 + x_2 = 1$ (a descent decision boundary)

NON-LINEAR DECISION BOUNDARY

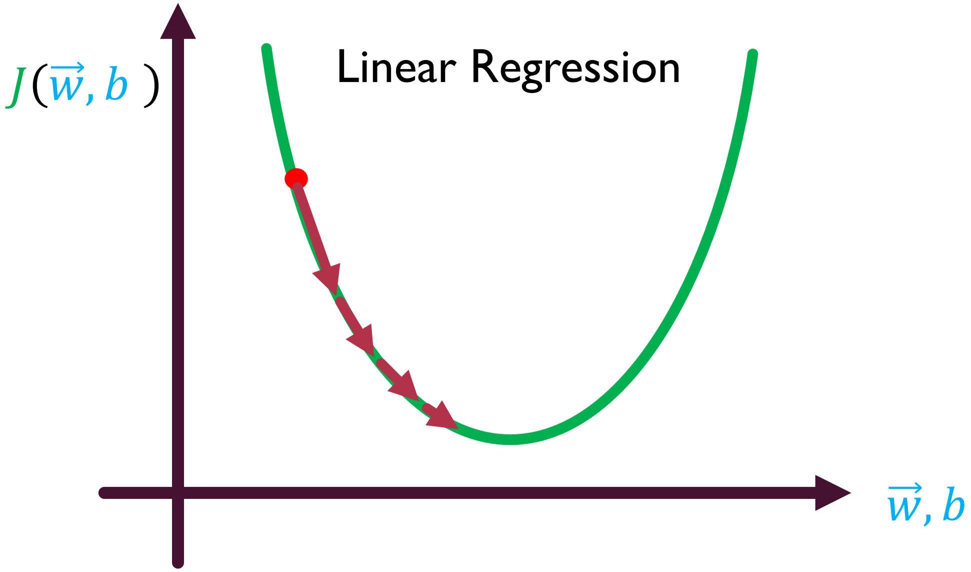


- $f_{w,b}(\vec{x}) = g(z) = g\left(w_1 x_1{}^2 + w_2 x_2{}^2 + b\right)$
- $z = w_1 x_1{}^2 + w_2 x_2{}^2 + b = 0$
- $w_1 = 1, w_2 = 1, b = -1$
  - $z = 1 \times x_1{}^2 + 1 \times x_2{}^2 - 1 = 0$
  - $x_1{}^2 + x_2{}^2 = 1$ (a descent decision boundary)
  - $x_1{}^2 + x_2{}^2 < 1 \rightarrow \hat{y} = 0$
  - $x_1{}^2 + x_2{}^2 > 1 \rightarrow \hat{y} = 1$

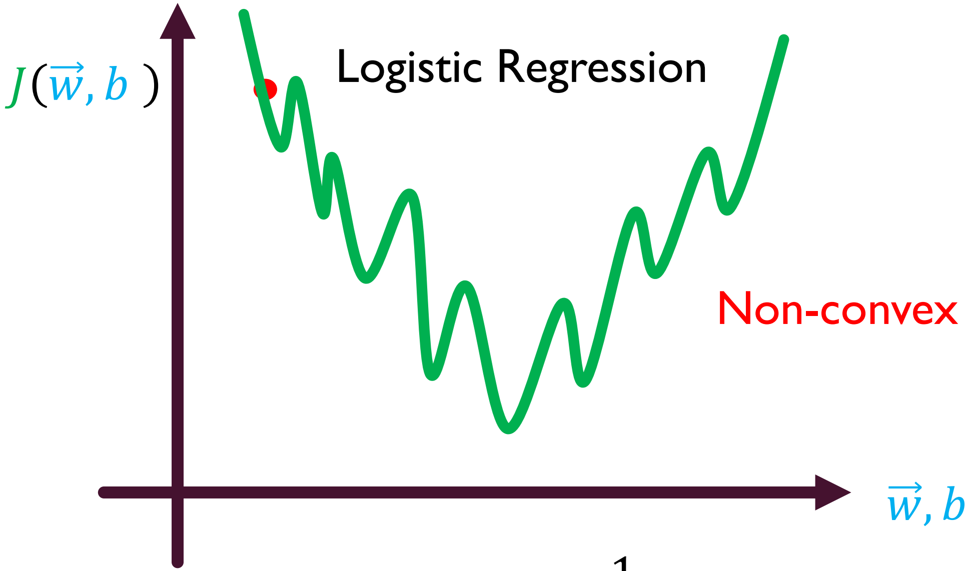
COST FUNCTION

- Can we use the squared error cost like Linear regression?

