Linear Regression

Outline

- Univariate linear regression
- Gradient descent
- Multivariate linear regression
- Polynomial regression
- Regularization

Classification vs. Regression

- Previously, we looked at *classification* problems where we used ML algorithms (e.g., *k*NN, decision trees, perceptrons) to predict *discrete*-valued (categorical with no numerical relationship) outputs

  - Given email, predict ham or spam
  - Given medical info, predict diabetes or not
  - Given tweets, predict positive or negative sentiment
  - Given Titanic passenger info, predict survival or not
  - Given images of handwritten numbers, predict intended digit

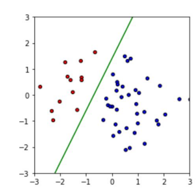
---

- Here, we look at *regression* problems where we use ML algorithms (e.g., linear regression) to predict *real*-valued outputs

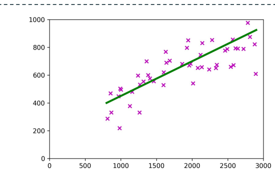
  - Given student info, predict exam scores
  - Given physical attributes, predict age
  - Given medical info, predict blood pressure
  - Given real estate ad, predict housing price
  - Given review text, predict numerical rating

Classification vs. Regression

- Previously, we looked at *classification* problems where we used ML algorithms (e.g., *k*NN, decision trees, perceptrons) to predict *discrete*-valued (categorical with no numerical relationship) outputs



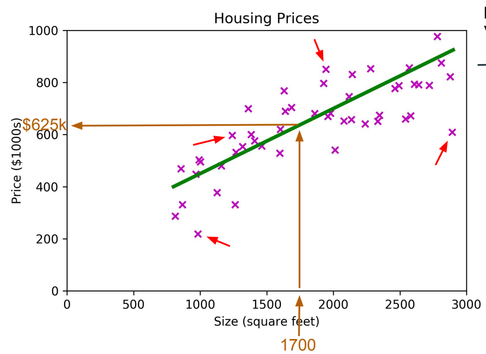
- Here, we look at *regression* problems where we use ML algorithms (e.g., linear regression) to predict *real*-valued outputs



Linear Regression (fitting a straight line)

Linear Regression (fitting a straight line)

Univariate Linear Regression



Feature Vectors X	Real Valued Labels y
1960	841,075.25
1250	590,999.99
960	210,500.86
2900	600,000.00
...	...

Hypothesis Function

- Our *hypothesis* function *h* maps *x*'s to predicted *y* values
- For now, we'll predict that *y* is a linear function of *x*:

$$h(x) = w_0 + w_1 x$$

Predicted Label

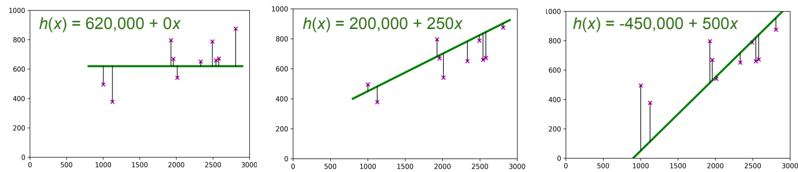
Data Point

With perceptrons, we named our bias term *b*. Here we name our intercept term w_0.

Sometimes the "weight" parameters are named theta, Θ, rather than *w*

Cost Function: Mean Squared Error

Objective: of all possible lines, find the one that *minimizes* the distance between the predicted *y* values (on the line) and the true *y* values

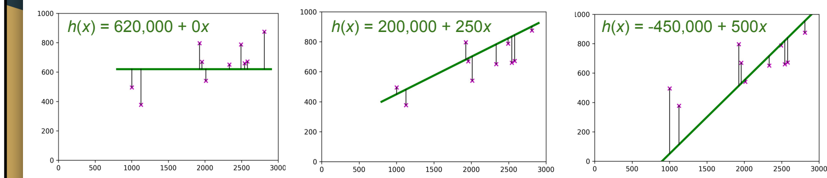


Cost Function: Mean Squared Error

$$J(w)=\frac{1}{2n}\sum_{i=1}^{n}(h(x^{(i)})-y^{(i)})^2$$

$$J(w)=\frac{1}{2n}\sum_{i=1}^{n}((w_0+w_1x^{(i)})-y^{(i)})^2$$

Objective: of all possible lines, find the one that *minimizes* the distance between the predicted *y* values (on the line) and the true *y* values



