

# Machine Learning

Lecture 2  
17-03-2022

Dr. Arif Mahmood  
Professor ITU

E-mail: arif.mahmood@itu.edu.pk

[Website](#) [Google Scholar](#)

Office: 6-th Floor CS Department, ITU

# Nearest Neighbor Classifiers

- Give label to an unknown instance as the label of its nearest neighbor
  - 1-Nearest Neighbor
  - 2-Nearest Neighbors
  - 3-Nearest Neighbors
- K-Nearest Neighbors

A man is known by the company he keeps.  
A man is judged by the company he keeps.  
Birds of a feather flock together.

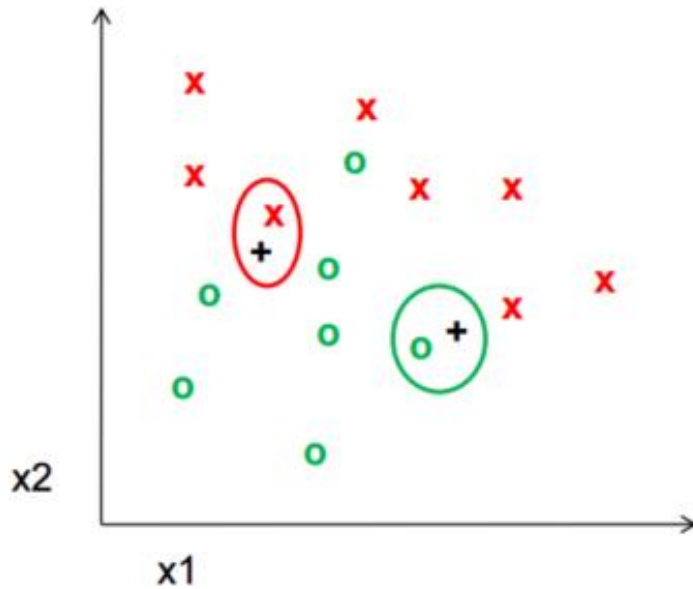
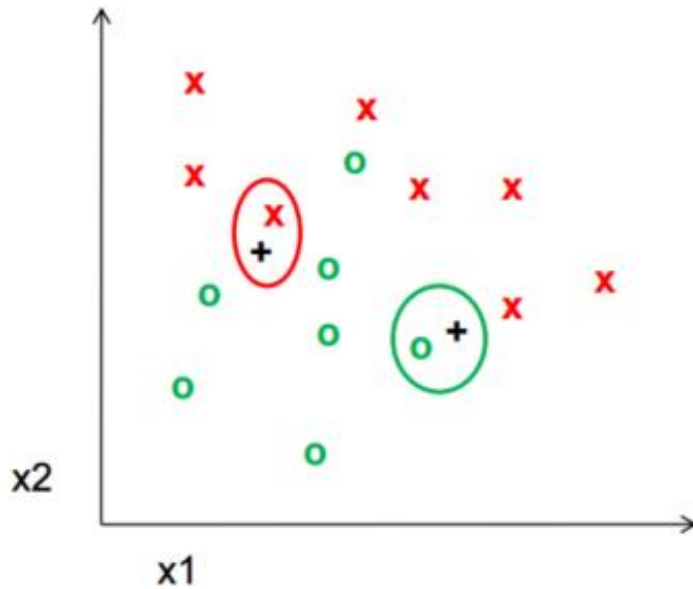
# 1-nearest neighbour

Task: classify the test set of “+”

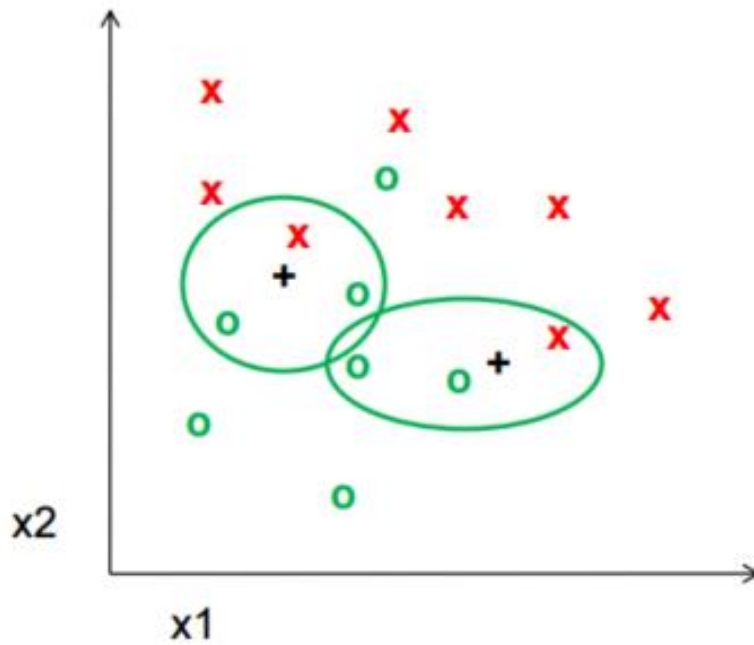
The labels for the training set are GREEN and RED

The examples are 2-dimensional

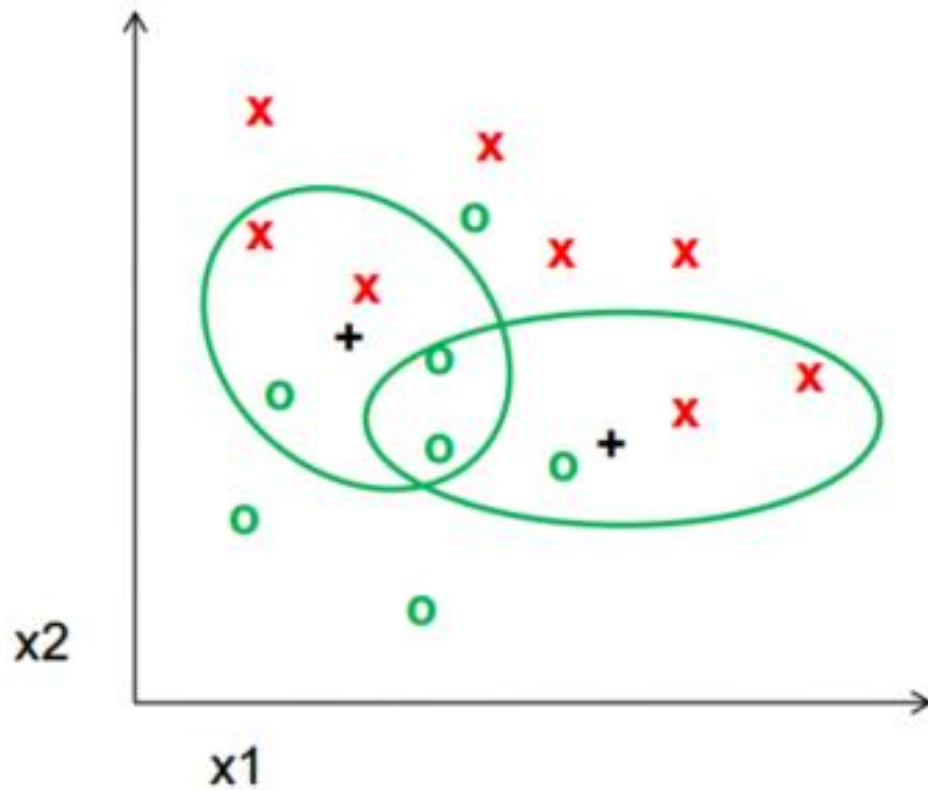
## Use L2/Euclidean distance



# 3-nearest neighbour



# 5-nearest neighbour



# The Task: Supervised Learning

- Given a set of labelled examples (the *training set*), determine/predict the labels of a set of unlabelled examples (the *test set*)

- Training set:

Train Example 1:  $(x_1^{(1)}, x_2^{(1)}, \dots, x_m^{(1)})$  Label:  $y^{(1)}$

Train Example 2:  $(x_1^{(2)}, x_2^{(2)}, \dots, x_m^{(2)})$  Label:  $y^{(2)}$

...

Train Example N:  $(x_1^{(N)}, x_2^{(N)}, \dots, x_m^{(N)})$  Label:  $y^{(N)}$

- Test set:

Test Example 1:  $(x_1^{(N+1)}, x_2^{(N+1)}, \dots, x_m^{(N+1)})$  Label:  $y^{(N+1)}$

Test Example 2:  $(x_1^{(N+2)}, x_2^{(N+2)}, \dots, x_m^{(N+2)})$  Label:  $y^{(N+2)}$

...

Test Example K:  $(x_1^{(N+K)}, x_2^{(N+K)}, \dots, x_m^{(N+K)})$  Label:  $y^{(N+K)}$

# Task: Face Recognition

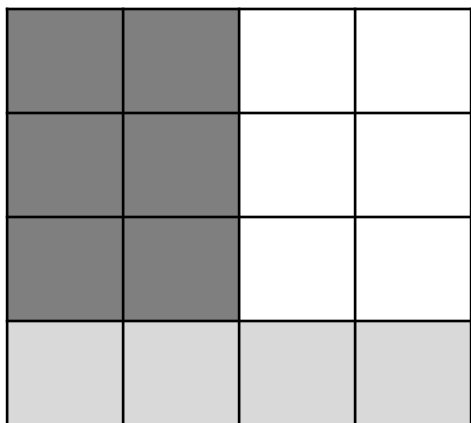
- Training set: photos of musicians with names (“labels”)
- Test set: photos of musicians whose name we want to figure out
  - Note: generally, we *will* know the labels for the test set, but we pretend we don’t. We can then predict the labels using our algorithm and compare the answers the algorithm gives to the correct answers to figure out the performance of our algorithm.
- An estimate for the performance of the algorithm on *new data*: the proportion of the examples in the test set that were correctly classified