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



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

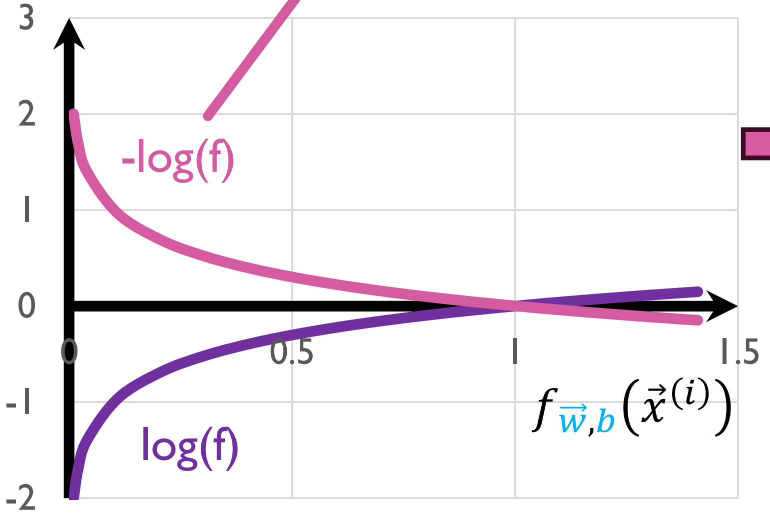


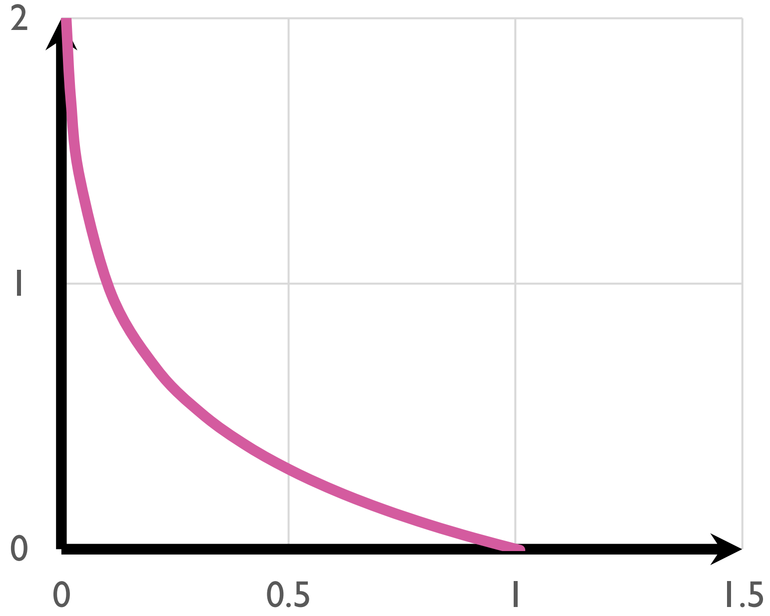
$$f_{w,b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w}.\vec{x}+b)}}$$

LOGISTIC COST FUNCTION

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

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



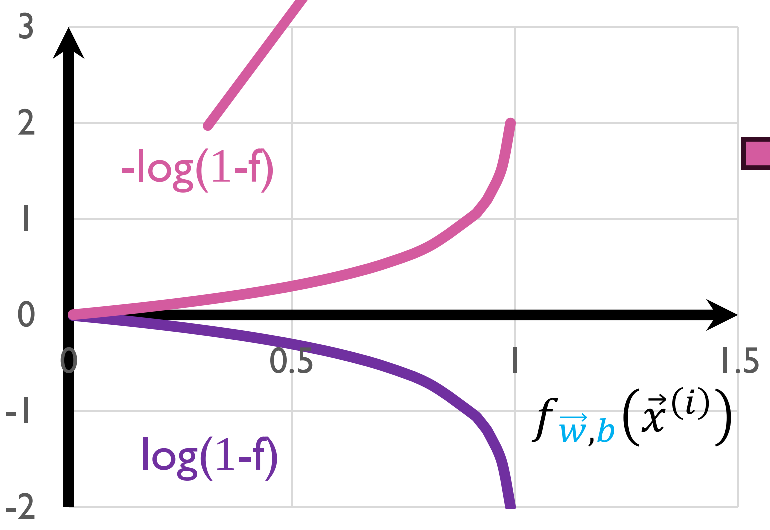


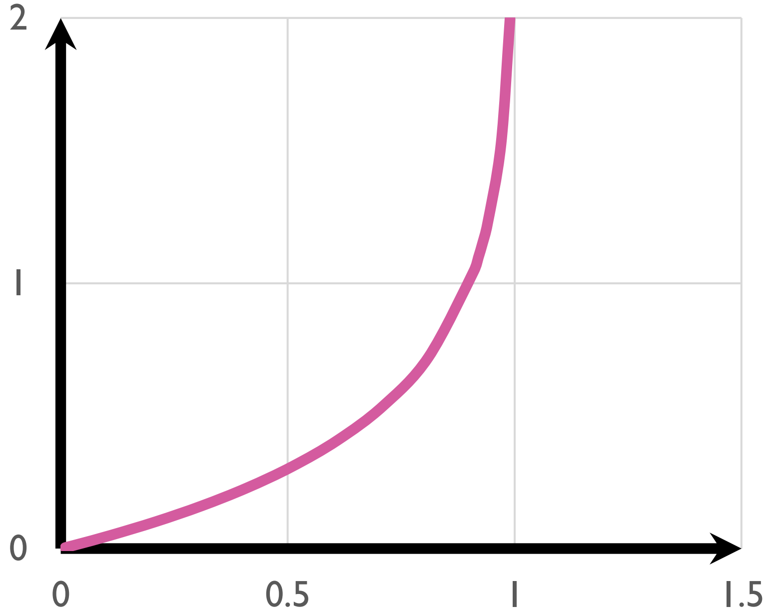
- If $y^{(i)} = 1$
  - If $f_{\vec{w},b}\left(\vec{x}^{(i)}\right) \to 1$ then Loss $\to 0$
  - If $f_{\vec{w},b}\left(\vec{x}^{(i)}\right) \to 0$ then Loss $\to \infty$
- Loss is lowest if our model predicts $f_{\vec{w},b}\left(\vec{x}^{(i)}\right)$ close to $y^{(i)} = 1$.
- Loss is extremely high if our model predicts $f_{\vec{w},b}\left(\vec{x}^{(i)}\right)$ far from $y^{(i)} = 1$.

LOGISTIC COST FUNCTION

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

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





- If $y^{(i)} = 0$
  - If $f_{\vec{w},b}\left(\vec{x}^{(i)}\right) \to 0$ then Loss $\to 0$
  - If $f_{\vec{w},b}\left(\vec{x}^{(i)}\right) \to 1$ then Loss $\to \infty$
- Loss is lowest if our model predicts $f_{\vec{w},b}\left(\vec{x}^{(i)}\right)$ close to $y^{(i)} = 0$.
- Loss is extremely high if our model predicts $f_{\vec{w},b}\left(\vec{x}^{(i)}\right)$ far from $y^{(i)} = 0$.

LOGISTIC COST FUNCTION

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

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

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

- To recap we defined cost function as an averaged summation of little loss terms $\left(\hat{y}^{(i)}, y^{(i)}\right) = \left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right)$.
- It can be shown that this cost function is a convex function with no local minima and can reach a global minimum.
- This cost function is derived from maximum likelihood estimation.