Cost Function: Mean Squared Error

$$J(w)=\frac{1}{2n}\sum_{i=1}^{n}(h(x^{(i)})-y^{(i)})^2$$

$$J(w)=\frac{1}{2n}\sum_{i=1}^{n}((w_0+w_1x^{(i)})-y^{(i)})^2$$

Objective: of all possible lines, find the one that *minimizes* the distance between the predicted *y* values (on the line) and the true *y* values

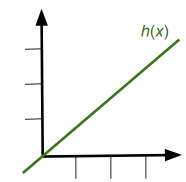
In other words, find w_0 and w_1 that minimize the cost function $J(w)$ for our n training examples

Cost Function (Simplified Example: w_0=0)

- As a simplification for the moment, let's set w_0 to be zero
- This means that our line will pass through the origin
- Our hypothesis is then $h(x) = 0 + w_1x = w_1x$
- Our cost function is then $J(w_1)=\frac{1}{2n}\sum_{i=1}^{n}(h(x^{(i)})-y^{(i)})^2$

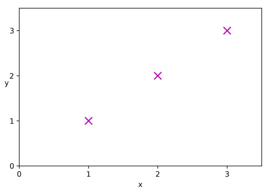
$$J(w_1)=\frac{1}{2n}\sum_{i=1}^{n}((w_1x^{(i)})-y^{(i)})^2$$

- Our goal is to find w_1 that minimizes $J(w_1)$

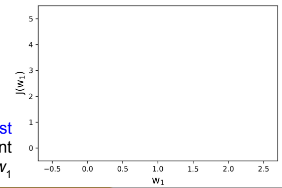


Cost Function
(Simplified Example: w_0=0)

Suppose we have the following three training data points



Let's consider the cost associated with different values of w_1

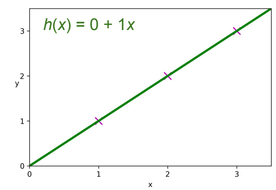


Cost Function
(Simplified Example: w_0=0)

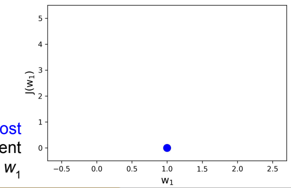
$$J(w_1)=\frac{1}{2n}\sum_{i=1}^{n}((w_1x^{(i)})-y^{(i)})^2$$
$$=\frac{1}{2n}\sum_{i=1}^{n}((1x^{(i)})-y^{(i)})^2$$
$$=\frac{1}{2*3}\sum_{i=1}^{3}((1x^{(i)})-y^{(i)})^2$$
$$=\frac{1}{2*3}((1-1)^2+(2-2)^2+(3-3)^2)\ =\ 0$$

Suppose we have the following three training data points

w_1 = 1



Let's consider the cost associated with different values of w_1

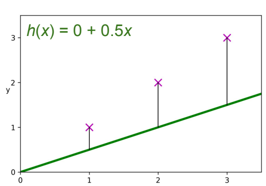


Cost Function
(Simplified Example: w_0=0)

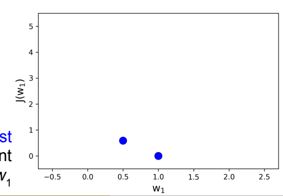
$$J(w_1)=\frac{1}{2n}\sum_{i=1}^{n}((w_1x^{(i)})-y^{(i)})^2$$
$$=\frac{1}{2n}\sum_{i=1}^{n}((0.5x^{(i)})-y^{(i)})^2$$
$$=\frac{1}{2*3}\sum_{i=1}^{3}((0.5x^{(i)})-y^{(i)})^2$$
$$=\frac{1}{2*3}((0.5-1)^2+(1-2)^2+(1.5-3)^2)\approx 0.58$$