# What Justin Bieber Looks like to a Computer

83 99 74 70 72 68 80 89 84 84 89 81 61 49 50 39 44 55 31 34 161 186 174 155 150 187 182 195 190 162 151 144 139 128 128 126 125 123 120 120 123 119 168 231 212 130 85 8  
5 44 43 62 27 26 34 34 53 39 59 64 25 32 54 32 85 68 82 88 53 77 55 77 74 82 81 89 86 77 73 64 52 51 51 33 59 83 76 63 147 148 122 141 166 188 202 194 169 150 146 140 1  
9 127 126 128 127 124 123 125 126 141 215 217 137 82 69 33 34 49 28 19 32 30 28 29 40 39 31 24 33 33 43 36 63 58 71 54 68 77 65 79 72 84 75 64 70 68 54 49 57 56 72 89  
6 76 77 132 113 151 172 184 194 193 175 150 147 142 90 96 100 101 100 98 98 103 107 104 181 195 130 90 79 61 46 29 25 17 27 37 28 45 42 28 32 36 18 36 32 34 59 58 72 6  
2 64 78 76 91 93 95 83 79 71 61 66 59 59 58 61 91 108 78 174 164 156 164 181 190 202 194 163 155 151 149 33 38 41 44 46 46 45 47 49 50 73 93 131 128 85 74 65 36 17 10 1  
8 40 46 29 33 59 54 44 41 65 65 64 72 77 68 80 83 72 87 93 101 106 95 89 83 72 71 68 63 51 63 92 47 165 189 174 174 172 188 201 199 180 154 149 151 151 26 28 27 28 30 3  
0 27 27 28 26 23 35 96 126 98 98 72 70 63 57 50 35 21 22 65 61 76 109 102 102 105 93 110 92 87 89 90 97 110 116 104 96 98 107 94 68 59 56 58 61 64 64 137 200 189 186 18  
5 174 190 206 195 172 156 156 148 145 27 28 28 28 31 32 32 32 25 26 33 44 80 108 77 62 39 28 48 34 51 69 44 78 94 98 87 66 54 50 35 23 77 99 78 69 116 159 163 149 115 6  
0 46 74 75 54 42 38 56 69 74 127 175 182 188 182 194 183 194 202 182 165 160 153 146 142 33 40 47 52 58 63 65 67 65 82 87 78 83 100 74 78 43 52 34 23 6 13 59 70 74 42 3  
0 19 40 63 73 105 122 102 122 136 145 147 97 52 35 23 11 13 36 62 66 60 84 109 126 132 178 167 186 180 187 190 196 188 164 158 163 155 150 146 104 115 125 132 138 141 1  
41 142 126 153 172 141 80 82 70 75 74 59 61 42 24 16 15 52 21 12 67 88 106 123 128 153 121 118 114 150 127 70 63 29 11 36 32 17 28 33 45 89 90 115 114 101 139 154 180 1  
80 175 190 189 170 150 148 159 158 153 150 135 144 148 149 151 151 151 150 145 158 203 179 85 96 92 58 67 57 61 56 58 37 14 55 48 58 76 58 76 68 49 69 50 65 88 114 117  
126 118 48 20 61 72 53 54 42 29 57 95 99 100 96 122 119 154 174 178 189 174 159 152 144 149 155 154 153 122 125 123 118 119 121 122 123 126 144 196 179 84 103 91 34 45  
68 65 55 45 47 48 52 65 64 82 69 75 67 68 74 85 110 112 111 127 139 117 73 5 53 84 68 60 61 50 52 91 52 80 92 118 127 141 171 177 178 153 151 163 153 152 162 158 156 12  
5 121 120 123 125 123 121 120 123 155 199 159 104 118 57 48 40 48 47 63 53 51 54 66 66 88 75 82 97 103 108 106 105 128 110 115 118 138 129 63 12 67 67 37 47 47 54 73 77  
86 68 91 131 109 143 170 175 148 136 152 143 154 151 161 170 181 122 120 121 122 122 122 122 122 119 184 202 137 138 127 106 93 62 50 39 53 69 46 46 64 69 90 67 66 52  
53 52 51 110 128 93 94 92 132 123 37 34 86 50 40 58 53 81 99 95 107 75 145 112 149 159 177 163 131 143 145 174 156 165 157 172 177 120 120 120 120 120 121 121 122 131 1  
95 187 107 156 92 80 68 60 42 43 57 51 58 72 60 66 85 80 60 51 47 64 59 89 116 85 124 125 135 100 12 65 73 43 58 64 51 79 94 130 132 105 159 138 162 183 150 154 152 140  
158 163 187 182 182 185 180 118 120 120 119 119 121 120 118 147 209 174 148 94 90 64 75 73 66 62 81 96 80 58 45 77 89 70 72 56 82 85 79 87 94 94 131 124 118 39 32 96 7  
5 44 69 80 75 74 120 136 162 148 180 181 153 185 147 149 156 159 177 184 189 192 202 192 178 118 118 117 117 119 120 118 115 160 208 177 97 95 62 73 59 63 79 80 94 113  
97 69 55 77 72 73 71 77 87 114 123 110 109 85 135 92 35 20 86 97 47 41 62 72 74 90 98 152 152 132 134 149 107 149 165 141 168 189 181 198 194 209 199 205 192 117 116 11  
5 117 119 118 117 117 176 196 106 79 75 73 82 87 71 73 98 80 129 106 97 41 91 77 66 89 80 97 113 147 163 132 40 19 11 61 96 118 97 44 66 50 66 122 93 110 142 108 104 13  
2 135 113 115 169 163 188 193 210 202 203 201 197 197 195 115 113 114 116 114 118 124 182 115 78 89 101 86 114 84 95 106 80 101 115 116 88 94 93 73 64 73 90 99 106  
118 164 147 69 92 95 94 116 106 61 43 75 65 100 101 155 97 113 124 102 121 90 150 139 174 175 198 202 202 194 188 187 160 173 183 113 111 112 115 113 111 119 130 130 88  
90 100 113 103 102 104 115 97 105 87 117 101 123 103 76 90 92 89 81 97 99 113 154 139 70 61 89 111 108 89 66 82 82 123 88 140 154 132 156 137 97 112 119 129 146 162 18  
7 193 184 174 159 150 159 138 139 155 99 106 100 100 111 99 101 154 100 104 110 107 115 115 107 123 112 119 109 100 99 121 121 103 82 103 77 81 78 96 103 106 107 113 62  
43 82 108 104 82 64 114 114 121 108 143 174 157 153 142 120 111 111 107 123 158 179 177 137 138 129 142 134 145 131 141 51 41 47 48 41 49 141 108 97 107 118 117 125 12  
7 120 132 123 113 114 82 96 109 115 115 102 70 105 86 93 100 118 133 119 136 84 34 69 115 110 100 128 125 82 116 110 137 170 206 179 126 112 122 137 128 153 153 168 132  
130 140 143 119 137 144 130 135 24 18 25 25 58 131 107 106 119 129 128 135 139 132 135 132 120 125 87 93 100 98 128 139 92 88 57 114 103 111 101 135 154 125 61 60 8  
6 84 92 87 99 95 79 140 150 190 182 167 150 145 104 142 108 155 154 149 147 133 125 147 121 122 140 128 129 24 27 30 21 24 162 102 127 120 130 135 131 138 145 143 143 1  
87 131 115 101 94 108 95 141 167 124 130 121 108 135 116 115 141 140 142 116 94 94 102 121 122 132 110 131 151 166 163 171 156 157 132 88 143 74 113 157 158 155 147 123  
128 120 131 126 131 133 23 28 19 29 56 125 120 118 122 134 138 135 136 142 151 152 141 146 118 117 97 102 110 159 159 133 142 135 136 140 138 133 103 94 102 122 128 13  
2 147 162 175 177 174 164 151 130 134 151 172 180 126 90 134 73 90 147 160 155 131 144 118 127 121 131 125 129 29 36 46 52 156 117 116 135 126 139 144 143 135 134 148 1  
62 149 149 124 129 125 118 147 170 109 146 122 155 130 169 153 113 86 93 96 110 135 152 168 182 178 156 141 137 126 121 117 129 193 208 151 71 121 57 84 148 157 153 142  
154 123 128 126 121 119 122 89 94 91 115 122 116 139 132 133 138 140 146 138 135 150 153 165 149 115 137 149 158 179 172 183 150 106 129 156 183 146 142 123 133 145 14  
1 147 147 135 145 144 127 125 119 122 124 120 129 185 214 163 49 79 60 94 128 144 170 168 120 153 121 124 119 123 142 154 155 141 137 146 122 125 139 133 135 129 127 140 14  
2 144 160 161 178 170 132 154 126 150 170 175 128 71 53 48 58 93 162 145 134 106 114 109 109 120 114 136 125 118 141 116 133 117 125 136 199 203 163 105 44 57 87 120 14  
2 169 152 138 125 129 121 125 121 122 112 105 106 144 113 150 129 131 138 119 118 113 161 141 178 181 178 172 154 182 92 52 45 30 33 30 45 67 27 38 71 107 108 116 108 1  
11 117 114 118 119 120 122 125 128 127 124 127 136 197 219 157 54 73 81 79 129 141 139 193 166 118 137 120 121 119 120 121 119 120 98 99 105 147 102 136 121 118 131 112 117 117 148  
178 187 153 184 196 219 212 126 41 23 32 36 33 25 30 39 33 35 58 92 106 110 116 114 111 119 121 120 122 124 127 127 124 127 136 197 219 162 42 50 107 82 122 136 140 15  
9 196 132 130 127 136 119 118 97 101 92 138 111 126 110 116 124 97 92 121 124 171 193 160 174 155 222 236 166 68 38 37 32 32 35 30 46 27 27 35 46 84 112 117 111 115 126  
122 121 122 123 127 127 126 127 135 199 220 171 41 34 122 87 123 127 146 129 173 169 115 127 127 137 128 101 102 103 130 113 121 124 108 107 96 115 132 136 182 148 124  
114 112 171 234 212 108 26 36 27 32 35 27 41 28 28 30 23 51 80 101 111 114 116 115 122 122 124 126 128 127 130 136 198 219 172 50 31 99 92 127 123 131 152 150 176 131



# Images $\longleftrightarrow$ Vectors



60	60	255	255
60	60	255	255
60	60	255	255
128	128	128	128



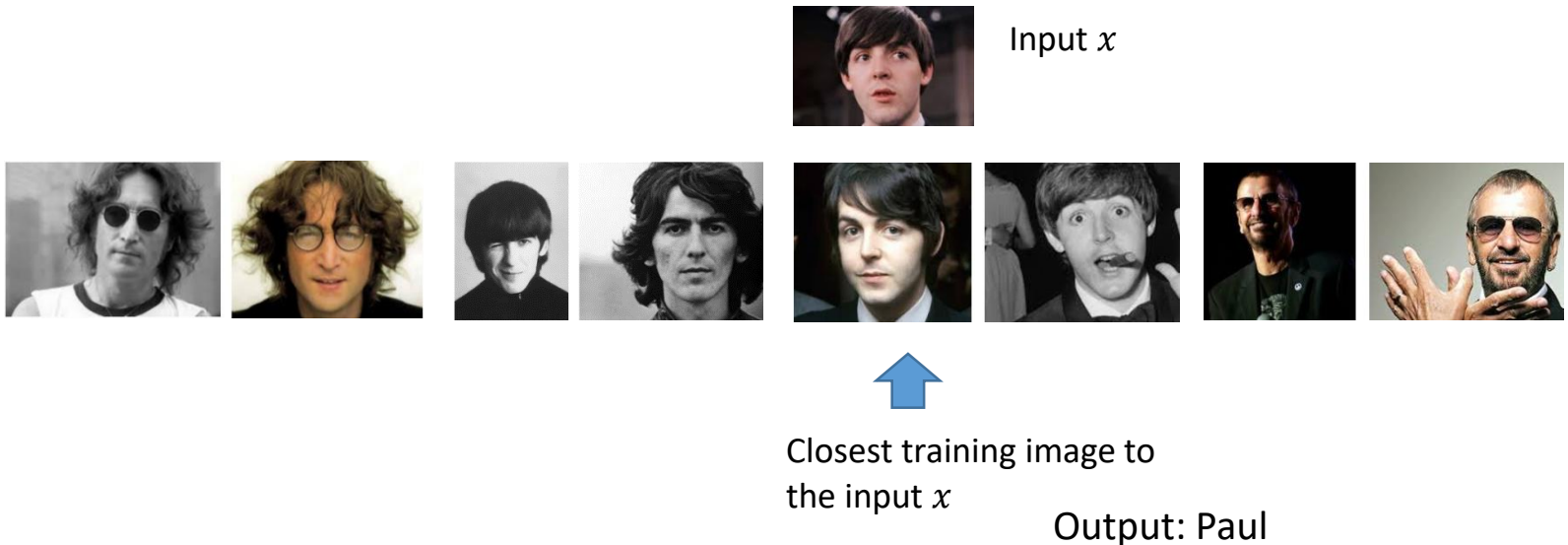
60
60
255
255
60
60
255
255
60
60
255
255
128
128
128
128

# The Face Recognition Task

- Training set:
  - $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)})\}$ 
    - $x^{(i)}$  is a  $k$ -dimensional vector consisting of the intensities of all the pixels in the  $i$ -th photo ( $20 \times 20$  photo  $\rightarrow x^{(i)}$  is 400-dimensional)
    - $y^{(i)}$  is the *label* (i.e., name)
- Test phase:
  - We have an input vector  $x$ , and want to assign a label  $y$  to it
    - Whose photo is it?

