Linear Regression & Gradient Descent

ML Club Week 2

Discord Server!

GitHub!!!

Linear Regression - Overview

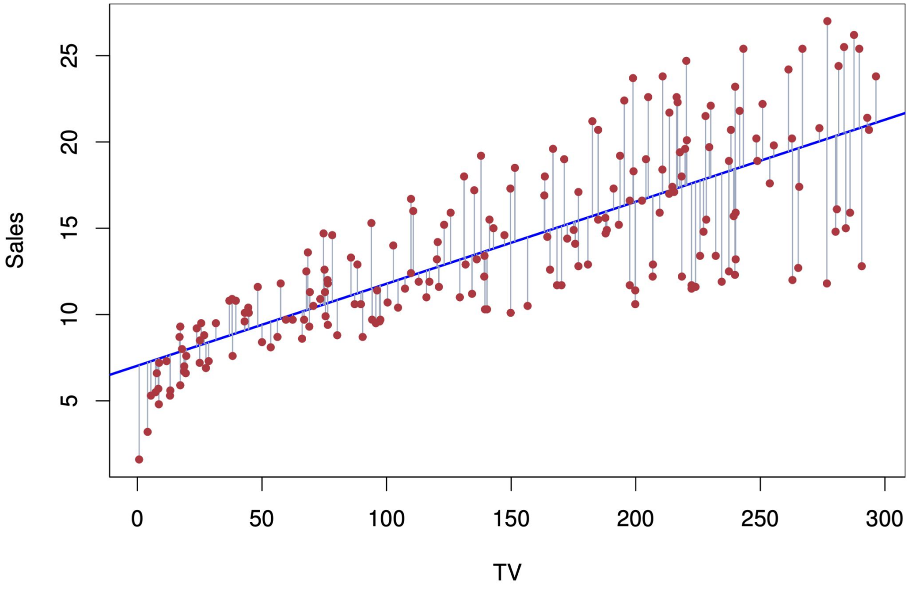
- Supervised ML
- Compute linear relationship: Dependent y - Predictor(s) x(s)
- ***Simple*** linear regression: x (1 Predictor)
- ***Multiple*** linear regression: xs (>= 2 Predictors)

Simple Linear Regression

$$y = wx + b$$

output weight input bias

(linear relationship between x and y)



Multiple Linear Regression

Multi features

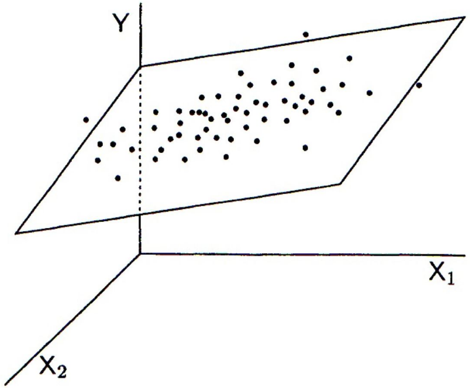
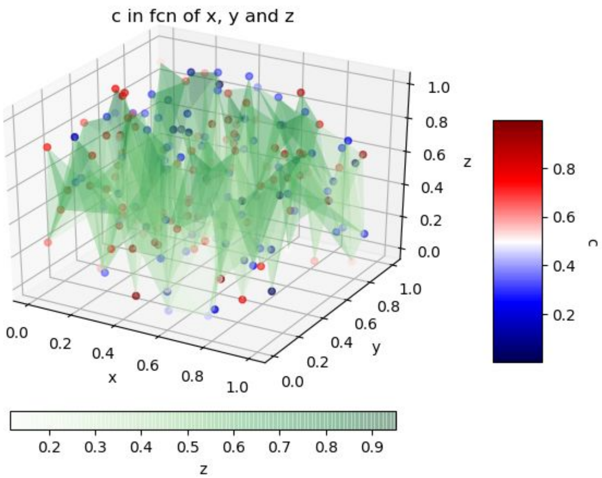