Suppose we have the following three training data points

w_1 = 0.5



Let's consider the cost associated with different values of w_1

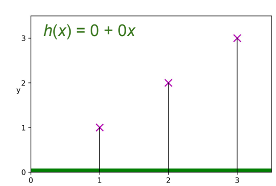


Cost Function
(Simplified Example: w_0=0)

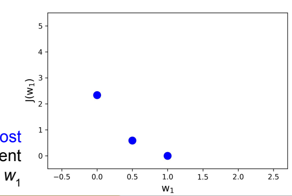
$$J(w_1)=\frac{1}{2n}\sum_{i=1}^{n}((w_1x^{(i)})-y^{(i)})^2$$
$$=\frac{1}{2n}\sum_{i=1}^{n}((0x^{(i)})-y^{(i)})^2$$
$$=\frac{1}{2*3}\sum_{i=1}^{3}((0x^{(i)})-y^{(i)})^2$$
$$=\frac{1}{2*3}((0-1)^2+(0-2)^2+(0-3)^2)\approx 2.33$$

Suppose we have the following three training data points

w_1 = 0



Let's consider the cost associated with different values of w_1

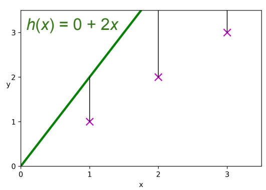


Cost Function
(Simplified Example: w_0=0)

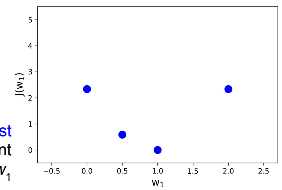
$$J(w_1) = \frac{1}{2n}\sum_{i=1}^{n}((w_1 x^{(i)}) - y^{(i)})^2$$
$$= \frac{1}{2n}\sum_{i=1}^{n}((2x^{(i)}) - y^{(i)})^2$$
$$= \frac{1}{2*3}\sum_{i=1}^{3}((2x^{(i)}) - y^{(i)})^2$$
$$= \frac{1}{2*3}((2-1)^2 + (4-2)^2 + (6-3)^2) \approx 2.33$$

Suppose we have the following three training data points

$w_1 = 2$

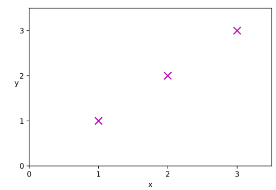


Let's consider the cost associated with different values of w_1

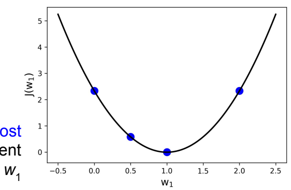


Cost Function
(Simplified Example: w_0=0)

Suppose we have the following three training data points

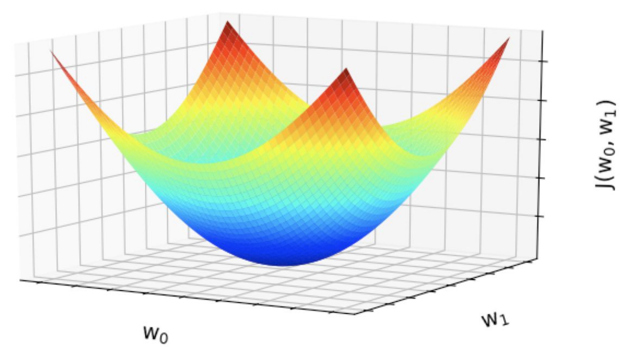


Let's consider the cost associated with different values of w_1

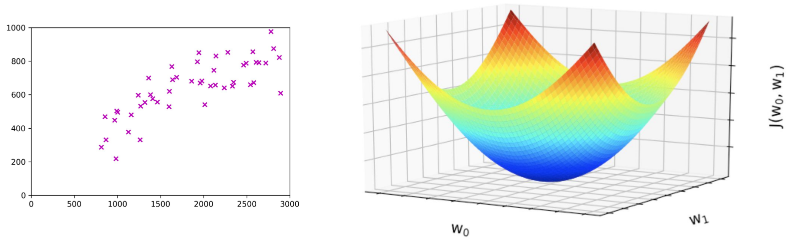


Cost Function: Two parameters (w_0, w_1)

