Linear Regression

James G. Shanahan [1,2]

[1]Church and Duncan Group,

[2]School of Informatics, Computing and Engineering, Indiana University

EMAIL: James_DOT_Shanahan_AT_gmail_DOT_com

Lecture Outline

- **Introduction**
- **Linear regression from a statistical perspective**
  - Simple linear regression
  - Assessing the quality of LR model
  - Multiple linear regression (and challenges)
- **Linear regression from scratch**
  - LR via Gradient Descent via brute force
  - LR via Normal Equation (Closed-form analytical optimization)
  - LR via Gradient Descent (Numerical optimization)
  - LR via gradient descent in graph form
- **Summary**

Lecture Outline

- **Introduction**
- **Linear regression from a statistical perspective**
  - Simple linear regression
  - Assessing the quality of LR model
  - Multiple linear regression (and challenges)
- **Linear regression from scratch**
  - LR via Gradient Descent via brute force
  - LR via Normal Equation (Closed-form analytical optimization)
  - LR via Gradient Descent (Numerical optimization)
  - LR via gradient descent in graph form
- **Summary**

Lecture Outline

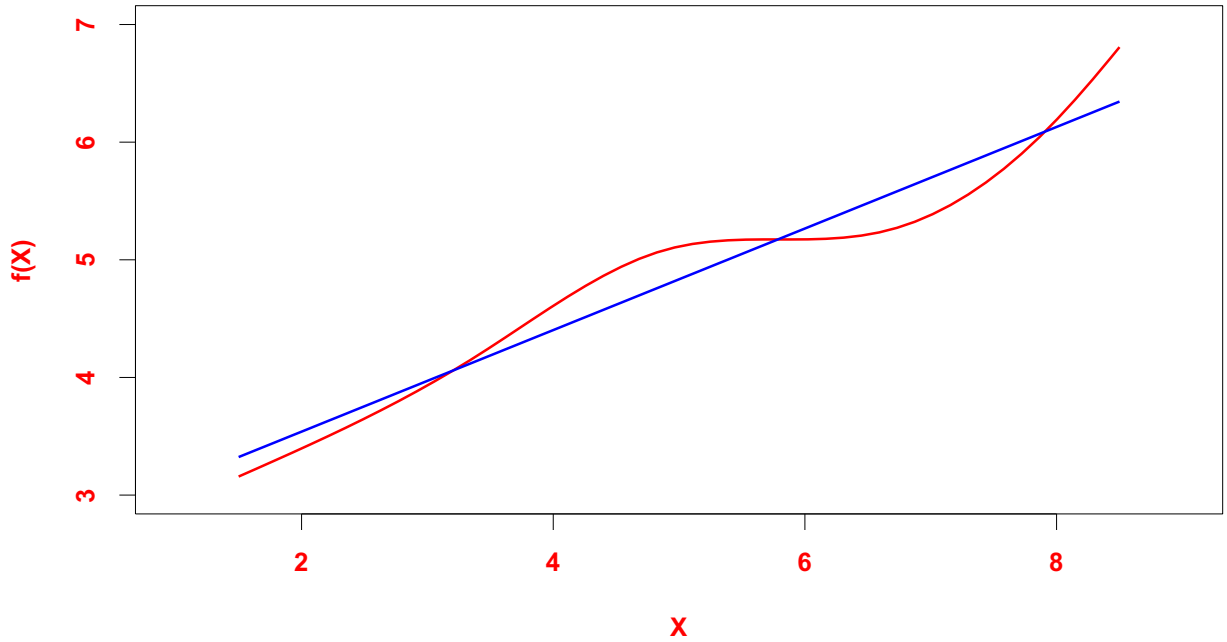
- **Introduction**
- **Linear regression from a statistical perspective**
  - Simple linear regression
  - Assessing the quality of LR model
  - Multiple linear regression (and challenges)
- **Linear regression from scratch**
  - LR via Gradient Descent via brute force
  - LR via Normal Equation (Closed-form analytical optimization)
  - LR via Gradient Descent (Numerical optimization)
  - LR via gradient descent in graph form
- **Summary**

Linear regression

- Linear regression is a simple approach to supervised learning. It assumes that the dependence of Y on $X_1, X_2, \ldots X_p$ is linear.

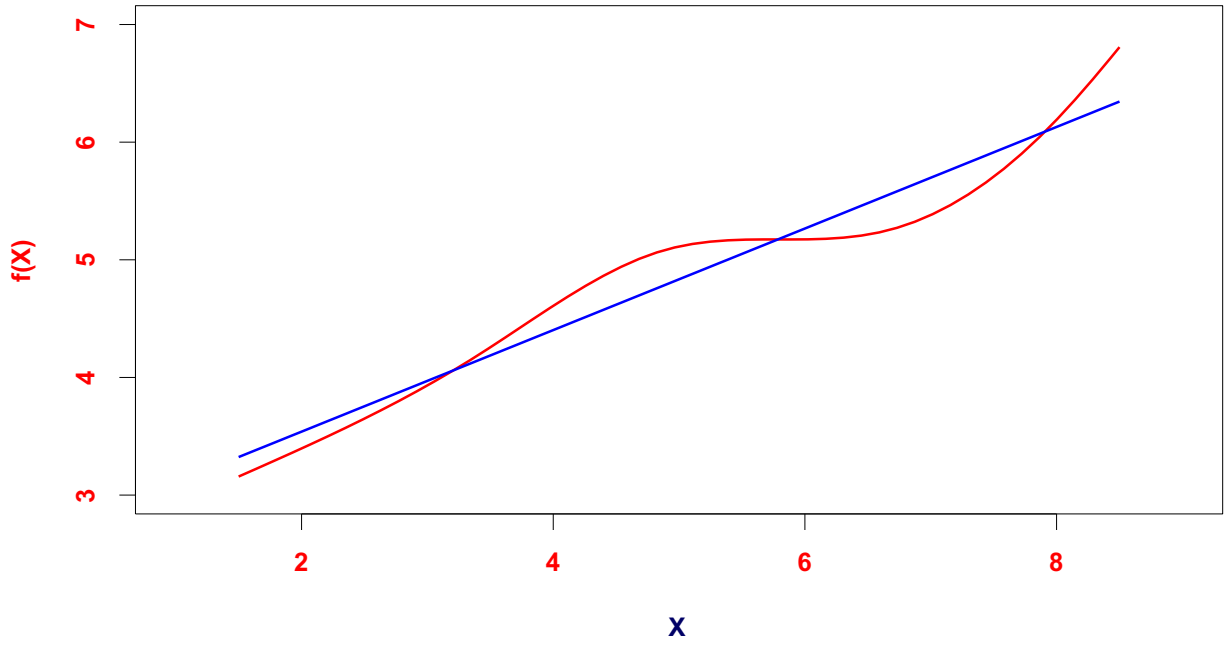
Linear regression

- Linear regression is a simple approach to supervised learning. It assumes that the dependence of Y on $X_1, X_2, \ldots X_p$ is linear.
- True regression functions are never linear!



Linear regression

- Linear regression is a simple approach to supervised learning. It assumes that the dependence of Y on $X_1, X_2, \ldots X_p$ is linear.
- True regression functions are never linear!



- although it may seem overly simplistic, linear regression is extremely useful both conceptually and practically.

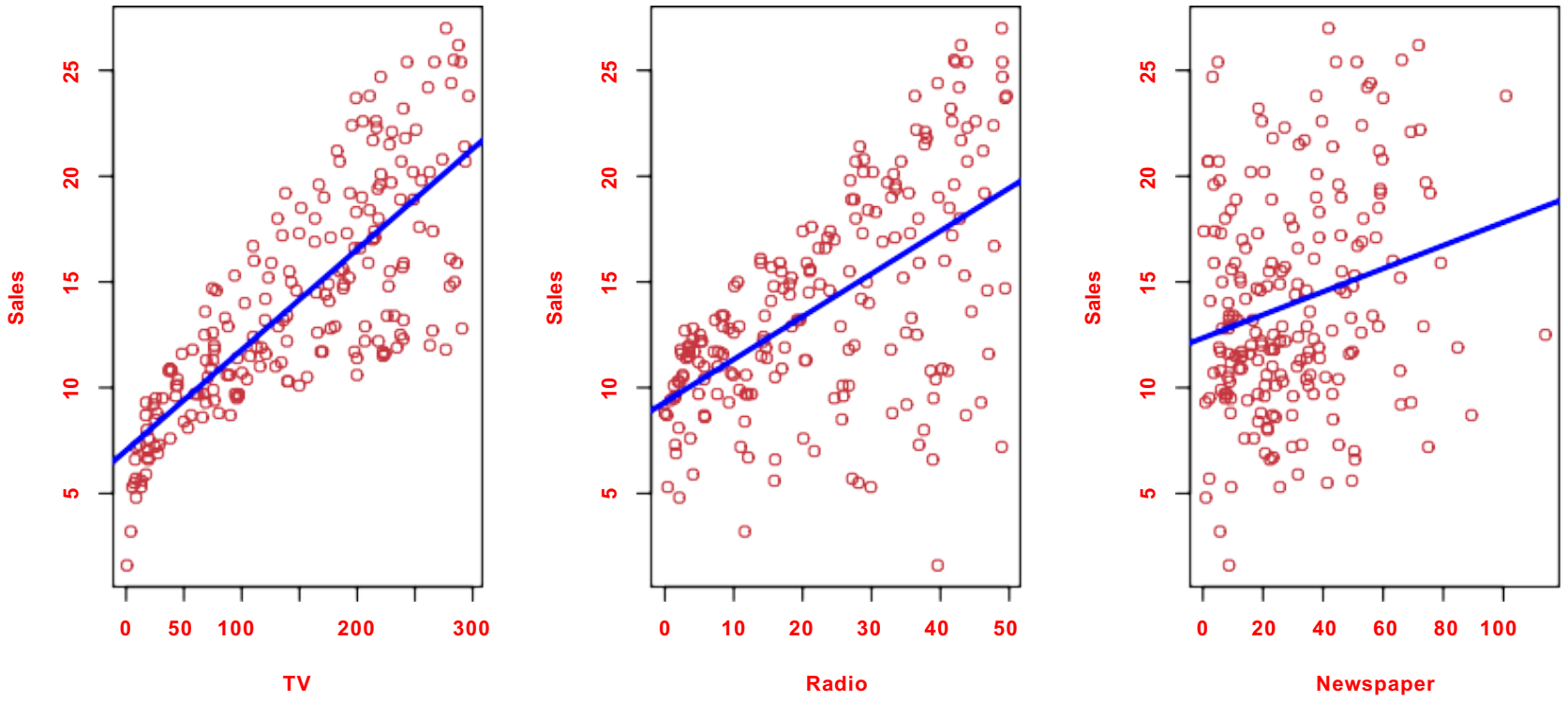
Linear regression for the advertising data

Consider the advertising data shown on the next slide.

Questions we might ask:

- Is there a relationship between advertising budget and sales?
- How strong is the relationship between advertising budget and sales?
- Which media contribute to sales?
- How accurately can we predict future sales?
- Is the relationship linear?
- Is there synergy among the advertising media?

Advertising data



Simple linear regression using a single predictor *X*.

- We assume a model

$$Y = \beta_0 + \beta_1 X + \epsilon,$$

where β_0 and β_1 are two unknown constants that represent the *intercept* and *slope*, also known as *coefficients* or *parameters*, and ϵ is the error term.

- Given some estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ for the model coefficients, we predict future sales using

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x,$$

where $\hat{y}$ indicates a prediction of Y on the basis of $X = x$. The *hat* symbol denotes an estimated value.

Estimation of the parameters by least squares

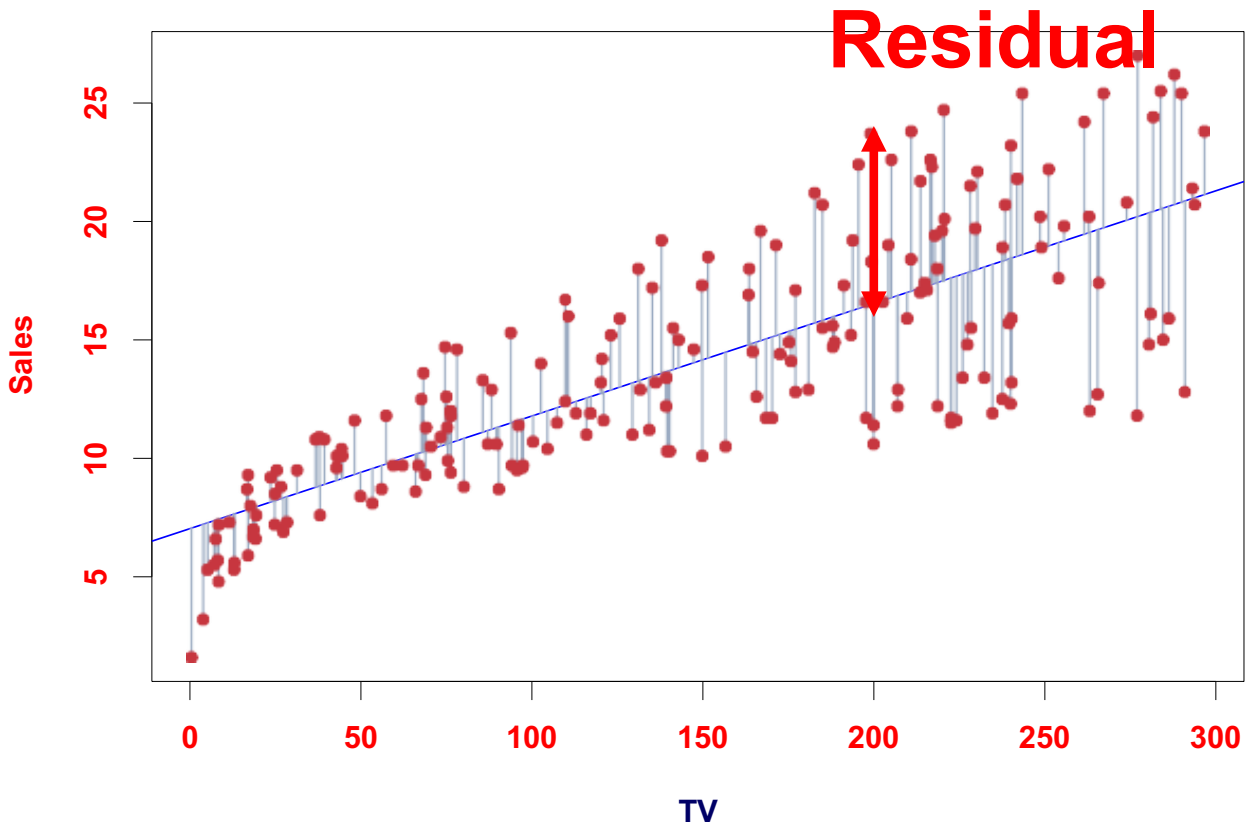
- Let $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ be the prediction for Y based on the ith value of X. Then $e_i = y_i - \hat{y}_i$ represents the ith *residual*
- We define the *residual sum of squares* (RSS) as

$$\text{RSS} = e_1^2 + e_2^2 + \cdots + e_n^2,$$

or equivalently as

$$\text{RSS} = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \ldots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2.$$

Example: advertising data



The least squares fit for the regression of **sales** onto **TV**.
In this case a linear fit captures the essence of the relationship, although it is somewhat deficient in the left of the plot.

Estimation of the parameters by least squares

- Let $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ be the prediction for Y based on the ith value of X. Then $e_i = y_i - \hat{y}_i$ represents the ith *residual*
- We define the *residual sum of squares* (RSS) as

$$\text{RSS} = e_1^2 + e_2^2 + \cdots + e_n^2,$$

or equivalently as

$$\text{RSS} = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \ldots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2.$$

- The least squares approach chooses $\hat{\beta}_0$ and $\hat{\beta}_1$ to minimize the RSS. The minimizing values can be shown to be

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x},$$

Analytical solution
Via the Normal Equation

where $\bar{y} \equiv \frac{1}{n}\sum_{i=1}^{n} y_i$ and $\bar{x} \equiv \frac{1}{n}\sum_{i=1}^{n} x_i$ are the sample means.

Lecture Outline

- **Introduction**
- **Linear regression from a statistical perspective**
  - Simple linear regression
  - Assessing the quality of LR model
  - Multiple linear regression (and challenges)
- **Linear regression from scratch**
  - LR via Gradient Descent via brute force
  - LR via Normal Equation (Closed-form analytical optimization)
  - LR via Gradient Descent (Numerical optimization)
  - LR via gradient descent in graph form
- **Summary**

Assess quality of LR model

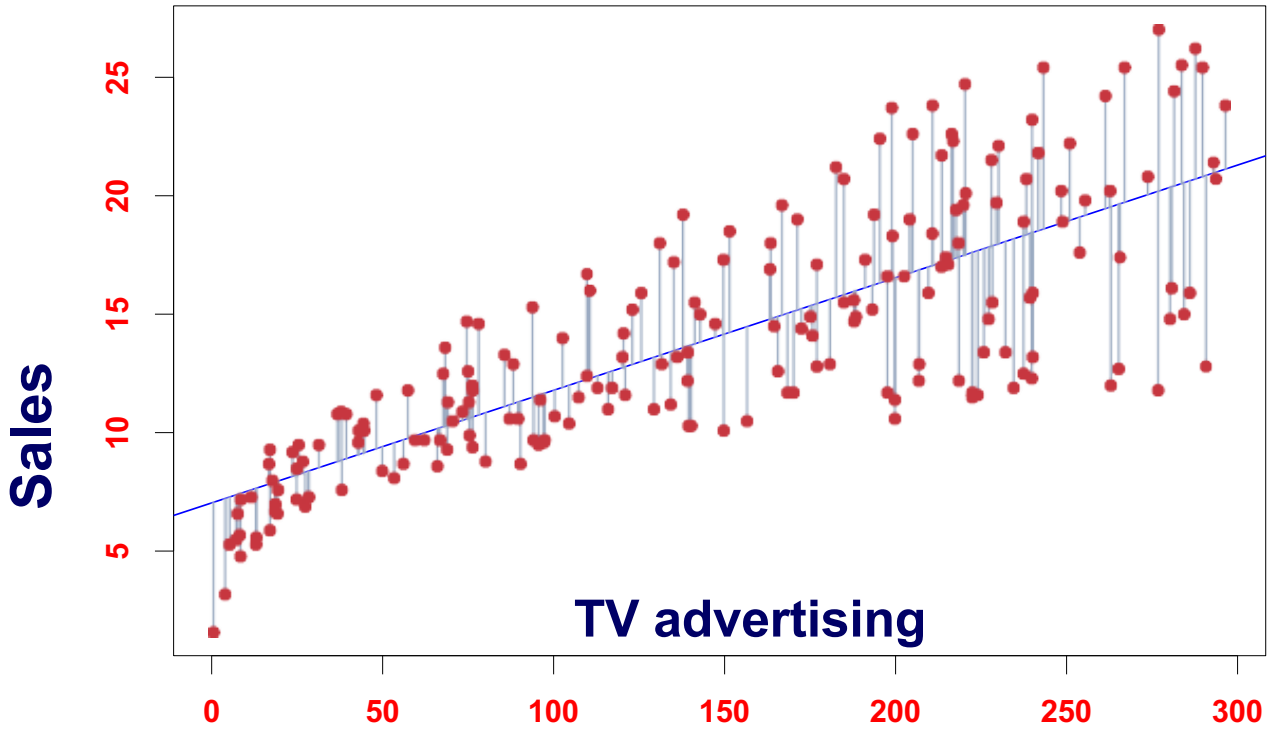
- **Individual variable (coefficient) importance**
- **Overall**

Assessing the Accuracy of the Coefficient Estimates

- The standard error of an estimator reflects how it varies under repeated sampling. We have

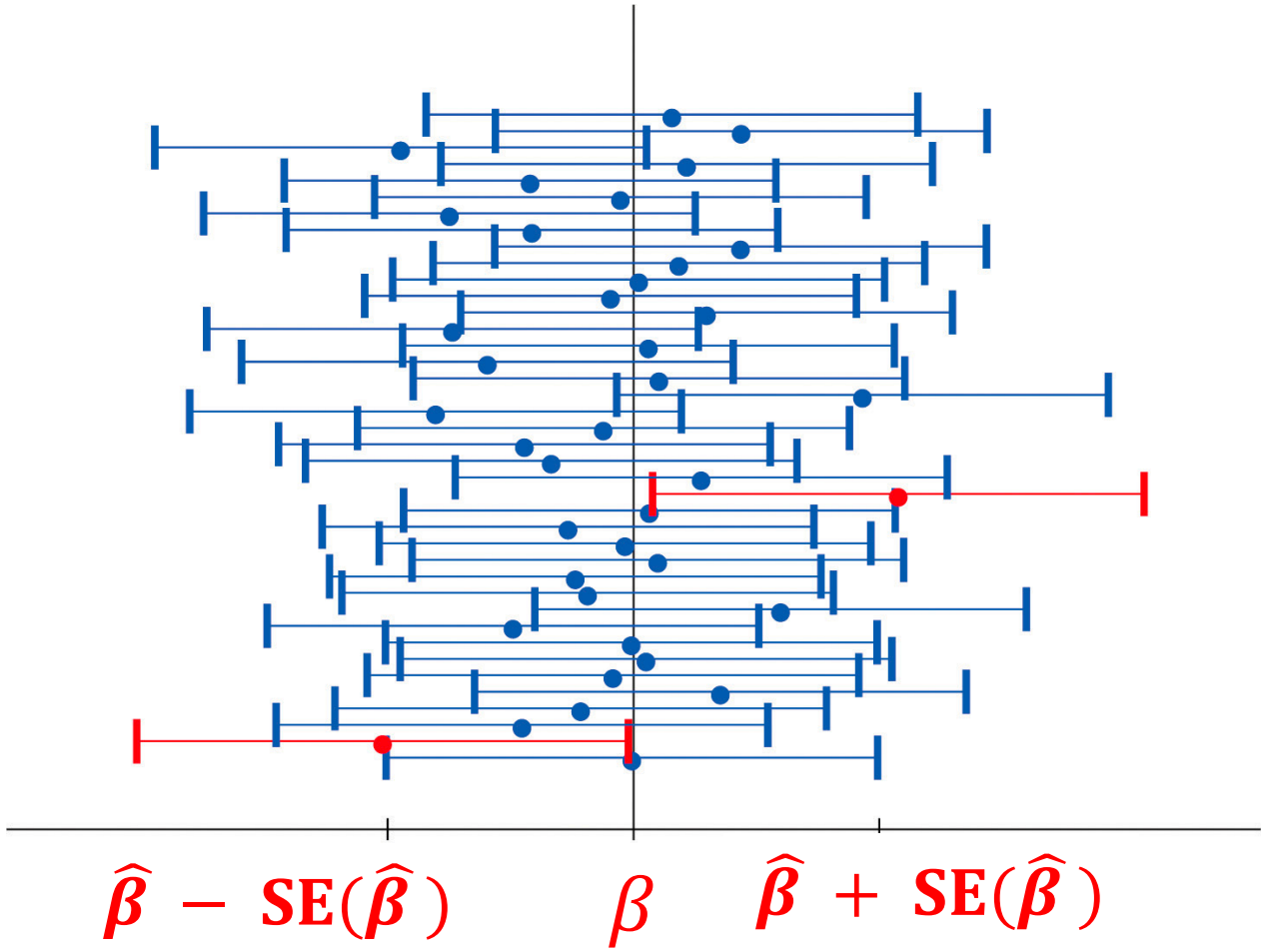
$$\mathrm{SE}(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \quad \mathrm{SE}(\hat{\beta}_0)^2 = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}\right],$$

where $\sigma^2 = \mathrm{Var}(\epsilon)$



Sampling

- True β
- 95% of samples will contain the True β



Assessing the Accuracy of the Coefficient Estimates

- The standard error of an estimator reflects how it varies under repeated sampling. We have

$$\mathrm{SE}(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \quad \mathrm{SE}(\hat{\beta}_0)^2 = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}\right],$$

  where $\sigma^2 = \mathrm{Var}(\epsilon)$

- These standard errors can be used to compute *confidence intervals.* A 95% confidence interval is defined as a range of values such that with 95% probability, the range will contain the true unknown value of the parameter. It has the form

$$\hat{\beta}_1 \pm 2 \cdot \mathrm{SE}(\hat{\beta}_1).$$

Confidence intervals – continued

That is, there is approximately a 95% chance that the interval

$$\left[\hat{\beta}_1 - 2 \cdot \mathrm{SE}(\hat{\beta}_1),\ \hat{\beta}_1 + 2 \cdot \mathrm{SE}(\hat{\beta}_1)\right]$$

will contain the true value of β_1 (under a scenario where we got repeated samples like the present sample)

For the advertising data, the 95% confidence interval for β_1 is $[0.042, 0.053]$

- **Zero is not in the 95% conf interval**

Hypothesis testing

- Standard errors can also be used to perform *hypothesis tests* on the coefficients. The most common hypothesis test involves testing the *null hypothesis* of

H_0 : There is no relationship between X and Y

versus the *alternative hypothesis*

H_A : There is some relationship between X and Y.

- Mathematically, this corresponds to testing

$$H_0 : \beta_1 = 0$$

versus

$$H_A : \beta_1 \neq 0,$$

since if $\beta_1 = 0$ then the model reduces to $Y = \beta_0 + \epsilon$, and X is not associated with Y.

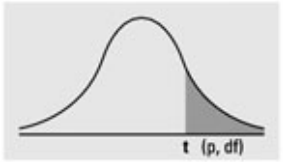
Hypothesis testing – continued

- To test the null hypothesis, we compute a ***t-statistic***, given by

$$t = \frac{\hat{\beta}_1 - 0}{\mathrm{SE}(\hat{\beta}_1)},$$

- This will have a t-distribution with $n - 2$ degrees of freedom, assuming $\beta_1 = 0$.
- Using statistical software, it is easy to compute the probability of observing any value equal to $|t|$ or larger. We call this probability the ***p-value***.

Numbers in each row of the table are values on a *t*-distribution with (*df*) degrees of freedom for selected right-tail (greater-than) probabilities (*p*).



df/p	0.40	0.25	0.10	0.05	0.025	0.01	0.005	0.0005
1	0.324920	1.000000	3.077684	6.313752	12.70620	31.82052	63.65674	636.6192
2	0.288675	0.816497	1.885618	2.919986	4.30265	6.96456	9.92484	31.5991
3	0.276671	0.764892	1.637744	2.353363	3.18245	4.54070	5.84091	12.9240
4	0.270722	0.740697	1.533206	2.131847	2.77645	3.74695	4.60409	8.6103
5	0.267181	0.726687	1.475884	2.015048	2.57058	3.36493	4.03214	6.8688
6	0.264835	0.717558	1.439756	1.943180	2.44691	3.14267	3.70743	5.9588
7	0.263167	0.711142	1.414924	1.894579	2.36462	2.99795	3.49948	5.4079
8	0.261921	0.706387	1.396815	1.859548	2.30600	2.89646	3.35539	5.0413
9	0.260955	0.702722	1.383029	1.833113	2.26216	2.82144	3.24984	4.7809
10	0.260185	0.699812	1.372184	1.812461	2.22814	2.76377	3.16927	4.5869
11	0.259556	0.697445	1.363430	1.795885	2.20099	2.71808	3.10581	4.4370
12	0.259033	0.695483	1.356217	1.782288	2.17881	2.68100	3.05454	43178
13	0.258591	0.693829	1.350171	1.770933	2.16037	2.65031	3.01228	4.2208
14	0.258213	0.692417	1.345030	1.761310	2.14479	2.62449	2.97684	4.1405
15	0.257885	0.691197	1.340606	1.753050	2.13145	2.60248	2.94671	4.0728
16	0.257599	0.690132	1.336757	1.745884	2.11991	2.58349	2.92078	4.0150
17	0.257347	0.689195	1.333379	1.739607	2.10982	2.56693	2.89823	3.9651
18	0.257123	0.688364	1.330391	1.734064	2.10092	2.55238	2.87844	3.9216
19	0.256923	0.687621	1.327728	1.729133	2.09302	2.53948	2.86093	3.8834
20	0.256743	0.686954	1.325341	1.724718	2.08596	2.52798	2.84534	3.8495
21	0.256580	0.686352	1.323188	1.720743	2.07961	2.51765	2.83136	3.8193
22	0.256432	0.685805	1.321237	1.717144	2.07387	2.50832	2.81876	3.7921
23	0.256297	0.685306	1.319460	1.713872	2.06866	2.49987	2.80734	3.7676
24	0.256173	0.684850	1.317836	1.710882	2.06390	2.49216	2.79694	3.7454
25	0.256060	0.684430	1.316345	1.708141	2.05954	2.48511	2.78744	3.7251
26	0.255955	0.684043	1.314972	1.705618	2.05553	2.47863	2.77871	3.7066
27	0.255858	0.683685	1.313703	1.703288	2.05183	2.47266	2.77068	3.6896
28	0.255768	0.683353	1.312527	1.701131	2.04841	2.46714	2.76326	3.6739
29	0.255684	0.683044	1.311434	1.699127	2.04523	2.46202	2.75639	3.6594
30	0.255605	0.682756	1.310415	1.697261	2.04227	2.45726	2.75000	3.6460
z	0.253347	0.674490	1.281552	1.644854	1.95996	2.32635	2.57583	3.2905

Results for the advertising data

	Coefficient	Std. Error	t-statistic	p-value
Intercept	7.0325	0.4578	15.36	< 0.0001
TV	0.0475	0.0027	17.67	< 0.0001

- **The chance of seeing this $\widehat{\beta}$ when the true β =0 is 0.0001**

- **95% confidence interval contains zero, t < 2**

- **The chance of seeing this $\widehat{\beta}$ when β =0 is 0.0001**

Assessing the Overall Accuracy of the Model

- We compute the *Residual Standard Error*

$$\text{RSE} = \sqrt{\frac{1}{n-2}\text{RSS}} = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2},$$

where the *residual sum-of-squares* is $RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.