TRAINING LOGISTIC REGRESSION

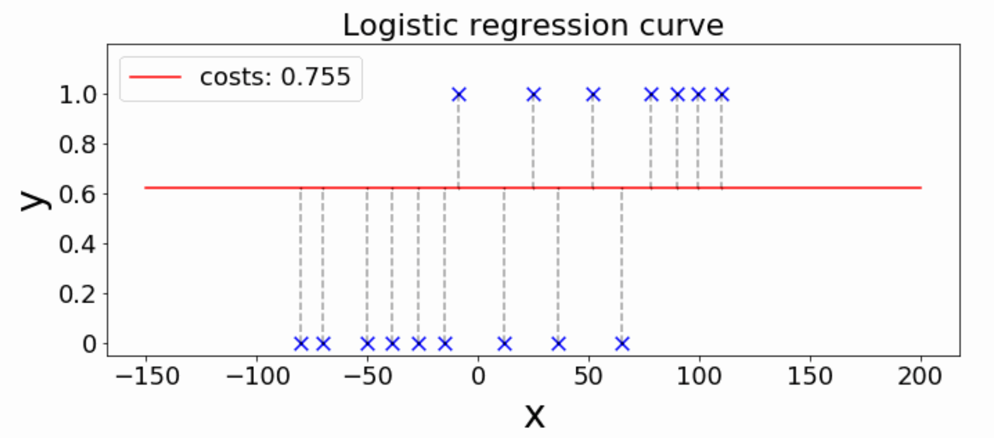
- Gradient descent for logistic regression:
- $J(\vec{w}, b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right)\right]$
- Repeat for $\mathrm{j} = 1, 2, \ldots, \mathrm{n}$ {

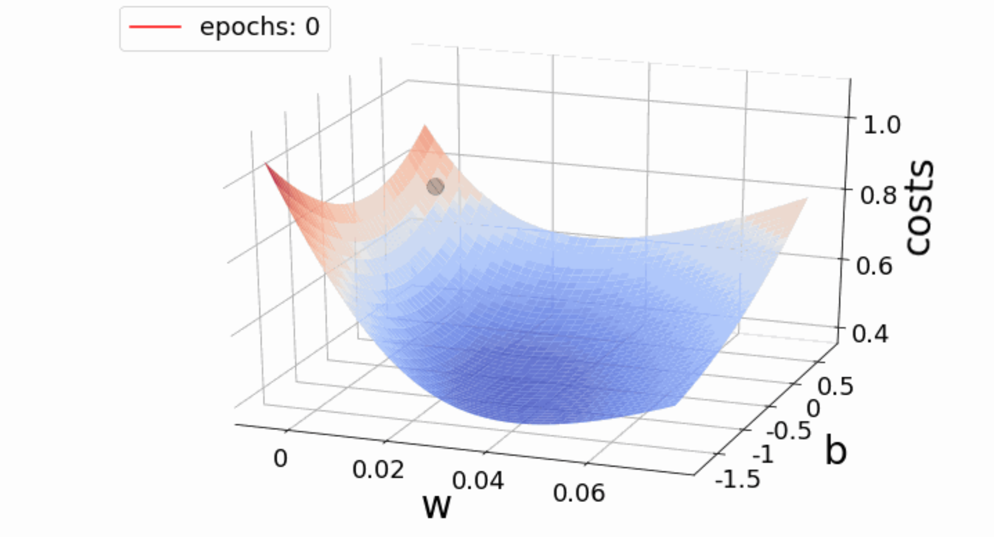
$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b); \qquad \frac{\partial}{\partial w_j} J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)x^{(i)}$$

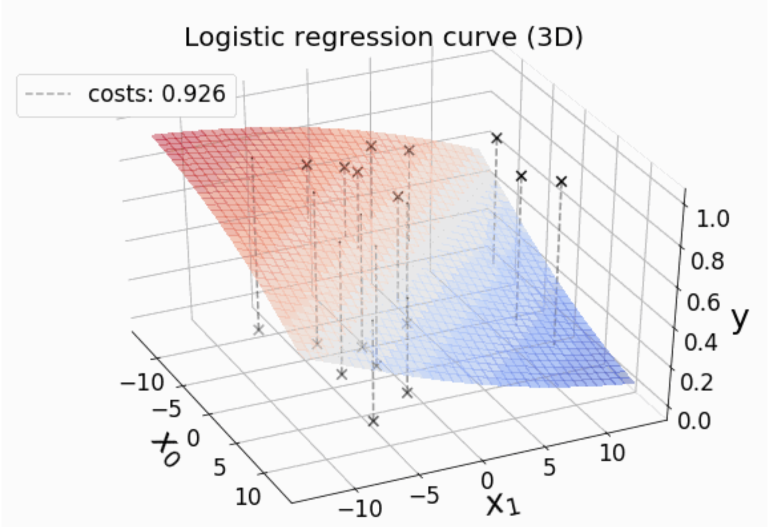
$$b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b); \qquad \frac{\partial}{\partial b} J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)$$

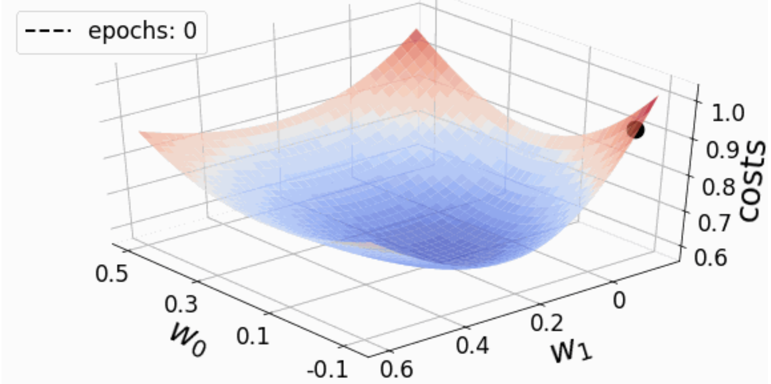
} simultaneous updates

LOGISTIC REGRESSION EXAMPLES (ONE AND TWO W-VECTOR)