# Face Recognition using 1-Nearest Neighbors (1NN)

- Training set:  $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)})\}$
- Input:  $x$
- 1-Nearest Neighbor algorithm:
  - Find the training photo/vector  $x^{(i)}$  that's as “close” as possible to  $x$ , and output the label  $y^{(i)}$



# Are the two images $a$ and $b$ close?

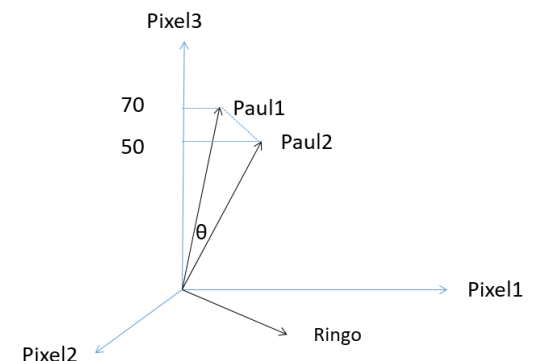
- Key idea: think of the images as *vectors*
  - Reminder: to turn an image into a vector, simply “flatten” all the pixels into a 1D vector
- Is the distance between the endpoints of vectors  $a$  and  $b$  small?

$$|a - b| = \sqrt{\sum_i (a_i - b_i)^2} \text{ small}$$

- Is the cosine of the angle between the vectors  $a$  and  $b$  large?

$$\cos \theta_{ab} = \frac{a \cdot b}{|a||b|} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2} \sqrt{\sum_i b_i^2}} \text{ large}$$

By the law of cosines



# k-Nearest Neighbour Classification

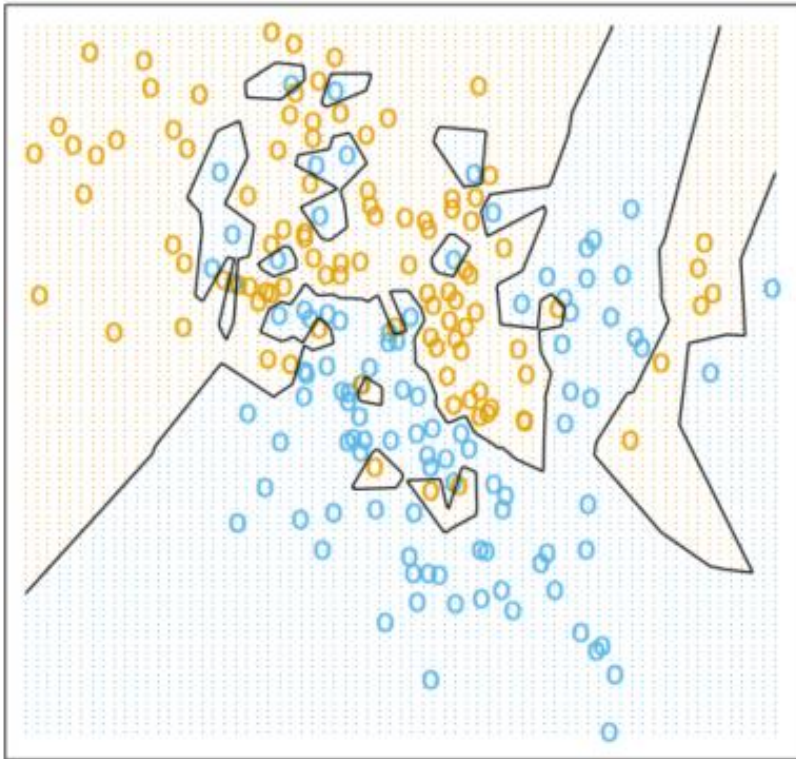
- For an example  $x$ 
  - Find the  $k$  closest examples (neighbours) to  $x$  in the training set
  - Output the plurality label for the  $k$  closest examples
- Can use various distance functions:
  - Euclidian (L2):  $\text{dist}(a, b) = \sqrt{\sum_i (a_i - b_i)^2}$  (default)
  - L-infinity:  $\text{dist}(a, b) = \max_i |a_i - b_i|$
  - L-zero:  $\text{dist}(a, b) = \#\{a_i \neq b_i\}$
  - Negative cosine:  $\text{dist}(a, b) = -\frac{a \cdot b}{|a||b|}$

# How do we determine K?

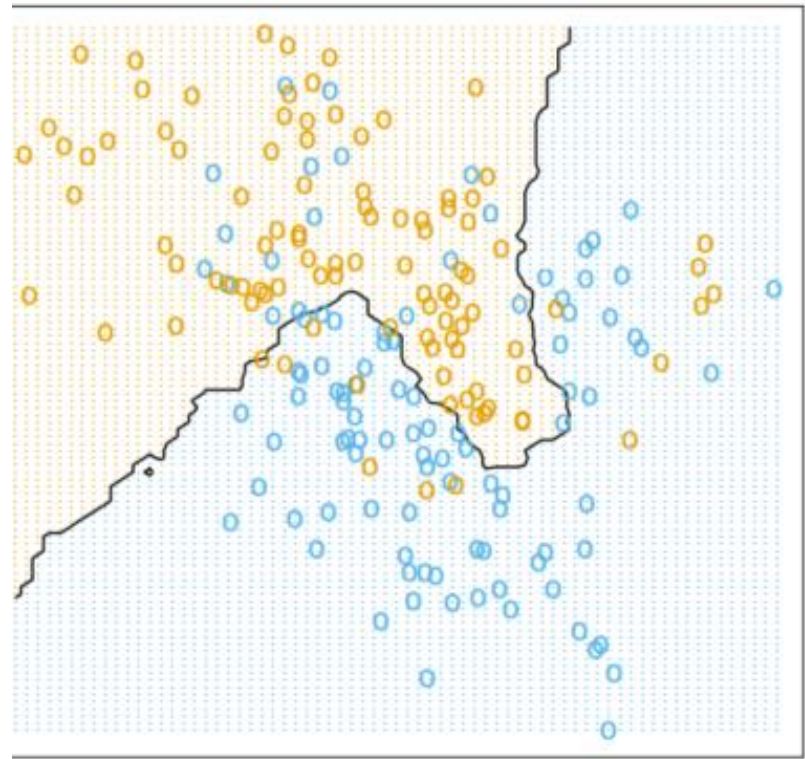
- Try different values, and see which works best on the test set?
  - Could do that, but then we are selecting the best K for our particular test set. This means that the performance on our test set is now an overestimate of how well we'd do on *new data*
- Solution: set aside a *validation set* (which is separate from both the training and the test set), and select the K for the best performance on the validation set, but report the results on the test set
  - Generally, the performance on the validation set will be better than on the test set
  - What about the performance on the *training set*?

# What does the best K say about the data?

1-Nearest Neighbor Classifier



15-Nearest Neighbor Classifier



Large  $k$ : relatively simple boundary, no small “islands” in the data. Small changes in  $x$  do not generally change the label

Small  $k$ : a complex boundary between the labels. Small changes in  $x$  often change the labels

# Why not let $K$ be very small?

- Great for the performance on the training set!
  - Perfect performance guaranteed for  $k = 1$
- If the test data does not look exactly like the training data, the performance on the test data will be worse for  $k$  that is too small
  - The training data could be noisy (e.g., in the orange region, data points are sometimes blue with probability 5%, randomly)
  - This is an example of *overfitting* – building a classifier that works well on the training set, but does not generalize well to the test set



Why not let  $K$  be very large?

# Distance Functions

- For images, why might the cosine distance make sense?
- For images, why might the Euclidean distance make sense?

Training set of housing prices (Portland, OR)	Size in feet <sup>2</sup> ( $x$ )	Price (\$) in 1000's ( $y$ )
	2104	460
	1416	232
	1534	315
	852	178
	...	...

Notation:

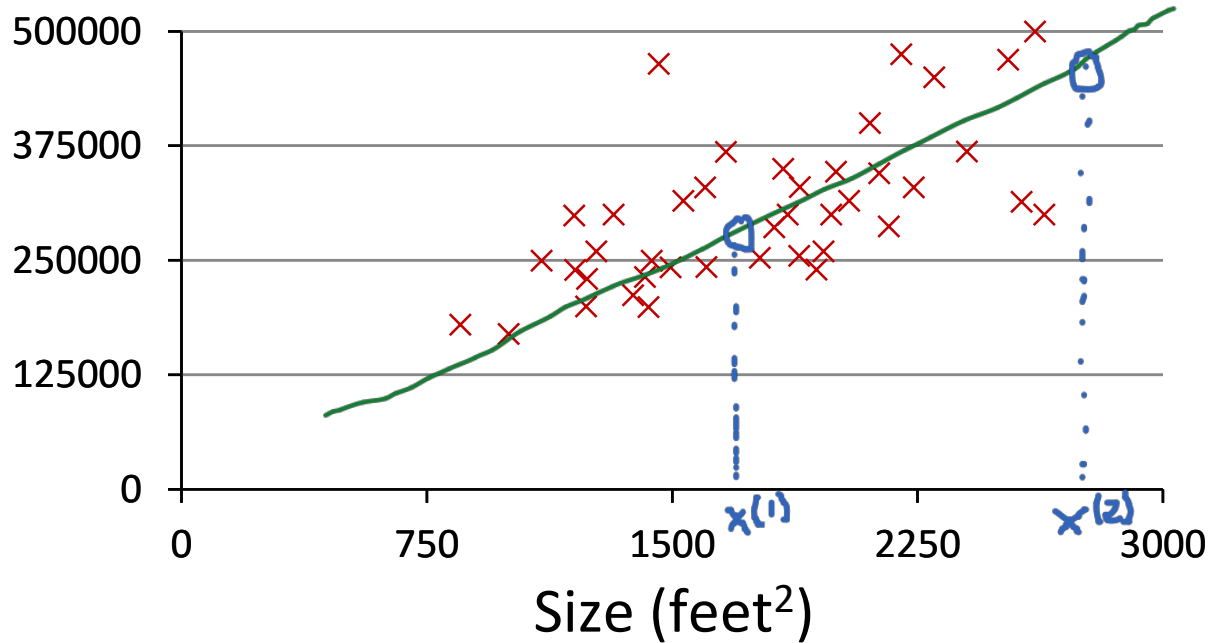
**m** = Number of training examples

**x**'s = “input” variable / features

**y**'s = “output” variable / “target” variable

# Housing Prices (Portland, OR)

Price  
(in 1000s of  
dollars)