Cost Function: Two parameters (w_0, w_1)



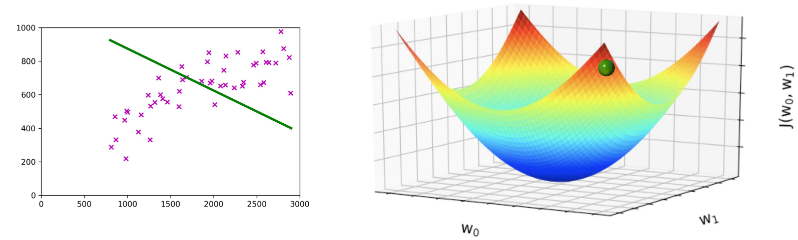
Cost Function: Two parameters (w_0, w_1)



Data and Hypothesis

Cost Function

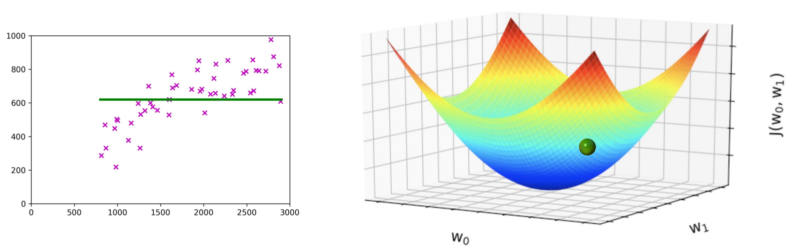
Cost Function: Two parameters (w_0, w_1)



Data and Hypothesis

Cost Function

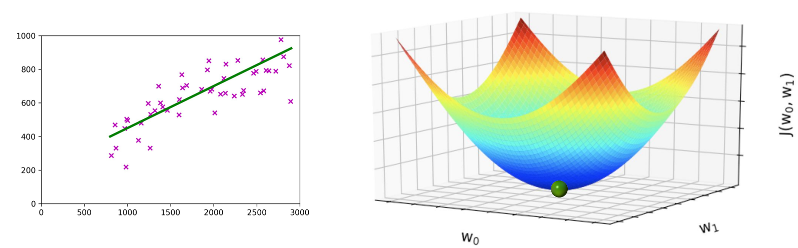
Cost Function: Two parameters (w_0, w_1)



Data and Hypothesis

Cost Function

Cost Function: Two parameters (w_0, w_1)



Data and Hypothesis

Cost Function

Outline

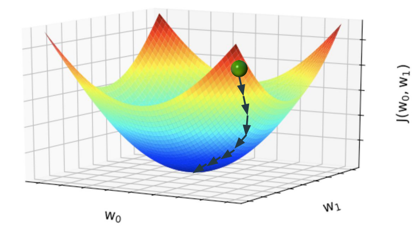
- Univariate linear regression
- Gradient descent
- Multivariate linear regression
- Polynomial regression
- Regularization

Gradient Descent

We want to find the line that best fits the data, i.e., we want to find w_0 and w_1 that minimize the cost, $J(w_0, w_1)$

Gradient Descent Algorithm

- Start with some w_0 and w_1 (e.g., $w_0 = 0$ and $w_1 = 0$)
- Keep changing w_0 and w_1 to reduce the cost $J(w_0, w_1)$ until hopefully we end up at a minimum

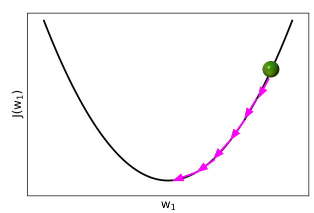


Gradient Descent (Simplified Example: w_0=0)

We want to find the line (passing through the origin) that best fits the data, i.e., we want to find w_1 that minimize the cost, $J(w_1)$

Gradient Descent Algorithm

- Start with some w_1 (e.g., $w_1 = 0$)
- Keep changing w_1 to reduce the cost $J(w_1)$ until hopefully we end up at a minimum