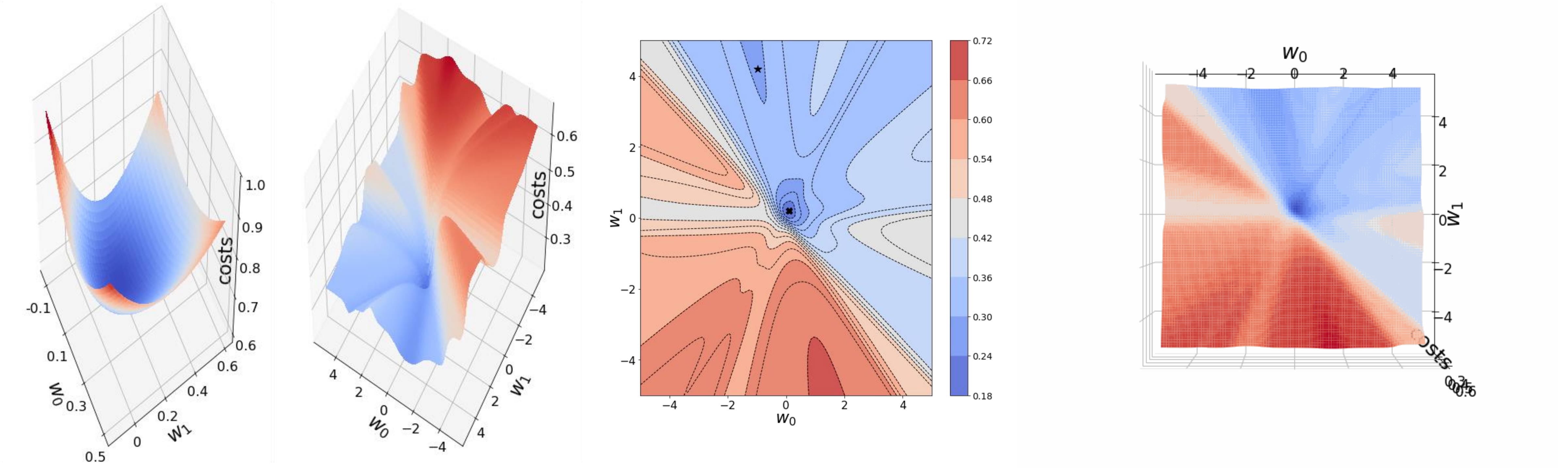






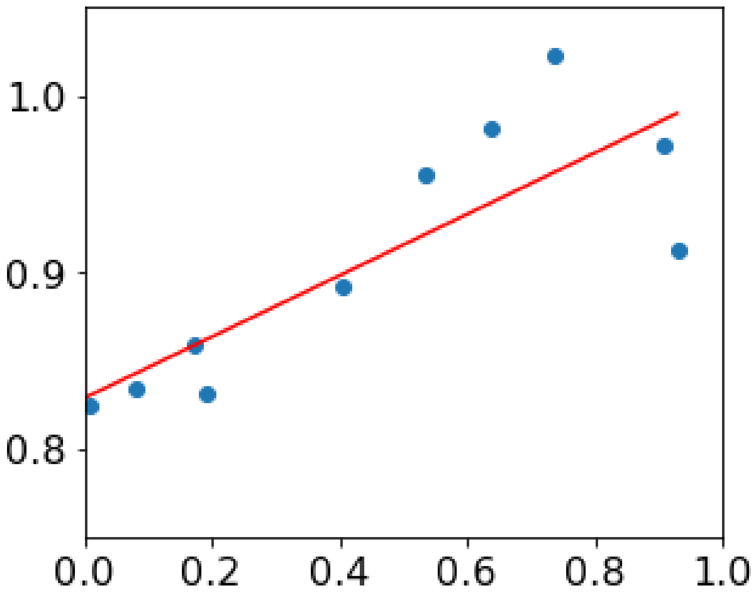


LOGISTIC LOSS VS SQUARED ERROR COST FUNCTION



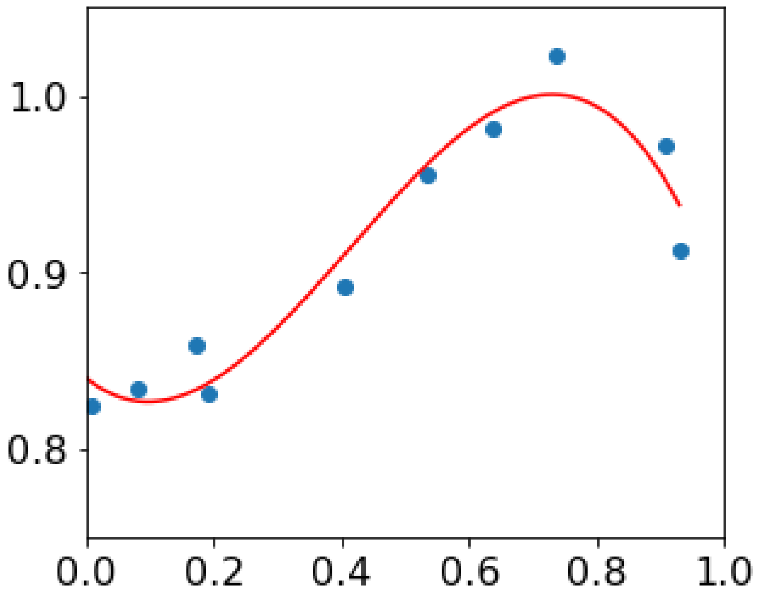
https://towardsdatascience.com/animations-of-logistic-regression-with-python-31f8c9cb420