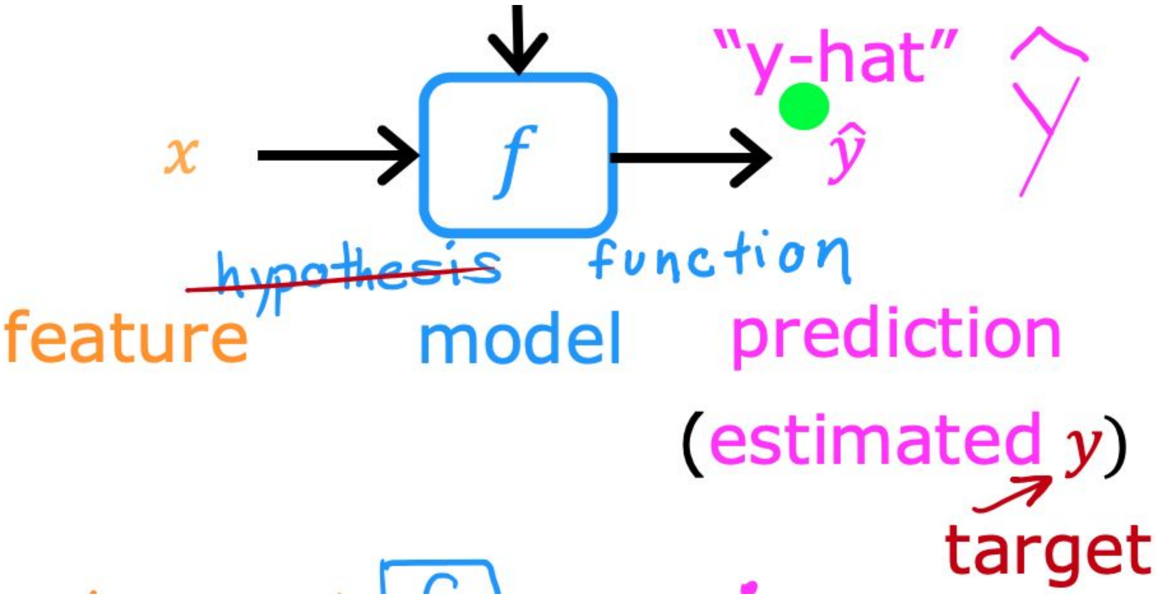
Figure 4.1: *The regression of* Y *onto* X_1 *and* X_2 *as a scatterplot in variable space.*



2 variable (= feature)

more

Regression



x
f
"y-hat"
$\hat{y}$
~~hypothesis~~ function
feature
model
prediction
(estimated y)
target

Dataset example

Living area (feet^2)	#bedrooms	Price (1000$s)
x_1	x_2	y
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

Convention: X is $n \times d$ <u>design matrix</u> of sample pts
y is n-vector of scalar labels

$$
\begin{array}{cc}
\begin{bmatrix}
X_{11} & X_{12} & \dots & X_{1j} & \dots & X_{1d} \\
X_{21} & X_{22} & & X_{2j} & & X_{2d} \\
\vdots & & & & & \\
X_{i1} & X_{i2} & & X_{ij} & & X_{id} \\
\vdots & & & & & \\
X_{n1} & X_{n2} & & X_{nj} & & X_{nd}
\end{bmatrix}
&
\begin{matrix} \vec{x}^{(1)} \\ \\ \\ \leftarrow \text{ point } X_i^\top \\ \\ \\ \end{matrix}
\end{array}
\qquad
\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}
$$