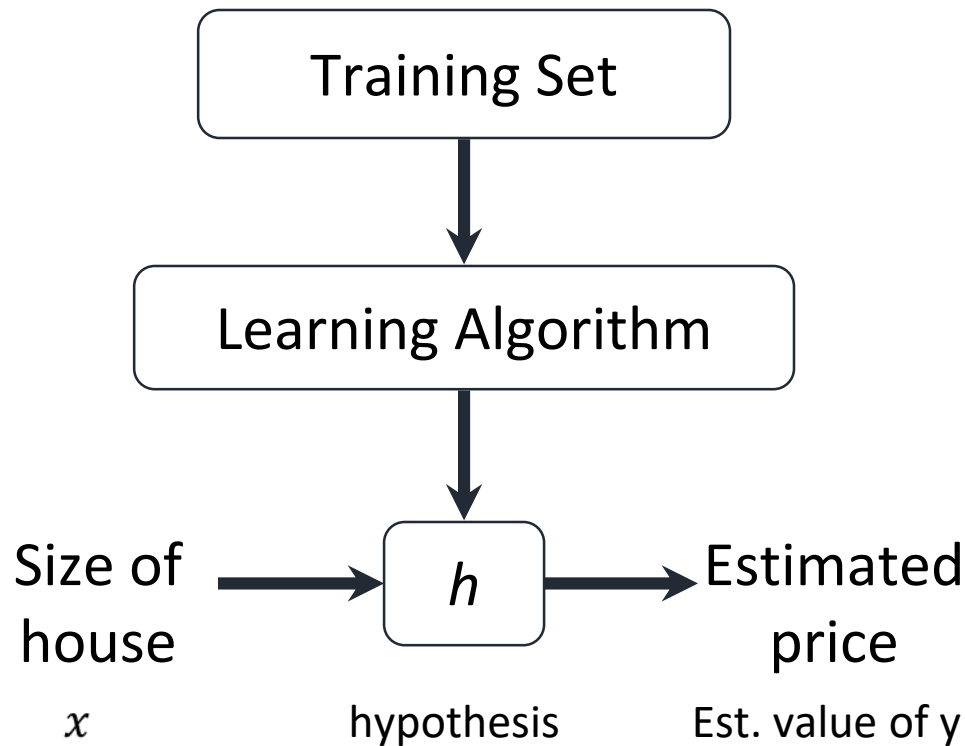


## Supervised Learning

Given the “right answer” for each example in the data.

## Regression Problem

Predict real-valued output



H maps x's to y's.

## How do we represent $h$ ?

- We represent hypotheses about the data using the parameters  $\theta = (\theta_0, \theta_1)$
- If the data is correctly predicted according to hypothesis  $h_\theta$ , then  $y \approx h_\theta(x) = \theta_0 + \theta_1 x$
- The learning algorithm finds the best hypothesis  $h_\theta$  for the training set
- We can then estimate the values of  $y$  for the test set using that  $h_\theta$
- If  $h_\theta(x)$  is a linear function of a real number  $x$ , this procedure is called linear regression.

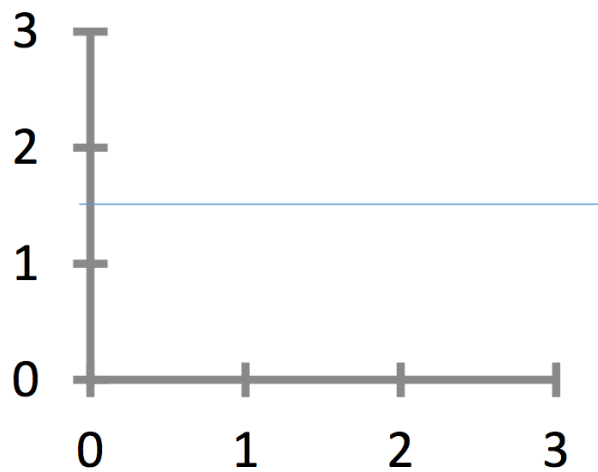
Training Set	Size in feet <sup>2</sup> (x)	Price (\$) in 1000's (y)
	2104	460
	1416	232
	1534	315
	852	178
	...	...

Hypothesis:  $h_{\theta}(x) = \theta_0 + \theta_1 x$

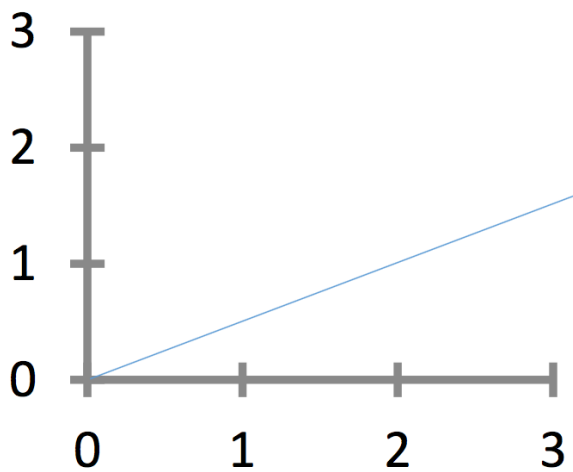
$\theta_i$ 's: Parameters

How to choose  $\theta_i$ 's ?

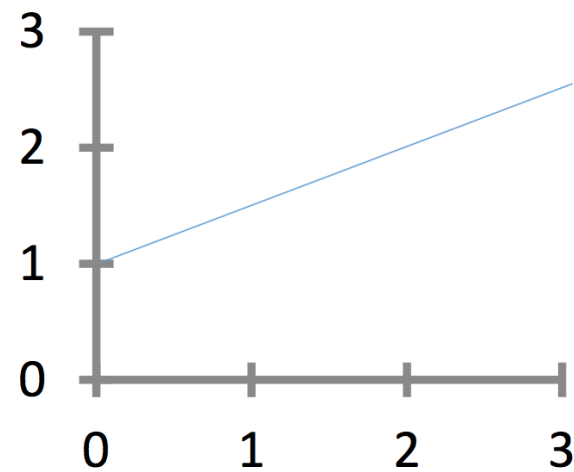
$$h_{\theta}(x) = \theta_0 + \theta_1 x$$



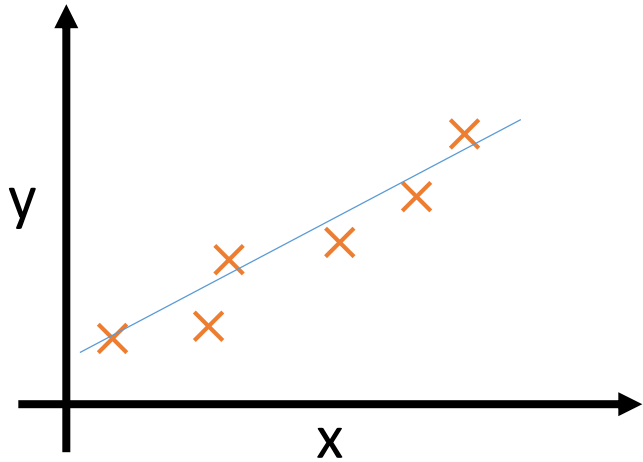
$$\theta_0 = 1.5$$
$$\theta_1 = 0$$



$$\theta_0 = 0$$
$$\theta_1 = 0.5$$



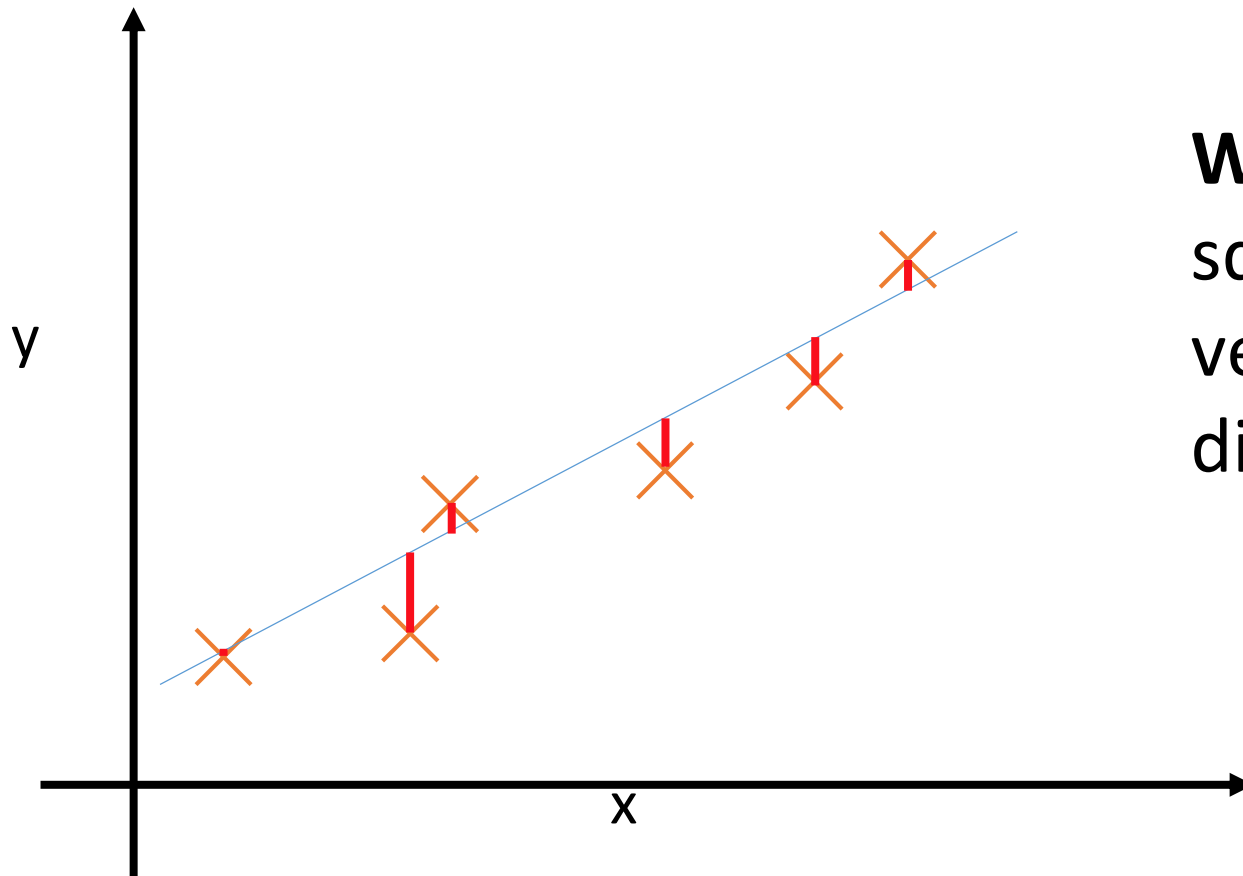
$$\theta_0 = 1$$
$$\theta_1 = 0.5$$



**But what does  
“close” mean?**

Idea: Choose  $\theta_0, \theta_1$  so that  
 $h_{\theta}(x)$  is close to  $y$  for our  
training examples  $(x, y)$