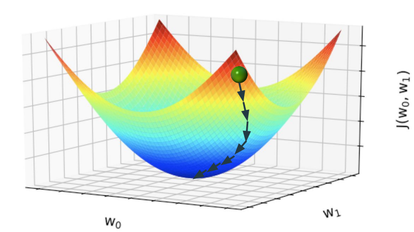


Gradient Descent Algorithm

Repeat until convergence:

$$w_0 = w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$$

$$w_1 = w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$$



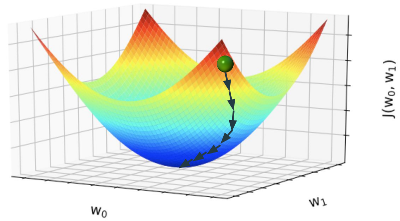
Gradient Descent Algorithm

Repeat until convergence:

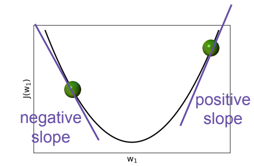
$$w_0 = w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$$

$$w_1 = w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$$

α is the ***step size*** or ***learning rate***



Gradient Descent Algorithm

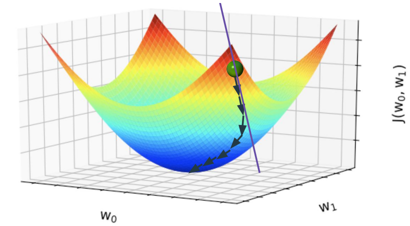


Repeat until convergence:

$$w_0 = w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$$

$$w_1 = w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$$

The partial derivative indicates the slope, i.e., the direction to step



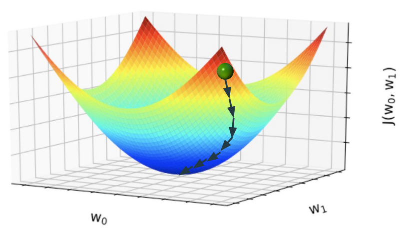
Gradient Descent Algorithm

Repeat until convergence:

$$w_0 = w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$$

$$w_1 = w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$$

Execute two updates in parallel rather than sequentially



Gradient Descent Algorithm

Repeat until convergence:

$$w_0 = w_0 - \alpha \boxed{\frac{\partial}{\partial w_0} J(w_0, w_1)}$$

$$w_1 = w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$$

$$\frac{\partial}{\partial w_0} J(w_0, w_1)$$

$$= \frac{\partial}{\partial w_0} \frac{1}{2n} \sum_{i=1}^{n} (h(x^{(i)}) - y^{(i)})^2$$

$$= \frac{\partial}{\partial w_0} \frac{1}{2n} \sum_{i=1}^{n} ((w_0 + w_1 x^{(i)}) - y^{(i)})^2$$

$$= \frac{1}{n} \sum_{i=1}^{n} ((w_0 + w_1 x^{(i)}) - y^{(i)})$$

$$= \frac{1}{n} \sum_{i=1}^{n} (h(x^{(i)}) - y^{(i)})$$

Gradient Descent Algorithm

Repeat until convergence:

$$w_0 = w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$$

$$w_1 = w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$$

$$\frac{\partial}{\partial w_1} J(w_0, w_1)$$

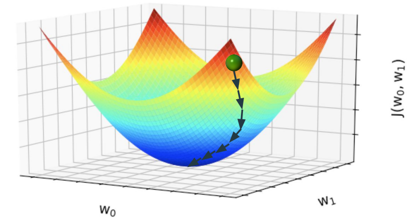
$$= \frac{\partial}{\partial w_1} \frac{1}{2n} \sum_{i=1}^{n} (h(x^{(i)}) - y^{(i)})^2$$

$$= \frac{\partial}{\partial w_1} \frac{1}{2n} \sum_{i=1}^{n} ((w_0 + w_1 x^{(i)}) - y^{(i)})^2$$

$$= \frac{1}{n} \sum_{i=1}^{n} ((w_0 + w_1 x^{(i)}) - y^{(i)})\, x^{(i)}$$

$$= \frac{1}{n} \sum_{i=1}^{n} (h(x^{(i)}) - y^{(i)})\, x^{(i)}$$

Gradient Descent Algorithm



Initialize:

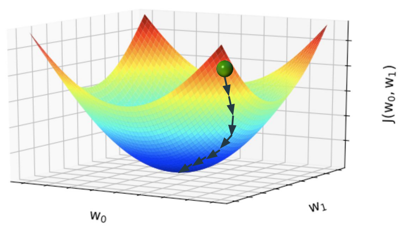
$$w_0 = 0 \qquad w_1 = 0$$

Repeat until convergence:

$$w_0 = w_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} ((w_0 + w_1 x^{(i)}) - y^{(i)})$$

$$w_1 = w_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} ((w_0 + w_1 x^{(i)}) - y^{(i)}) x^{(i)}$$

Gradient Descent Algorithm



Initialize:

$$w_0 = 0 \qquad w_1 = 0$$

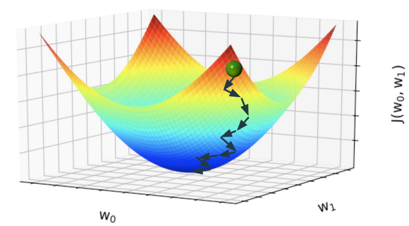
Repeat until convergence:

$$w_0 = w_0 - \alpha \boxed{\frac{1}{n} \sum_{i=1}^{n}} ((w_0 + w_1 x^{(i)}) - y^{(i)})$$

$$w_1 = w_1 - \alpha \boxed{\frac{1}{n} \sum_{i=1}^{n}} ((w_0 + w_1 x^{(i)}) - y^{(i)}) x^{(i)}$$

With ***batch*** gradient descent, we consider *all* data points each time we update a weight parameter

Gradient Descent Algorithm



Initialize:

$$w_0 = 0 \qquad w_1 = 0$$

Repeat until convergence, iterating over each data point (x, y):

$$w_0 = w_0 - \alpha((w_0 + w_1 x) - y)$$

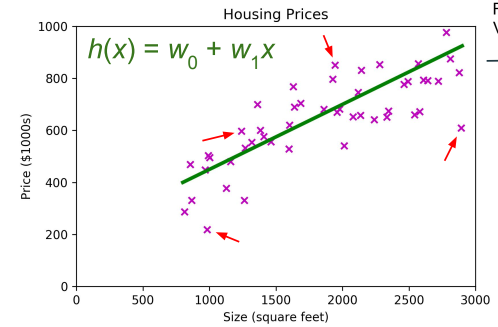
$$w_1 = w_1 - \alpha((w_0 + w_1 x) - y) x$$

With ***stochastic*** gradient descent, we consider *a single* data point each time we update a weight parameter

Outline

- Univariate linear regression
- Gradient descent
- Multivariate linear regression
- Polynomial regression
- Regularization

Univariate Linear Regression



Feature Vectors X	Real Valued Labels y
1960	841,075.25
1250	590,999.99
960	210,500.86
2900	600,000.00
...	...

Multivariate Linear Regression

Feature Vectors X				Real Valued Labels y
x_1 Size (feet^2)	x_2 Number bedrooms	x_3 Lot (feet^2)	x_4 Age (years)	
1960	3	19,000	12	841,075.25
1250	3	10,700	65	590,999.99
960	2	12,035	41	210,500.86
2900	5	15,431	23	600,000.00
...	...	...	...	...

$h(\boldsymbol{x}) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$

Multivariate Linear Regression

Suppose we have *d* features, then our hypothesis is given by:

$h(\boldsymbol{x}) = w_0 + w_1x_1 + w_2x_2 + \dots + w_dx_d$

To simplify our notation, we define a new feature x_0 that always has the value of 1. Then we can write our hypothesis as:

$h(\boldsymbol{x}) = w_0x_0 + w_1x_1 + w_2x_2 + \dots + w_dx_d$

$h(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}$

$h(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$

where ***w*** and ***x*** are *d*+1 dimensional vectors

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_d \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_d \end{bmatrix}$$

Multivariate Linear Regression

Univariate (d = 1)

Hypothesis: $h(\mathbf{x}) = w_0 + w_1 x$

Cost function: $J(w_0, w_1) = \frac{1}{2n} \sum_{i=1}^{n} (h(x^{(i)}) - y^{(i)})^2$