- *R-squared* or fraction of variance explained is

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

where $\text{TSS} = \sum_{i=1}^{n}(y_i - \bar{y})^2$ is the *total sum of squares*.

- It can be shown that in this simple linear regression setting that $R^2 = r^2$, where r is the correlation between X and Y:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}.$$

Advertising data results

Quantity	Value
Residual Standard Error	3.26
R^2	0.612
F-statistic	312.1

R: Linear Regression via lm()

example.lm ()

Output Window

```
> colnames(dataEx1)=c("time", "temperature")

>  lm.temp <- lm(temperature ~ time, data=as.data.frame(dataEx1))

>  summary(lm.temp)

Call:
lm(formula = temperature ~ time, data = as.data.frame(dataEx1))

Residuals:
         1          2          3          4          5
-1.490e-16 -9.000e-01  1.200e+00  3.000e-01 -6.000e-01

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.100      0.995   0.101  0.92628
time           1.900      0.300   6.333  0.00796 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9487 on 3 degrees of freedom
Multiple R-squared: 0.9304, Adjusted R-squared: 0.9072
F-statistic: 40.11 on 1 and 3 DF,  p-value: 0.00796
```

Pay attention to

1. **Residual standard error**
2. **Or Deviance (SSE),**
3. **And variable significanc**

Residuals = ($WX^i - y^i$)

Variable significance

Residual standard error

```
> deviance(lm.temp)
[1] 2.7
```

Residual standard error =σ=Sqrt(deviance/m-n-1)

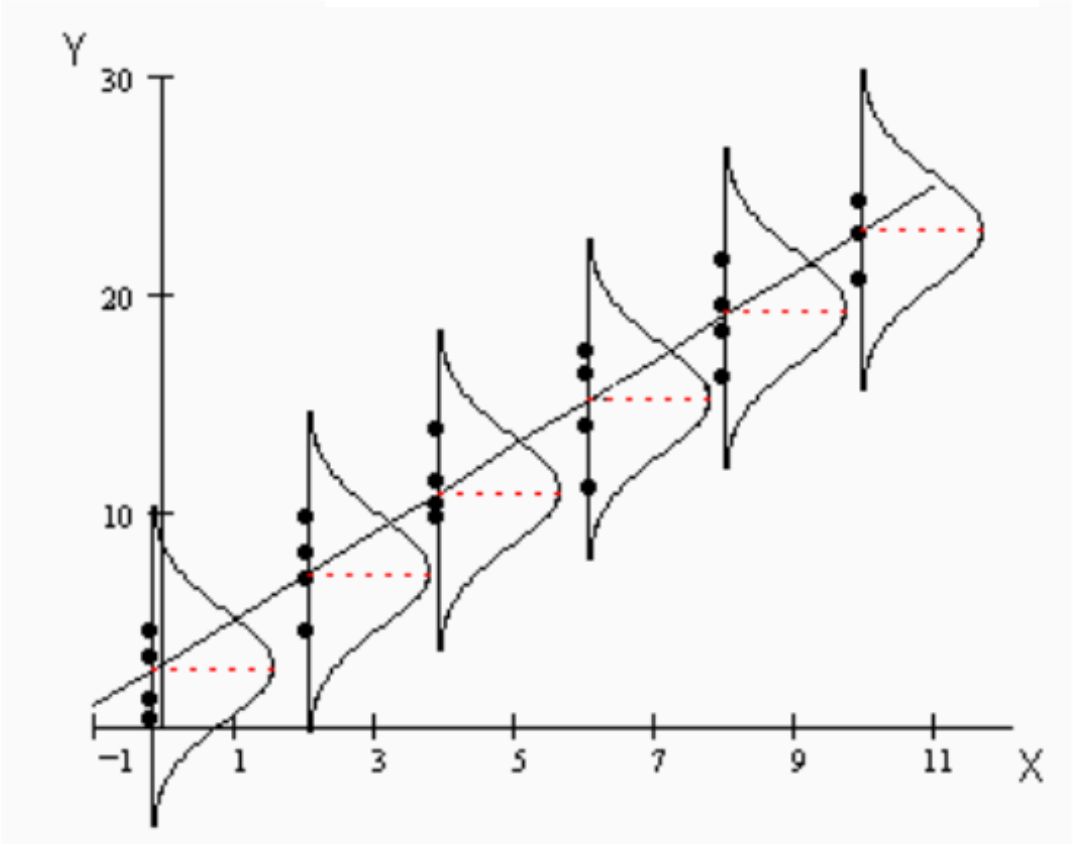
$$\sigma = \sqrt{\left(\frac{1}{m-n-1}\right)\sum_{i=1}^{m}\left(WX^i - y^i\right)^2} = \sqrt{1/3 * 2.7}$$

residualStandardError=sqrt((t(lm.temp$residuals) %*% lm.temp$residuals)/3)

LR assumptions: Expected value of y given x

- **There is a normally distributed subpopulation of responses for each value of the explanatory variable. These subpopulations all have a common variance.**

$$y \mid x \sim N(\mu_{y|x}, \sigma_e)$$



Standard Error for slope, Intercept

Standard error for Slope:	$s_{\hat{b}} = \frac{s_e}{\sqrt{n-1}\, s_x}$
Standard error for Predicted Mean:	$s_{\hat{y}} = s_e \sqrt{\frac{1}{n} + \frac{(x-\bar{x})^2}{\sum (x_i - \bar{x})^2}}$
Standard error for the Intercept:	$s_a = s_e \sqrt{\frac{1}{n} + \frac{(\bar{x})^2}{\sum (x_i - \bar{x})^2}}$
Standard error for a Predicted Value:	$s_{y_p} = s_e^2 \left(1 + \frac{1}{n} + \frac{(x-\bar{x})^2}{\sum (x_i - \bar{x})^2} \right)$

Reference: Kennedy, Joh B. and Adam M. Neville, Basic Statistical Methods for Engineers and Scientists, 3rd, Harper and Row, 1986.

Lecture Outline

- **Introduction**
- **Linear regression from a statistical perspective**
  - Simple linear regression
  - Assessing the quality of LR model
  - Multiple linear regression (and challenges)
- **Linear regression from scratch**
  - LR via Gradient Descent via brute force
  - LR via Normal Equation (Closed-form analytical optimization)
  - LR via Gradient Descent (Numerical optimization)
  - LR via gradient descent in graph form
- **Summary**

Multiple Linear Regression

- Here our model is

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$

- We interpret β_j as the *average* effect on Y of a one unit increase in X_j, *holding all other predictors fixed*. In the advertising example, the model becomes

$$\texttt{sales} = \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio} + \beta_3 \times \texttt{newspaper} + \epsilon.$$

Interpreting regression coefficients

- The ideal scenario is when the predictors are uncorrelated — a ***balanced design***:
  - Each coefficient can be estimated and tested separately.
  - Interpretations such as ***"a unit change in X_j is associated with a β_j change in Y, while all the other variables stay fixed"***, are possible.
- Correlations amongst predictors cause problems:
  - The variance of all coefficients tends to increase, sometimes dramatically
  - Interpretations become hazardous — when X_j changes, everything else changes.
- ***Claims of causality*** should be avoided for observational data.

woes of (interpreting) regression coefficients

"Data Analysis and Regression" Mosteller and Tukey 1977

- a regression coefficient β_j estimates the expected change in Y per unit change in X_j, *with all other predictors held fixed*. But predictors usually change together!
- Example: Y total amount of change in your pocket; $X_1 = \#$ of coins; $X_2 = \#$ of pennies, nickels and dimes. By itself, regression coefficient of Y on X_2 will be > 0. But how about with X_1 in model?
- $Y =$ number of tackles by a football player in a season; W and H are his weight and height. Fitted regression model is $\hat{Y} = b_0 + .50W - .10H$. How do we interpret $\hat{\beta}_2 < 0$?

quotes by famous Statisticians

"Essentially, all models are wrong, but some are useful"

George Box

Estimation and Prediction for Multiple Regression

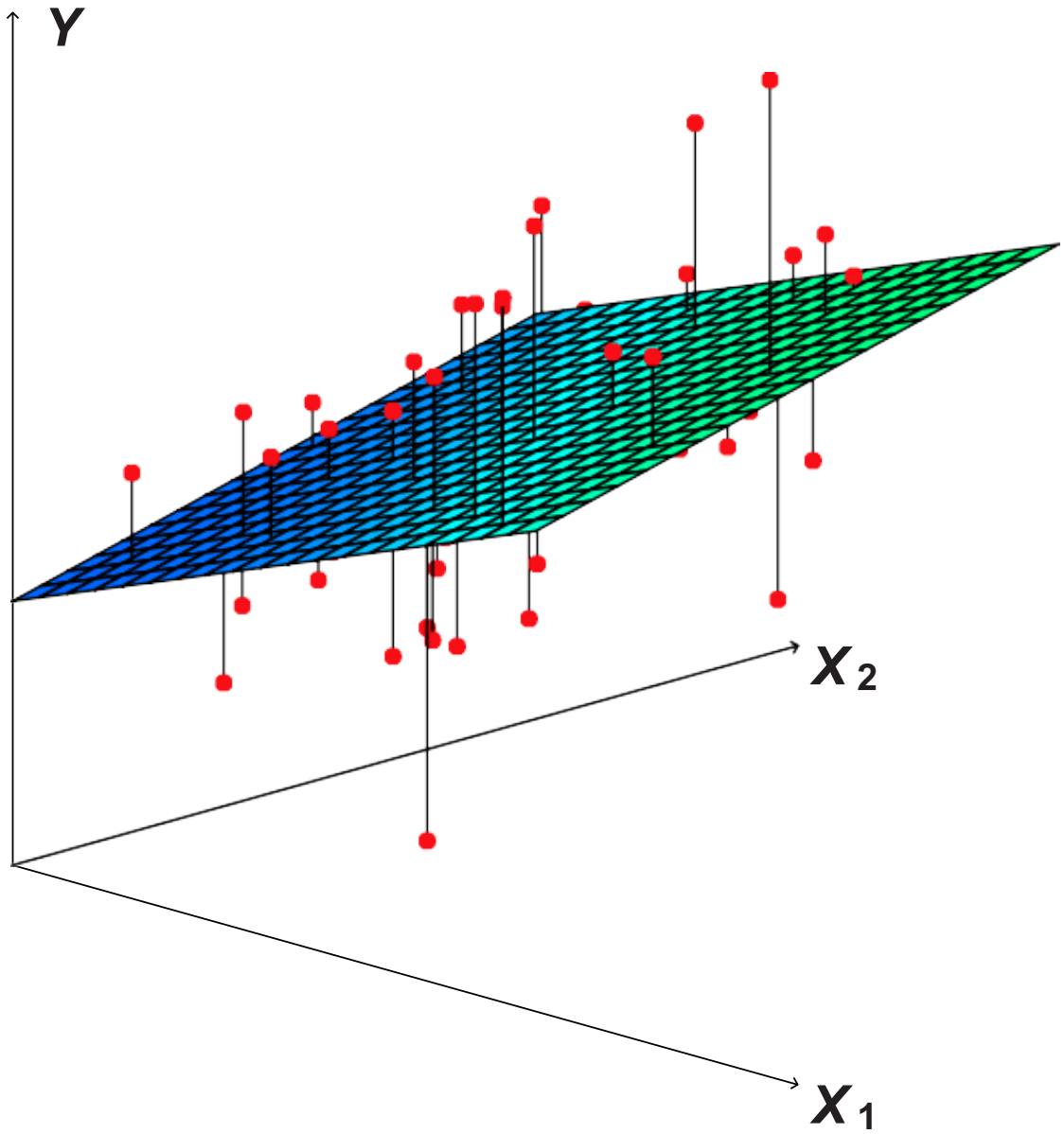
- Given estimates $\hat{\beta}_0, \hat{\beta}_1, \ldots \hat{\beta}_p$, we can make predictions using the formula

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p.$$

- We estimate $\beta_0, \beta_1, \ldots, \beta_p$ as the values that minimize the sum of squared residuals

$$\begin{aligned} \text{RSS} &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip})^2. \end{aligned}$$

This is done using standard statistical software. The values $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ that minimize RSS are the multiple least squares regression coefficient estimates.


Y
X 2
X 1

Results for advertising data

	Coefficient	Std. Error	t-statistic	p-value
Intercept	2.939	0.3119	9.42	< 0.0001
TV	0.046	0.0014	32.81	< 0.0001
radio	0.189	0.0086	21.89	< 0.0001
newspaper	-0.001	0.0059	-0.18	0.8599

Correlations:

	TV	radio	newspaper	sales
TV	1.0000	0.0548	0.0567	0.7822
radio		1.0000	0.3541	0.5762
newspaper			1.0000	0.2283
sales				1.0000

Some important questions

1. *Is at least one of the predictors $X_1, X_2, \ldots, X_p$ useful in predicting the response?*
2. *Do all the predictors help to explain Y, or is only a subset of the predictors useful?*
3. *How well does the model fit the data?*
4. *Given a set of predictor values, what response value should we predict, and how accurate is our prediction?*

Forward selection

- Begin with the ***null model*** — a model that contains an intercept but no predictors.
- Fit p simple linear regressions and add to the null model the variable that results in the lowest RSS.
- Add to that model the variable that results in the lowest RSS amongst all two-variable models.
- Continue until some stopping rule is satisfied, for example when all remaining variables have a p-value above some threshold.

Backward selection

- Start with all variables in the model.
- Remove the variable with the largest p-value — that is, the variable that is the least statistically significant.
- The new $(p - 1)$-variable model is fit, and the variable with the largest p-value is removed.
- Continue until a stopping rule is reached. For instance, we may stop when all remaining variables have a significant p-value defined by some significance threshold.

What we did not cover

- Qualitative (think categorical) predictors
- Variable interactions
- Outliers
- Non-constant variance of error terms High leverage points
- Collinearity
- See ISLR text Section 3.33

Generalizations of the Linear Model

In much of the rest of this course, we discuss methods that expand the scope of linear models and how they are fit:

- ***Classification problems:*** logistic regression, support vector machines
- ***Non-linearity:*** kernel smoothing, splines and generalized additive models; nearest neighbor methods.
- ***Interactions:*** Tree-based methods, bagging, random forests and boosting (these also capture non-linearities)
- ***Regularized fitting:*** Ridge regression and lasso

Lecture Outline

- **Introduction**
- **Linear regression from a statistical perspective**
  - Simple linear regression
  - Assessing the quality of LR model
  - Multiple linear regression (and challenges)
- **Linear regression from scratch**
  - LR via Gradient Descent via brute force
  - LR via Normal Equation (Closed-form analytical optimization)
  - LR via Gradient Descent (Numerical optimization)
  - LR via gradient descent in graph form
- **Summary**

Machine Learning: Regression

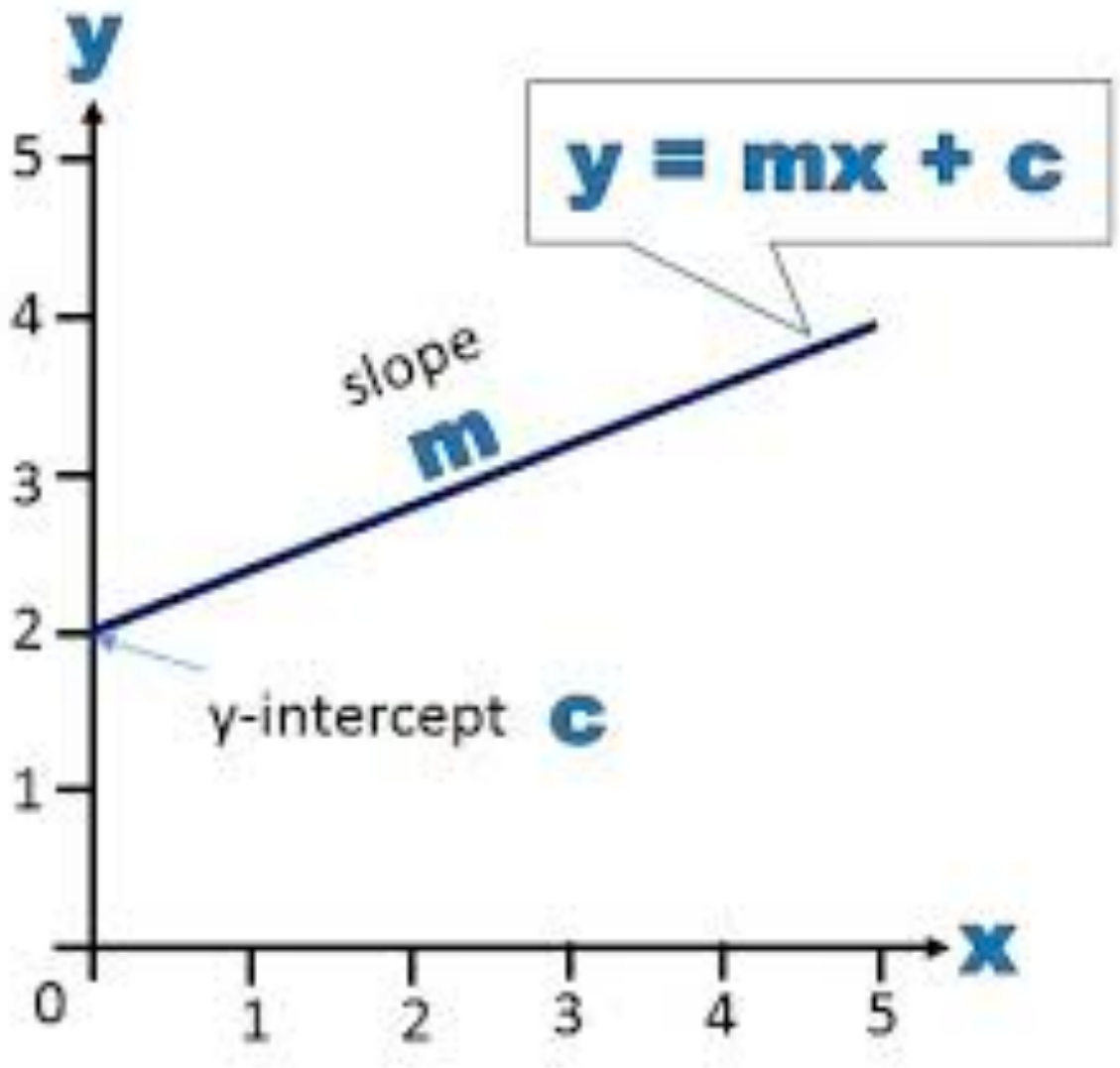
Machine Learning (ML):"*a computer program that improves its performance at some task through experience*" [Mitchell 1997]

GIVEN: Input data is a table of attribute values and associated class values (in the case of supervised learning)
GOAL: Approximate $f(x_1,\ldots,x_n)$->y

Mimimize $MSE = \frac{1}{n}\sum \text{Residual}^i = \frac{1}{n}\sum \left(WX^i - y^i\right)^2$ **Y is real valued**

Instance\Attr	x_1	x_2	…	x_n	y
1	3	0	..	7	73
2					76
…	…	…	…	…	…
L (aka m)	0	4	...	8	97

Equation of a line



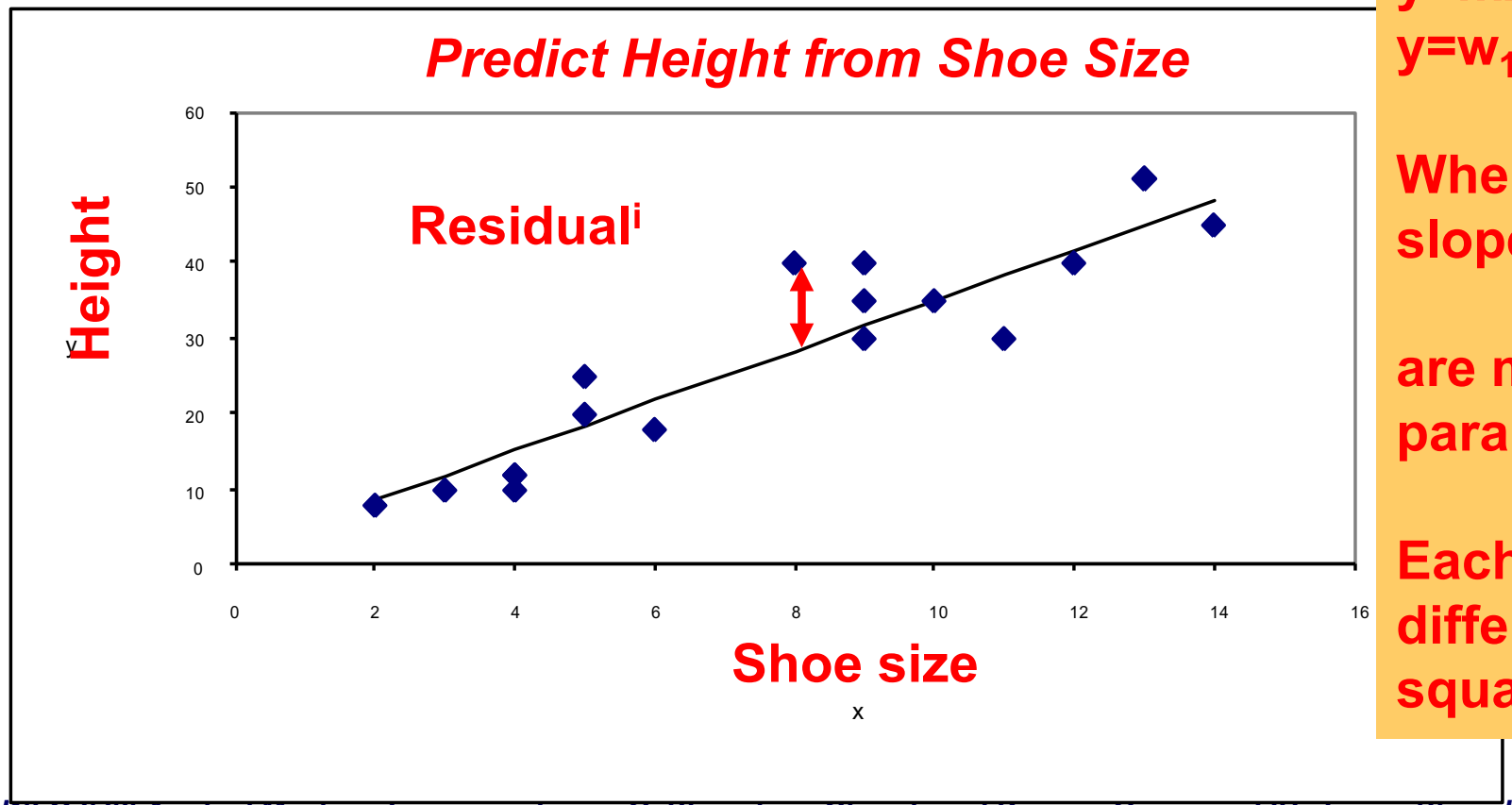
$$f(x) = mx + b$$

In math an Equation says 1,(

Equation of a line

$$\text{Residual}^{\text{i}} = \left(WX^i - y^i\right) \longrightarrow \text{Residual}^{\text{i}} = \left(WX^i - y^i\right)^2$$

Squared error loss gives us a twice differentiable function and thus we can use convex optimization



y=mx + b
$y=w_1 * x + w_0$

Where w1, w0, slope, intercept

are model parameters

Each pair yields a different sum of squares error

Linear Regression

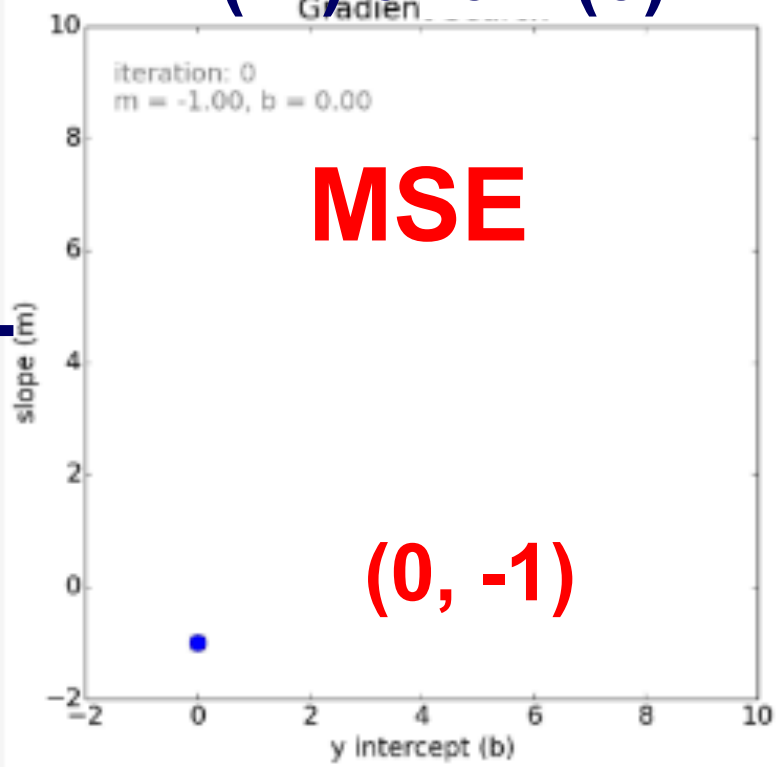
- **Next we look at a data modeling approach that has a convex objective function, that of linear regression.**

- **We will exploit this fact to develop 2 learning algorithms**
  - These algorithms can be scaled up through parallelization in a map-reduce framework such as Spark

- **Linear Regression**
  - Bruteforce
  - closed form
  - gradient descent

Version Space for linear regression

Model Space F(W) aka F(θ)

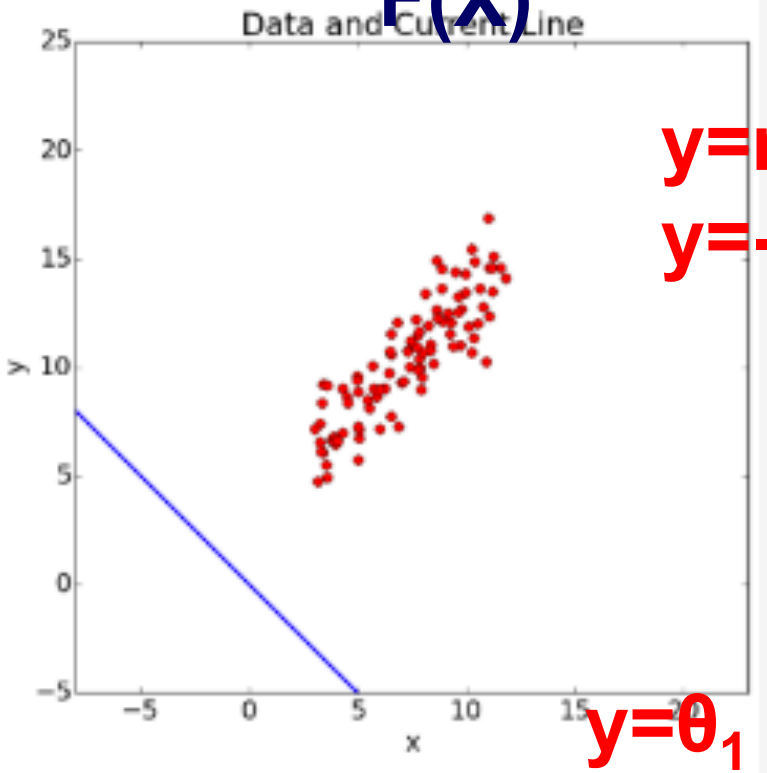


MSE

(0, -1)