OVERFITTING



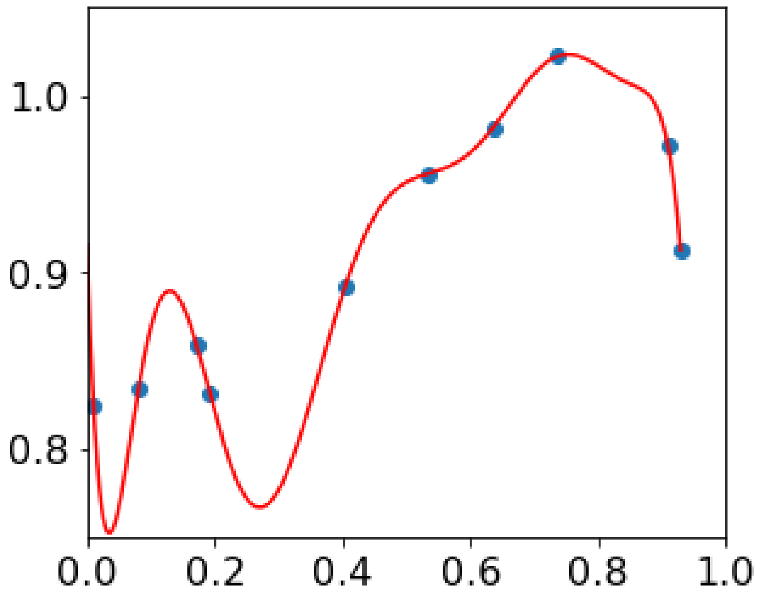
$w_1x_1 + b$

- Underfit
- Does not fit the training set well.
- High Bias



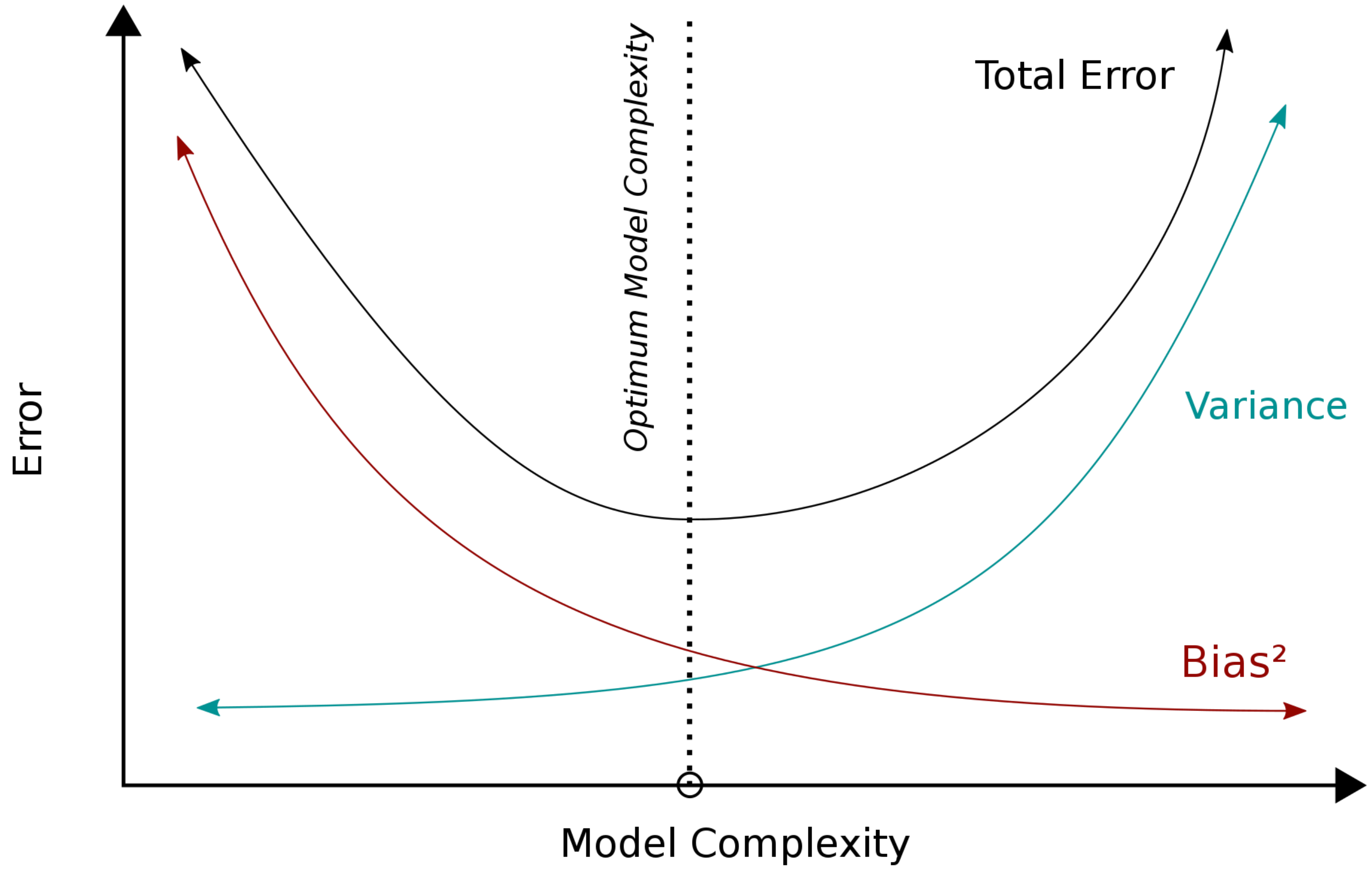
$w_1x_1 + w_2{x_1}^2 + w_3{x_1}^3 + b$

- Fit
- Fits training set very well.
- Generalization

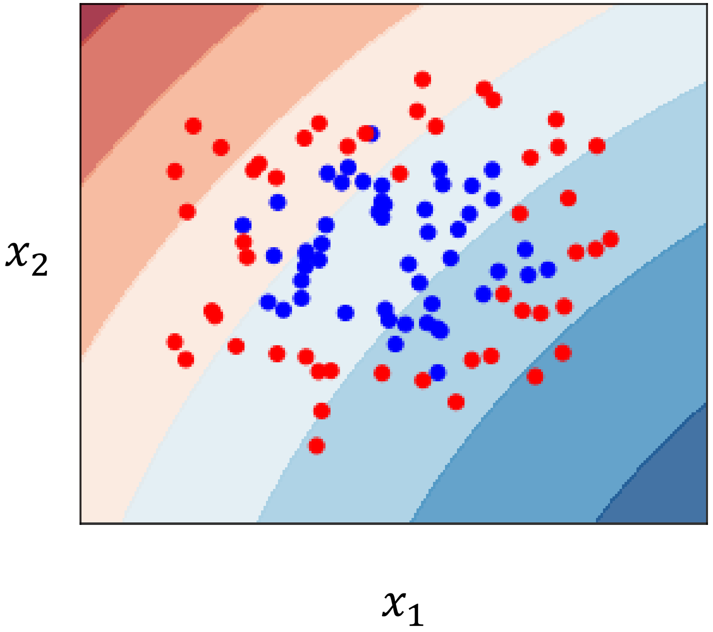


$w_1x_1 + w_2{x_1}^2 + w_3{x_1}^3 + w_4{x_1}^4 + w_5{x_1}^5 + w_6{x_1}^6 + w_7{x_1}^7 + w_8{x_1}^8 + w_9{x_1}^9 + b$

- Overfit
- Fits the training set extremely well.
- High variance