Gradient descent (repeated update):

$$w_0 = w_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (h(x^{(i)}) - y^{(i)})$$

$$w_1 = w_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} (h(x^{(i)}) - y^{(i)})\, x^{(i)}$$

Multivariate (d ≥ 1)

Hypothesis: $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$

Cost function: $J(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^{n} (h(x^{(i)}) - y^{(i)})^2$

Gradient descent (repeated update):

$$w_0 = w_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (h(x^{(i)}) - y^{(i)})\, x_0^{(i)}$$

$$w_1 = w_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} (h(x^{(i)}) - y^{(i)})\, x_1^{(i)}$$

$$\cdots$$

$$w_d = w_d - \alpha \frac{1}{n} \sum_{i=1}^{n} (h(x^{(i)}) - y^{(i)})\, x_d^{(i)}$$

Feature Scaling

Feature Vectors
X

Real Valued Labels
y

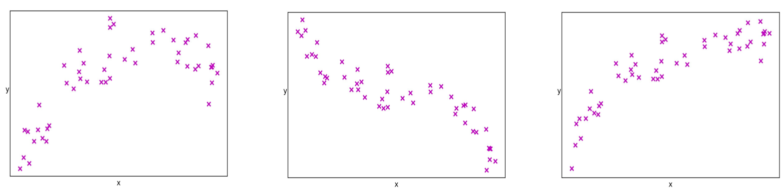
x_1 Size (feet^2)	x_2 Number bedrooms	x_3 Lot (feet^2)	x_4 Age (years)	
1960	3	19,000	12	841,075.25
1250	3	10,700	65	590,999.99
960	2	12,035	41	210,500.86
2900	5	15,431	23	600,000.00
...	...	...	...	...

- Features may have very different ranges!
- Don't forget to perform *feature scaling*, e.g., subtract each feature's mean and divide by each feature's standard deviation.
- Then features will have the same scale.

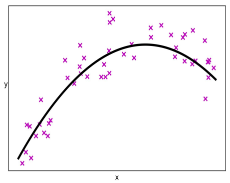
Outline

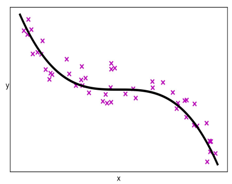
- Univariate linear regression
- Gradient descent
- Multivariate linear regression
- Polynomial regression
- Regularization

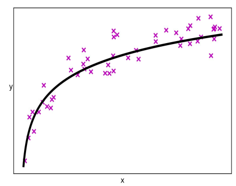
Polynomial Regression



Polynomial Regression





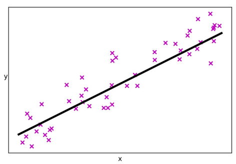


Polynomial Regression

x_1 = (Size)

Feature Vectors X | Real Valued Labels y

x_1 Size	y
1.9×10^3	841,075.25
1.2×10^3	590,999.99
0.9×10^3	210,500.86
2.9×10^3	600,000.00
...	...



$h(x) = w_0 + w_1 x_1$

$h(x) = w_0 + w_1(\text{Size})$

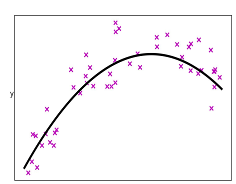
Polynomial Regression

x_1 = (Size)

x_2 = $(\text{Size})^2$

Feature Vectors X | Real Valued Labels y

x_1 Size	x_2 Size^2	y
1.9×10^3	3.8×10^6	841,075.25
1.2×10^3	1.5×10^6	590,999.99
0.9×10^3	0.9×10^6	210,500.86
2.9×10^3	8.4×10^6	600,000.00
...	...	...



$h(x) = w_0 + w_1 x_1 + w_2 x_2$

$h(x) = w_0 + w_1(\text{Size}) + w_2(\text{Size})^2$

Polynomial Regression

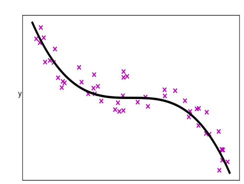
x_1 = (Size)

x_2 = $(\text{Size})^2$

x_3 = $(\text{Size})^3$

Feature Vectors X | Real Valued Labels y

x_1 Size	x_2 Size^2	x_3 Size^3	y
1.9×10^3	3.8×10^6	7.5×10^9	841,075.25
1.2×10^3	1.5×10^6	1.9×10^9	590,999.99
0.9×10^3	0.9×10^6	0.8×10^9	210,500.86
2.9×10^3	8.4×10^6	24.3×10^9	600,000.00
...	...	...	...