**We choose:**  
squared  
vertical  
distance

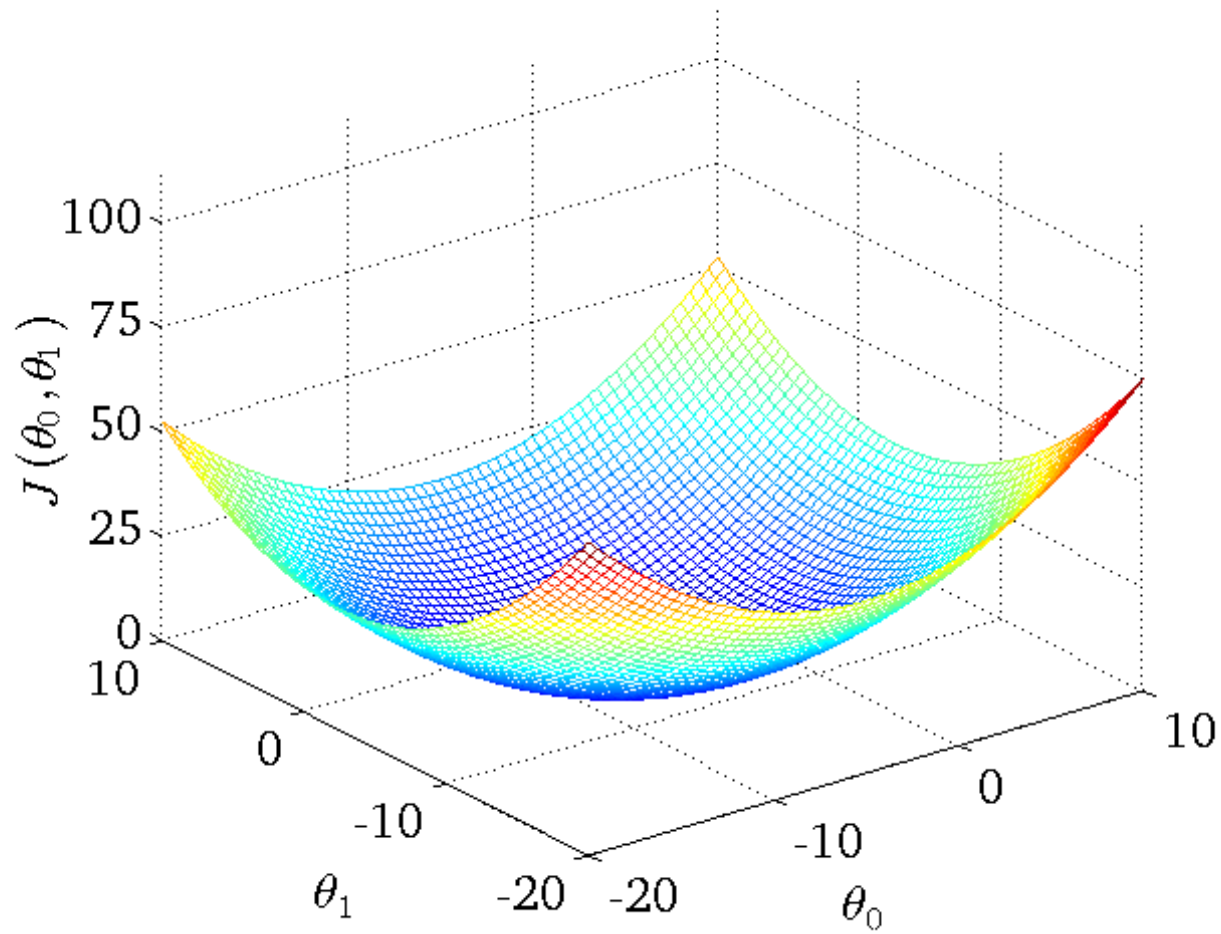
Hypothesis:  $h_{\theta}(x) = \theta_0 + \theta_1 x$

Parameters:  $\theta_0, \theta_1$

Cost Function:  $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$

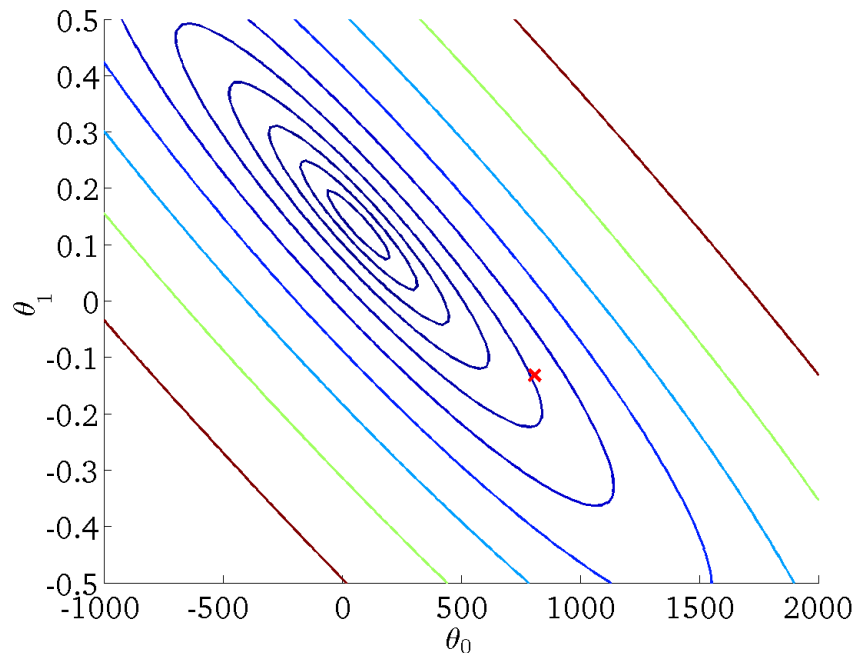
Goal:  $\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1)$

# Cost Function Surface Plot



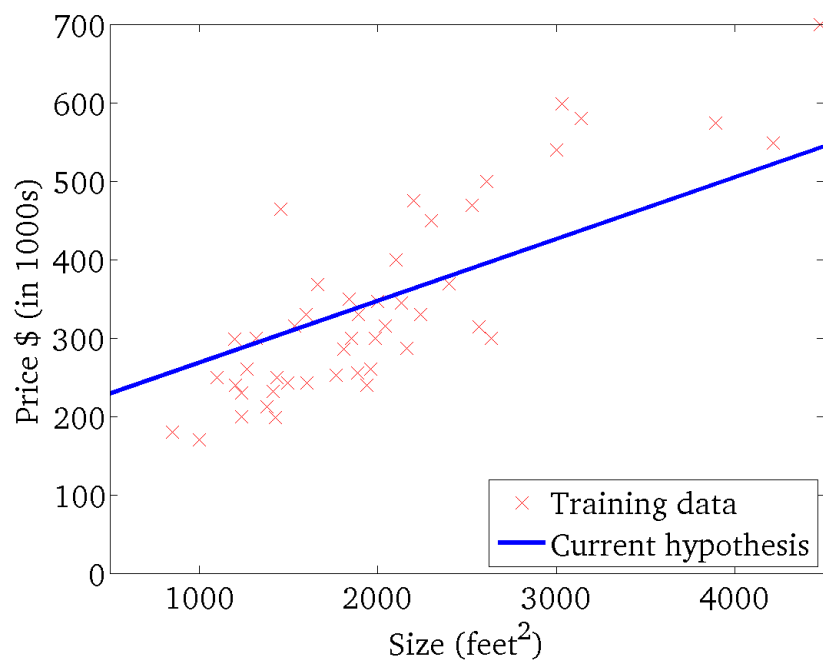
# Contour Plots

- For a function  $F(x, y)$  of two variables, assigned different colours to different values of  $F$
- Pick some values to plot
- The result will be *contours* – curves in the graph along which the values of  $F(x, y)$  are constant



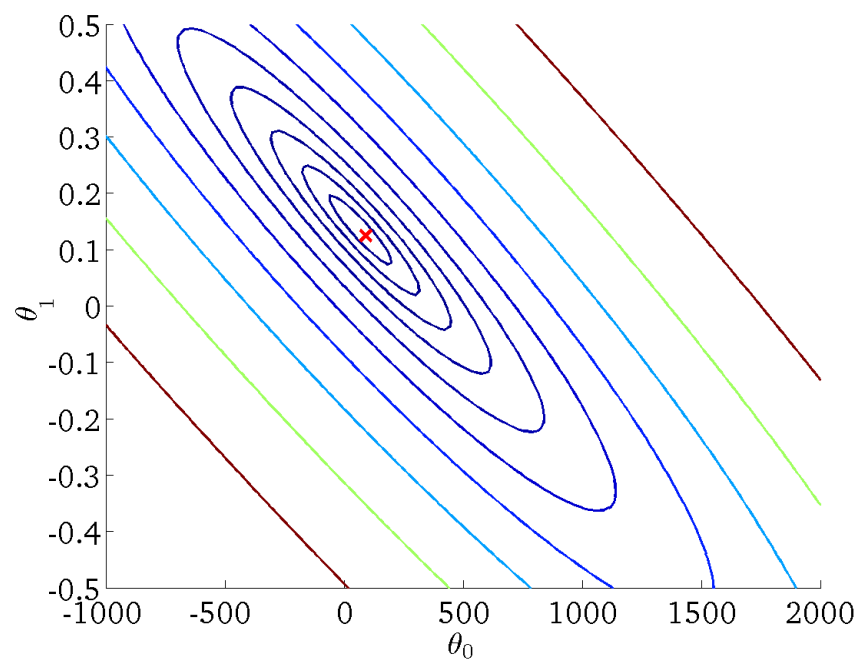
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$  this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

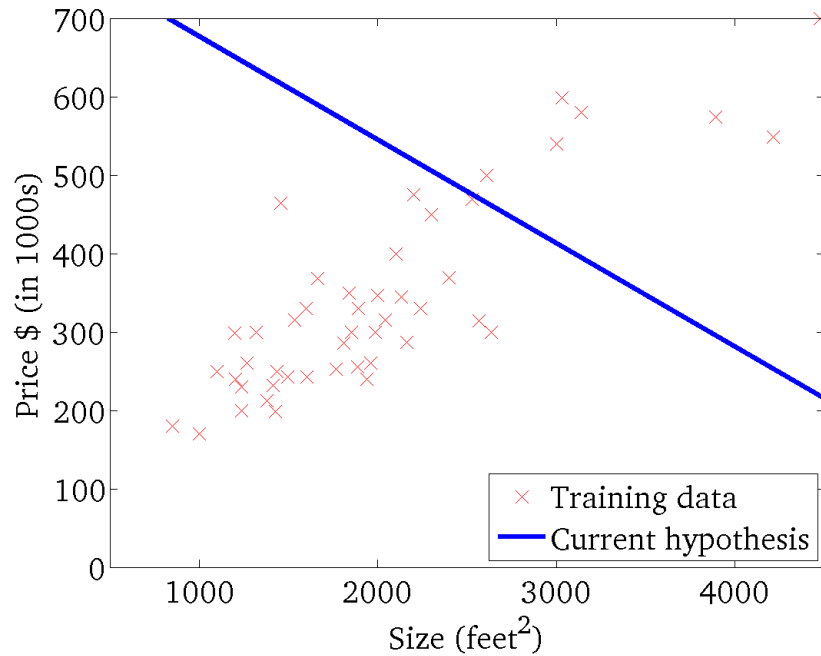
(function of the parameters  $\theta_0, \theta_1$ )



# Cost Function Contour Plot

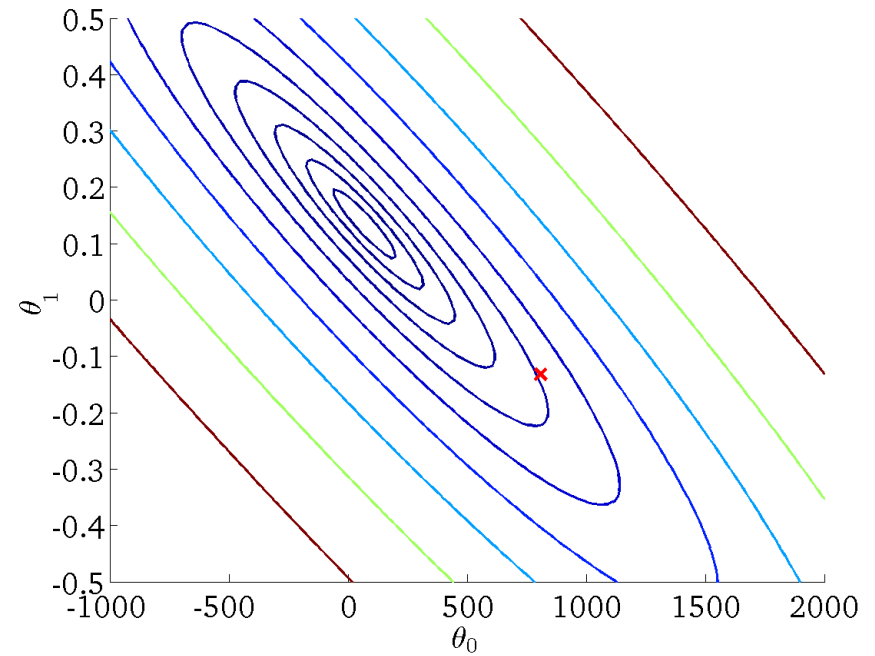
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$  this is a function of  $x$ )