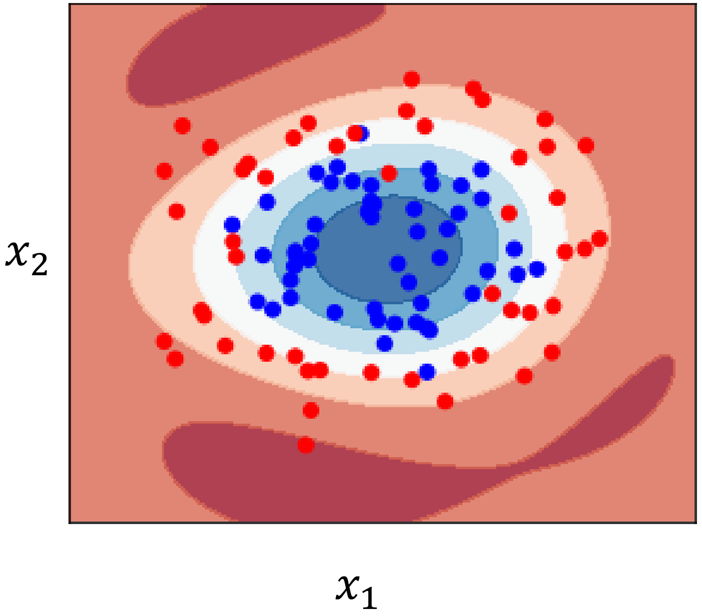

Optimum Model Complexity
Total Error
Variance
Bias²
Error
Model Complexity

OVERFITTING IN CLASSIFICATION

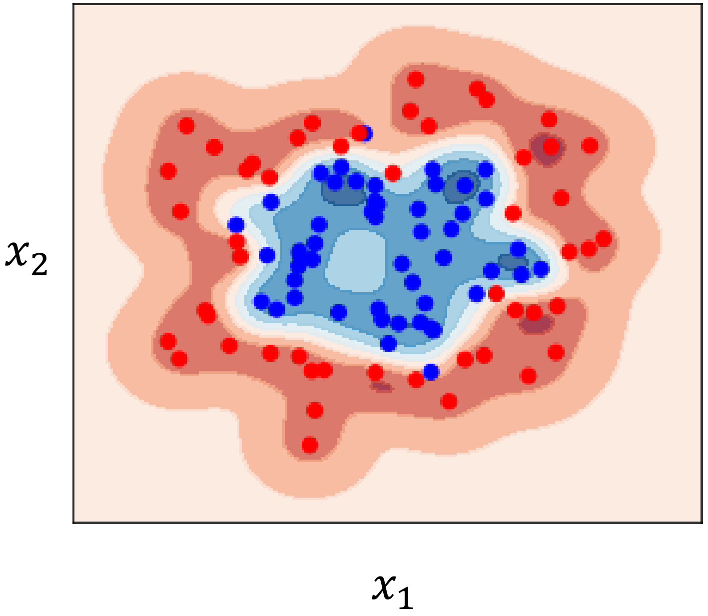


Underfit

High Bias



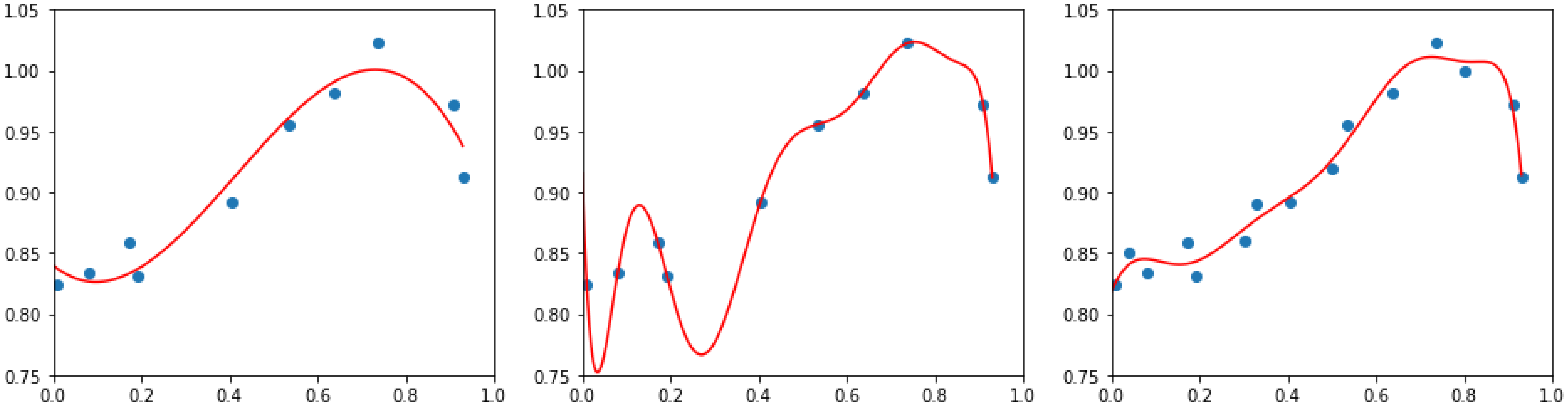
Fit



Overfit

High Variance

SOLUTION TO OVERFITTING



$w_1 x_1 + w_2 x_1{}^2 + w_3 x_1{}^3 + b$

$w_1 x_1 + w_2 x_1{}^2 + w_3 x_1{}^3 + w_4 x_1{}^4 + w_5 x_1{}^5 + w_6 x_1{}^6 + w_7 x_1{}^7 + w_8 x_1{}^8 + w_9 x_1{}^9 + b$