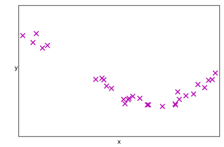
$h(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$

$h(x) = w_0 + w_1(\text{Size}) + w_2(\text{Size})^2 + w_3(\text{Size})^3$

Outline

- Univariate linear regression
- Gradient descent
- Multivariate linear regression
- Polynomial regression
- Regularization

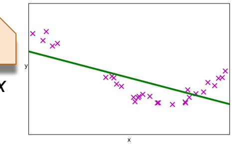
Overfitting



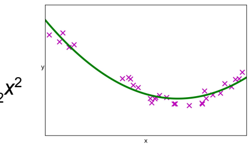
Overfitting may occur when we have too many features and the learned hypothesis fits the training data very well but fails to generalize to new examples.

Underfit

$w_0 + w_1x$

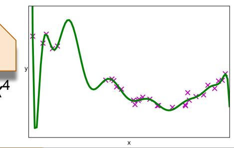


$w_0 + w_1x + w_2x^2$

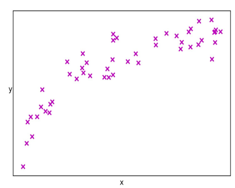


Overfit

$w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$



Addressing Overfitting

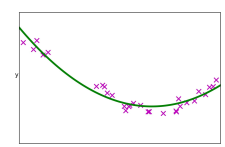


x_1 = size of house
x_2 = number of bedrooms
x_3 = lot size
x_4 = age of house
x_5 = parking spaces
x_6 = distance to schools
x_7 = neighborhood crime rate
…
x_d

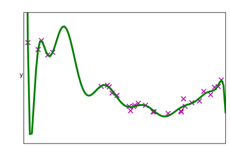
To address overfitting:

- Reduce the number of features
- *Regularization*. Keep all the features but reduce the magnitude/values of parameters ***w***.

Regularization



$w_0 + w_1x + w_2x^2$



$w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$

Suppose we penalize w_3 and w_4 in order to make them very small.

$$J(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h(x^{(i)}) - y^{(i)}\right)^2 + 1{,}000{,}000 w_3 + 1{,}000{,}000 w_4$$

$w_3 \approx 0$
$w_4 \approx 0$

Regularization

- Smaller values for the parameters $w_1, w_2, w_3, \ldots, w_d$ lead to simpler hypotheses that are less prone to overfitting.
- We modify our cost function so that it not only

  (1) finds a good fitting hypothesis (penalizes error of hypothesis on training data)

  but also

  (2) considers the complexity of the hypothesis (penalizing more complex hypotheses and favoring simpler hypotheses)

$$J(\mathbf{w}) = \frac{1}{2n}\left[\sum_{i=1}^{n}\left(h(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{d} w_j^2\right]$$

λ is regularization parameter

Gradient Descent $\quad w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\mathbf{w})$

Linear Regression

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j}\left(\frac{1}{2n}\sum_{i=1}^{n}\left(h(x^{(i)}) - y^{(i)}\right)^2\right) \qquad w_j = w_j - \alpha \frac{1}{n}\sum_{i=1}^{n}\left(h(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

Regularized Linear Regression

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j}\left(\frac{1}{2n}\left(\sum_{i=1}^{n}\left(h(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{d} w_j^2\right)\right)$$

$$w_j = w_j - \alpha \frac{1}{n}\left(\sum_{i=1}^{n}\left(h(x^{(i)}) - y^{(i)}\right)x_j^{(i)} + \lambda w_j\right)$$

$$w_j = w_j\left(1 - \alpha \frac{\lambda}{n}\right) - \alpha \frac{1}{n}\sum_{i=1}^{n}\left(h(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

Overview