$\uparrow$ feature column X_{*j}

$\uparrow$ y

Notations

$\vec{x}^{(i)}$ = features of ith training sample

x_j = jth feature

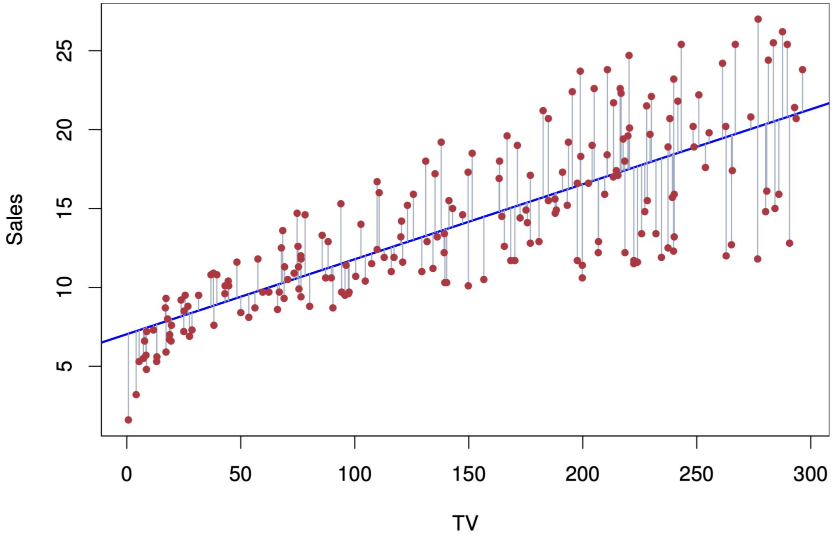
$\vec{w} = \left[w_1\ w_2\ w_3\ \dots\ w_n\right]^{\mathsf{T}}$

$f_{w,b}(x) = w_1 x_1 + w_2 x_2 + w_n x_n + b$

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

How to find the optimal w?

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



minimize the Cost!

high cost = discrepancy between y and ŷ

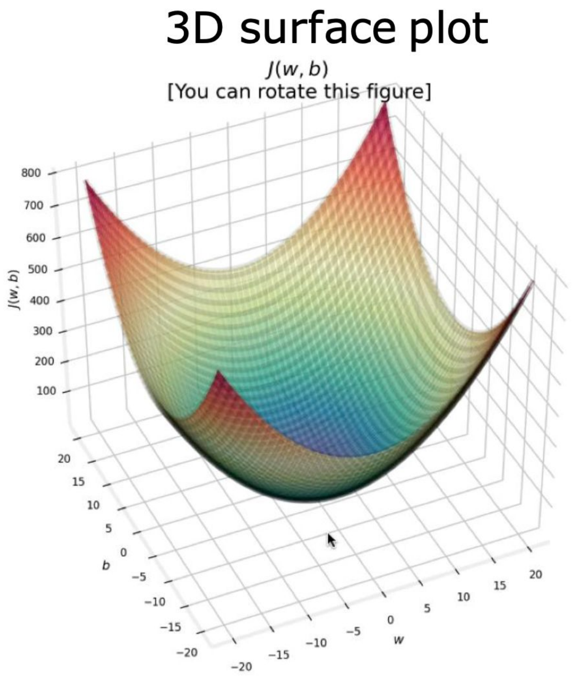
low cost = prediction is pretty accurate

Parameters (w) & Cost Functions (Mean Squared Error)

- MSE: Average of squared errors between predicted and actual values.
- *How close the model’s predictions are to observed values*.
- Goal: Estimate the optimal set of weights (w) that minimize **cost** (MSE).

$$J(w, b) = \mathrm{MSE} = \overset{\text{Mean}}{\boxed{\frac{1}{n}\sum_{i=1}^{n}}}\Big(\overset{\text{Error}}{\boxed{Y_i - \hat{Y}_i}}\Big)^{\overset{\text{Squared}}{\boxed{2}}}$$

J(w, b) - one feature and one bias



MSE cost is always convex!
(regardless of how many features)

Gradient Descent

Have some function $J(w, b)$ for linear regression or any function

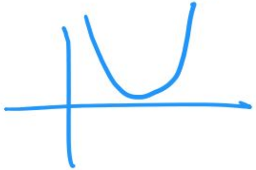
Want $\min_{w,b} J(w, b)$ $\qquad \min_{w_1, \dots, w_n, b} J(w_1, w_2, \dots, w_n, b)$