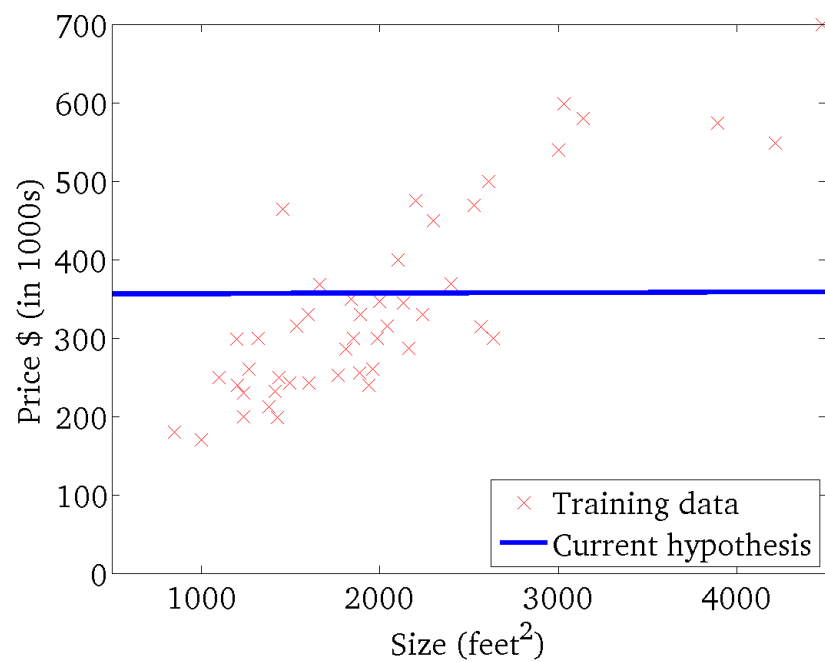
$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )



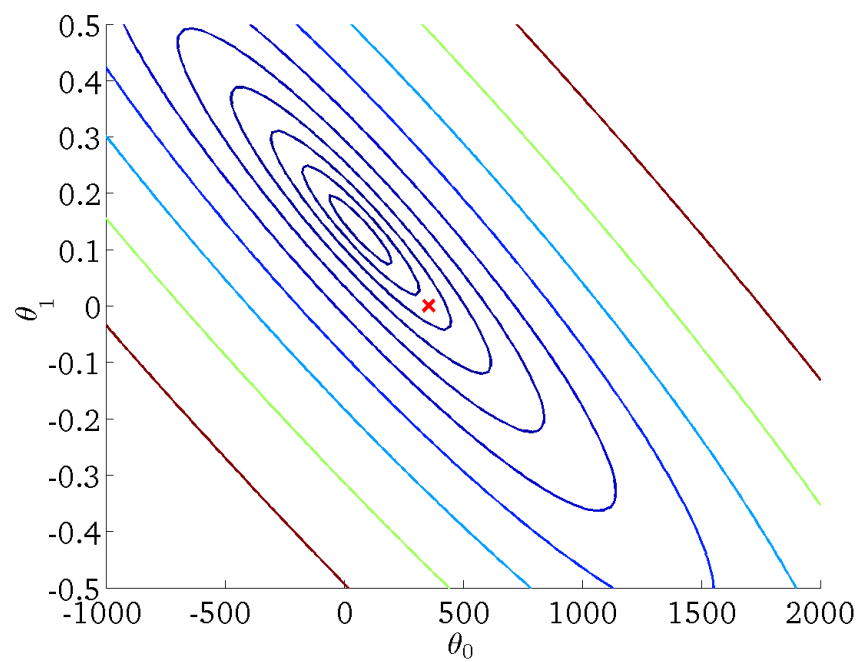
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$  this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )



Have some function  $J(\theta_0, \theta_1)$

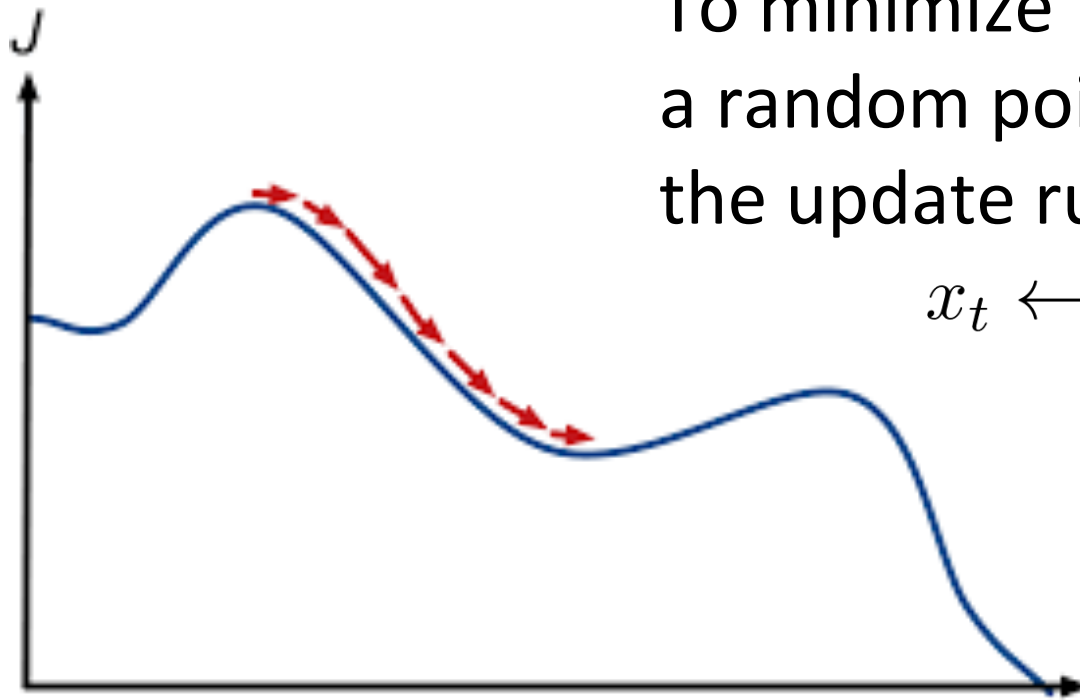
Want  $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

## Outline:

- Start with some  $\theta_0, \theta_1$
- Keep changing  $\theta_0, \theta_1$  to reduce  $J(\theta_0, \theta_1)$   
until we hopefully end up at a minimum



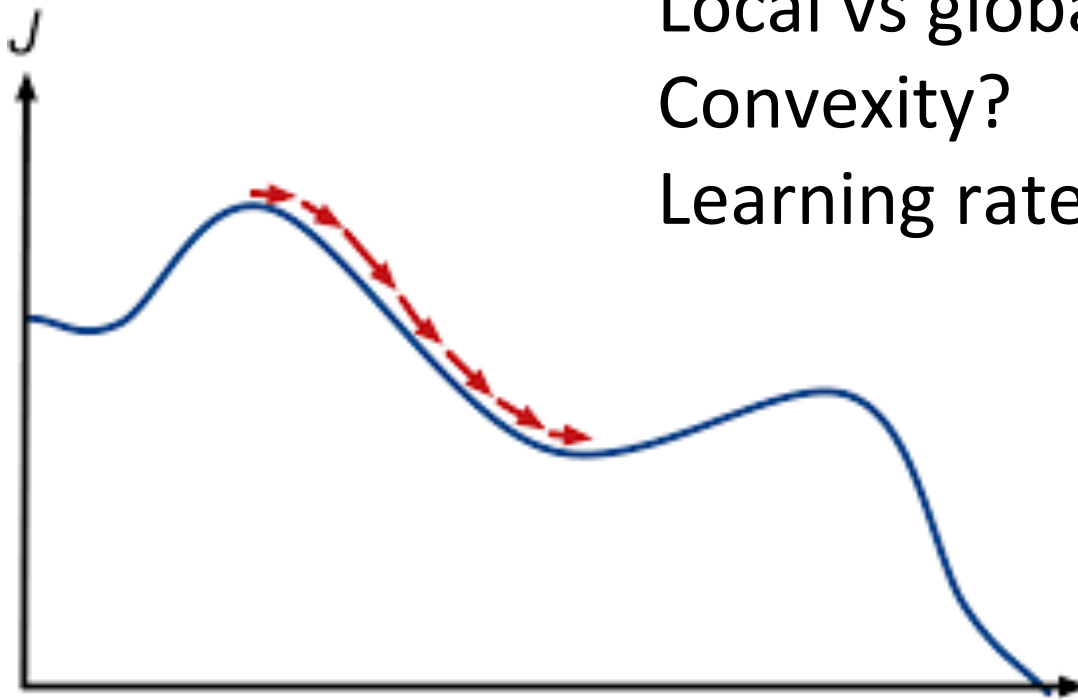
# Gradient Descent in 1D



To minimize  $f(x)$ , we start with a random point and iterate with the update rule:

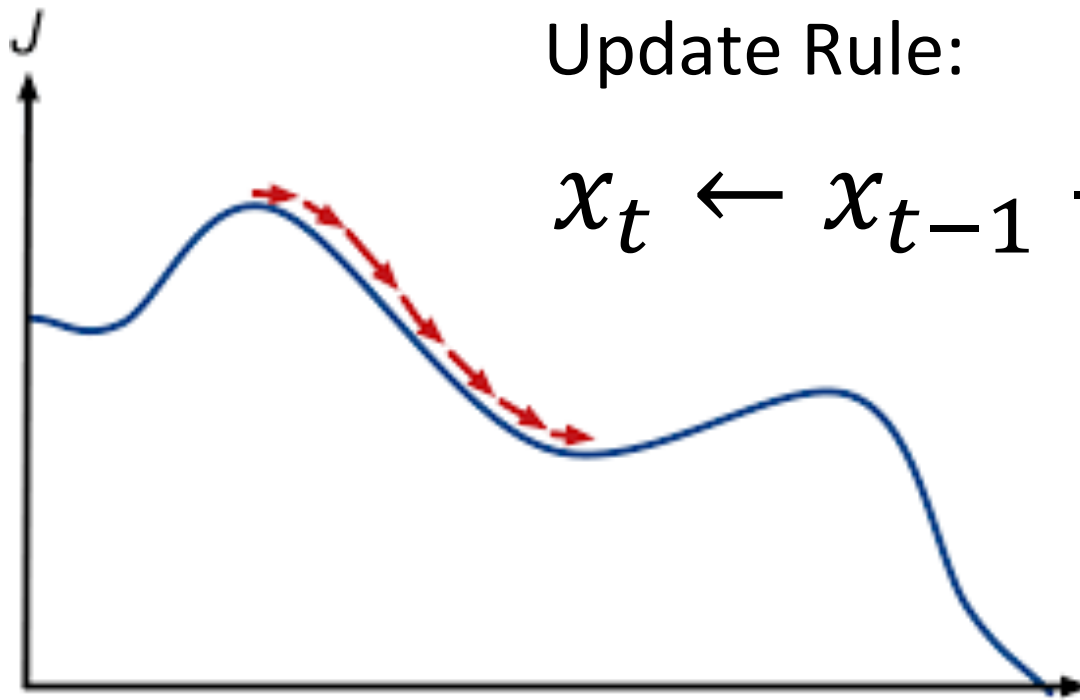
$$x_t \leftarrow x_{t-1} - \alpha \frac{df}{dx}(x_{t-1})$$