$w_4, w_5, w_6, w_7, w_8, w_9 \neq 0$

$w_1 x_1 + w_2 x_1{}^2 + w_3 x_1{}^3 + w_4 x_1{}^4 + w_5 x_1{}^5 + w_6 x_1{}^6 + w_7 x_1{}^7 + w_8 x_1{}^8 + w_9 x_1{}^9 + b$

$w_4, w_5, w_6, w_7, w_8, w_9 \rightarrow 0$

- One solution is to collect more data.
- More data will force the model to adjust the parameters in order to get a better fit for the data.
- Sometimes there are no ways to gather more data and this method might not work.

SOLUTION TO OVERFITTING

Size x_1	Bedrooms x_2	Floors x_3	Age x_4	Avg income x_5	…	Distance to coffee shop x_{100}	Price y
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

All features
+
Insufficient data

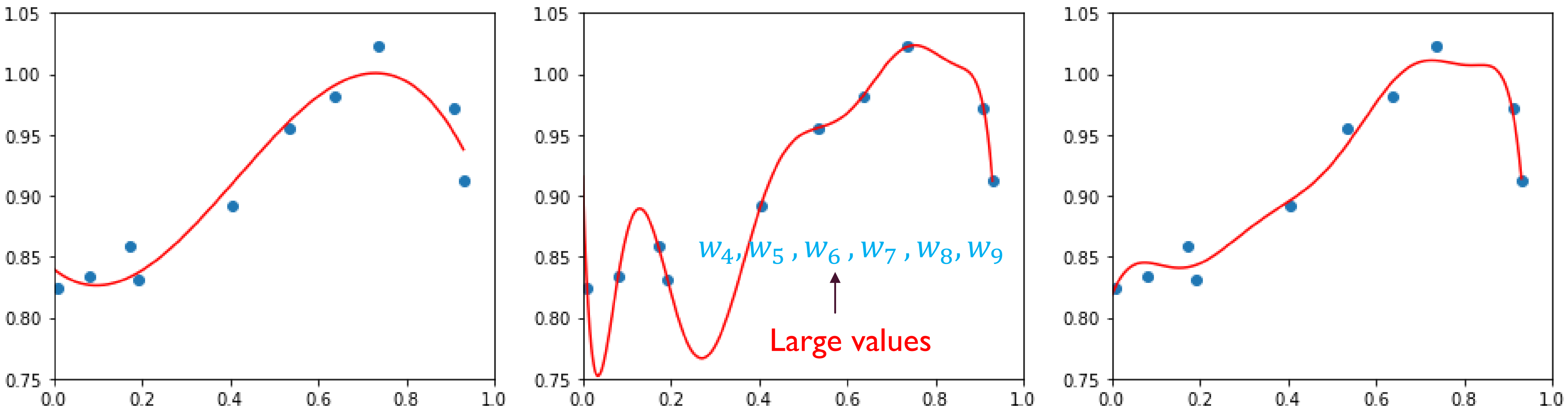
Over fit

selected features
+
(size, bedrooms, age)

Just right

- Another solution is to select only a few relevant features and therefore reduce the dimensionality of the model.
- Feature selection is dependent on the designer intuition. One disadvantage is that some of the ignored features might have been important.