Outline:

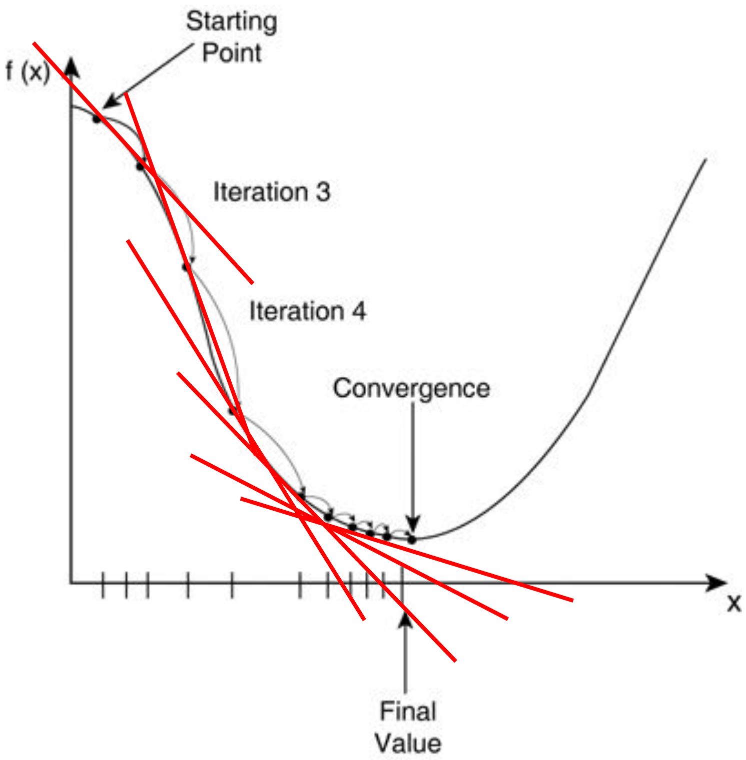
- Start with some w, b (set $w=0$, $b=0$)
- Keep changing w, b to reduce $J(w, b)$
- Until we settle at or near a minimum