# Things to consider:



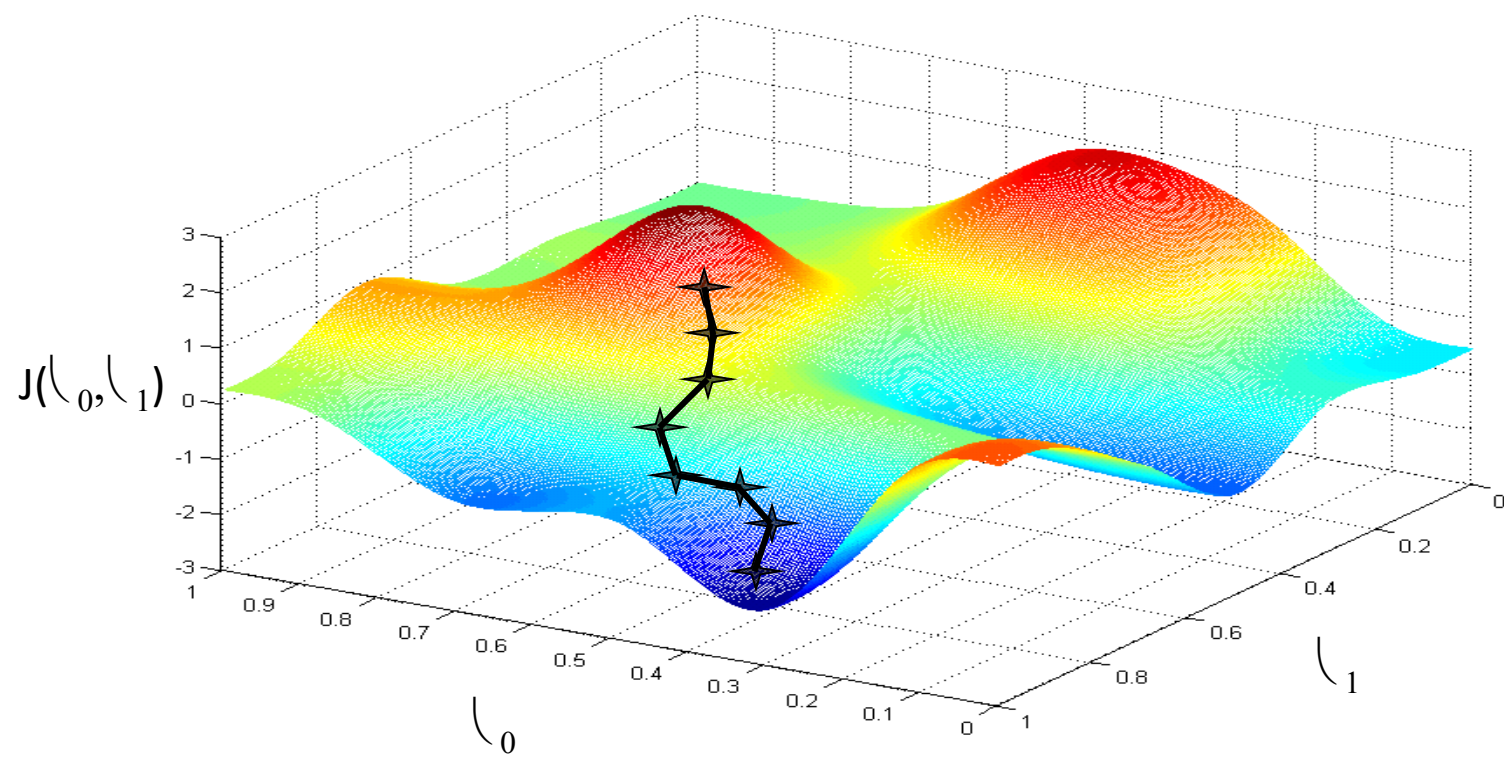
Local vs global minima?  
Convexity?  
Learning rate? ( $\alpha$ )

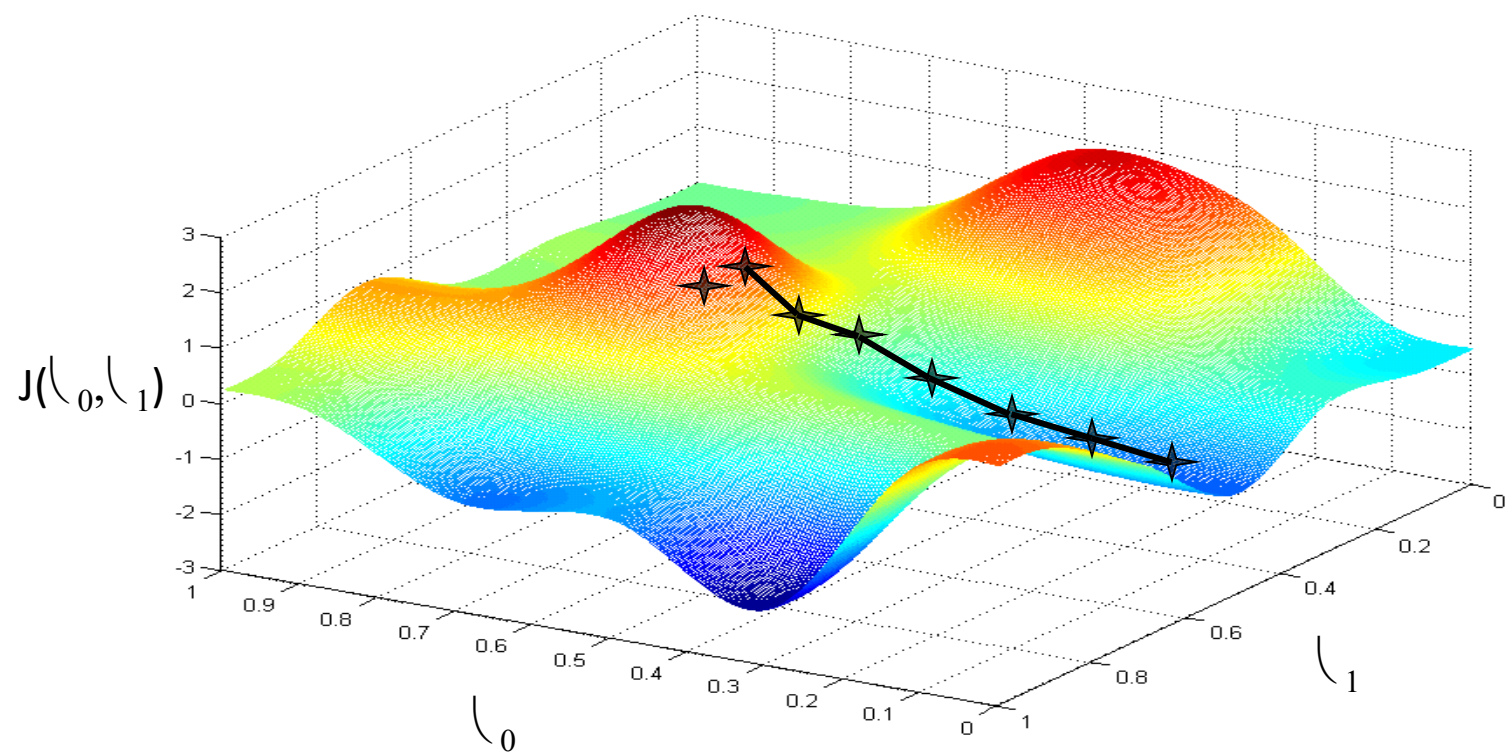
# Gradient Descent in Higher Dimensions



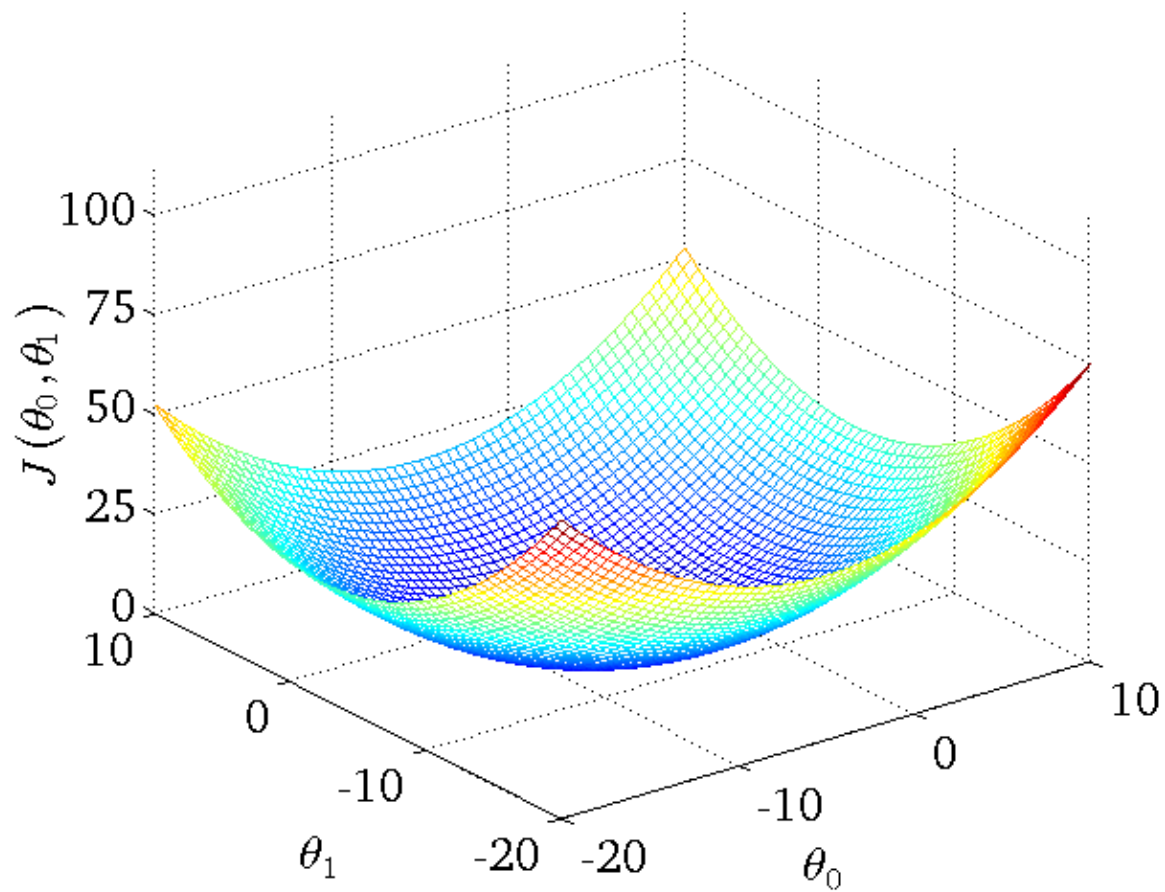
Update Rule:

$$x_t \leftarrow x_{t-1} - \alpha \nabla f(x_{t-1})$$





For Linear Regression,  $J$  is bowl-shaped (“convex”)