Slope

Problem Space F(X)



y=m * x + b

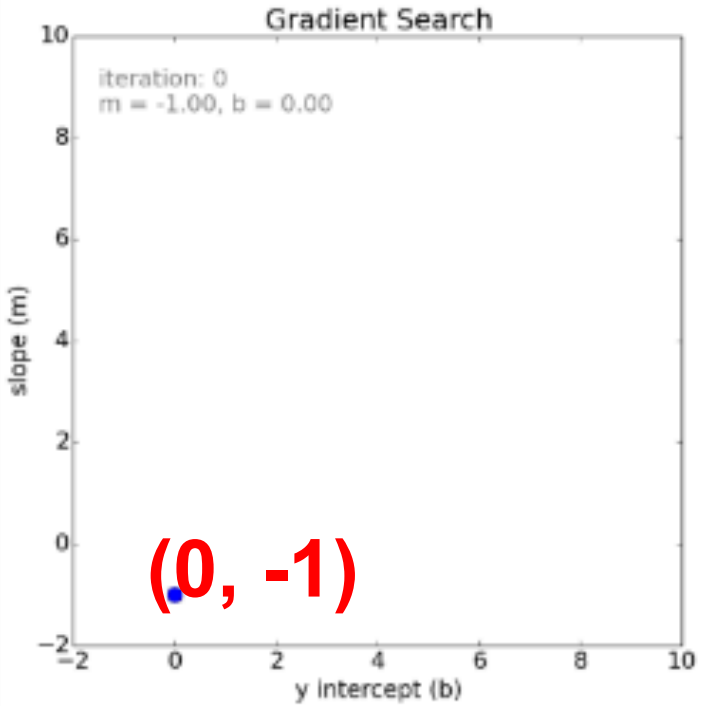
y=-1 * x + 0

$y=\theta_1 * x + \theta_0$

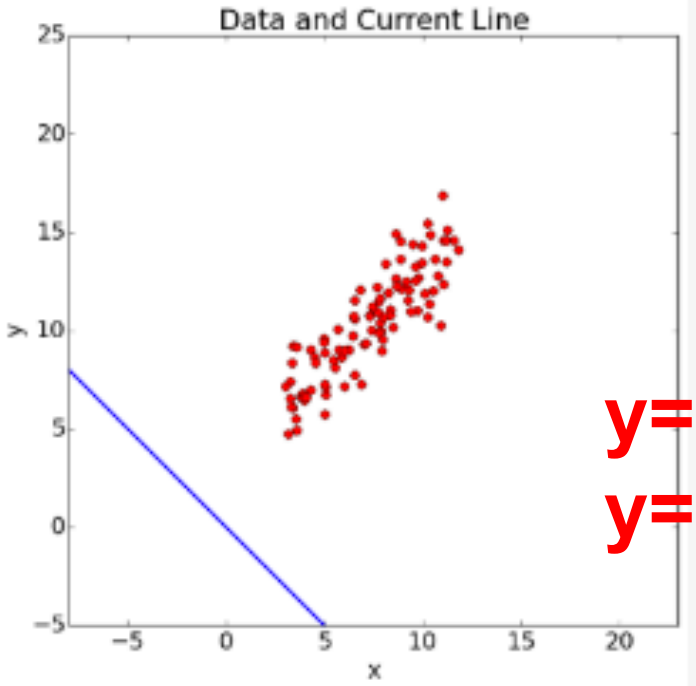
y=-1 * x + 0

Intercept

$$MSE = \frac{1}{n}\sum \text{Residual}^i = \frac{1}{n}\sum \left(WX^i - y^i\right)^2$$

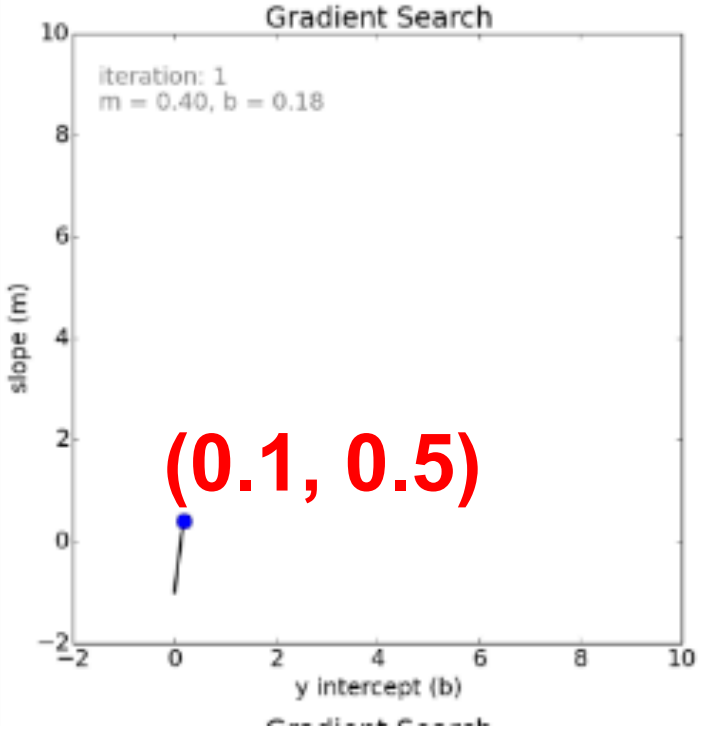


(0, -1)

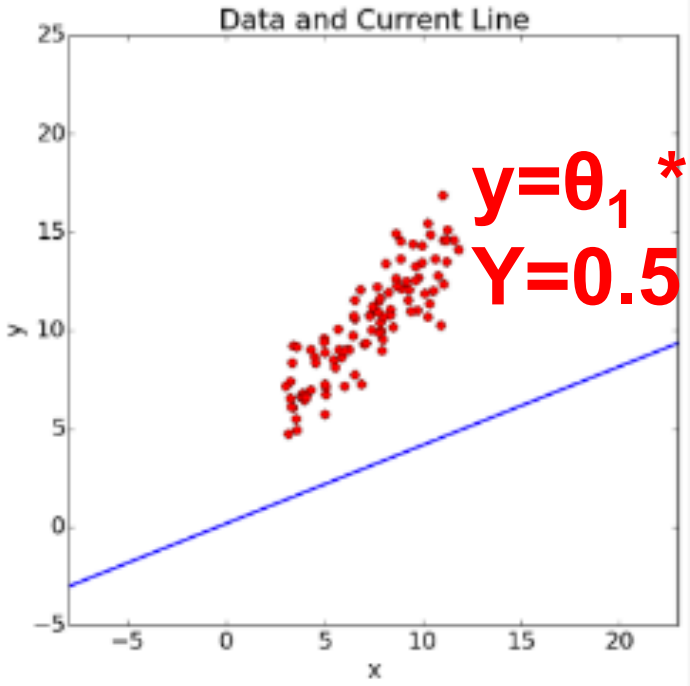


$y=\theta_1 * x + \theta$

$y=-1 * x + 0$



(0.1, 0.5)

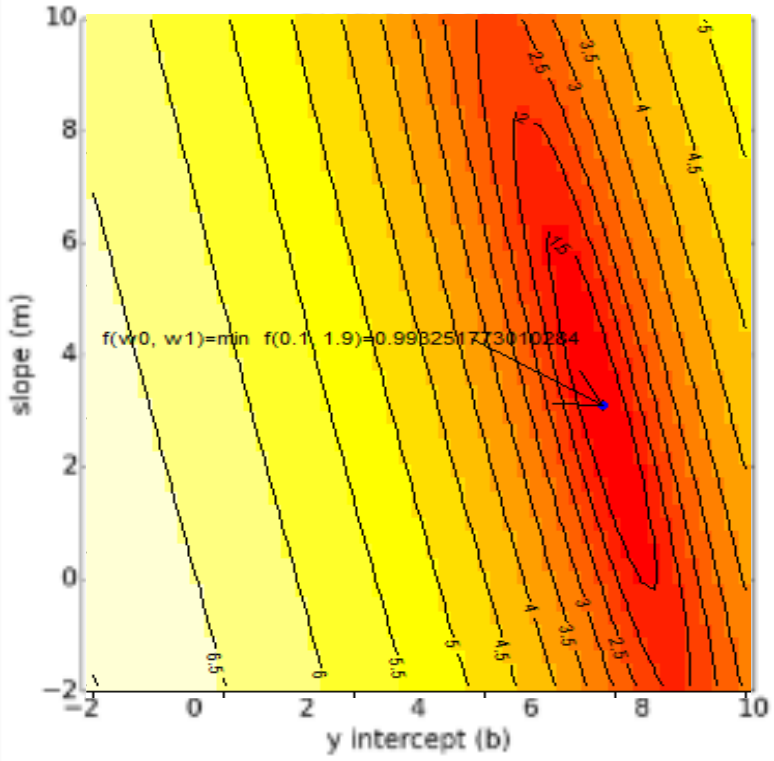


$y=\theta_1 * x + \theta_0$

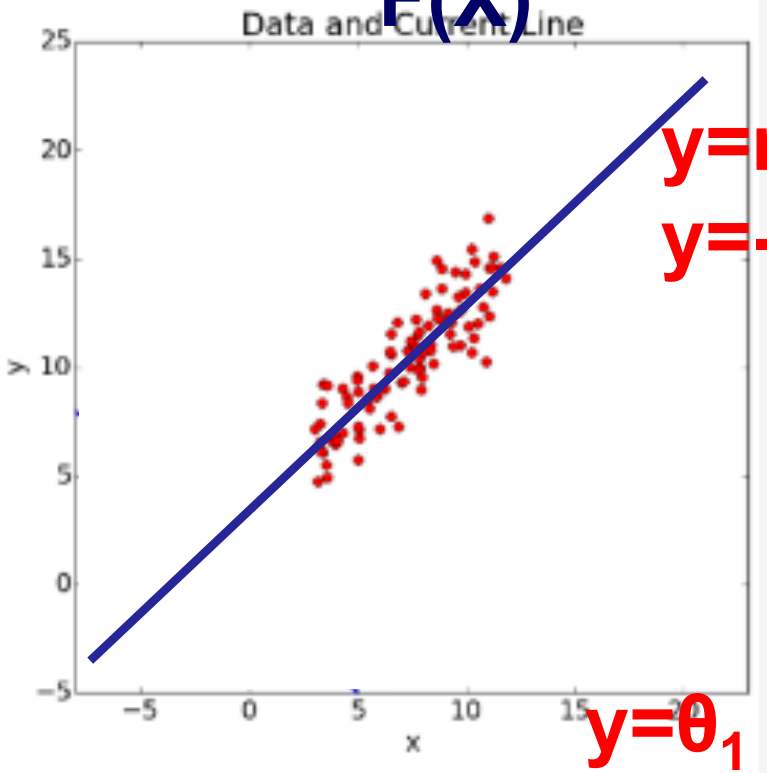
$Y=0.5 * x + 0.1$

Plot the error for different slopes and intercepts

MSE Surface



Problem Space F(X)



$y = m * x + b$

$y = -1 * x + 0$

$y = \theta_1 * x + \theta_0$

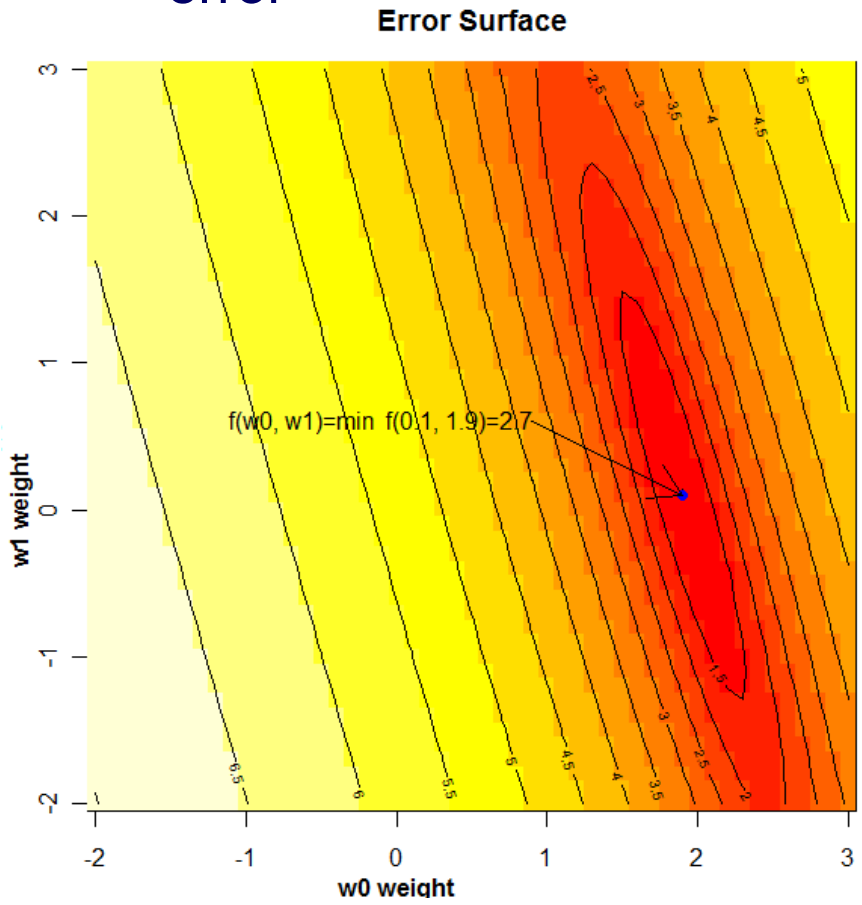
$y = -1 * x + 0$

Search the MSE Error surface using Calculus (gradient descent): Tensorflow, Caffe, Torch, Theano, etc

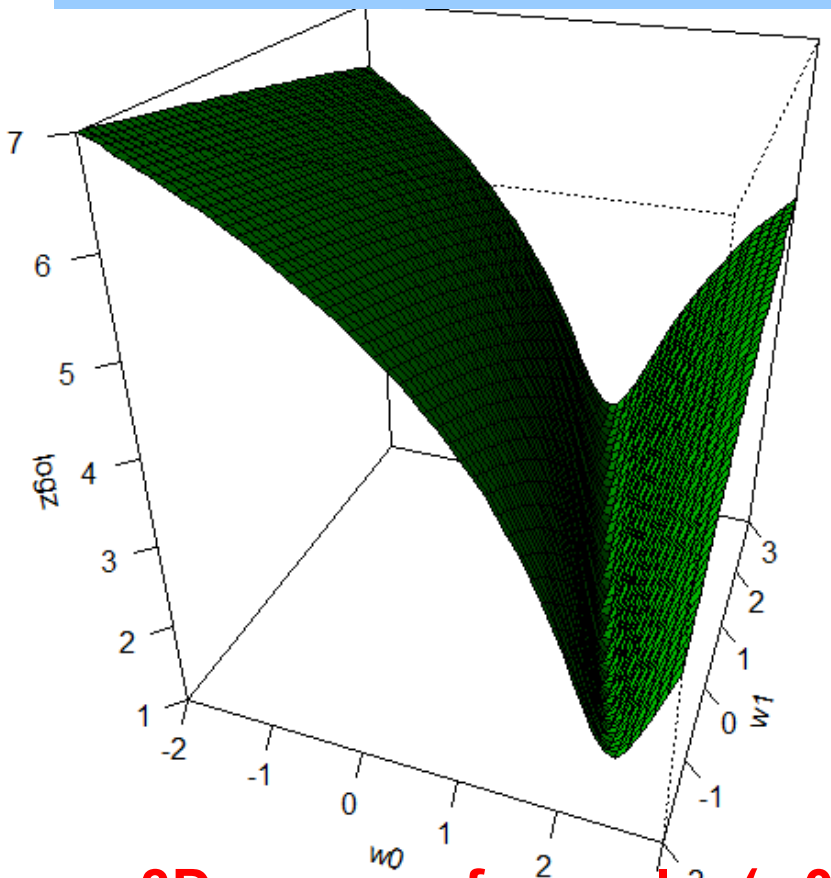
Brute Force Search of Weights

- Enumerate all possible coefficient combinations (in our case coefficient for the y-intercept (bias) and for the c-variable (time)
- Select the weight combination that minimizes the sum of square error

example.OLS_Heatmap()



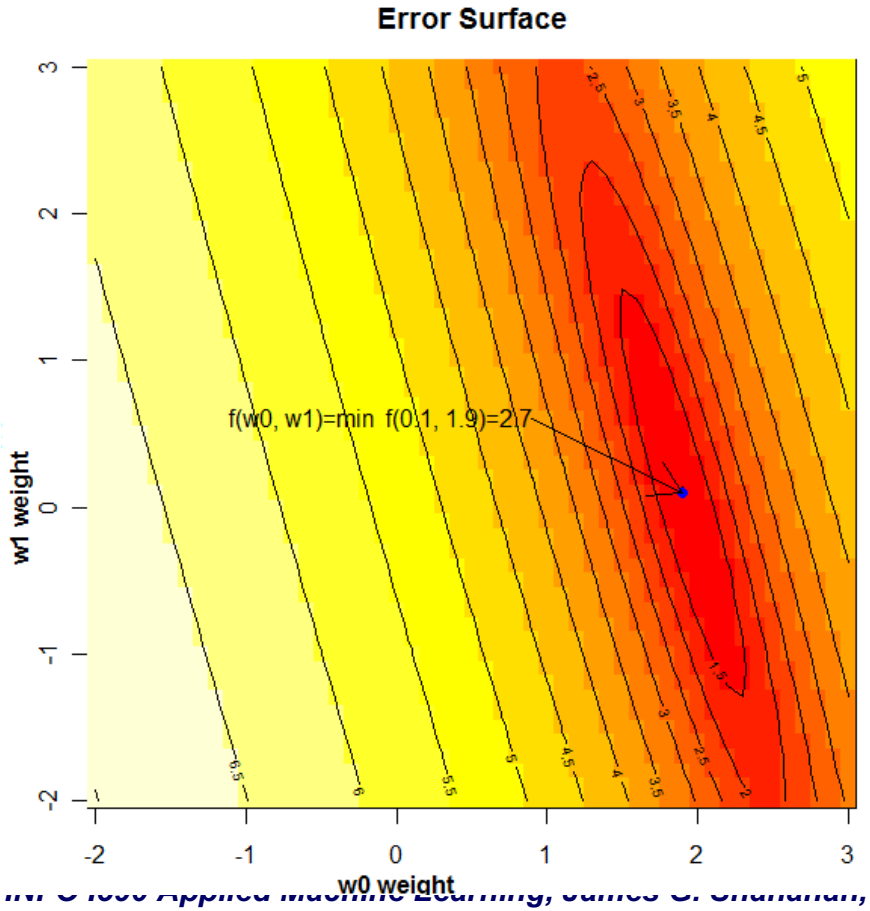
HeatMap with isolines overlayed

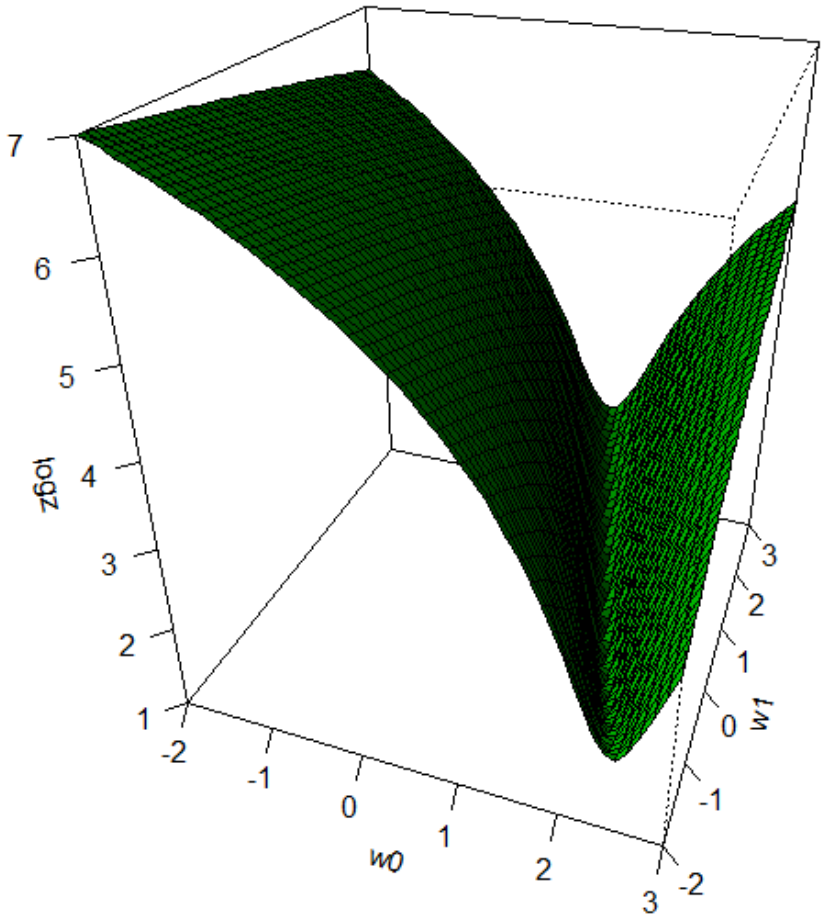


3D error surface z=log(w0+w1∗x)

Brute Force Search of Weights

- Very inefficient; at best we can only approximate the surface
- Not scaleable
- Avoid this approach…





Lecture Outline

- **Introduction**
- **Linear regression from a statistical perspective**
  - Simple linear regression
  - Assessing the quality of LR model
  - Multiple linear regression (and challenges)
- **Linear regression from scratch**
  - LR via Gradient Descent via brute force
  - LR via Normal Equation (Closed-form analytical optimization)
  - LR via Gradient Descent (Numerical optimization)
  - LR via gradient descent in graph form
- **Summary**

Maximum vs Minimum

$$f(x)= x^3 - 12x + 1$$

First derivative

$$f'(x) = 3x^2 - 12$$

FOC

f'(x=x*) =0 at maximum and minimum

These zero points are the roots!

Second derivative f''(x) = 6x

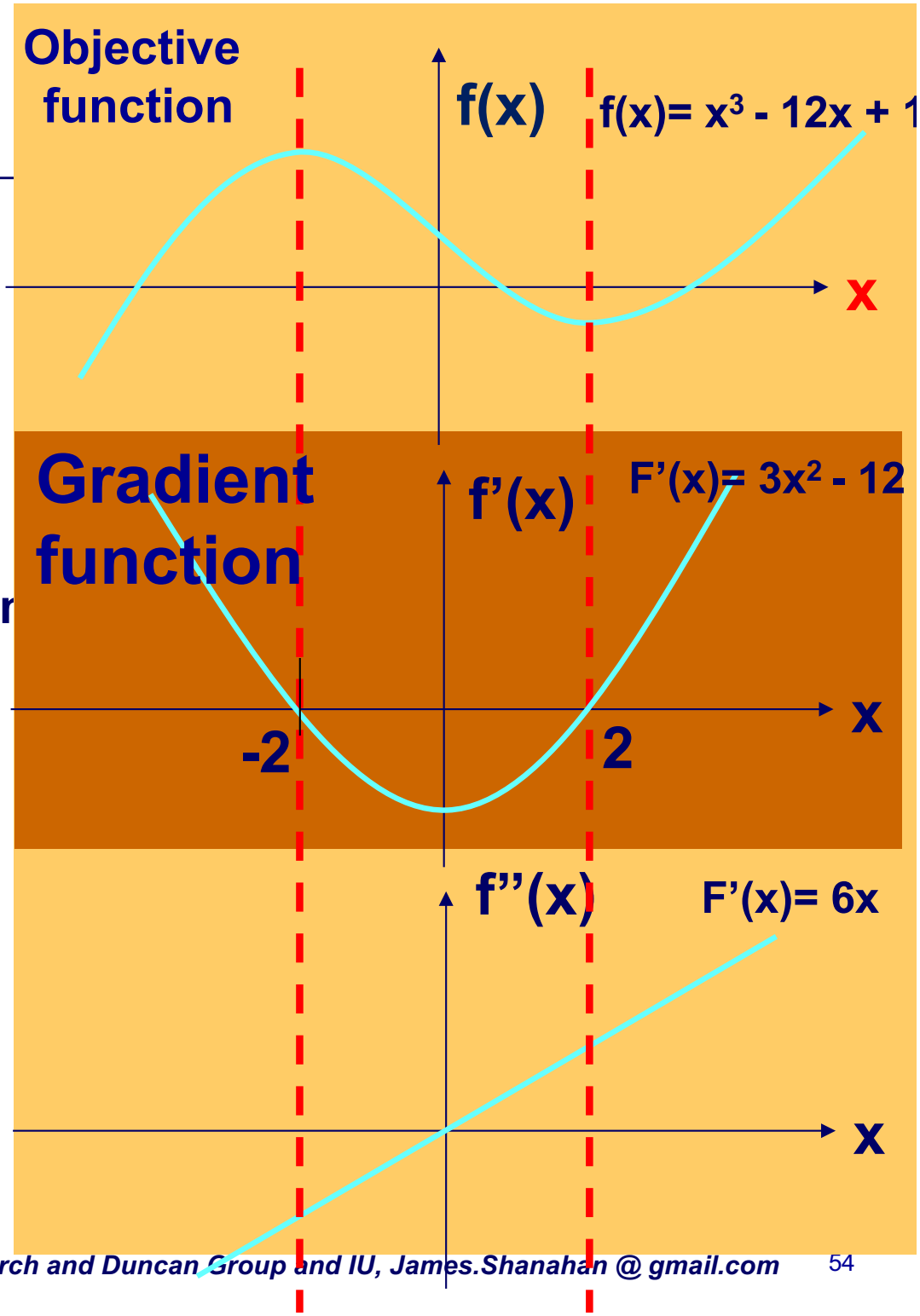
Gives the "rate of change of gradient"

SOC

If f''(x=x*) < 0 then maximum

f''(x=x*) > 0 then minimum

f''(x=x*) == 0 then ???



Y=m X x + b X 1
$y=\theta_1 x + \theta_0 1$

vs Minimum

$$f(\theta)= \theta^3 - 12\theta + 1$$

First derivative

$$f'(\theta) = 3\theta^2 - 12$$

In convex optimisation what can we say about roots?

$F'(\theta)= 3\theta^2 - 12$
$0 = 3\theta^2 - 12$
$\Theta = 2$ or -2

FOC

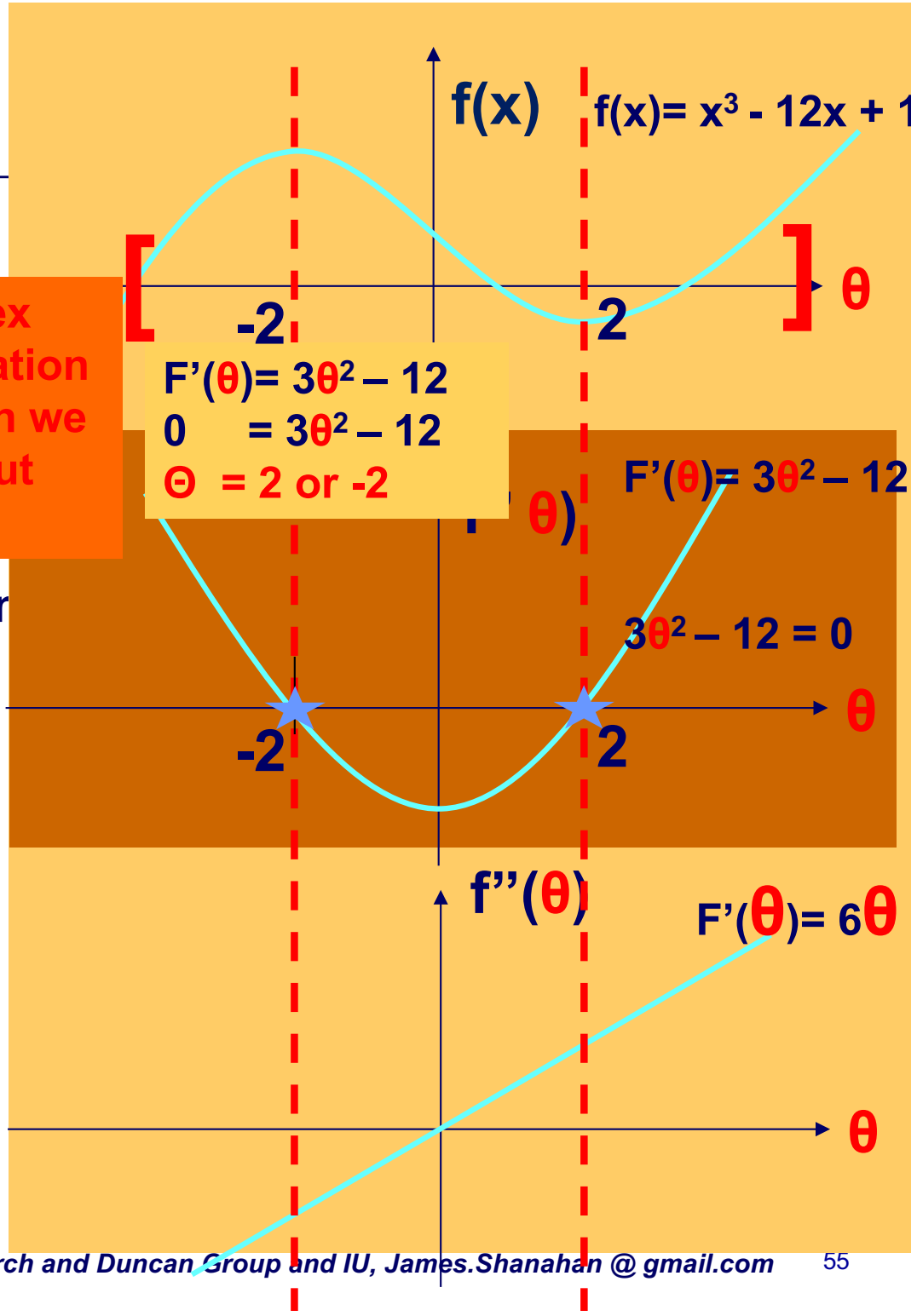
$f'(\theta=\theta^*) = 0$ at maximum and minimur
These zero points are the roots!