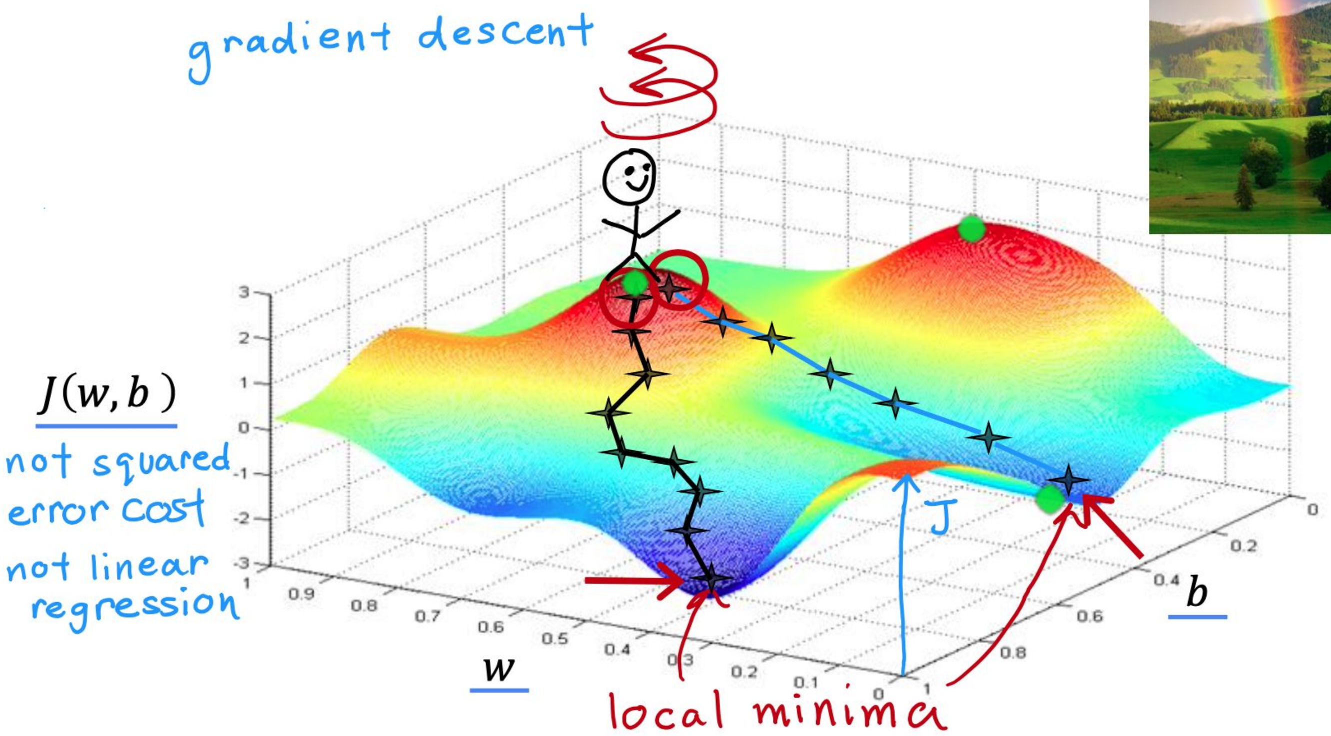
may have >1 minimum

J not always



Gradient Descent in 2D




gradient descent
$J(w,b)$
not squared error cost
not linear regression
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
w
J
0
0.2
0.4
0.6
0.8
1
b
local minima

$f(x) = wx + b$

Linear regression model

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

Cost function

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

Gradient descent algorithm

repeat until convergence {

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

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

}

next

(Optional)

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

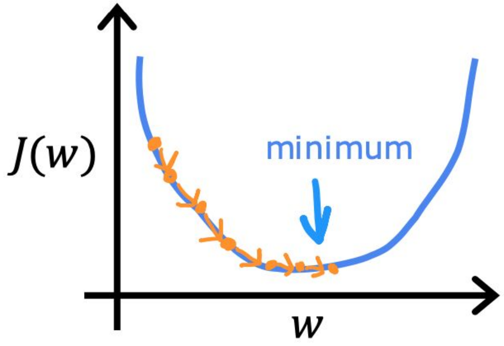
$$= \frac{1}{\cancel{2}m}\sum_{i=1}^{m}\left(wx^{(i)}+b-y^{(i)}\right)\cancel{2}x^{(i)} = \boxed{\frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)})-y^{(i)})x^{(i)}}$$

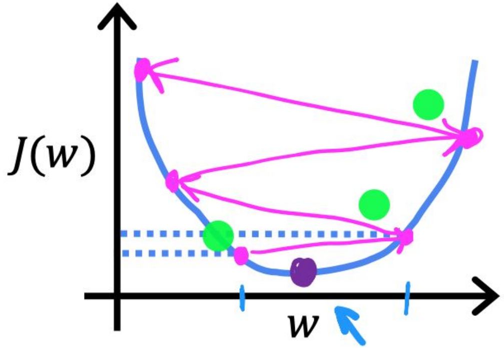
$$\frac{\partial}{\partial b}J(w,b) = \frac{\partial}{\partial b}\frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)})-y^{(i)}\right)^2 = \frac{\partial}{\partial b}\frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)}+b-y^{(i)}\right)^2$$