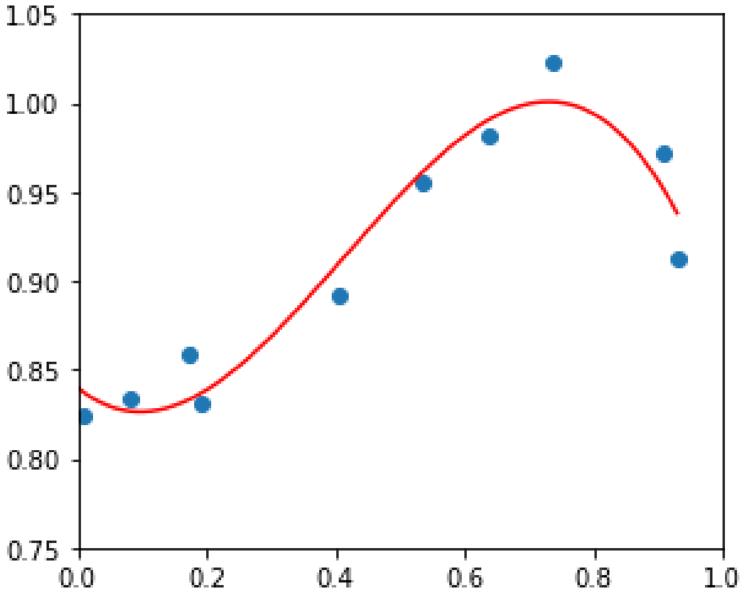
SOLUTION TO OVERFITTING



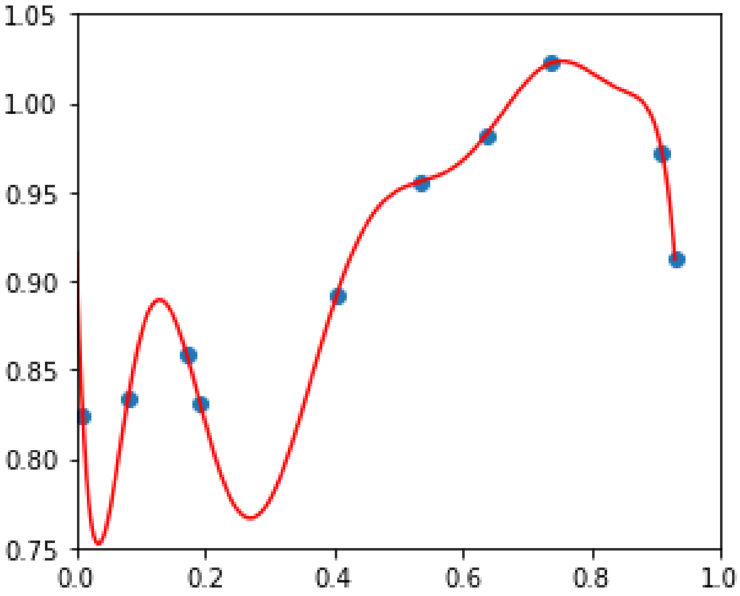
$$w_1 x_1 + w_2 {x_1}^2 + w_3 {x_1}^3 + w_4 {x_1}^4 + w_5 {x_1}^5 + w_6 {x_1}^6 + w_7 {x_1}^7 + w_8 {x_1}^8 + w_9 {x_1}^9 + b$$

- Another solution is to regularize the parameters of the model (w_j) so that no parameters gets too large.

COST FUNCTION WITH REGULARIZATION



$w_1 x_1 + w_2 x_1{}^2 + w_3 x_1{}^3 + b$



$w_1 x_1 + w_2 x_1{}^2 + w_3 x_1{}^3 + w_4 x_1{}^4 + w_5 x_1{}^5 + w_6 x_1{}^6 + w_7 x_1{}^7 + w_8 x_1{}^8 + w_9 x_1{}^9 + b$

- $w_4, w_5, w_6, w_7, w_8, w_9$ are large numbers.
- How can we enforce the model to keep them small.
- One way is to penalize the model from fitting to large values of w_j by adding some extra terms in cost function.

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}\left(x^{(i)}\right) - y^{(i)}\right)^2 + 1000\ w_4{}^2 + 1000\ w_5{}^2 + \cdots + 1000\ w_9{}^2$$

SOLUTION TO OVERFITTING

Size	Bedrooms	Floors	Age	Avg income	…	Distance to coffee shop	Price
x_1	x_2	x_3	x_4	x_5		x_{100}	y
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- In general, we do not know which parameters to penalize.
- We penalize all parameters and will let algorithm to decide which is more important.

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

REGULARIZED REGRESSION

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

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

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

} simultaneous updates

REGULARIZED REGRESSION

- Repeat for j = 1,2, … , n {

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

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

} simultaneous updates

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

$$\alpha = 0.01; \lambda = 10; m = 50 \rightarrow 1 - 0.01 \frac{10}{50} = 1 - 0.002 = 0.998$$

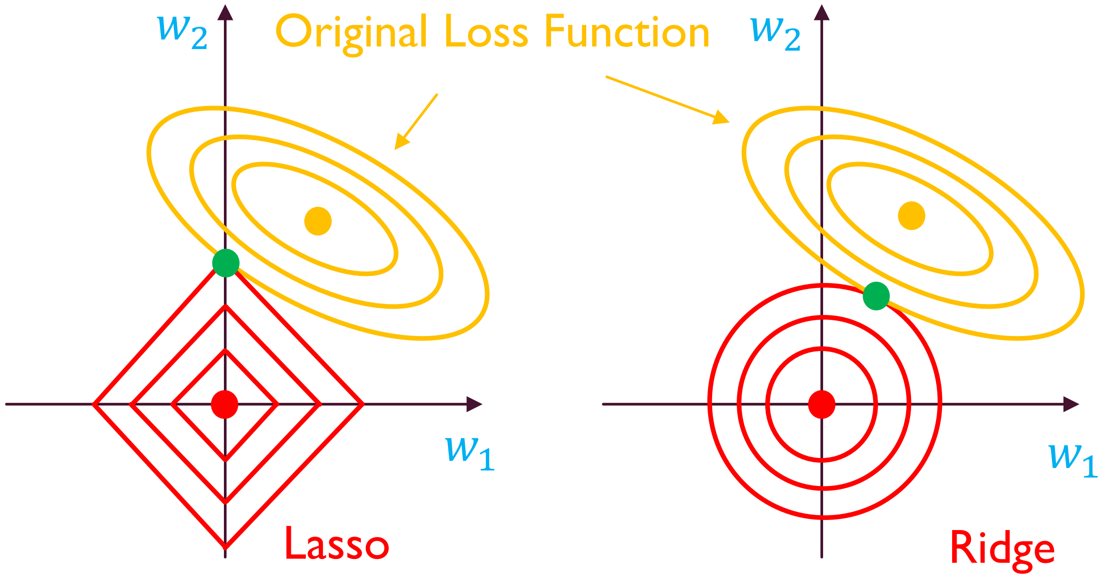
REGULARIZATION USING OTHER NORMS

L_1 norm (Lasso)

$$\frac{\lambda}{2m}\sum_{j=1}^{n}|w_j|$$

L_2 norm (Ridge)

$$\frac{\lambda}{2m}\sum_{j=1}^{n}{w_j}^2$$



- Lasso regression :
  - Introduces more sparsity to the model.
  - Is helpful for feature selection.
  - Is more computationally efficient as a model due to the sparse solutions.
- Ridge regression:
  - Pushes weights towards 0, but not actually 0.
  - In practice, usually performs betters.
- Usually, a combination of these two works best (Elastic Net).

REFERENCE

- Supervised Machine Learning: Regression and Classification, Andrew Ng, Stanford Online, DeepLearning.AI
- Animations of Logistic Regression with Python (https://towardsdatascience.com/animations-of-logistic-regression-with-python-31f8c9cb420).