Second derivative $f''(\theta) = 6\theta$

Gives the "rate of change of gradient"

SOC

If $f''(x=x^*) < 0$ then maximum
$f''(x=x^*) > 0$ then minimum
$f''(x=x^*) == 0$ then ???



Augmented Data

Instance\Attr	x_1	x_2	…	x_n	y
1	3	0	..	7	-1
2					+1
…	…	…	…	…	…
L (aka m)	0	4	…	8	-1

Augmented Dataset

Instance\Attr	x_0	x_1	x_2	…	x_n	y
1	1	-3	0	..	-7	-1
2	1					+1
…	1	…	…	…	…	…
L (aka m)	1	0	4	…	8	-1

Model as an Augmented weight vector

$$W = \begin{bmatrix} w_0 \\ w_1 \\ .. \\ w_n \end{bmatrix} = \begin{bmatrix} b \\ w_1 \\ .. \\ w_n \end{bmatrix}$$

Augmented Data vector

$$X = \begin{bmatrix} \\ x_1 \\ .. \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ .. \\ x_n \end{bmatrix}$$

Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta^T \cdot \mathbf{x}$$

- θ is the model's *parameter vector*, containing the bias term θ_0 and the feature weights θ_1 to θ_n.
- θ^T is the transpose of θ (a row vector instead of a column vector).
- $\mathbf{x}$ is the instance's *feature vector*, containing x_0 to x_n, with x_0 always equal to 1.
- $\theta^T \cdot \mathbf{x}$ is the dot product of θ^T and $\mathbf{x}$.
- h_θ is the hypothesis function, using the model parameters θ.

$$\mathrm{MSE}(\mathbf{X}, h_\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(\theta^T \cdot \mathbf{x}^{(i)} - y^{(i)}\right)^2$$

Single variable $$\frac{\partial}{\partial \theta_j} \mathrm{MSE}(\theta) = \frac{2}{m} \sum_{i=1}^{m} \left(\theta^T \cdot \mathbf{x}^{(i)} - y^{(i)}\right) x_j^{(i)}$$

LR closed-form solution: Normal Equation

$$\mathrm{MSE}(\mathbf{X}, h_\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(\theta^T \cdot \mathbf{x}^{(i)} - y^{(i)}\right)^2$$

MSE Loss Function (Objective)

$$\nabla_\theta \mathrm{MSE}(\theta) = \begin{pmatrix} \frac{\partial}{\partial\theta_0}\mathrm{MSE}(\theta) \\ \frac{\partial}{\partial\theta_1}\mathrm{MSE}(\theta) \\ \vdots \\ \frac{\partial}{\partial\theta_n}\mathrm{MSE}(\theta) \end{pmatrix} = \frac{2}{m}\mathbf{X}^T \cdot (\mathbf{X} \cdot \theta - \mathbf{y})$$

Gradient of partial derivatives

Solve for θ (aka W; aka β)

$$\hat{\theta} = \left(\mathbf{X}^T \cdot \mathbf{X}\right)^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$$

- $\hat{\theta}$ is the value of θ that minimizes the cost function.
- y is the vector of target values containing $y^{(1)}$ to $y^{(m)}$.

LR closed-form solution: Normal Equation

- **To find the value of θ that minimizes the cost function, there is a closed-form solution —in other words, a mathematical equation that gives the result directly. This is called the Normal Equation**

$$\hat{\theta} = \left(\mathbf{X}^T \cdot \mathbf{X}\right)^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$$

- $\hat{\theta}$ is the value of θ that minimizes the cost function.
- y is the vector of target values containing $y^{(1)}$ to $y^{(m)}$.

Now let's compute $\hat{\theta}$ using the Normal Equation. We will use the `inv()` function from NumPy's Linear Algebra module (`np.linalg`) to compute the inverse of a matrix, and the `dot()` method for matrix multiplication:

```
X_b = np.c_[np.ones((100, 1)), X]  # add x0 = 1 to each instance
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
```

The actual function that we used to generate the data is $y = 4 + 3x_0 + \text{Gaussian noise}$. Let's see what the equation found:

```
>>> theta_best
array([[ 4.21509616],
       [ 2.77011339]])
```

Chain rule

From Wikipedia, the free encyclopedia

For other uses, see Chain rule (disambiguation).

In calculus, the **chain rule** is a formula for computing the derivative of the composition of two or more functions. That is, if f is a function and g is a function, then the chain rule expresses the derivative of the composite function $f \circ g$ in terms of the derivatives of f and g. For example, the chain rule for $f \circ g(x) = f[g(x)]$ is

$$\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}.$$

In integration, the counterpart to the chain rule is the substitution rule.

The simplest form of the chain rule is for real-valued functions of one real variable. It says that if g is a function that is differentiable at a point c (i.e. the derivative $g'(c)$ exists) and f is a function that is differentiable at $g(c)$, then the composite function $f \circ g$ is differentiable at c, and the derivative is[2]

$$(f \circ g)'(c) = f'(g(c)) \cdot g'(c).$$

The rule is sometimes abbreviated as

$$(f \circ g)' = (f' \circ g) \cdot g'.$$

If $y = f(u)$ and $u = g(x)$, then this abbreviated form is written in Leibniz notation as:

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}.$$

OLS Via Gradient Descent: The Gradient

$$W_{j,t+1} = W_{j,t} - \alpha * \nabla J_{w_j}(W_{j,t})$$

- In order to implement this algorithm, we have to work out what is the partial derivative term at time t on the right hand side $\nabla f_{wj}(W)=dF(W)/dw_i$.
- Assume we have only one training example (x, y), so that we can drop the sum in the definition of J.

$$\nabla J_{W_j}(W) = \frac{\partial}{\partial W_j} J(W) = \frac{\partial}{\partial W_j}\left(\frac{1}{2}\left(f_W(x)-y\right)^2\right)$$

Use chain rule df/du*du/dx

$$= 2 * \frac{1}{2}\left(f_W(x)-y\right)\frac{\partial}{\partial W_j}\left(f_W(x)-y\right)$$

Assume a single training example

$$= \left(f_W(x)-y\right)\frac{\partial}{\partial W_j}\left(\left(\sum_{i=0}^{n} w_i x_i\right)-y\right)$$

For a single w_j

$$= \left(f_W(x)-y\right)x_j$$

Recall

$$\frac{\partial}{\partial W_j}\left(\left(\sum_{i=0}^{n} w_i x_i\right)-y\right) = \frac{\partial}{\partial W_j}\left(w_0x_0 + w_1x_1 \ldots + w_jx_j + \ldots w_nx_n\right)$$

$$= 0+0+\ldots x_j + \ldots 0$$

Lecture Outline

- **Introduction**
- **Linear regression from a statistical perspective**
  - Simple linear regression
  - Assessing the quality of LR model
  - Multiple linear regression (and challenges)
- **Linear regression from scratch**
  - LR via Gradient Descent via brute force
  - LR via Normal Equation (Closed-form analytical optimization)
  - LR via Gradient Descent (Numerical optimization)
  - LR via gradient descent in graph form
- **Summary**

Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta^T \cdot \mathbf{x}$$

- θ is the model's *parameter vector*, containing the bias term θ_0 and the feature weights θ_1 to θ_n.
- θ^T is the transpose of θ (a row vector instead of a column vector).
- $\mathbf{x}$ is the instance's *feature vector*, containing x_0 to x_n, with x_0 always equal to 1.
- $\theta^T \cdot \mathbf{x}$ is the dot product of θ^T and $\mathbf{x}$.
- h_θ is the hypothesis function, using the model parameters θ.

$$\mathrm{MSE}(\mathbf{X}, h_\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(\theta^T \cdot \mathbf{x}^{(i)} - y^{(i)}\right)^2$$

Single variable $$\frac{\partial}{\partial \theta_j} \mathrm{MSE}(\theta) = \frac{2}{m} \sum_{i=1}^{m} \left(\theta^T \cdot \mathbf{x}^{(i)} - y^{(i)}\right) x_j^{(i)}$$

Gradient Descent in vector form

$$\mathrm{MSE}(\mathbf{X}, h_\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(\theta^T \cdot \mathbf{x}^{(i)} - y^{(i)}\right)^2$$

MSE Loss Function (Objective)

$$\nabla_\theta \mathrm{MSE}(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_0}\mathrm{MSE}(\theta) \\ \frac{\partial}{\partial \theta_1}\mathrm{MSE}(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n}\mathrm{MSE}(\theta) \end{pmatrix} = \frac{2}{m}\mathbf{X}^T \cdot (\mathbf{X} \cdot \theta - \mathbf{y})$$

Gradient of partial derivatives

$$\theta^{(\text{next step})} = \theta - \eta \nabla_\theta \mathrm{MSE}(\theta)$$

GD Update rule

Implementation

```
eta = 0.1  # learning rate
n_iterations = 1000
m = 100

theta = np.random.randn(2,1)  # random initialization

for iteration in range(n_iterations):
    gradients = 2/m * X_b.T.dot(X_b.dot(theta) - y)
    theta = theta - eta * gradients
```

Slope in 1D; derivative in n-Dims

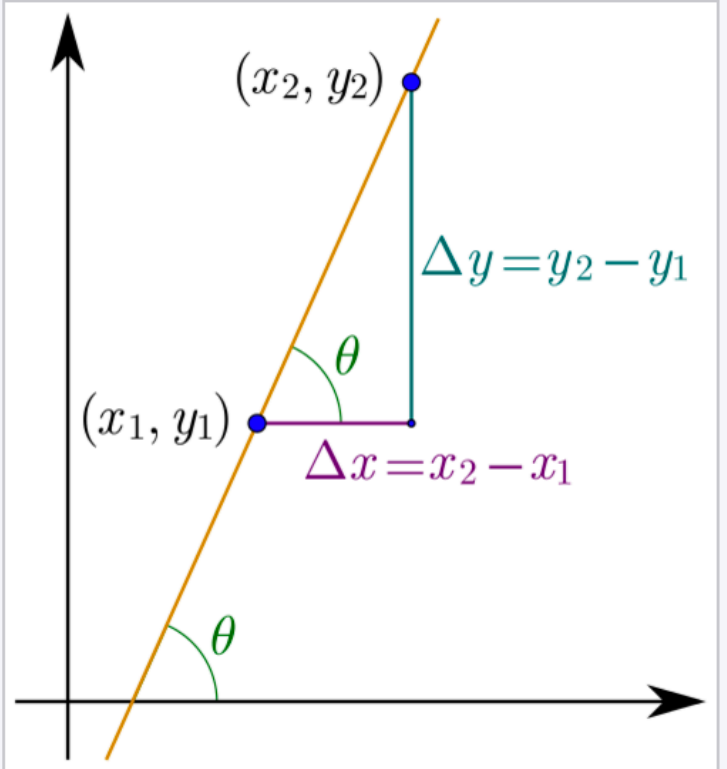
- The derivative of a function at a chosen input value describes the rate of change of the function near that input value.
- Helps us understand function optima (FOC &SOC)

- APPROACH 1: Numerical gradient (using the secant at the limit) via divided difference approach

$$\lim \frac{f(x+h)-f(x)}{h} = \lim \frac{(x+h)^2 - x^2}{h}$$

- APPROACH 2 : Analytical gradient via calculus

$$\frac{df(x^2)}{dx} = 2x$$



Slope of a linear function: $m = \frac{\Delta y}{\Delta x}$

Rate of change as a limit value

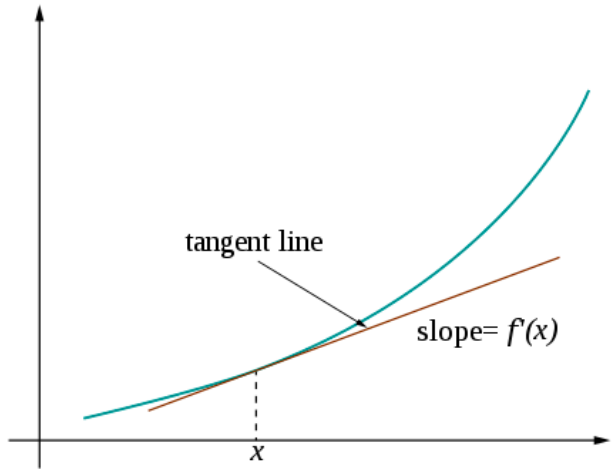


Figure 1. The tangent line at (*x*, *f*(*x*))

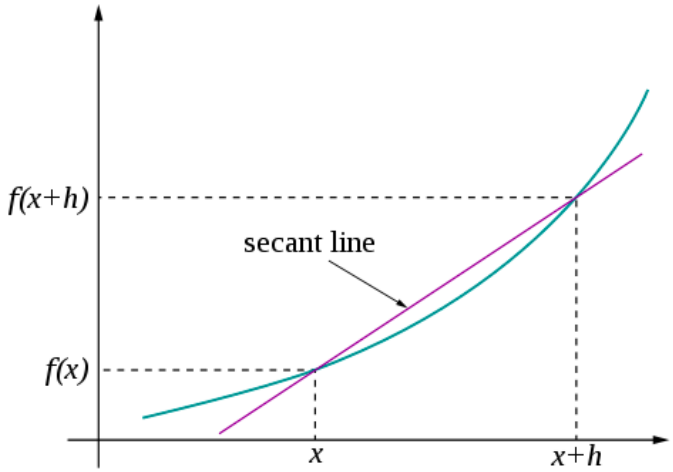
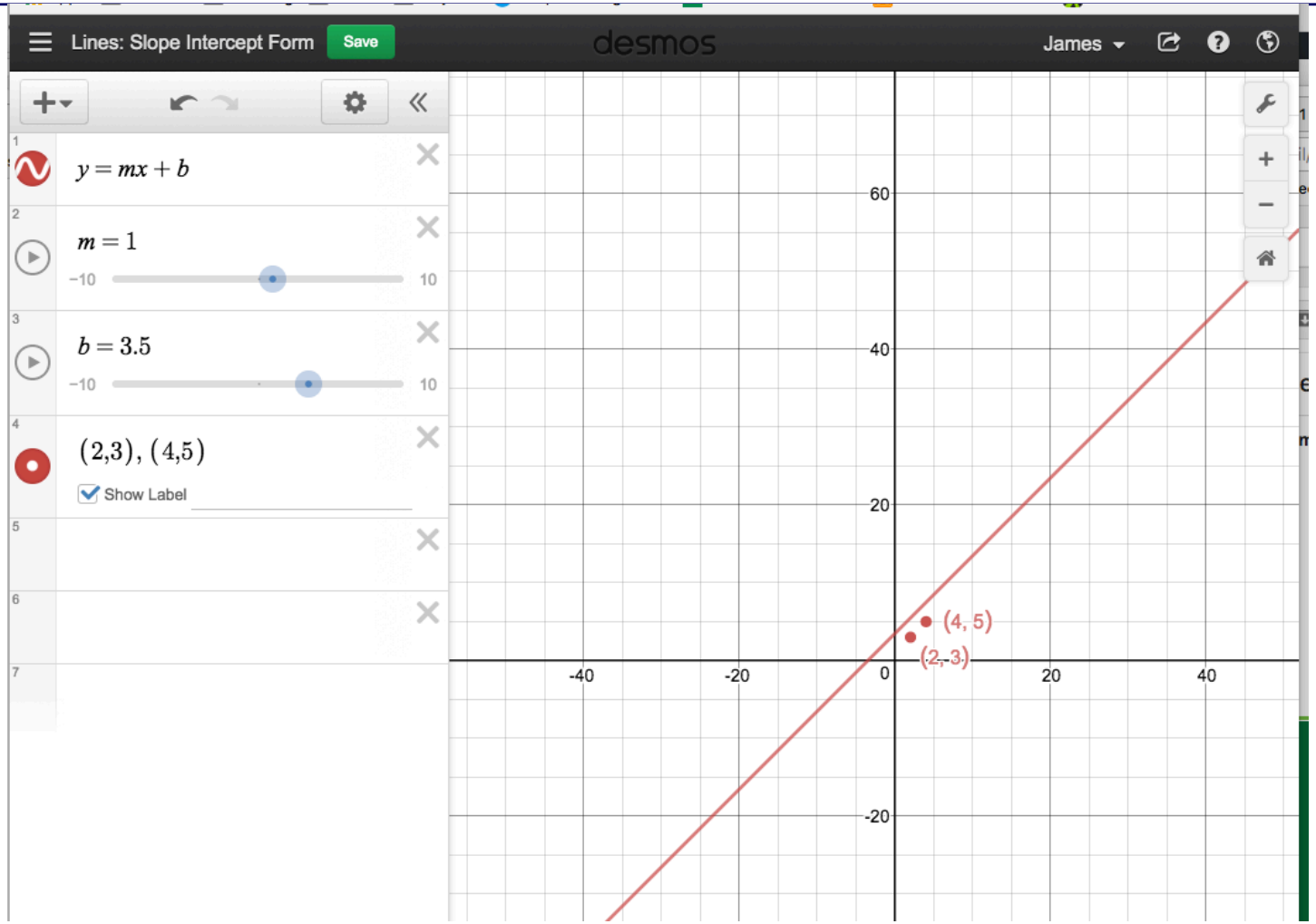


Figure 2. The secant to curve *y*= *f*(*x*) determined by points (*x*, *f*(*x*)) and (*x*+*h*, *f*(*x*+*h*))

https://en.wikipedia.org/wiki/Derivative

Line: slope and intercept



https://www.desmos.com/calculator/sxwozg33me

Slope in 1D; derivative in n-Dims

- The derivative of a function at a chosen input value describes the rate of change of the function near that input value.
- Helps us understand function optima (FOC &SOC)

- APPROACH 1: Numerical gradient (using the secant at the limit) via divided difference approach

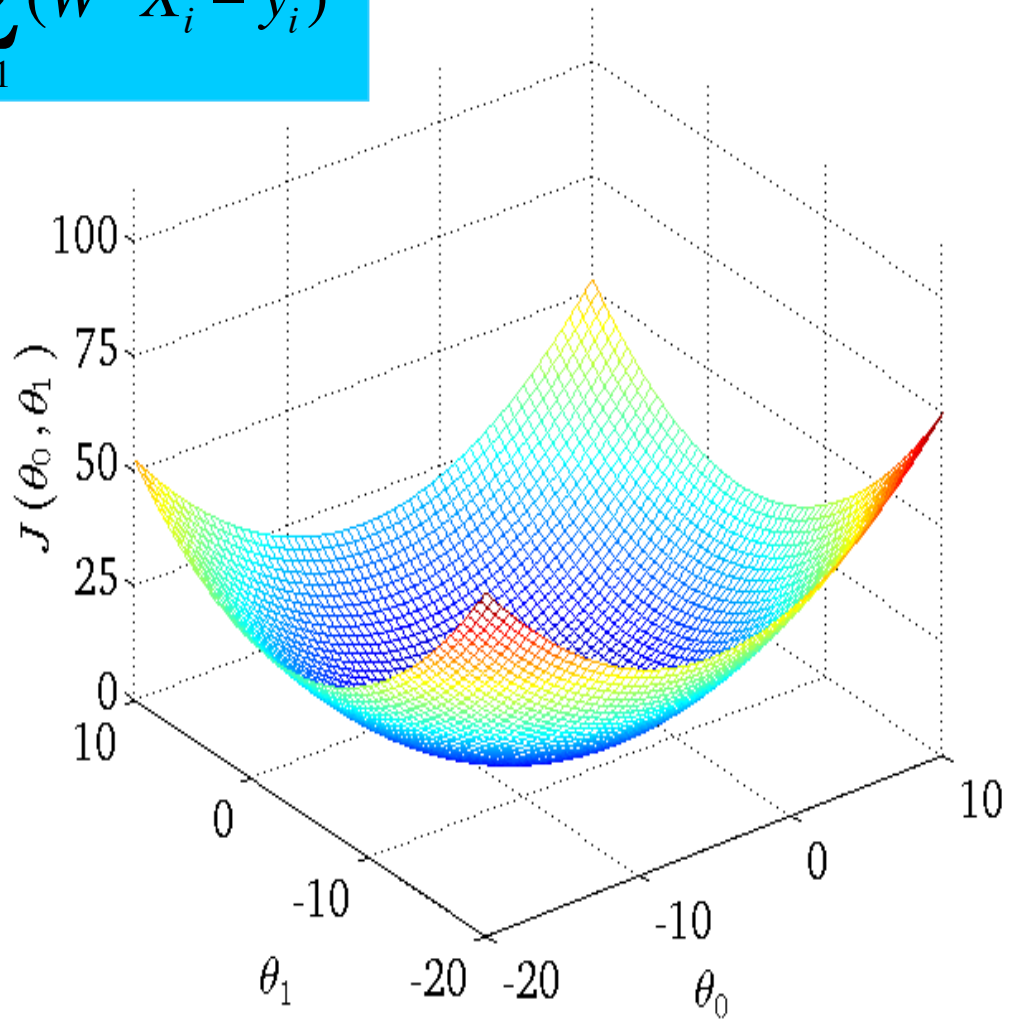
$$\lim \frac{f(x+h)-f(x)}{h} = \lim \frac{(x+h)^2 - x^2}{h}$$

- APPROACH 2 : Analytical gradient via calculus

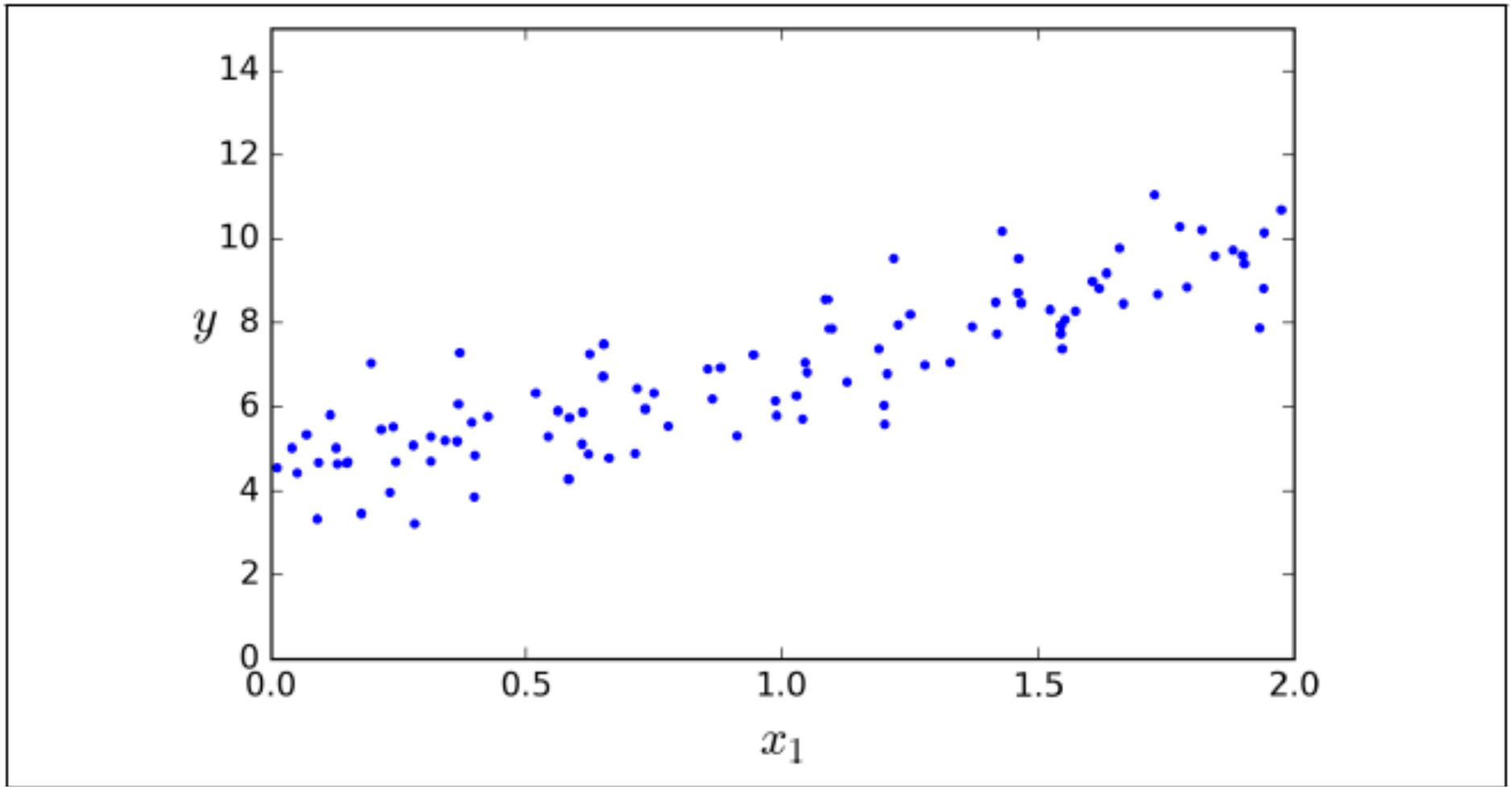
$$\frac{df(x)}{dx}$$

Error Surface for OLS: Price~Size

$$J_q(W, X_1^L) = \frac{1}{m}\sum_{i=1}^{m}(W^T X_i - y_i)^2$$



```
import numpy as np

X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)
```



Gradient Descent for LR Price~Size Iteration 0

Eqn Line y= aX + b

Y = w1X + W0 1

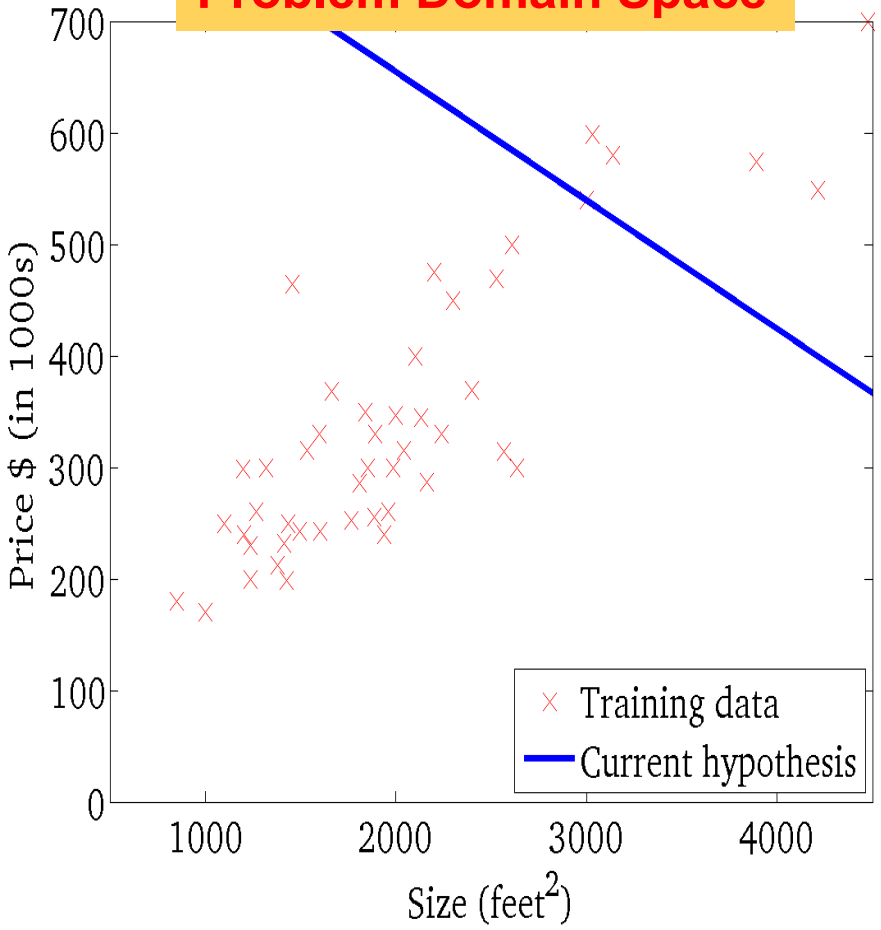
$J(\theta_0, \theta_1$

$$J_q(W, X_1^L) = \frac{1}{m}\sum_{i=1}^{m}(W^T X_i - y_i)^2$$

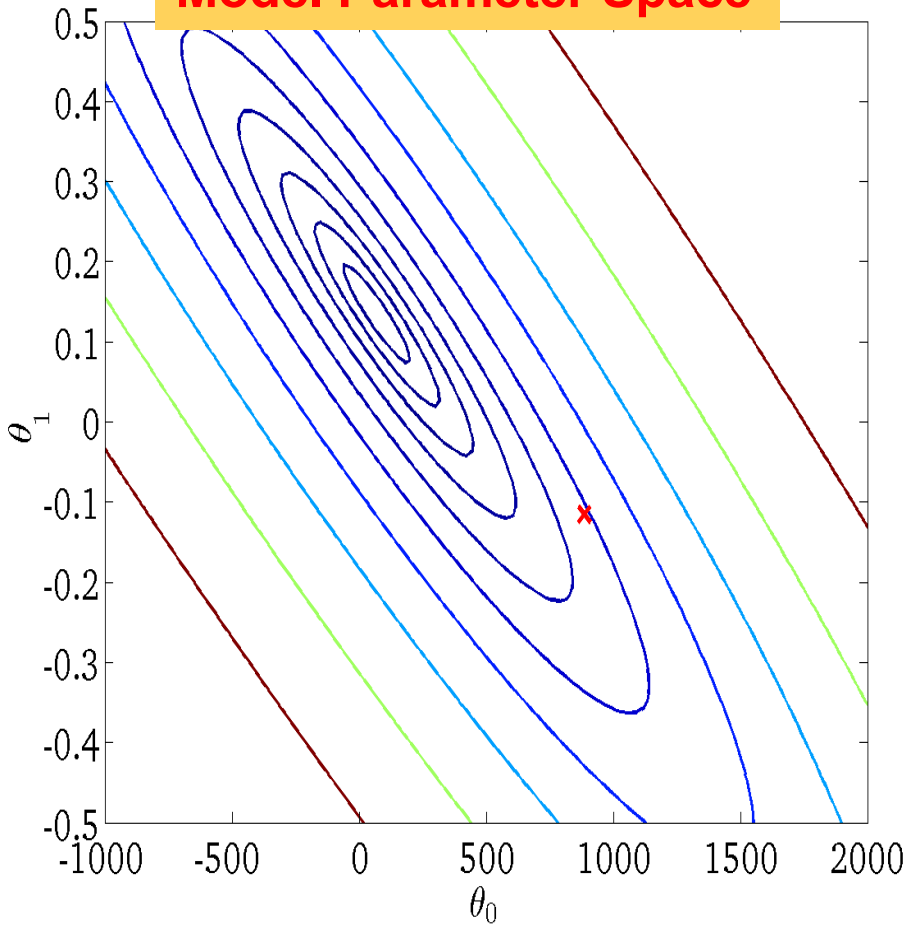
(for fixed θ_0, θ_1, this is a function of x)