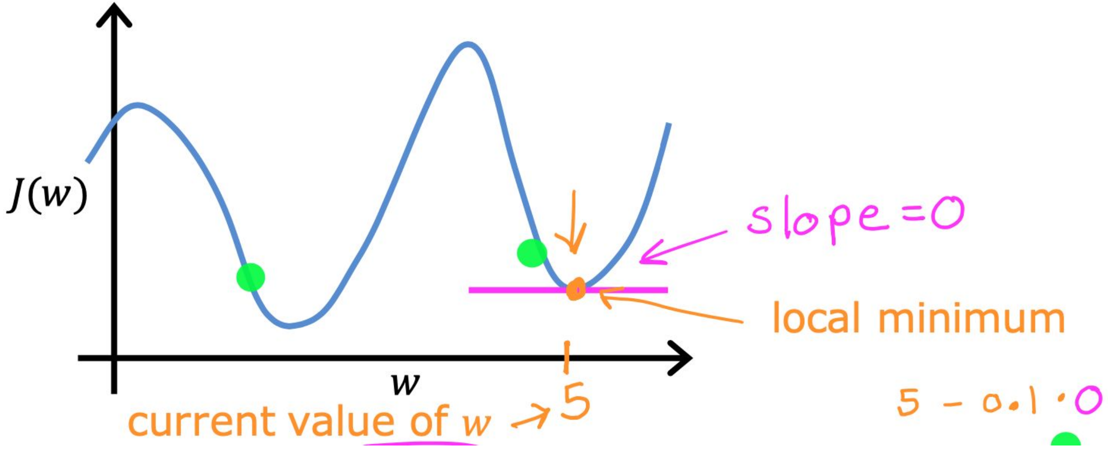
$$= \frac{1}{\cancel{2}m}\sum_{i=1}^{m}\left(wx^{(i)}+b-y^{(i)}\right)\cancel{2} = \boxed{\frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)})-y^{(i)})}$$

no $x^{(i)}$

Learning Rate



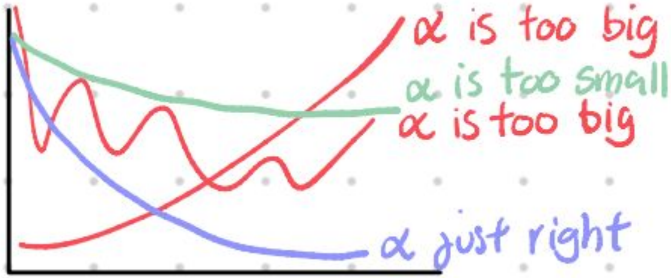




epsilon and learning rate

Choosing ε and α

- ε - automatic convergence test. stop if $J(\vec{w}, b) \leq \varepsilon$
- α should decrease if it's chosen as really small number.



multi features

repeat {

$$w_j = w_j - \alpha \boxed{\frac{\partial}{\partial w_j} J(w_1, \ldots w_n, b)}$$

$$b = b - \alpha \boxed{\frac{\partial}{\partial b} J(w_1, \ldots w_n, b)}$$

}

$$\left[\begin{array}{l} w_1 = w_1 - \frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right) x_1^{(i)} \\ \vdots \\ w_n = w_n - \frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right) x_n^{(i)} \end{array}\right.$$

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

Recap: Gradient

$$\nabla f(\mathbf{w}) = \left(\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \frac{\partial f}{\partial w_3} \right)$$

In calculus, the **gradient** is the vector of partial derivatives with respect to each unknown variable in a function

Gradient Descent

$$\theta := \theta - \alpha \nabla_\theta J(\theta)$$

$$Cost = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$Cost = \frac{1}{n}(\mathbf{y} - \mathbf{X}\hat{\mathbf{w}})^T(\mathbf{y} - \mathbf{X}\hat{\mathbf{w}})$$

Gradient (Matrix form)

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) \\
&= \frac{1}{2}\nabla_\theta \left((X\theta)^T X\theta - (X\theta)^T\vec{y} - \vec{y}^T(X\theta) + \vec{y}^T\vec{y}\right) \\
&= \frac{1}{2}\nabla_\theta \left(\theta^T(X^TX)\theta - \vec{y}^T(X\theta) - \vec{y}^T(X\theta)\right) \\
&= \frac{1}{2}\nabla_\theta \left(\theta^T(X^TX)\theta - 2(X^T\vec{y})^T\theta\right) \\
&= \frac{1}{2}\left(2X^TX\theta - 2X^T\vec{y}\right) \\
&= \boxed{X^TX\theta - X^T\vec{y}}
\end{aligned}
$$

(Stanford CS229)

Recall that we can express the linear regression cost function as:

$$Cost = \frac{1}{n}\mathbf{y}^T\mathbf{y} + \frac{1}{n}(-2\mathbf{y}^T\mathbf{X}\hat{\mathbf{w}} + \hat{\mathbf{w}}^T\mathbf{X}^T\mathbf{X}\hat{\mathbf{w}})$$

Thus, the gradient is:

$$Gradient \boxed{= \frac{-2}{n}\mathbf{X}^T(\mathbf{y} - \mathbf{X}\hat{\mathbf{w}})}$$

And our gradient descent updates look like:

$$\hat{\mathbf{w}}^{(j)} = \hat{\mathbf{w}}^{(j-1)} + \frac{2}{n}\mathbf{X}^T(\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}^{(j-1)})$$

(STA395)

Closed Form

In the third step, we used the fact that $a^T b = b^T a$, and in the fifth step used the facts $\nabla_x b^T x = b$ and $\nabla_x x^T A x = 2Ax$ for symmetric matrix A (for more details, see Section 4.3 of "Linear Algebra Review and Reference"). To minimize J, we set its derivatives to zero, and obtain the **normal equations**:

$$X^T X\theta = X^T \vec{y}$$

Thus, the value of θ that minimizes $J(\theta)$ is given in closed form by the equation

$$\theta = (X^T X)^{-1} X^T \vec{y}.^3$$

The bias term is implicitly included in the formulation given in the image through the construction of the X matrix and θ vector.

In linear regression, we typically have an input vector x and want to predict a target value y. The model's prediction is given by:

$\hat{y} = \theta^T x$

Here, x is a feature vector, and θ contains the weights that map the features to the predicted value.

To include the bias term, we augment the feature vector x with an additional constant term (usually 1), and correspondingly add an extra parameter to θ to represent the bias. So if the original feature vector was $[x_1, x_2, ..., x_n]$, the augmented vector becomes:

$x_{aug} = [1, x_1, x_2, ..., x_n]$

And θ becomes:

$\theta_{aug} = [\theta_0, \theta_1, \theta_2, ..., \theta_n]$

Where θ_0 represents the bias term.