(function of the parameters θ_0, θ_1)

Problem Domain Space



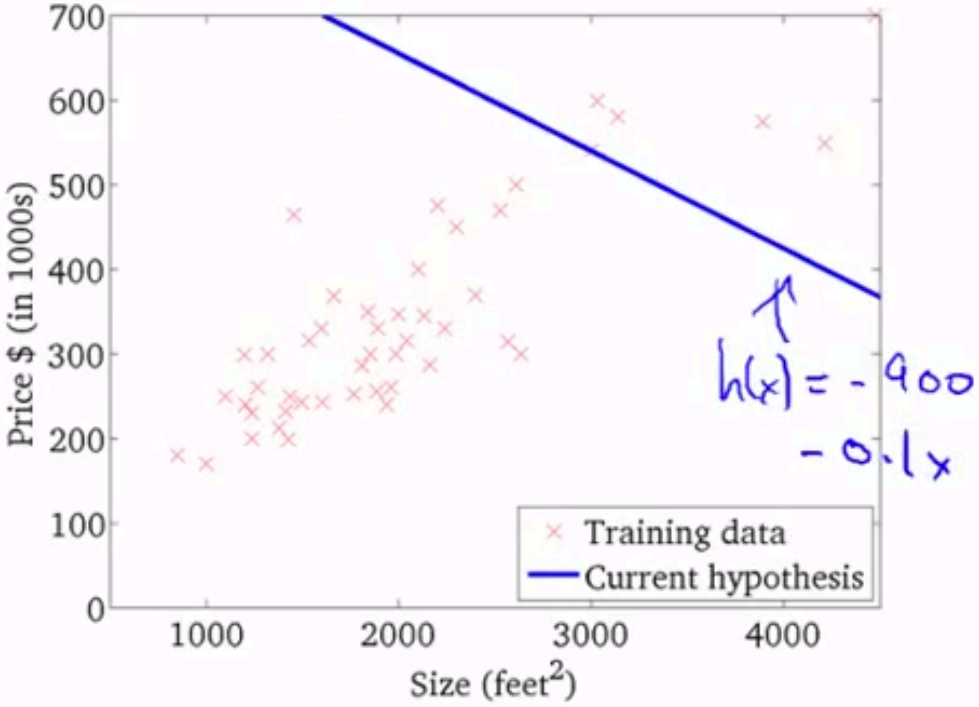
Model Parameter Space



Gradient Descent for LR Price~Size – Iteration 0

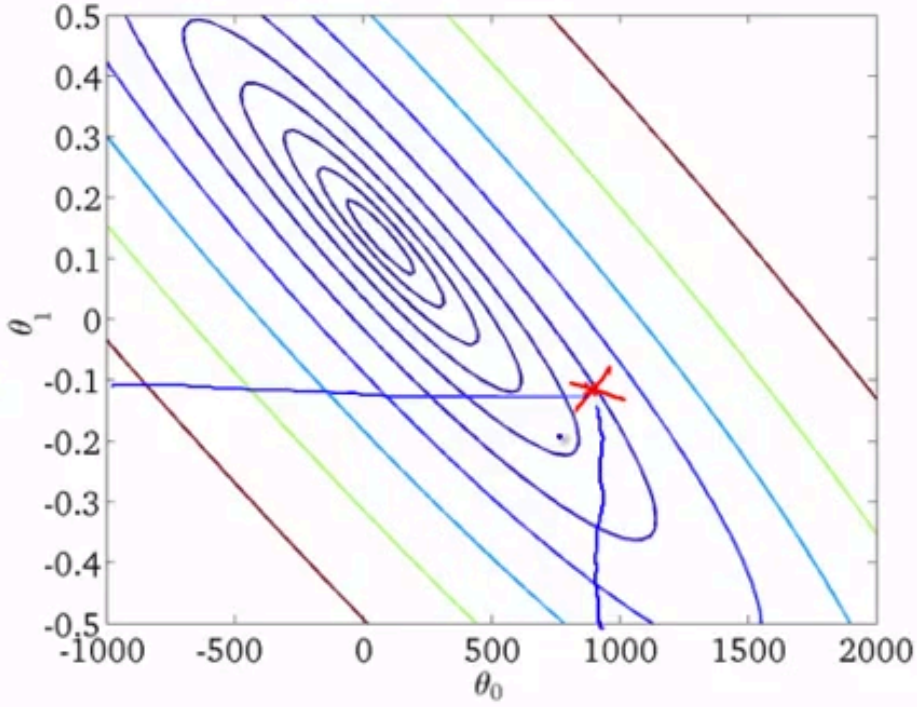
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

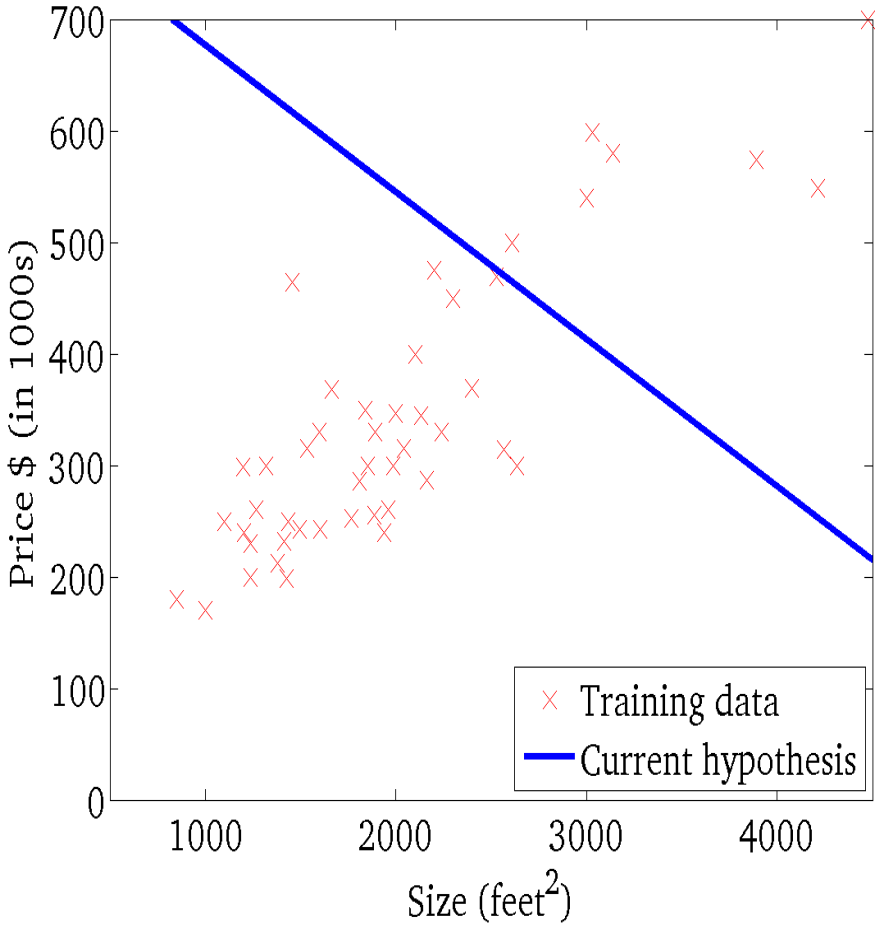


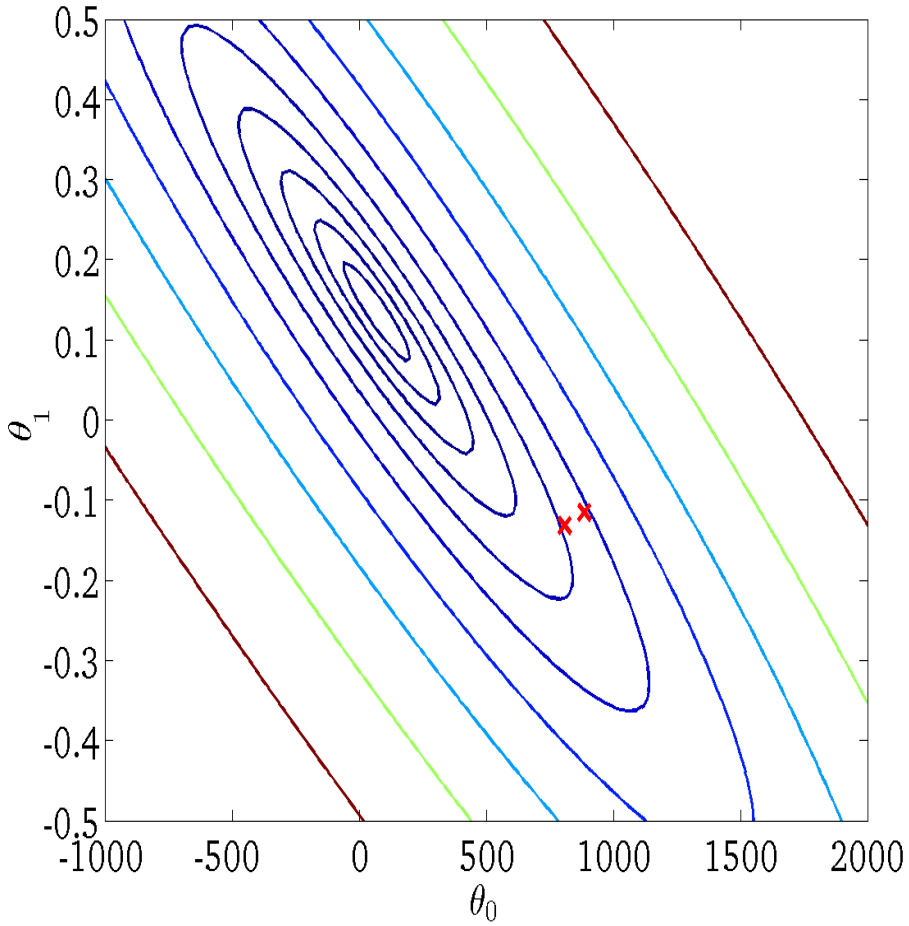
Andrew Ng

Gradient Descent for LR Price~Size – Iteration 1

$h_\theta(x)$ (for fixed θ_0, θ_1, this is a function of x)

$J(\theta_0, \theta_1)$ (function of the parameters θ_0, θ_1)

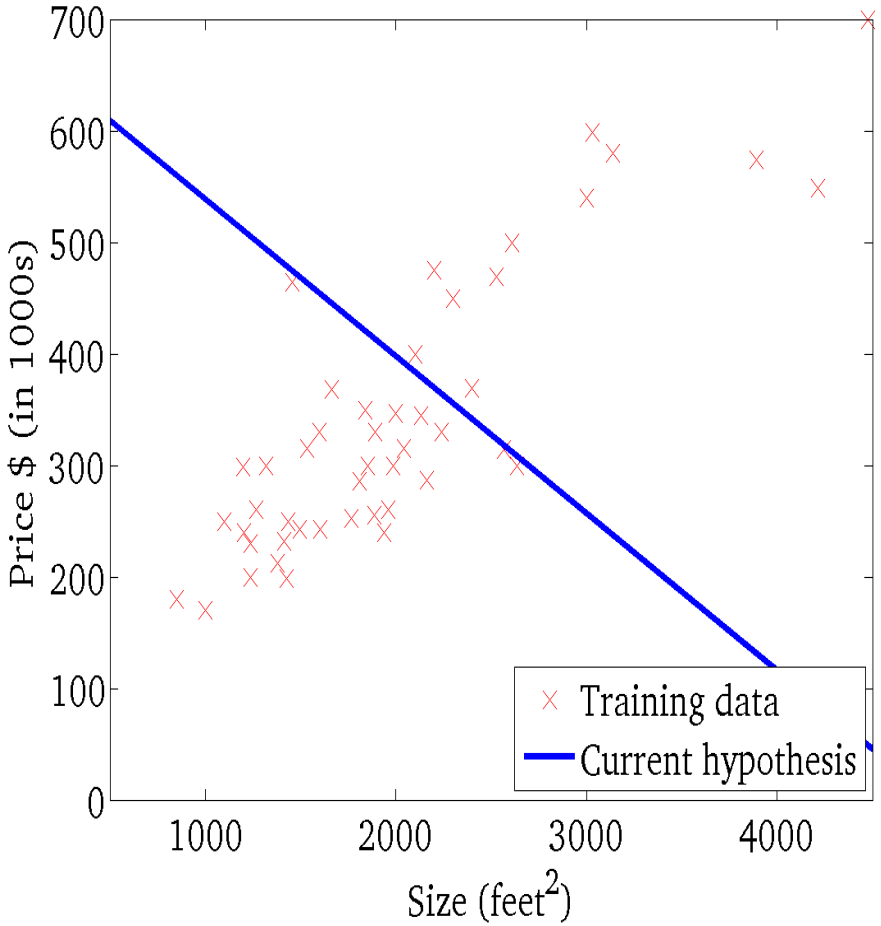


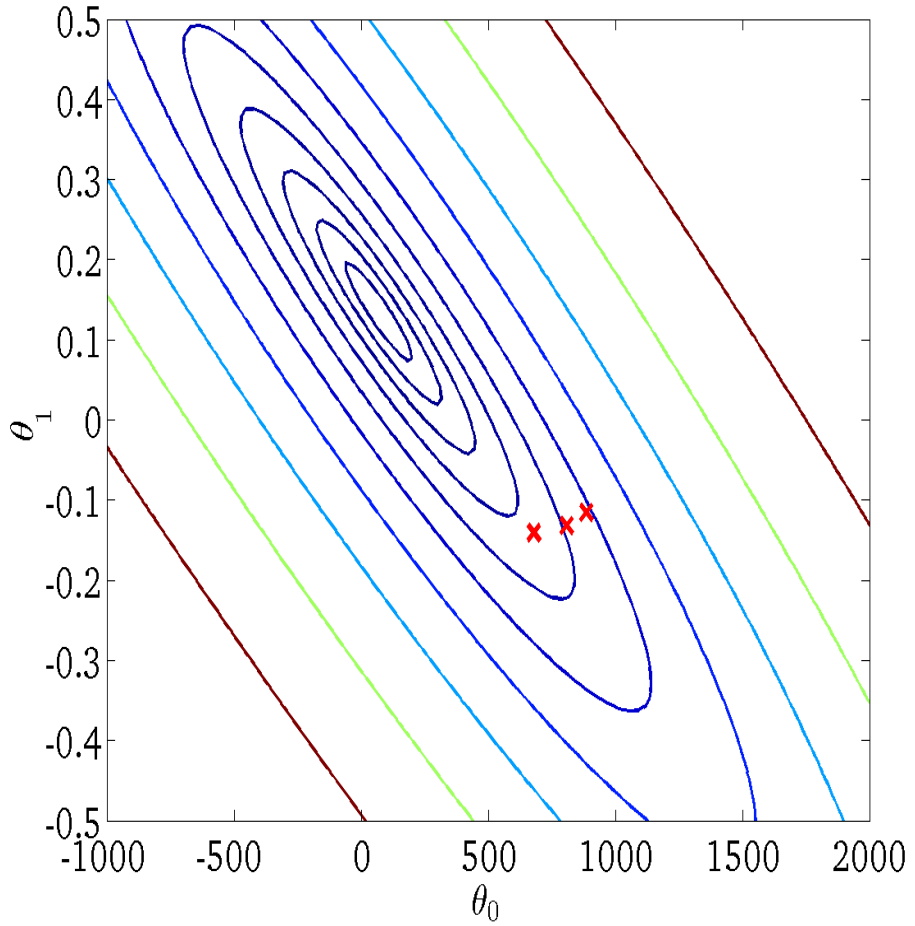


Gradient Descent for LR Price~Size – Iteration 2

$h_\theta(x)$ (for fixed θ_0, θ_1, this is a function of x)

$J(\theta_0, \theta_1)$ (function of the parameters θ_0, θ_1)

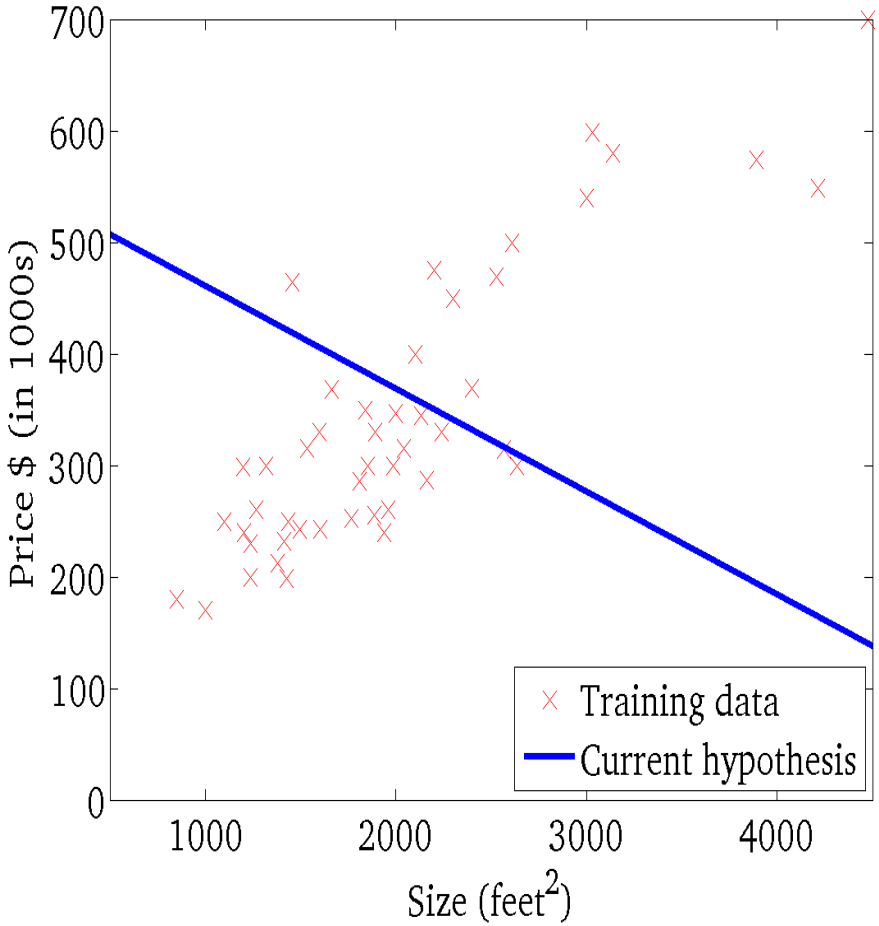


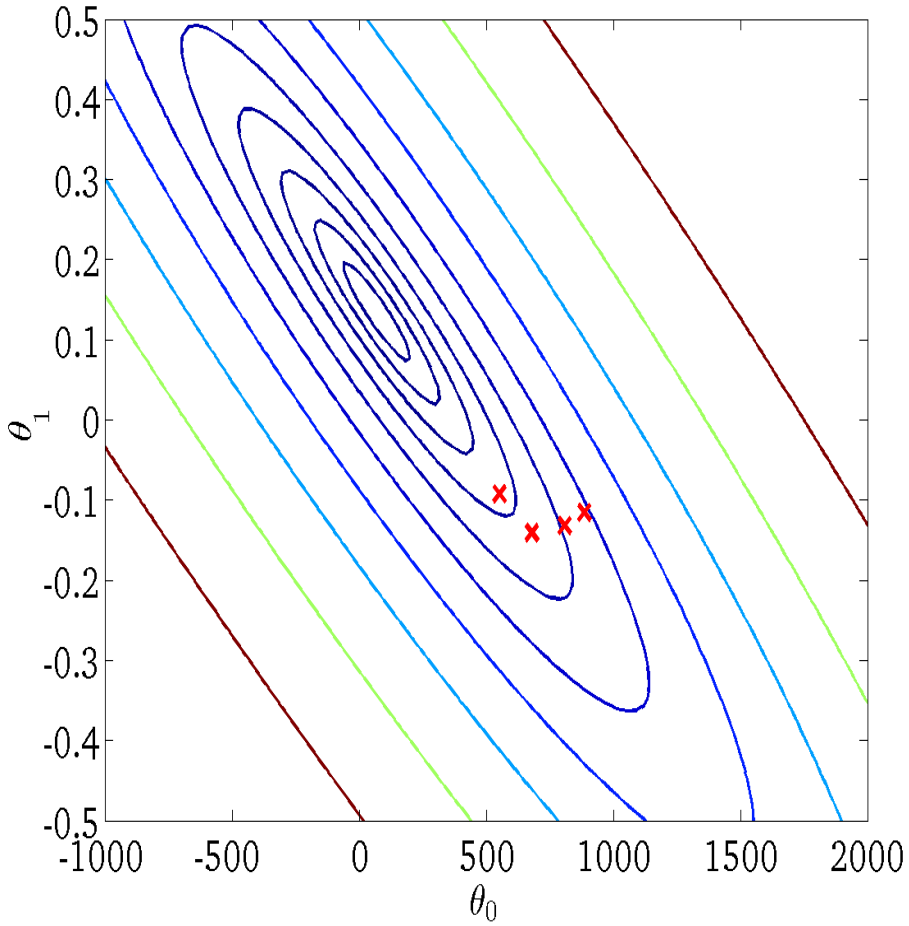


Gradient Descent for LR Price~Size – Iteration 3

$h_\theta(x)$ (for fixed θ_0, θ_1, this is a function of x)

$J(\theta_0, \theta_1)$ (function of the parameters θ_0, θ_1)

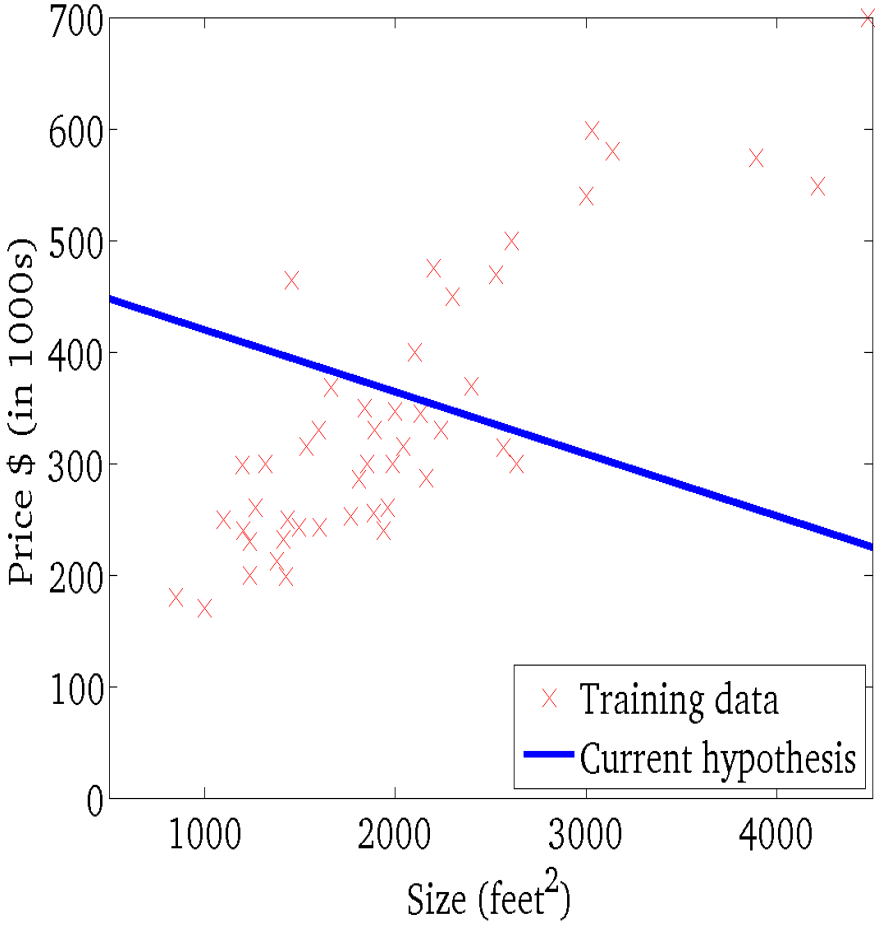


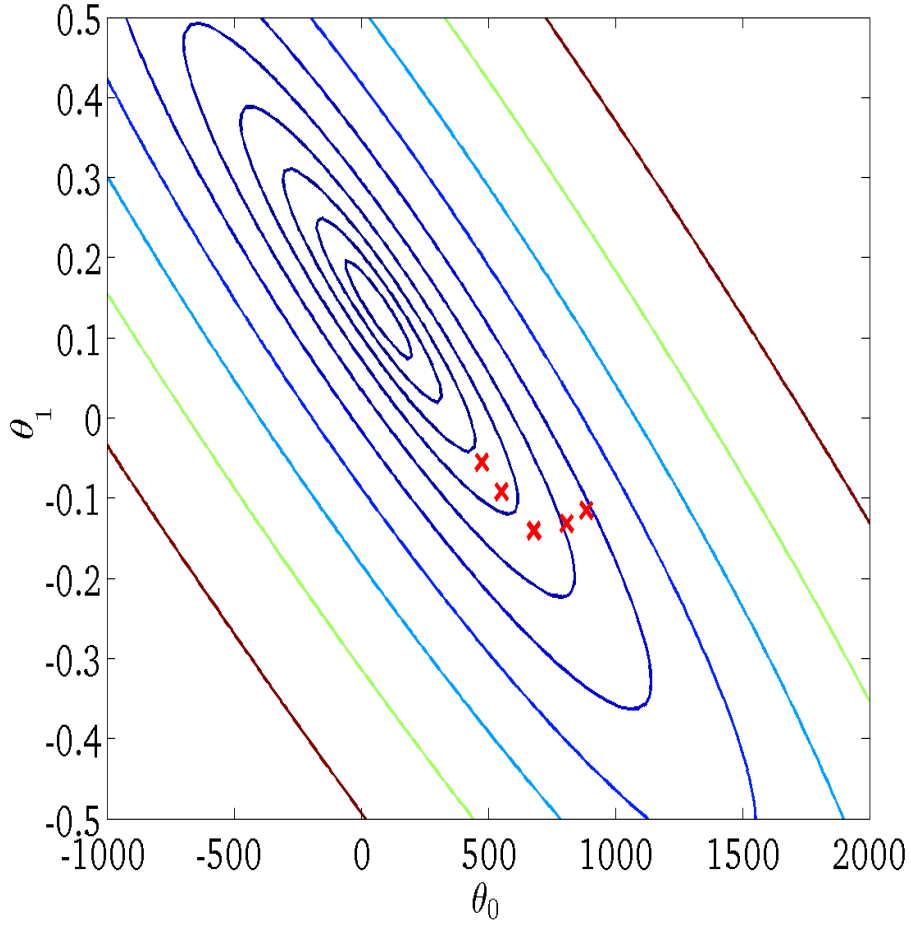


Gradient Descent for LR Price~Size – Iteration 4

$h_\theta(x)$ (for fixed θ_0, θ_1, this is a function of x)

$J(\theta_0, \theta_1)$ (function of the parameters θ_0, θ_1)