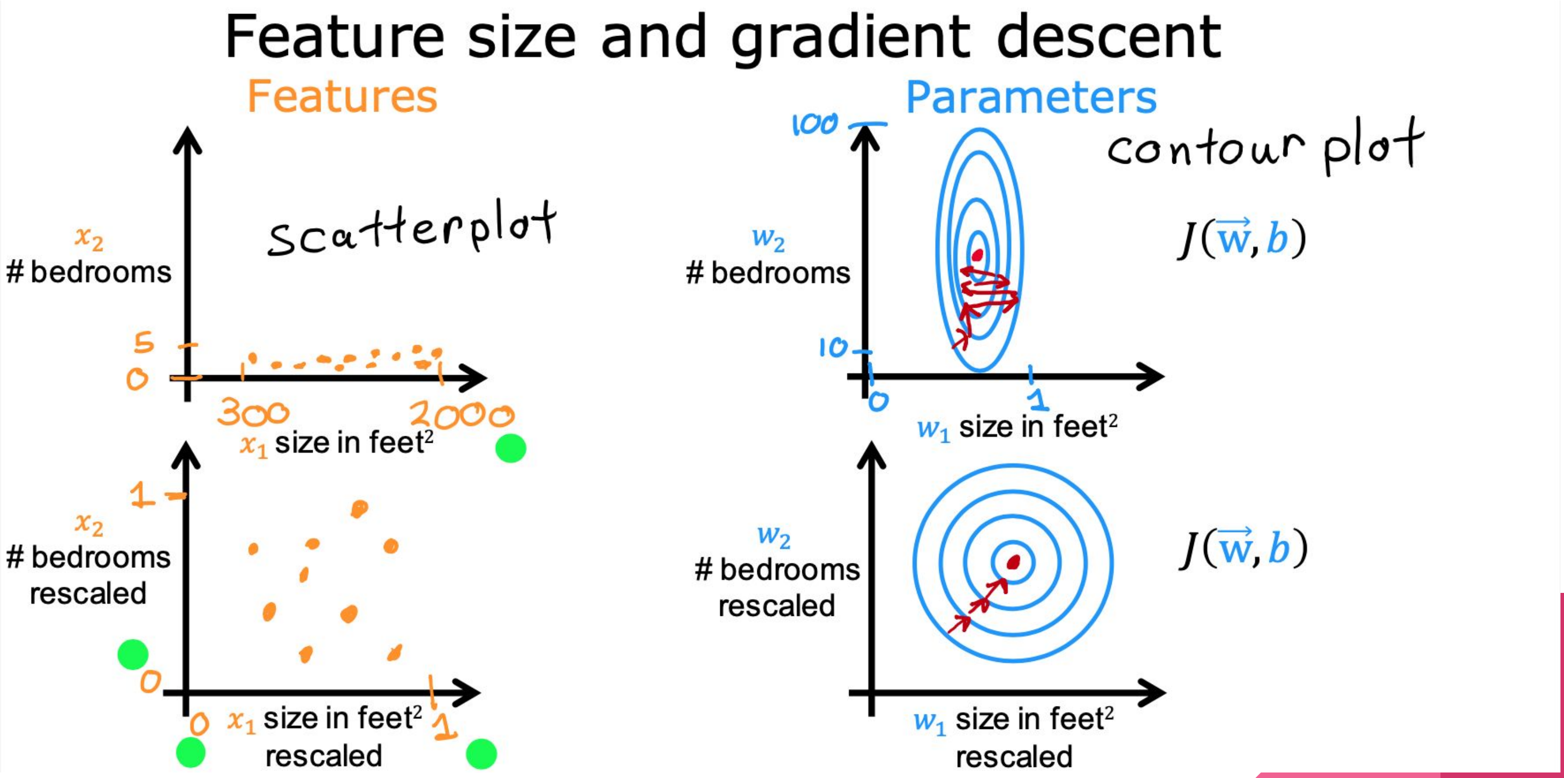
Now, the prediction is:

$\hat{y} = \theta_{aug}^T x_{aug} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n * x_n$

In the normal equation $\theta = (X^T X)^{-1} X^T y$, X is assumed to be the matrix of augmented feature vectors (with the added constant term), and θ is the augmented weight vector (including the bias). So the bias term is inherently part of this formulation, even though it's not explicitly called out.

Feature Scaling

Feature and Cost Function



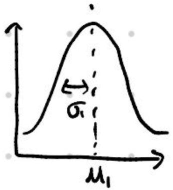
Ways of feature scaling

① divide by max : $300 < x_1 < 2000 \rightarrow 0.15 < x_1 < 1$

② mean normalization : $300 < x_1 < 2000$

do $\boxed{\dfrac{x_1 - \mu_1}{2000 - 300}} \rightarrow -0.18 < x_1 < 0.82$

③ z-score normalization : $300 < x_1 < 2000$



do $\boxed{x_1 = \dfrac{x_1 - \mu_1}{\sigma_1 \text{ (st.dev.)}}}$ $\quad -0.67 < x_1 < 3.1$

aim for $-1 \leq x_j \leq 1$ for each feature x_j