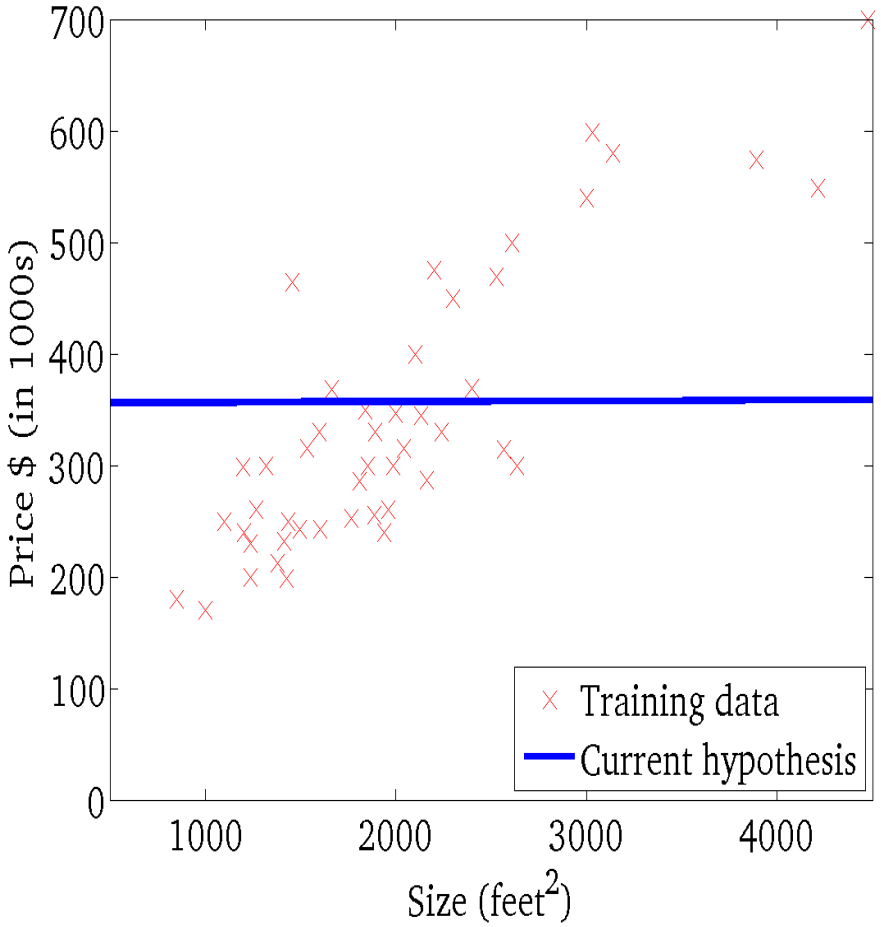


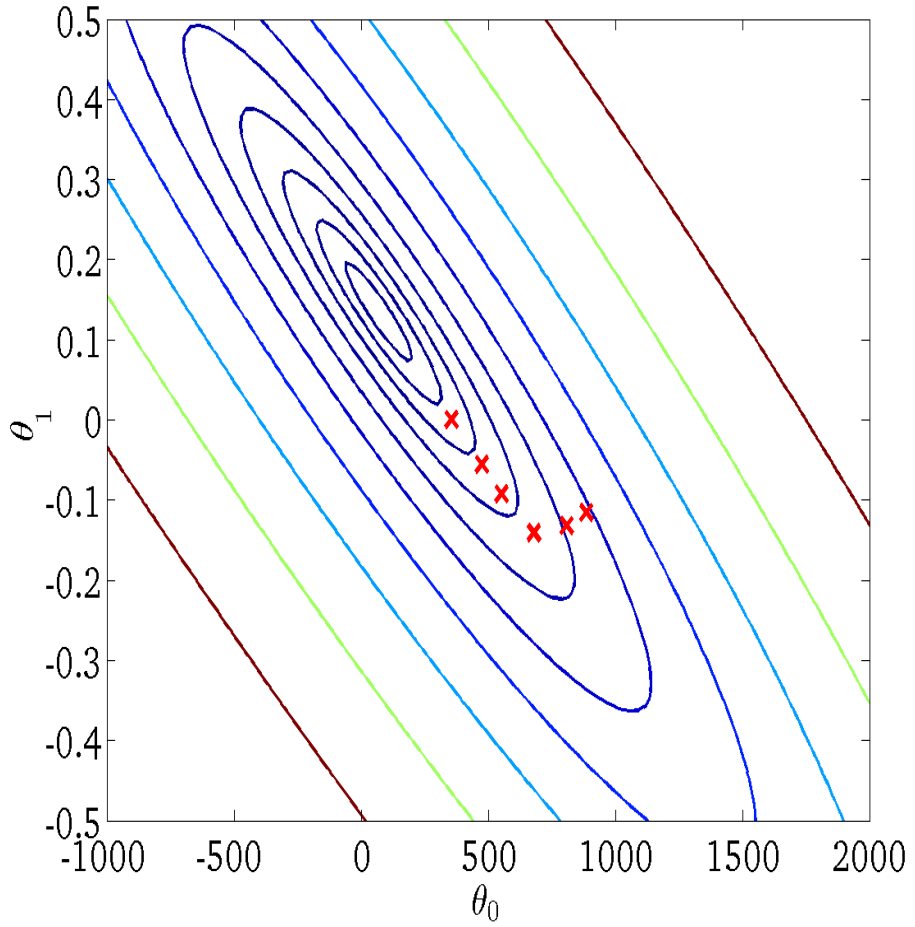


Gradient Descent for LR Price~Size – Iteration 5

$h_\theta(x)$ (for fixed θ_0, θ_1, this is a function of x)

$J(\theta_0, \theta_1)$ (function of the parameters θ_0, θ_1)

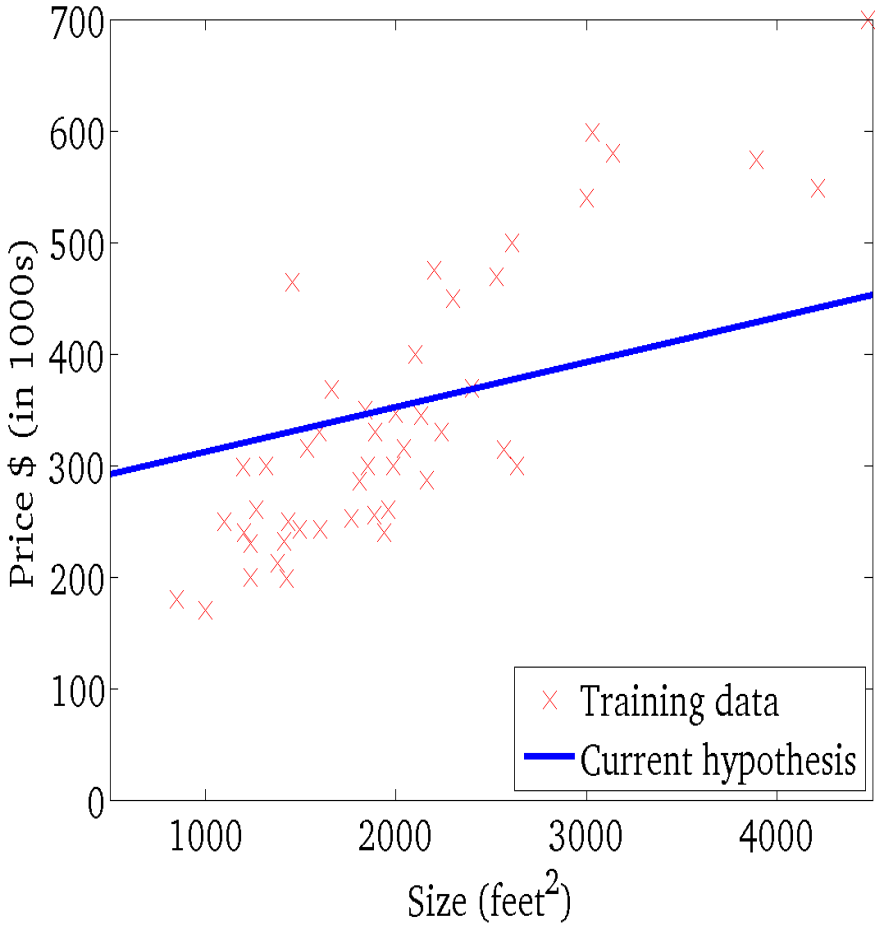


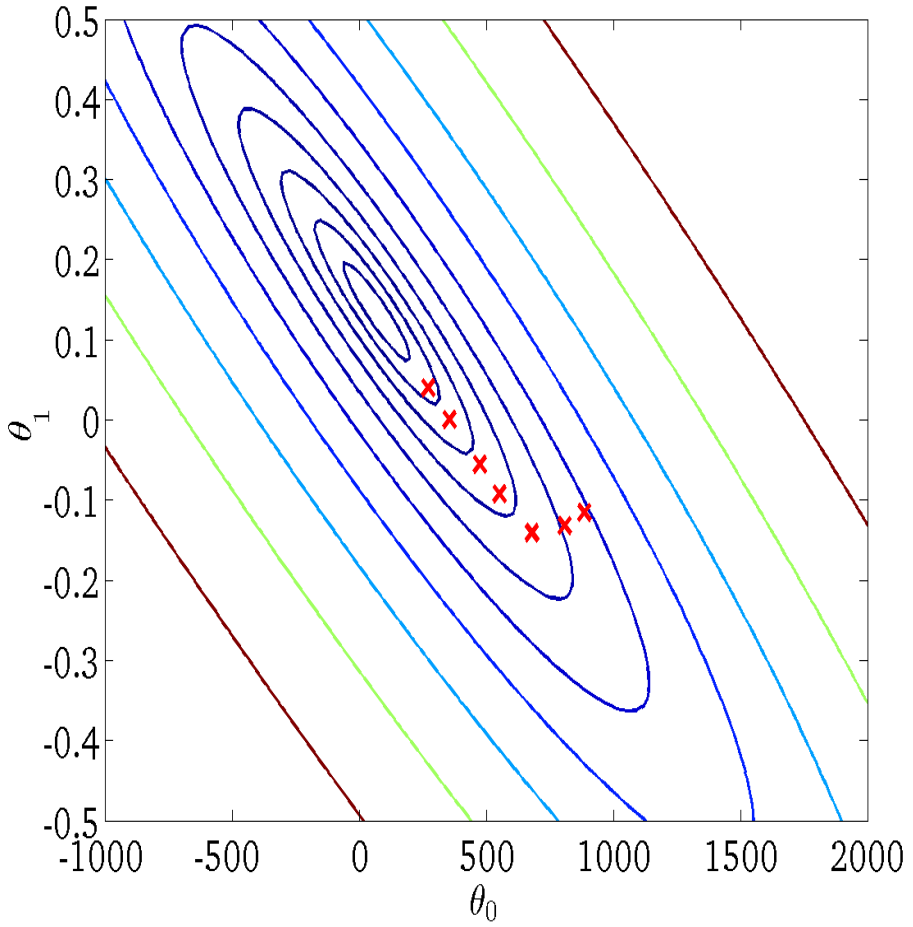


Gradient Descent for LR Price~Size – Iteration 6

$h_\theta(x)$ (for fixed θ_0, θ_1, this is a function of x)

$J(\theta_0, \theta_1)$ (function of the parameters θ_0, θ_1)

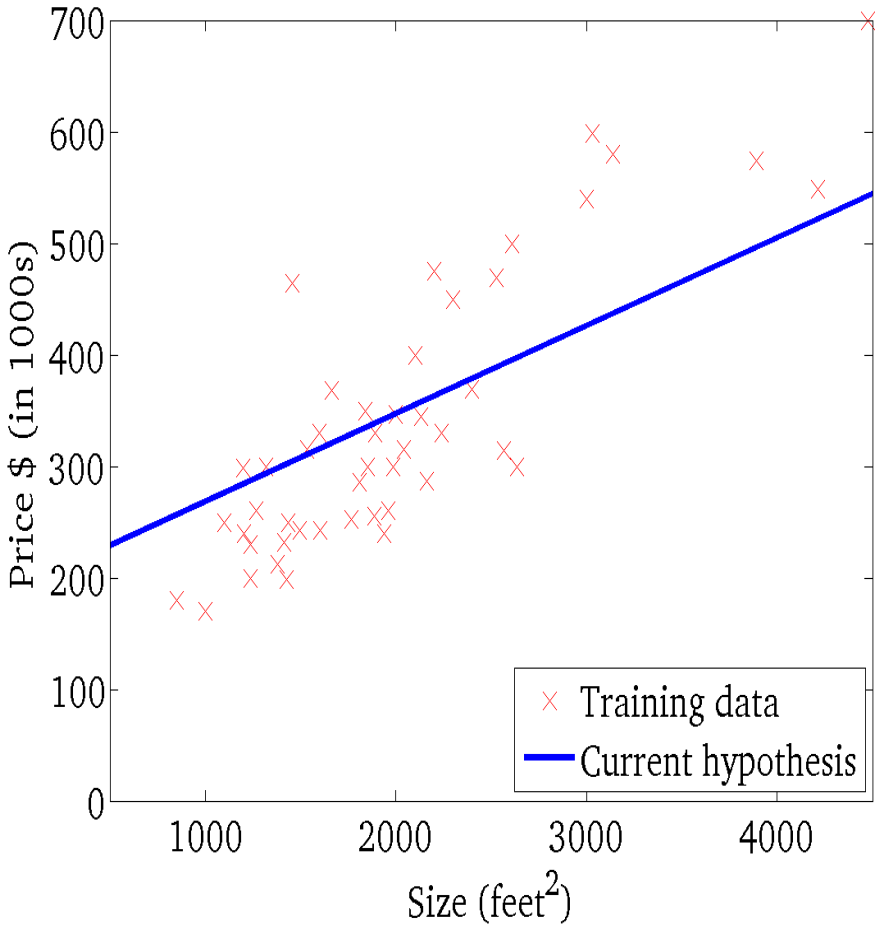


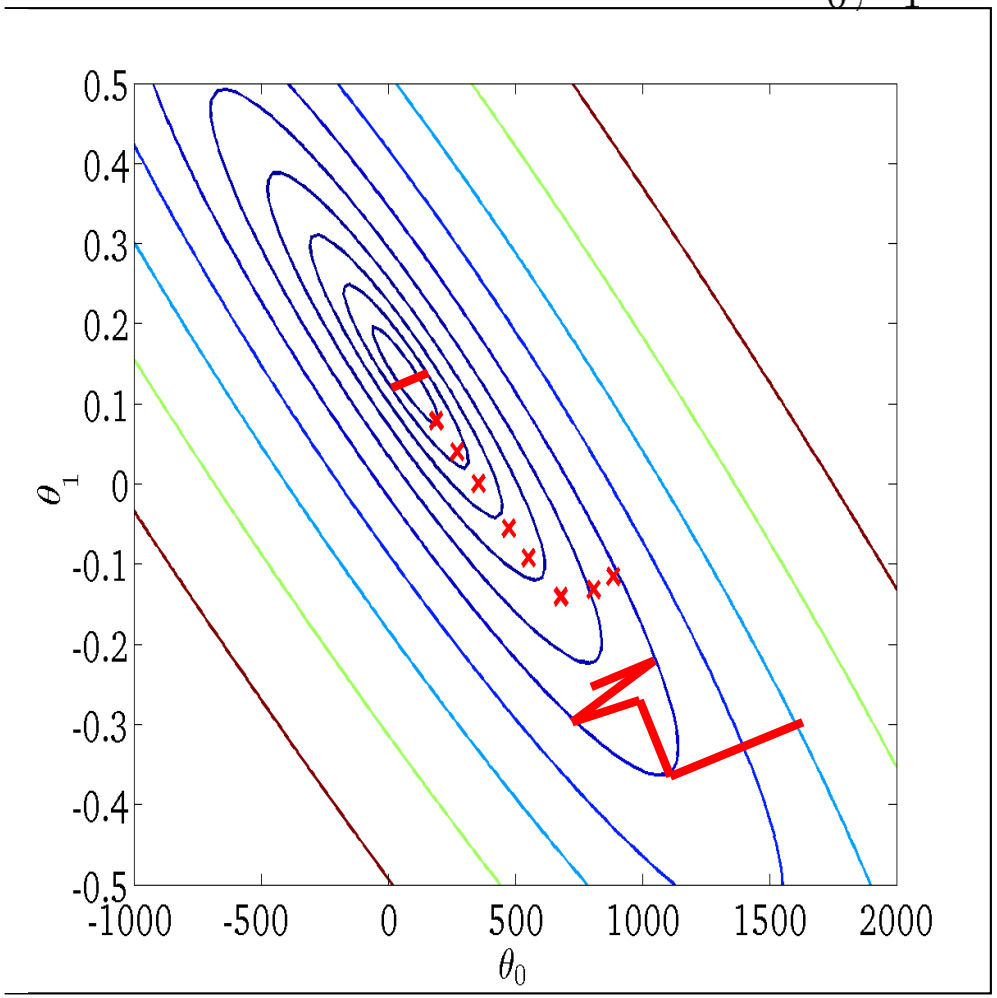


Gradient Descent for LR Price~Size – Iteration 7

$h_\theta(x)$ (for fixed θ_0, θ_1, this is a function of x)

$J(\theta_0, \theta_1)$ (function of the parameters θ_0, θ_1)

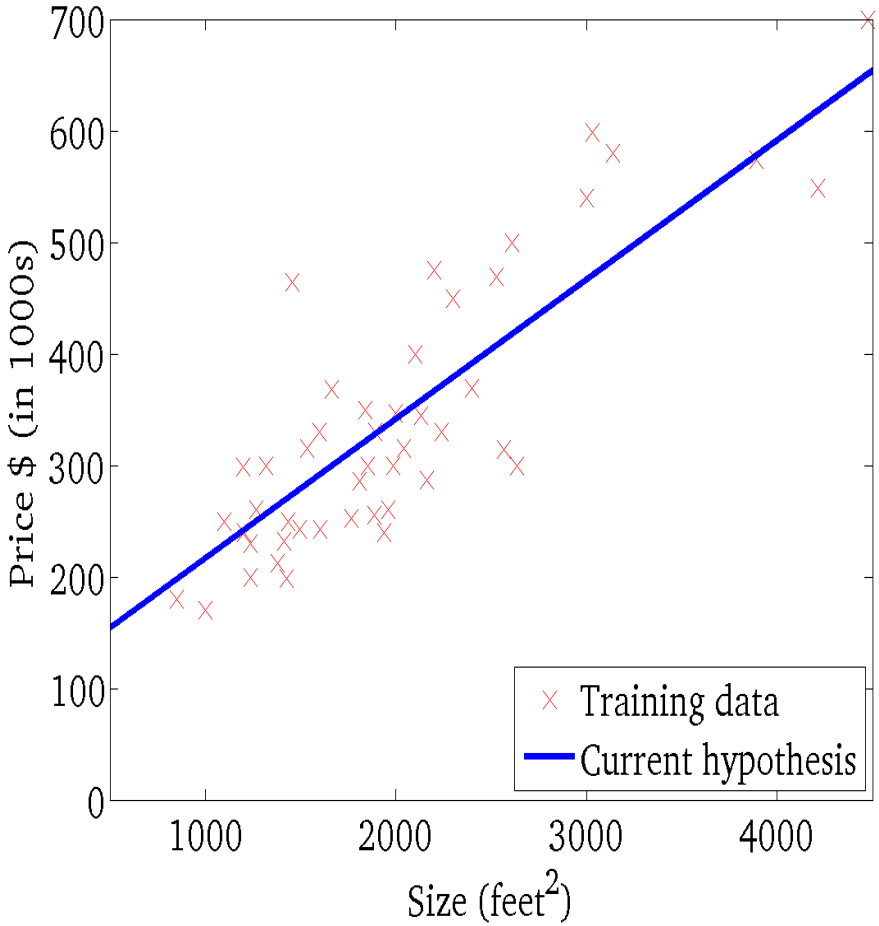


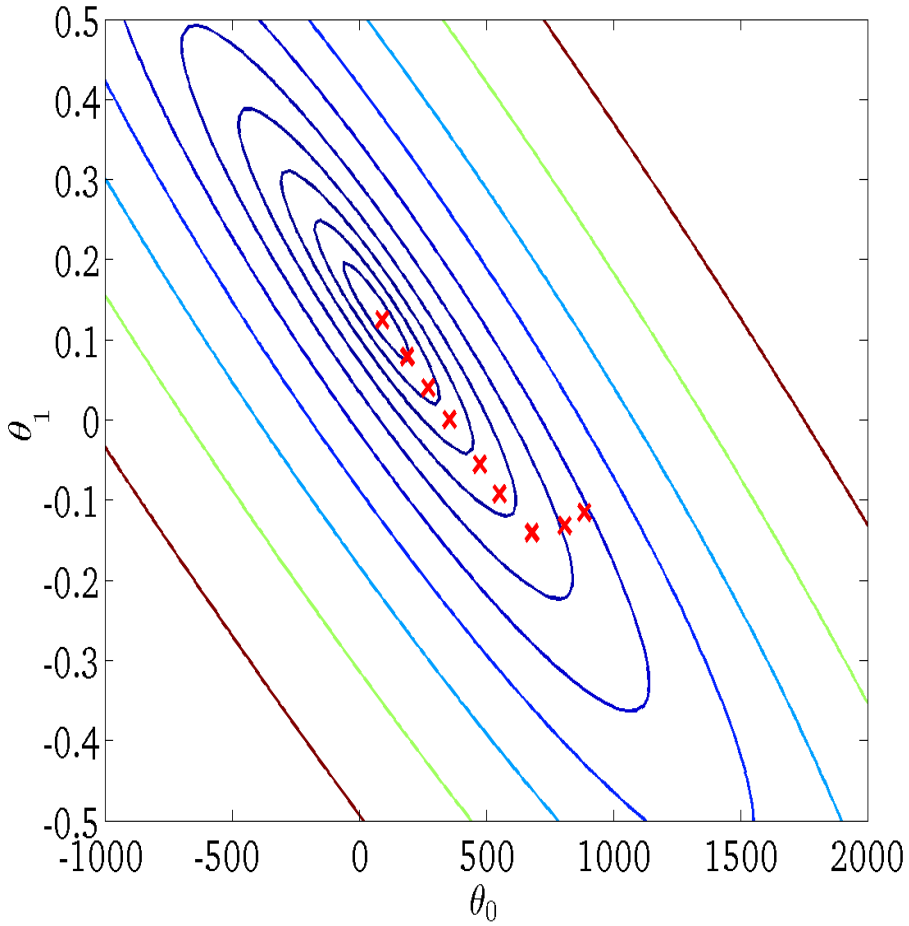


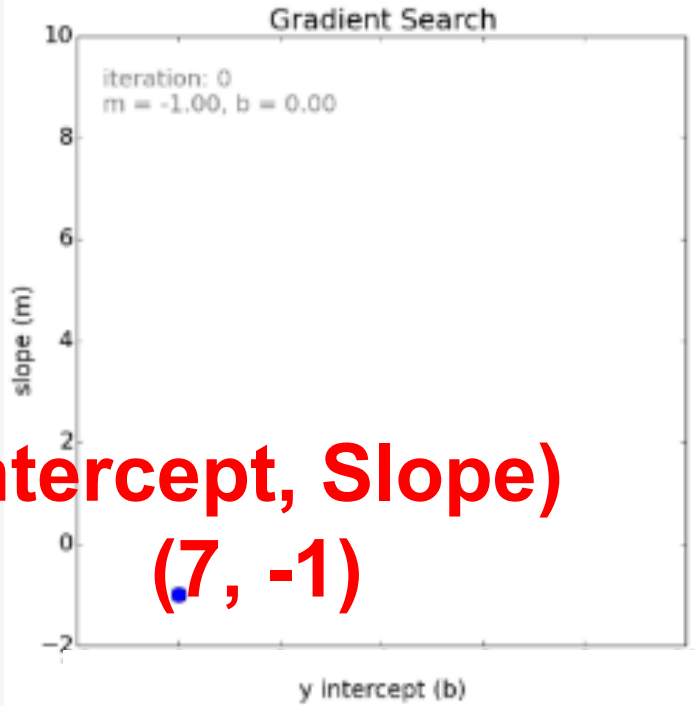
Gradient Descent for LR Price~Size – Converged after 9 steps

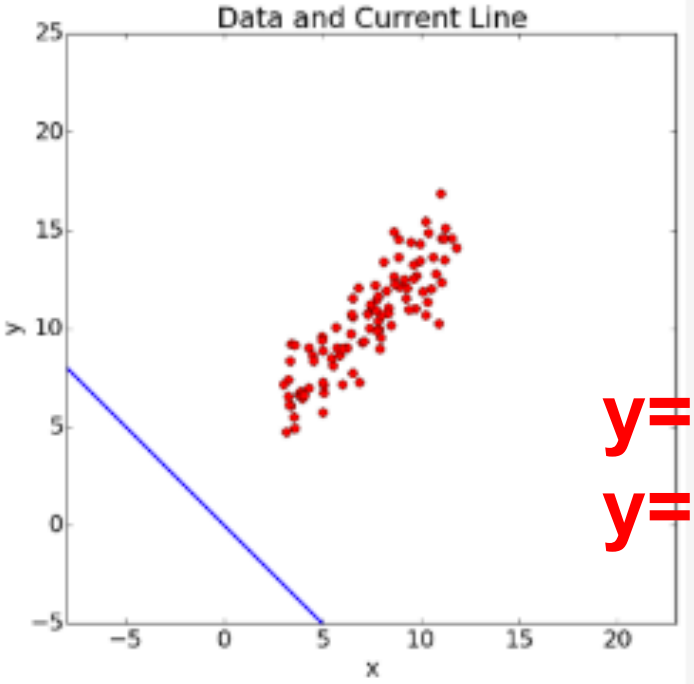
$h_\theta(x)$ (for fixed θ_0, θ_1, this is a function of x)

$J(\theta_0, \theta_1)$ (function of the parameters θ_0, θ_1)









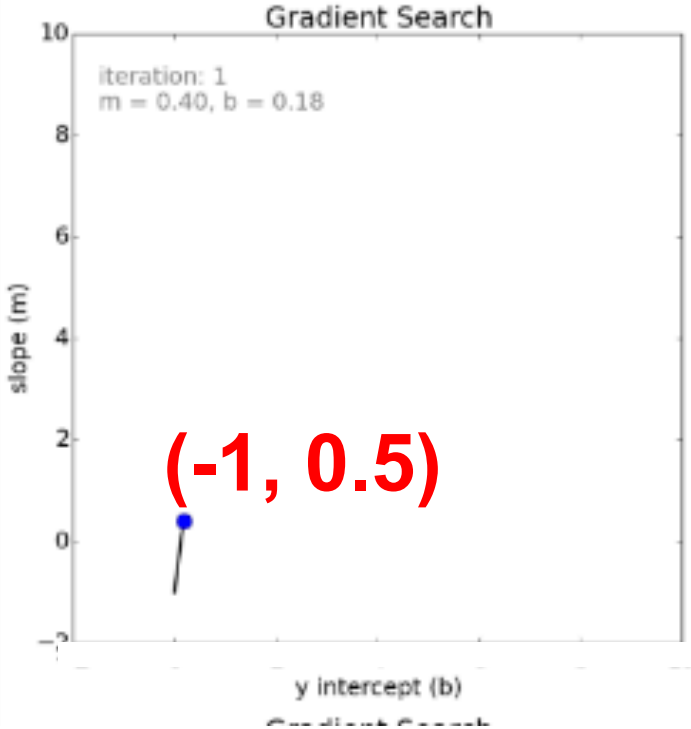
(Intercept, Slope)

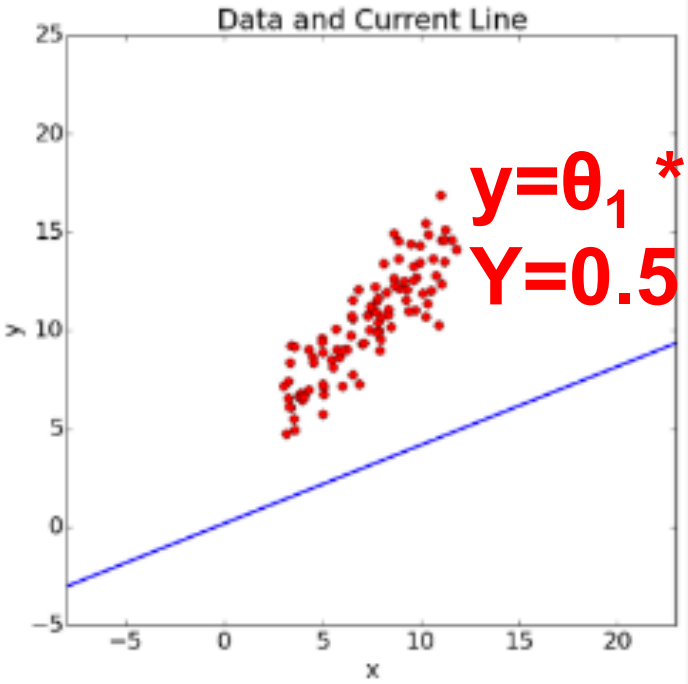
(7, -1)

$y=\theta_1 * x + \theta$

$y=-1 * x + 7$

Graphs not scale

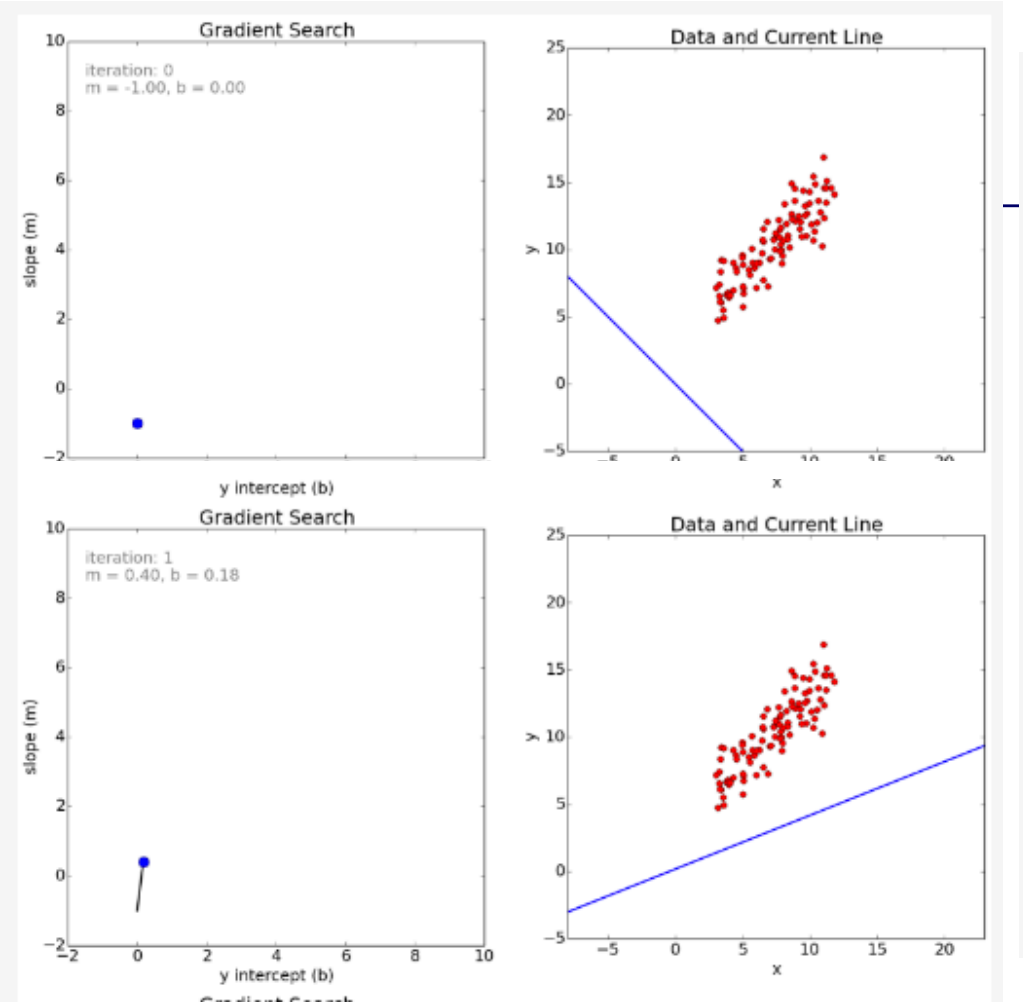


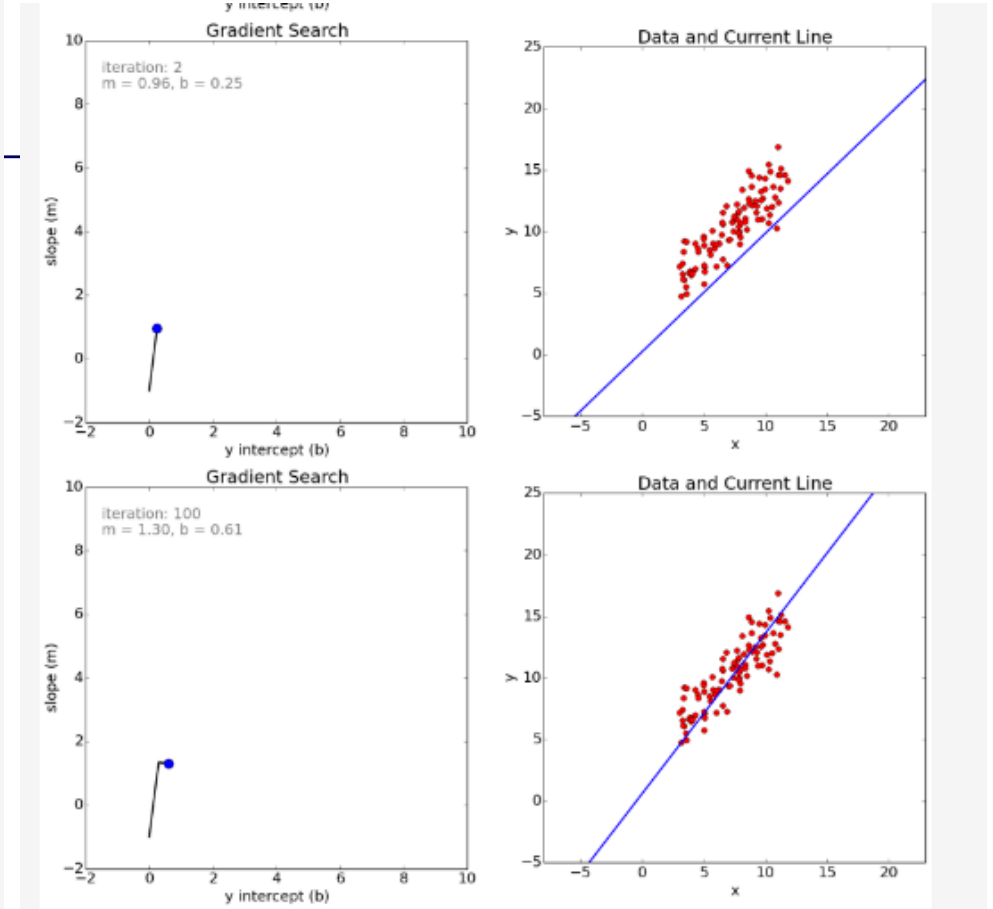


(-1, 0.5)

$y=\theta_1 * x + \theta_0$

$Y=0.5 * x - 1$


Gradient Search
iteration: 0
m = -1.00, b = 0.00
slope (m)
y intercept (b)
Data and Current Line
y
x
Gradient Search
iteration: 1
m = 0.40, b = 0.18
slope (m)
y intercept (b)
Data and Current Line
y
x


Gradient Search
iteration: 2
m = 0.96, b = 0.25
slope (m)
y intercept (b)
Data and Current Line
y
x
Gradient Search
iteration: 100
m = 1.30, b = 0.61
slope (m)
y intercept (b)
Data and Current Line
y
x

Lecture Outline

- **Introduction**
- **Linear regression from a statistical perspective**
  - Simple linear regression
  - Assessing the quality of LR model
  - Multiple linear regression (and challenges)
- **Linear regression from scratch**
  - LR via Gradient Descent via brute force
  - LR via Normal Equation (Closed-form analytical optimization)
  - LR via Gradient Descent (Numerical optimization)
  - LR via gradient descent in graph form
- **Summary**

Gradient Descent in vector form

$$\mathrm{MSE}(\mathbf{X}, h_\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(\theta^T \cdot \mathbf{x}^{(i)} - y^{(i)}\right)^2$$

MSE Loss Function (Objective)

$$\nabla_\theta \mathrm{MSE}(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_0}\mathrm{MSE}(\theta) \\ \frac{\partial}{\partial \theta_1}\mathrm{MSE}(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n}\mathrm{MSE}(\theta) \end{pmatrix} = \frac{2}{m}\mathbf{X}^T \cdot (\mathbf{X} \cdot \theta - \mathbf{y})$$

Gradient of partial derivatives

$$\theta^{(\text{next step})} = \theta - \eta \nabla_\theta \mathrm{MSE}(\theta)$$

GD Update rule

Implementation

```
eta = 0.1  # learning rate
n_iterations = 1000
m = 100

theta = np.random.randn(2,1)  # random initialization

for iteration in range(n_iterations):
    gradients = 2/m * X_b.T.dot(X_b.dot(theta) - y)
    theta = theta - eta * gradients
```

Linear Regression via GD

$$\frac{\partial E}{\partial W_{jk}} = \frac{\partial}{\partial W_{jk}} \frac{1}{2} \sum_{k \in K} (O_k - t_k)^2$$

All training data

$$\frac{\partial E}{\partial W_i} = \frac{\partial}{\partial W_i} \sum_{n=1:Train} \frac{1}{2} (X_{in} W^T - t_i)^2$$

One example

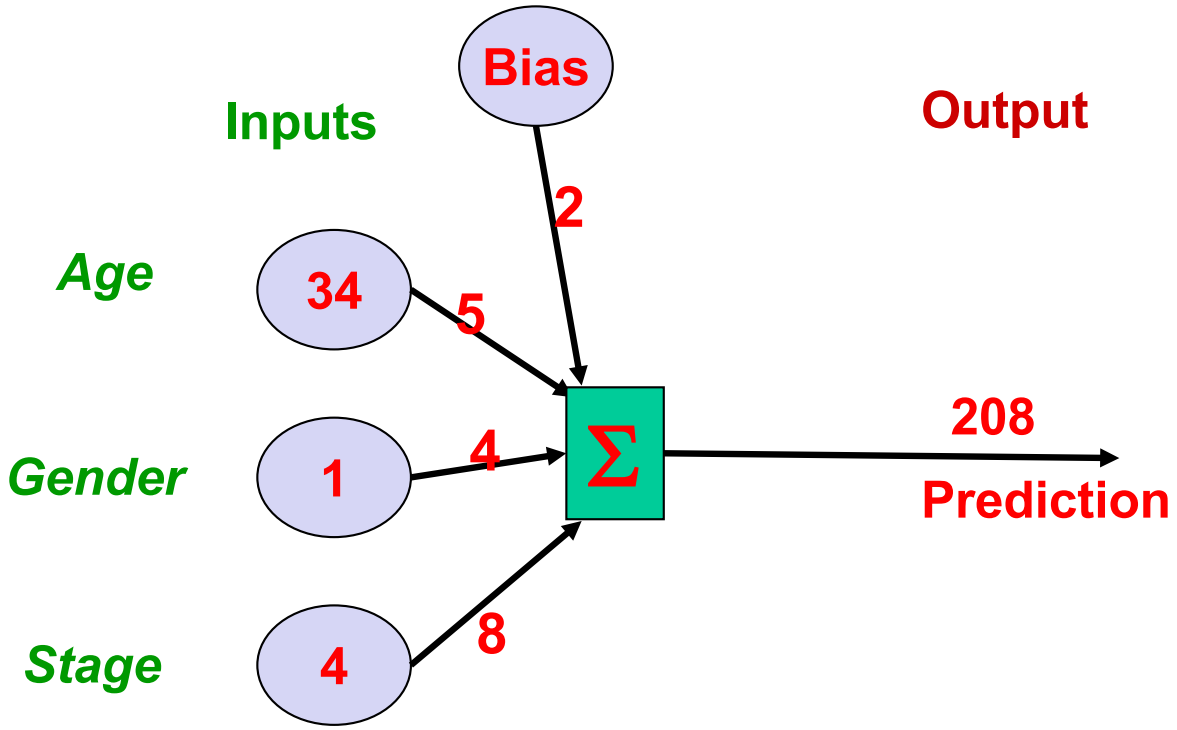
$$\frac{\partial E}{\partial W_i} = \frac{\partial}{\partial W_i} \frac{1}{2} (O_i - t_i)^2$$

$$\frac{\partial E}{\partial W_i} = (O_i - t_i) \frac{\partial}{\partial W_i} (O_i - t_i)$$

$$\frac{\partial E}{\partial W_i} = (O_i - t_i) \frac{\partial}{\partial W_i} (X_i W^T - t_i)$$

$$\frac{\partial E}{\partial W} = (O - t) X$$

Linear Regression Model



$$Y = X\beta + \varepsilon$$

Independent variables

x1, x2, x3

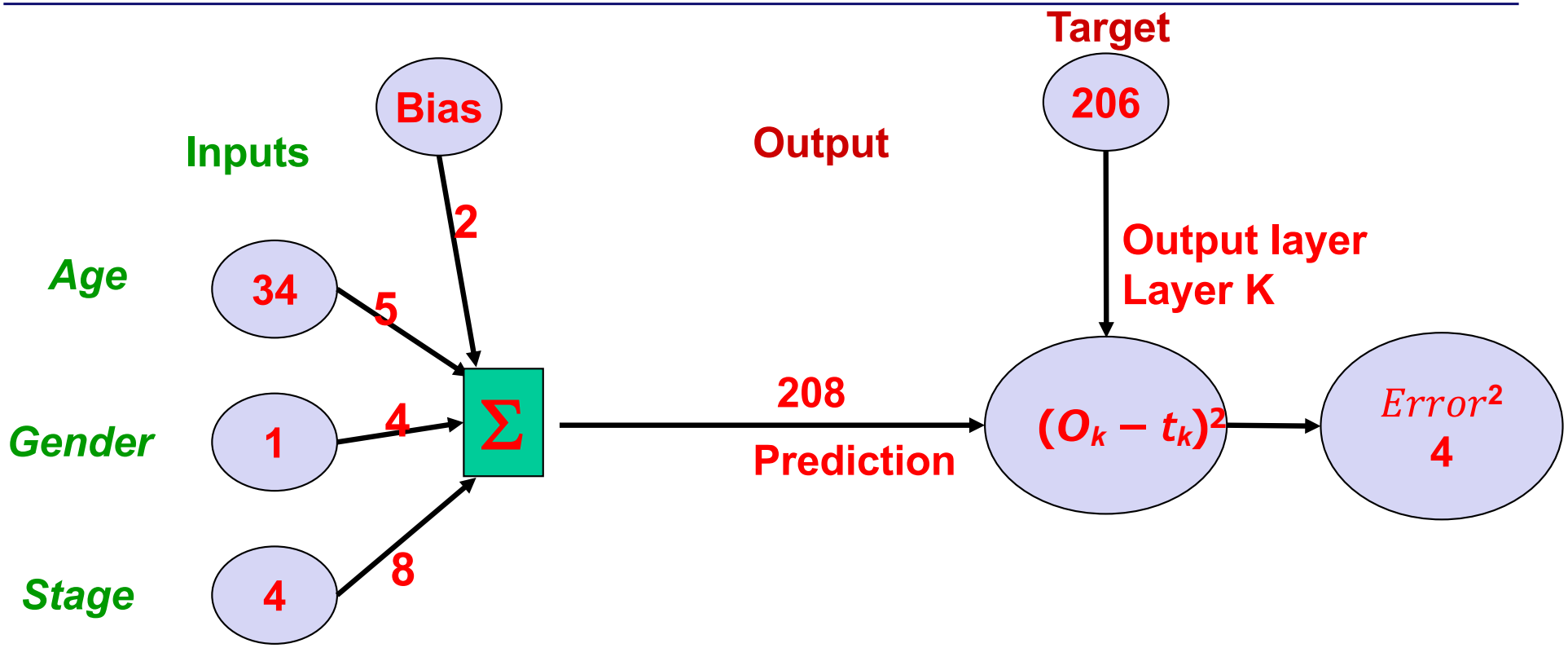
Coefficients

a, b, c; bias

Dependent variable

Prediction

Linear Regression Model: MSE



Independent variables

x1, x2, x3

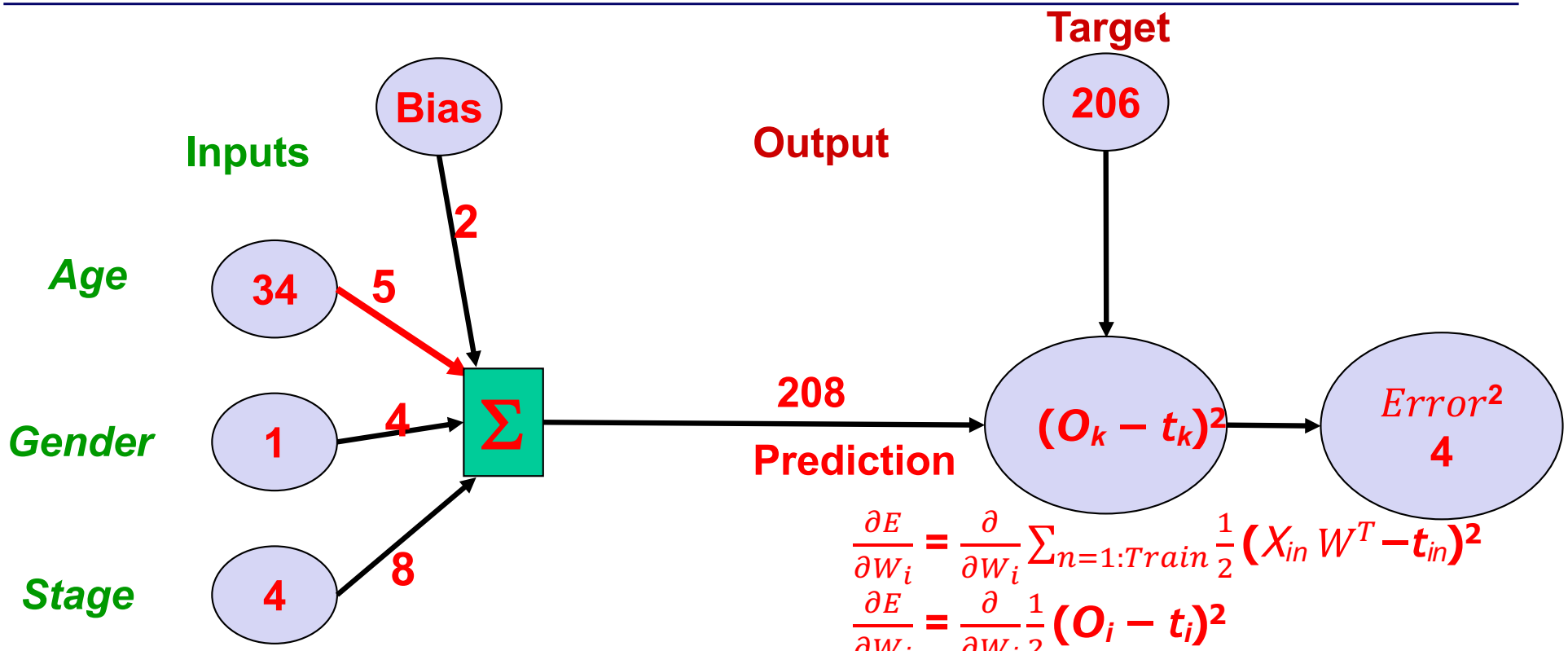
Coefficients

a, b, c; bias

Dependent variable

Prediction

Linear Regression Model: MSE



$$\frac{\partial E}{\partial W_i} = \frac{\partial}{\partial W_i} \sum_{n=1:Train} \frac{1}{2} (X_{in} W^T - t_{in})^2$$

$$\frac{\partial E}{\partial W_i} = \frac{\partial}{\partial W_i} \frac{1}{2} (O_i - t_i)^2$$

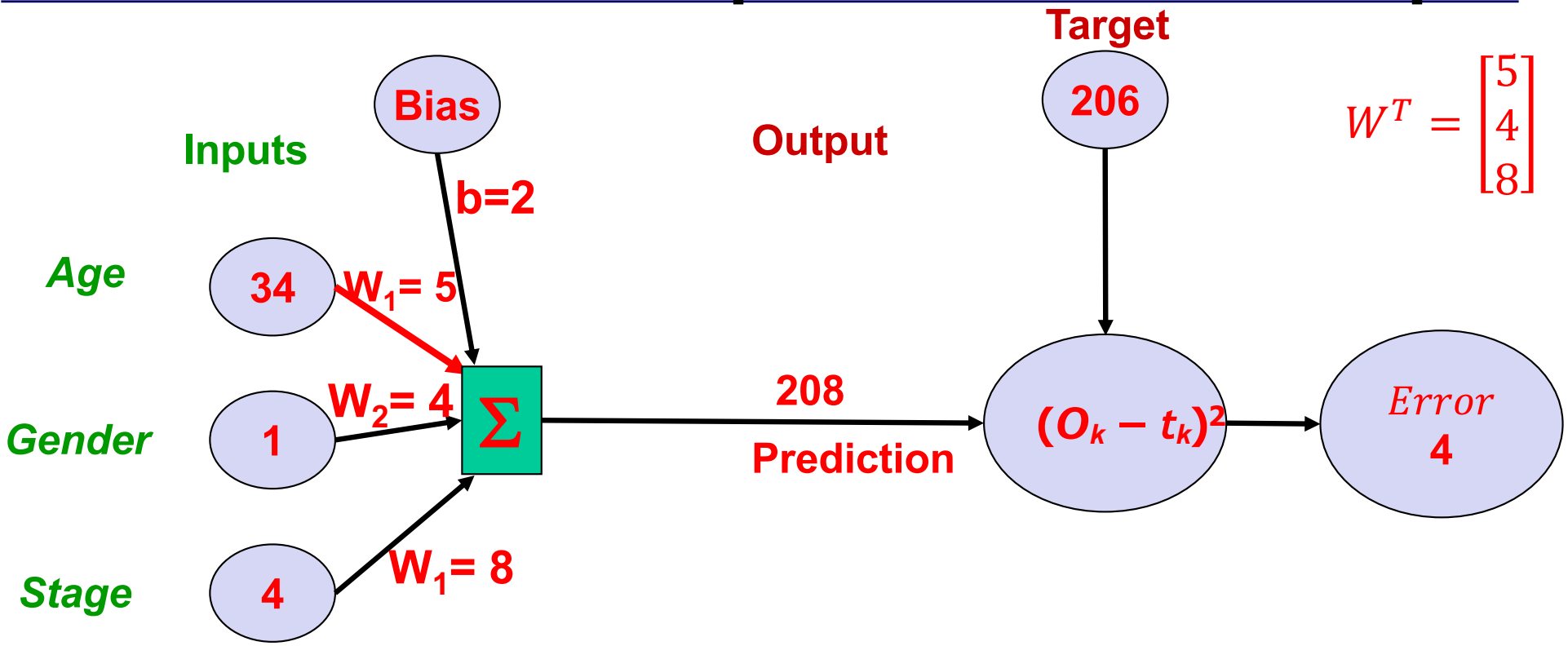
$$\frac{\partial E}{\partial W_i} = (O_i - t_i) \frac{\partial}{\partial W_i} (O_i - t_i)$$

$$\frac{\partial E}{\partial W_i} = (O_i - t_i) \frac{\partial}{\partial W_i} (X_i W^T - t_i)$$

$$\frac{\partial E}{\partial W_i} = (O_i - t_i) X_i$$

$$W = W - \alpha \frac{\partial E}{\partial W}$$

Linear Regression Model: Gradient update for one example

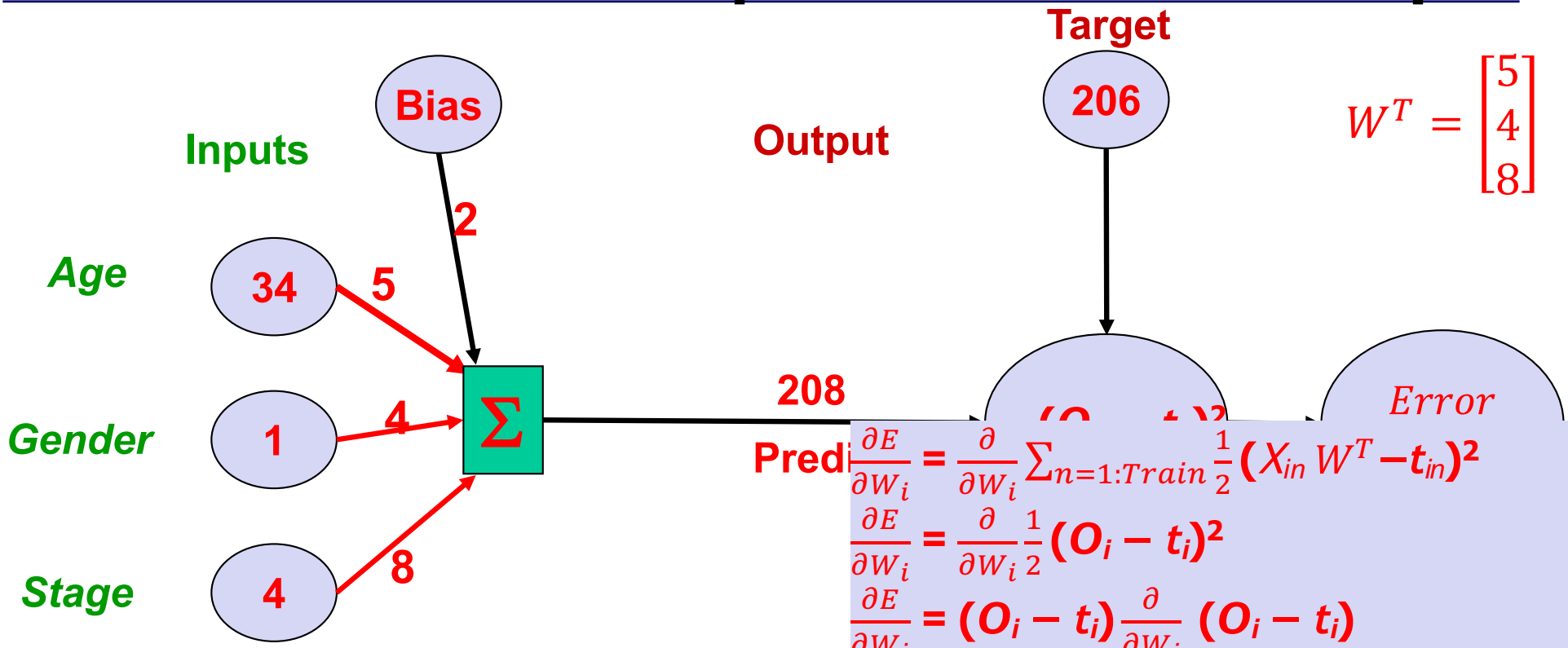


Quiz Prep example

Weight Gradient for this example[34,1,4]

Gradient for b

Linear Regression Model: Gradient update for one example



$$W^T = \begin{bmatrix} 5 \\ 4 \\ 8 \end{bmatrix}$$

$$\frac{\partial E}{\partial W_i} = \frac{\partial}{\partial W_i} \sum_{n=1:Train} \frac{1}{2} (X_{in} W^T - t_{in})^2$$

$$\frac{\partial E}{\partial W_i} = \frac{\partial}{\partial W_i} \frac{1}{2} (O_i - t_i)^2$$

$$\frac{\partial E}{\partial W_i} = (O_i - t_i) \frac{\partial}{\partial W_i} (O_i - t_i)$$

$$\frac{\partial E}{\partial W_i} = (O_i - t_i) \frac{\partial}{\partial W_i} (X_i W^T - t_i)$$

$$\frac{\partial E}{\partial W} = (O - t) X$$

$$W = W - \alpha \frac{\partial E}{\partial W}$$

$$\begin{bmatrix} 4.32 \\ 3.98 \\ 7.92 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 8 \end{bmatrix} - 0.01 \times 2 \times \begin{bmatrix} 34 \\ 1 \\ 4 \end{bmatrix}$$

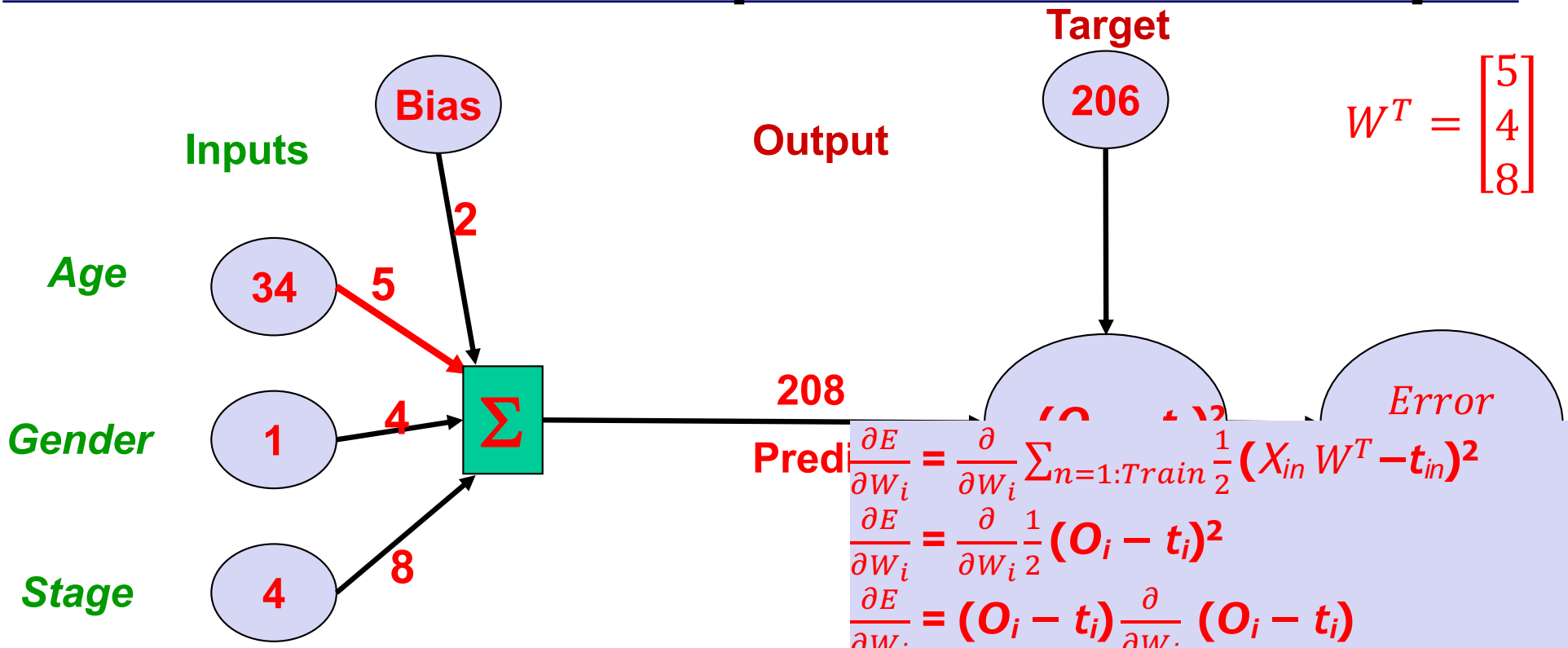
$$4.32 = 5 - 0.01 \times (2 \times 34) \text{ for } i = 1$$

Weight Gradient for this example[34,1,4]

Gradient for b = 2

Assume a learning rate α=0.01

Linear Regression Model: Gradient update for one example



$$W^T = \begin{bmatrix} 5 \\ 4 \\ 8 \end{bmatrix}$$

$$\frac{\partial E}{\partial W_i} = \frac{\partial}{\partial W_i} \sum_{n=1:Train} \frac{1}{2} (X_{in} W^T - t_{in})^2$$

$$\frac{\partial E}{\partial W_i} = \frac{\partial}{\partial W_i} \frac{1}{2} (O_i - t_i)^2$$

$$\frac{\partial E}{\partial W_i} = (O_i - t_i) \frac{\partial}{\partial W_i} (O_i - t_i)$$

$$\frac{\partial E}{\partial W_i} = (O_i - t_i) \frac{\partial}{\partial W_i} (X_i W^T - t_i)$$

$$\frac{\partial E}{\partial W} = (O - t) X$$

$$W = W - \alpha \frac{\partial E}{\partial W}$$

$$\begin{bmatrix} 4.32 \\ 3.98 \\ 7.92 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 8 \end{bmatrix} - 0.01 \times 2 \times \begin{bmatrix} 34 \\ 1 \\ 4 \end{bmatrix}$$

$$4.32 = 5 - 0.01 \times (2 \times 34) \text{ for } i = 1$$

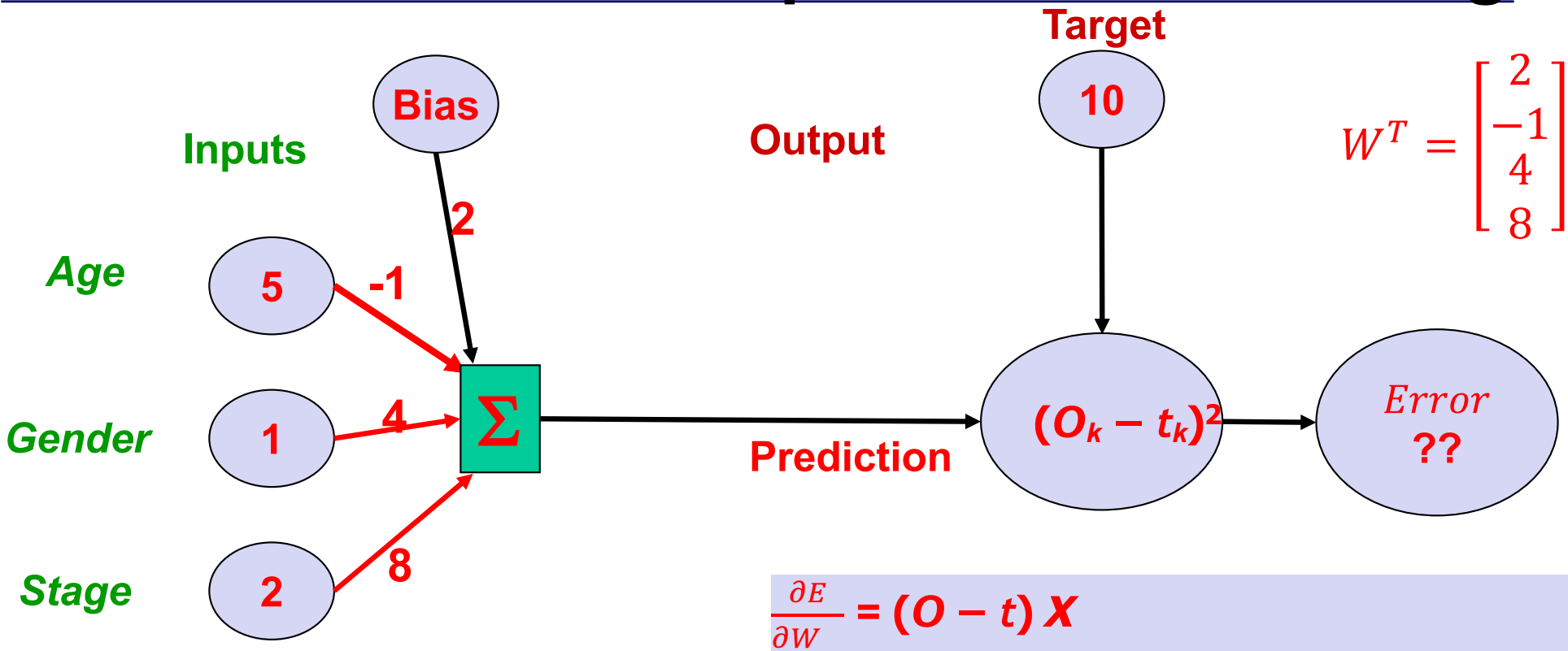
NOTE: the gradient is a weighted sum of all examples; here the weight is 2 for this example

Gradient for this example[34,1,4]

Gradient for b = 2

Assume a learning rate α=0.01

Quiz : calculate gradient descent update for this setting



$$W^T = \begin{bmatrix} 2 \\ -1 \\ 4 \\ 8 \end{bmatrix}$$

Gradient for this example[5, 1,2]

Gradient for bias?

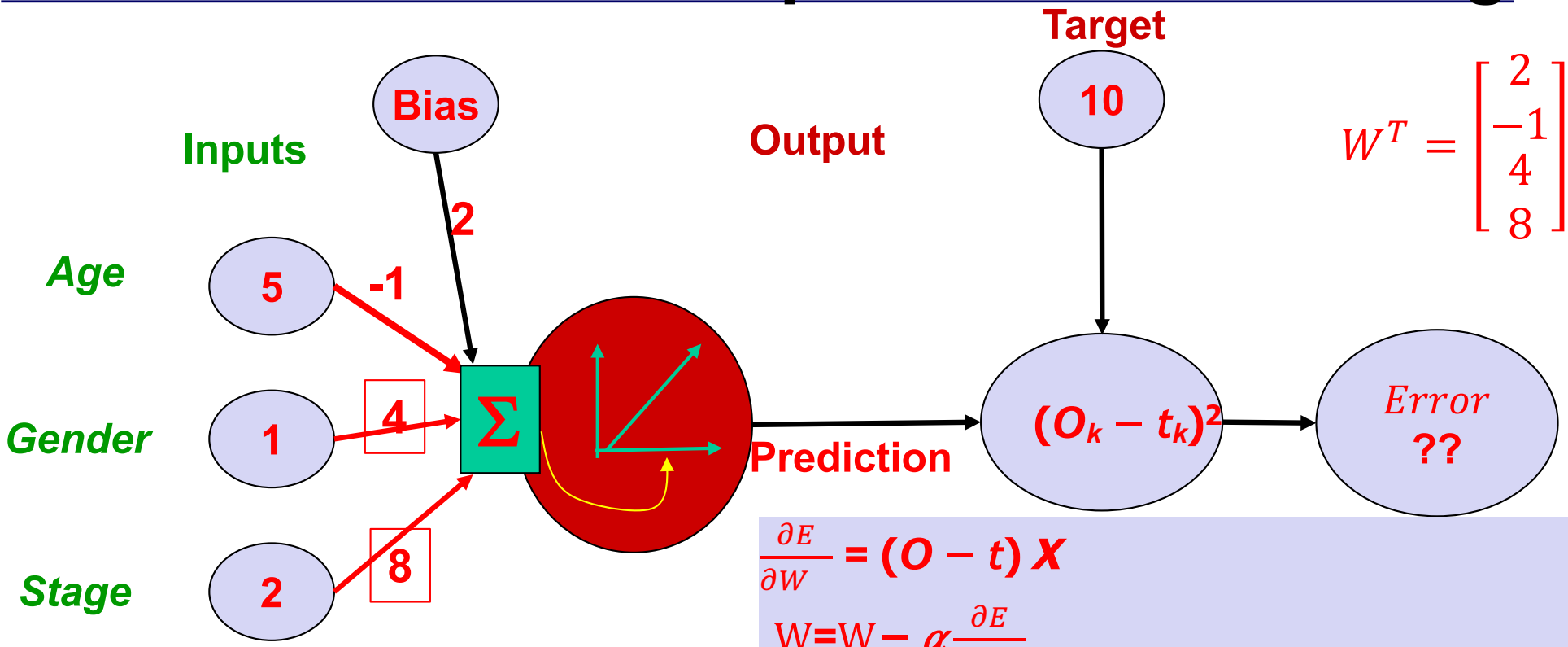
$$\frac{\partial E}{\partial W} = (O - t)\,X$$

$$W = W - \alpha \frac{\partial E}{\partial W}$$

$$\begin{bmatrix} ? \\ ? \\ ? \\ ? \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ 4 \\ 8 \end{bmatrix} - 0.01 \times ? \times \begin{bmatrix} 1 \\ 5 \\ 1 \\ 2 \end{bmatrix}$$

T1: https://canvas.instructure.com/courses/1159217/quizzes/2102928

Quiz: calculate gradient descent update for this setting



$$W^T = \begin{bmatrix} 2 \\ -1 \\ 4 \\ 8 \end{bmatrix}$$

Gradient for this example[5, 1,2]

Gradient for bias?

$$\frac{\partial E}{\partial w} = (O - t)\,X$$

$$W = W - \alpha \frac{\partial E}{\partial w}$$

$$\begin{bmatrix} ? \\ ? \\ ? \\ ? \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ 4 \\ 8 \end{bmatrix} - 0.01 \times ? \times \begin{bmatrix} 1 \\ 5 \\ 1 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 1.93 \\ -1.35 \\ 3.93 \\ 7.86 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ 4 \\ 8 \end{bmatrix} - 0.01 \times 7 \times \begin{bmatrix} 1 \\ 5 \\ 1 \\ 2 \end{bmatrix}$$

▽ the gradient chorus/mantra

- **What is the gradient for linear regression?**

- **Chorus**
  - The gradient is the weighted sum of the training data, where the weights are proportional to the error (for each example) !

weight *example*

$$\frac{\partial E}{\partial W} = (O - t)\,x$$

$$W = W - \alpha \times \frac{\partial E}{\partial W}$$

$$\begin{bmatrix} 4.32 \\ 3.98 \\ 7.92 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 8 \end{bmatrix} - 0.01 \times 2 \times \begin{bmatrix} 34 \\ 1 \\ 4 \end{bmatrix}$$

$$4.32 = 5 - 0.01 \times (2 \times 34) \text{ for } i = 1$$

▽ the gradient chorus/mantra

- **What is the gradient for linear regression?**

- **Chorus**
  - The gradient is the weighted sum of the training data, where the weights are proportional to the error (for each example) !

Linear Regression

  - The gradient is the weighted sum of the **input** data, where the weights are proportional to the error **or the error proxy** (for each example) !

NN

$$\frac{\partial E}{\partial W} = \text{weight} \cdot \text{example} \cdot (O - t)$$

$$W = W - \alpha \times \frac{\partial E}{\partial W}$$

$$\begin{bmatrix} 4.32 \\ 3.98 \\ 7.92 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 8 \end{bmatrix} - 0.01 \times 2 \times \begin{bmatrix} \mathbf{34} \\ \mathbf{1} \\ \mathbf{4} \end{bmatrix}$$

$$\mathbf{4.32} = 5 - 0.01 \times (2 \times 34) \text{ for } i = 1$$

Gradient Descent: in vectors and matrices

$$\frac{\partial E}{\partial W} = \frac{\partial}{\partial W}\sum_{i=1:Train}\frac{1}{2}(X_i W^T - t_i)^2 \ /Train$$

$$\frac{\partial E}{\partial W} = \left(\sum_{i=1:Train} X^T(X_i W^T - t_i)\right)/Train$$

$$\frac{\partial E}{\partial W} = X^T(XW^T - t)/Train$$

#weighted sum
Grad = (X^T x Error) /|Train|

$$W = W - \alpha\frac{\partial E}{\partial W}$$

W =. W. $-\alpha$.

$$\begin{bmatrix} \\ \end{bmatrix} = \begin{bmatrix} \\ \end{bmatrix} - 0.01 \times \begin{bmatrix} \\ \end{bmatrix} \begin{bmatrix} \frac{\partial E}{\partial W} \\ \dots \end{bmatrix} \times \begin{bmatrix} \\ .. \\ \\ \end{bmatrix}$$

```
107  # gradient descent
108  for (i in 1:num_iters) {
109    error <- (X %*% theta - y)
110    delta <- t(X) %*% error / length(y)
111    browser()
112    theta <- theta - alpha * delta
113    cost_history[i] <- cost(X, y, theta)
114    theta_history[[i]] <- theta
115  }
116
```

X^T

```
Browse[1]> t(X)[,1:15]
          [,1]      [,2]     [,3]      [,4]      [,5]      [,6]     [,7]    [,8]     [,9]     [,10]    [,11]    [,12]     [,13]      [,14]     [,15]
[1,]  1.000000  1.0000000  1.00000  1.000000 1.0000000  1.000000 1.000000 1.00000 1.000000  1.000000 1.000000 1.000000  1.000000  1.0000000  1.000000
[2,] -2.693694 -0.8262961 -3.99729 -2.214886 0.3054649 -1.858805 1.007021 4.87098 1.092815 -3.542627 2.043165 1.791803 -1.774161 -0.7668436 -1.955358
```

```
Browse[1]> error[1:15,]
 [1] -1.5567693412 -1.1069488315  0.0009026616 -2.3625949477 -3.0435404777 -2.7024735425 -4.5443853081 -6.7992027942 -4.5565287002  1.01129
[13] -0.8538891833 -2.2371164799 -2.1808640782
```

Error

```
Browse[1]> delta
          [,1]
[1,] -2.871905
[2,] -6.886667
```

$\frac{\partial E}{\partial W}$ gradient or delta

```
Browse[1]> - alpha * delta
            [,1]
[1,] 0.02871905
[2,] 0.06886667
```

Gradient

Gradient Descent: in vectors and matrices
Weighted sum of training data

$$\frac{\partial E}{\partial W} = \frac{\partial}{\partial W}\sum_{i=1:Train}\frac{1}{2}(X_i W^T - t_i)^2 \ /Train$$

$$\frac{\partial E}{\partial W} = (\sum_{i=1:Train} X^T(X_i W^T - t_i))/Train$$

$$\frac{\partial E}{\partial W} = X^T(XW^T - t)/Train$$

$X^T \times$ Error

$$W = W - \alpha\frac{\partial E}{\partial W}$$

$$\begin{bmatrix} \ \end{bmatrix} = \begin{bmatrix} \ \end{bmatrix} - 0.01 \times \begin{bmatrix} \ \ldots \ \end{bmatrix} \times \begin{bmatrix} .. \\ \ \end{bmatrix}$$

```
107 # gradient descent
108 for (i in 1:num_iters) {
109   error <- (X %*% theta - y)
110   delta <- t(X) %*% error / length(y)
111   browser()
112   theta <- theta - alpha * delta
113   cost_history[i] <- cost(X, y, theta)
114   theta_history[[i]] <- theta
115 }
116
```

X^T #Transposed augmented design matrix (2 input variable; 2 rows in transpose mode)

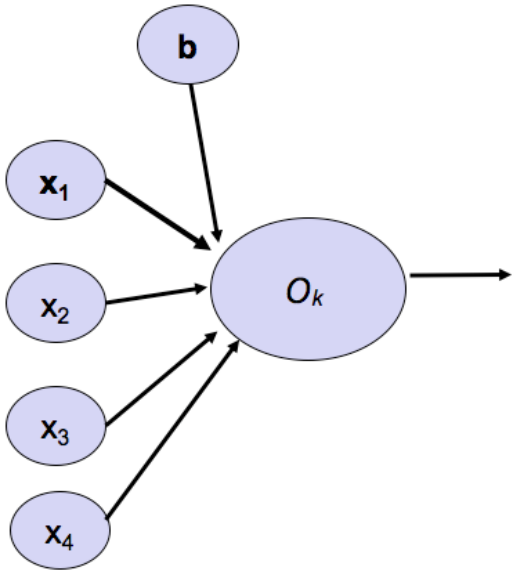
```
Browse[1]> t(X)[,1:15]
          [,1]      [,2]     [,3]      [,4]      [,5]      [,6]     [,7]     [,8]     [,9]     [,10]    [,11]    [,12]    [,13]     [,14]     [,15]
[1,]  1.000000  1.0000000  1.00000  1.000000 1.0000000  1.000000 1.000000 1.00000 1.000000  1.000000 1.000000 1.000000 1.000000  1.0000000  1.000000
[2,] -2.693694 -0.8262961 -3.99729 -2.214886 0.3054649 -1.858805 1.007021 4.87098 1.092815 -3.542627 2.043165 1.791803 -1.774161 -0.7668436 -1.955358
Browse[1]> error[1:15,]
 [1] -1.5567693412 -1.1069488315  0.0009026616 -2.3625949477 -3.0435404777 -2.7024735425 -4.5443853081 -6.7992027942 -4
[13] -0.8538891833 -2.2371164799 -2.1808640782
Browse[1]> delta
          [,1]
[1,] -2.871905
[2,] -6.886667
Browse[1]> - alpha * delta
            [,1]
[1,] 0.02871905
[2,] 0.06886667
```

$\frac{\partial E}{\partial W}$ **Gradient or Delta**

Error, i.e., the weights in the weighted sum (aka gradient)

Gradient

Activation matrices and model matrices for a single target variable



4 1 1 Prediction

150 4 4 = 150

X W P

$$W = W - \alpha \nabla_W = \times$$

Weighted Sum

E

Model

Data

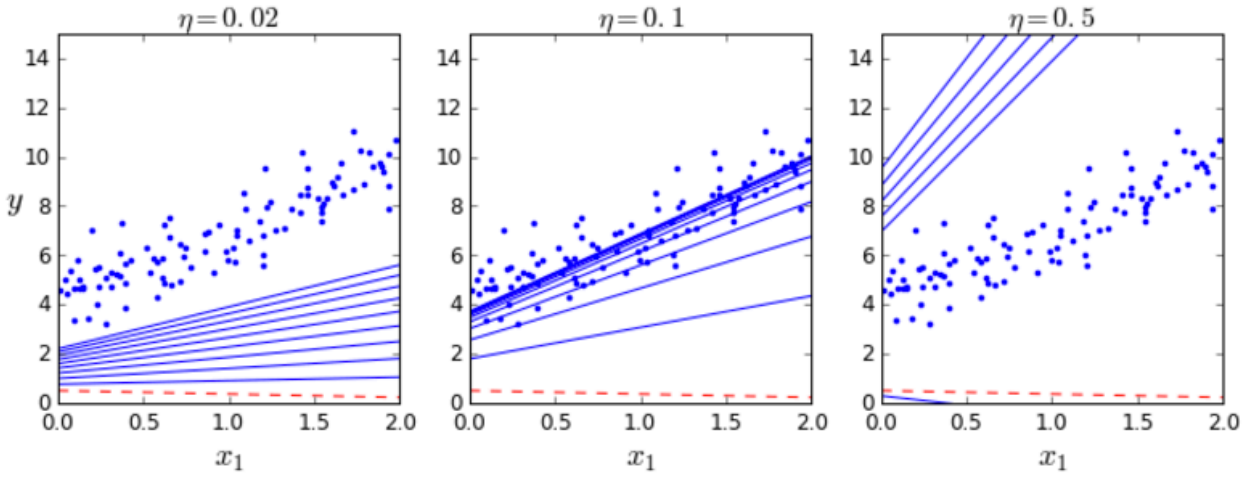
$$\begin{bmatrix} 4.32 \\ 3.98 \\ 7.92 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 8 \end{bmatrix} - 0.01 \times 2 \times \begin{bmatrix} 34 \\ 1 \\ 4 \end{bmatrix}$$

3.1 Different learning rates and their progress during the first 10 iterations of GD

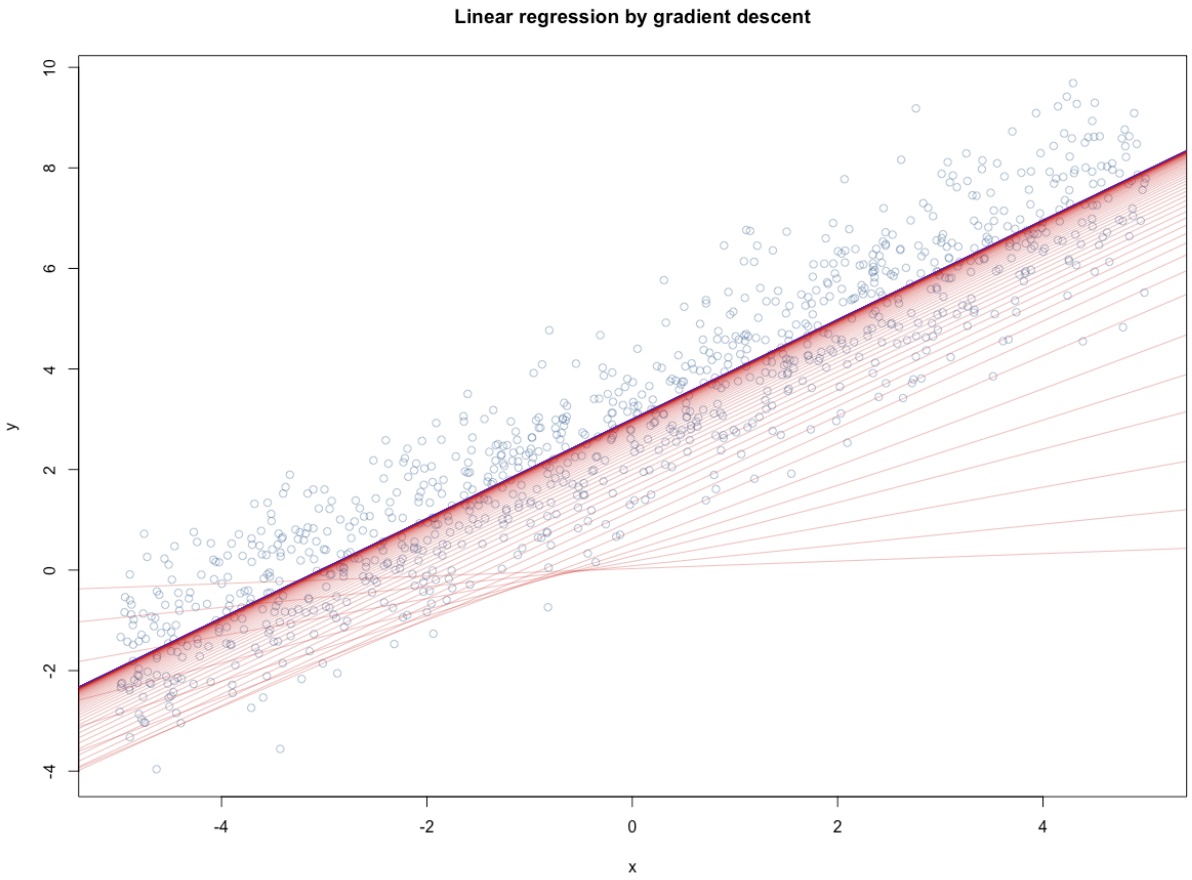
```
theta_path_bgd = []
#eta, the learning rate
def plot_gradient_descent(theta, eta, theta_path=None):
    m = len(X_b)
    plt.plot(X, y, "b.")
    n_iterations = 1000
    for iteration in range(n_iterations):
        if iteration < 10:
            y_predict = X_new_b.dot(theta)
            style = "b-" if iteration > 0 else "r--"
            plt.plot(X_new, y_predict, style)
        gradients = 2/m * X_b.T.dot(X_b.dot(theta) - y)
        theta = theta - eta * gradients
        if theta_path is not None:
            theta_path.append(theta)
    plt.xlabel("$x_1$", fontsize=18)
    plt.axis([0, 2, 0, 15])
    plt.title(r"$\eta = {}$".format(eta), fontsize=16)

rnd.seed(42)
theta = rnd.randn(2,1)  # random initialization

plt.figure(figsize=(10,4))
plt.subplot(131); plot_gradient_descent(theta, eta=0.02)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.subplot(132); plot_gradient_descent(theta, eta=0.1, theta_path=theta_path_bgd)
plt.subplot(133); plot_gradient_descent(theta, eta=0.5)

save_fig("gradient_descent_plot")
plt.show()
```

http://localhost:8890/notebooks/Dropbox/Projects/Target-2016-04/DeepLearning/Src/handson-ml-master/04_training_linear_models.ipynb#Linear-regression-using-batch-gradient-descent-3

```
Saving figure gradient_descent_plot
```



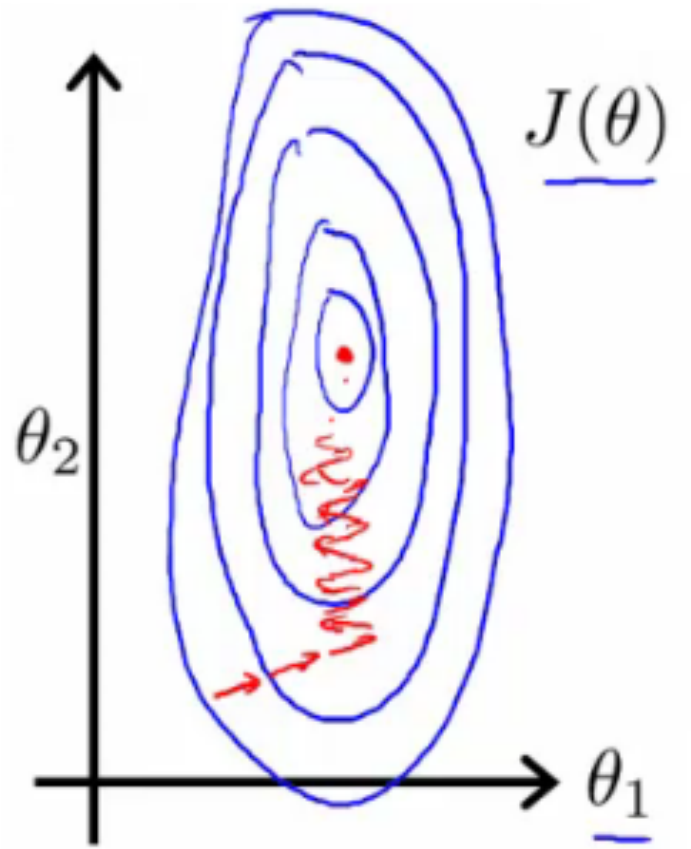
Linear Regression in R



https://www.dropbox.com/s/w8wzviukvcpl720/LinearRegressionClosed-Form-Gradient-Descent.R?dl=0

Sometimes we overshoot

Alpha our learning rate needs to be small enough to allow us to converge. Otherwise we will oscilliate. …can also decrease ALPHA over time… Gradient descent provides no guarantees to find the minimum but….



$$RSS = Variance \text{ of } \varepsilon$$

$$0 = \frac{\partial \sum \hat{\varepsilon}_i^2}{\partial W} = \frac{\partial \left(\sum_{j=1}^{n} \left(X_j W - y_i \right)^2 \right)}{\partial W}$$

$$\nabla J(W) = \left(\sum_{j=1}^{n} \left(X_j W - y_i \right) X_j \right)$$

$$W^{t+1} = W^t - \alpha^i \left(\sum_{j=1}^{n} \left(X_j W^t - y_i \right) X_j \right)$$

Learning Rate: α

- **Fixed learning rate**
  - While it is more common to run stochastic gradient descent as we have described it and with a fixed learning rate
- **Dynamic, decreasing learning rate**
  - by slowly letting the learning rate decrease to zero as the algorithm runs, it is also possible to ensure that the parameters will converge to the global minimum rather then merely oscillate around the minimum

- **Or it can be calculated**

$$\alpha = \frac{\mathbf{h}^{\mathbf{T}}\mathbf{h}}{\mathbf{h}^{\mathbf{T}}\mathbf{H}\mathbf{h}} = \frac{\nabla f(x_0)^{\mathbf{T}} \nabla f(x_0^{\mathbf{T}})}{\nabla f(x_0)^{\mathbf{T}} H \nabla f(x_0^{\mathbf{T}})}$$

Lecture Outline

- **Introduction**
- **Linear regression from a statistical perspective**
  - Simple linear regression
  - Assessing the quality of LR model
  - Multiple linear regression (and challenges)
- **Linear regression from scratch**
  - LR via Gradient Descent via brute force
  - LR via Normal Equation (Closed-form analytical optimization)
  - LR via Gradient Descent (Numerical optimization)
  - LR via gradient descent in graph form
- **Summary**

End of lecture