# Gradient Descent Example

Hypothesis:  $h_{\theta}(x) = \theta_0 + \theta_1 x$

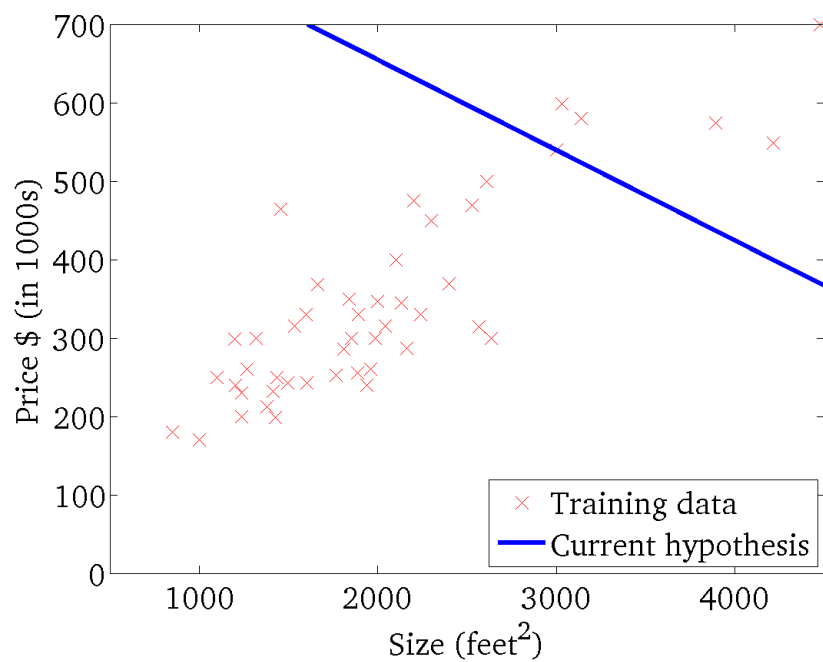
Parameters:  $\theta_0, \theta_1$

Cost Function:  $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$

Goal:  $\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1)$

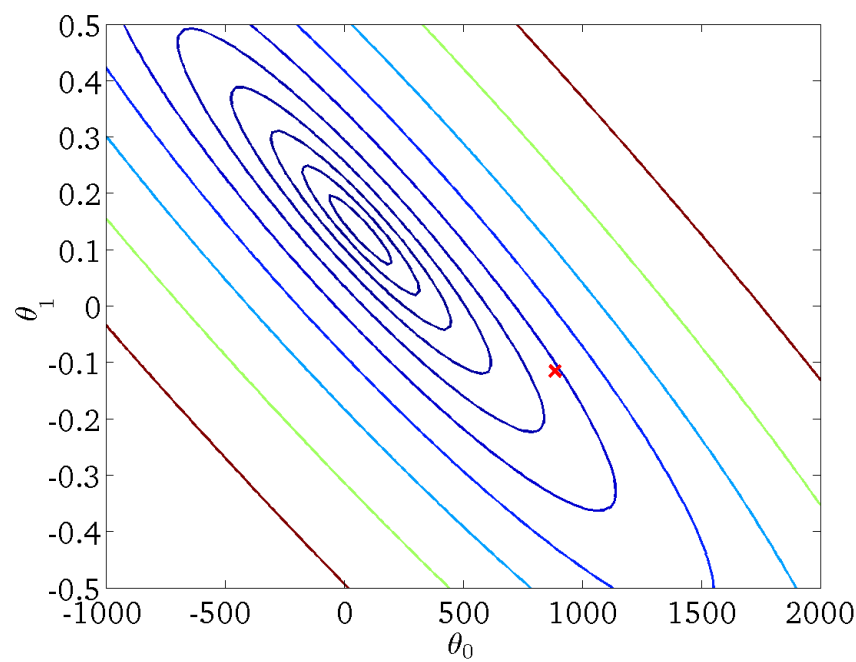
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$  this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

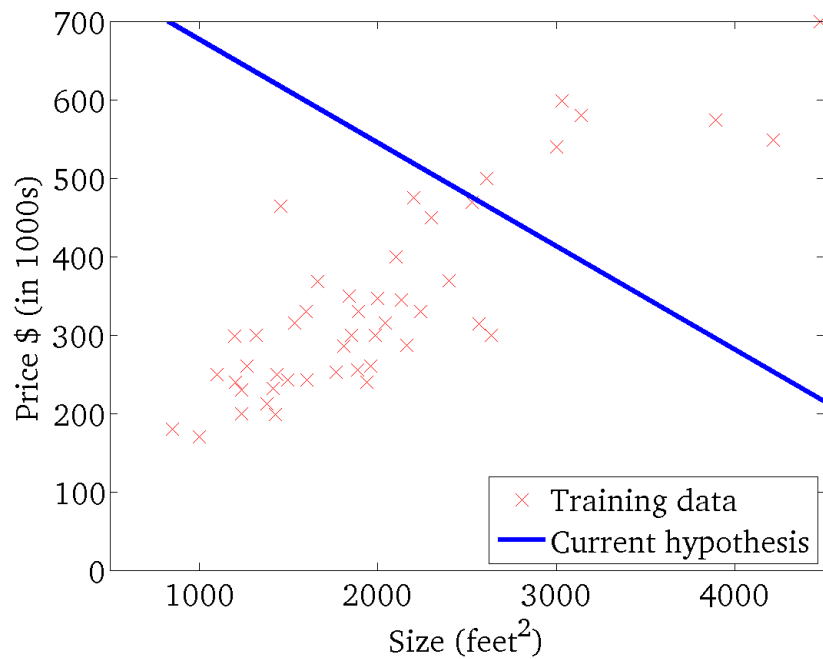
(function of the parameters  $\theta_0, \theta_1$ )





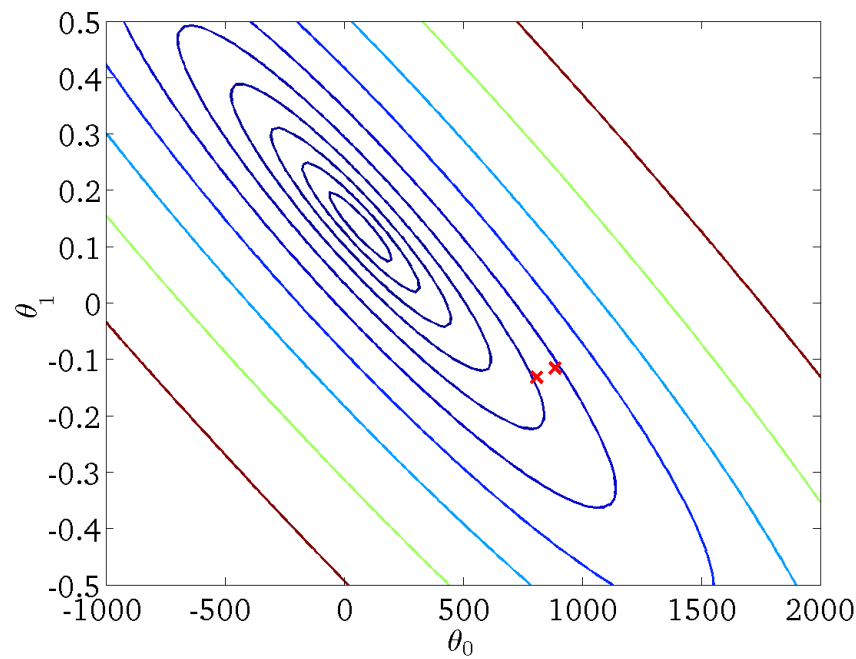
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$  this is a function of  $x$ )



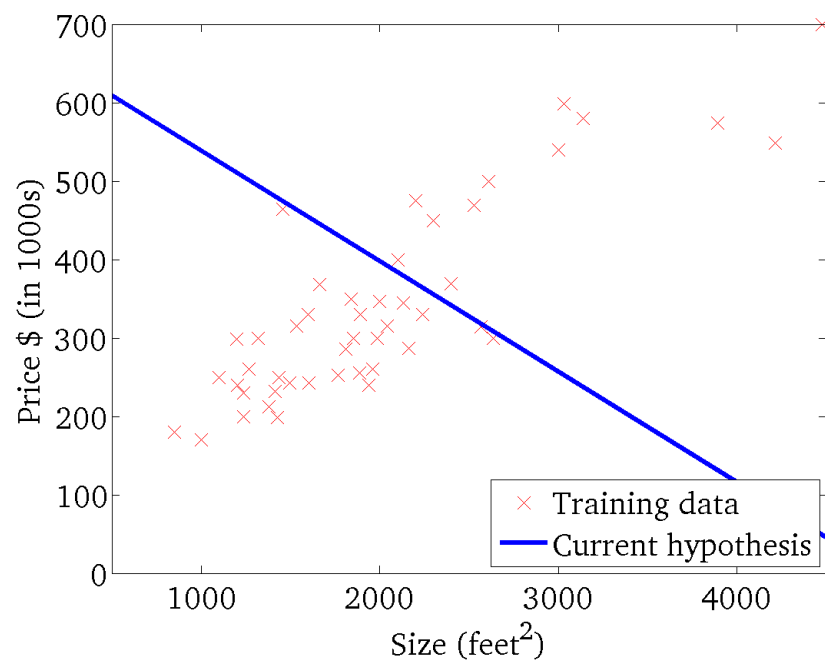
$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )



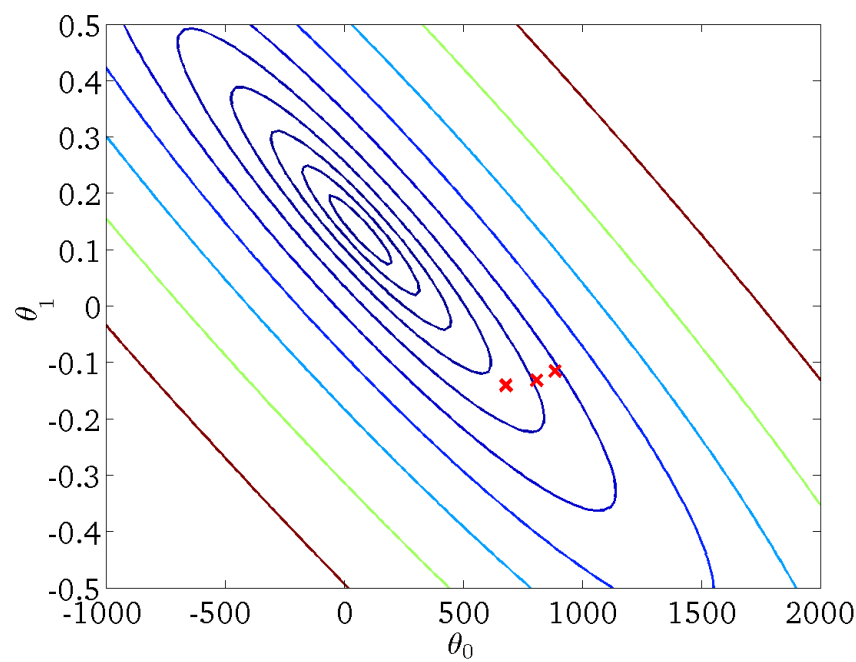
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$  this is a function of  $x$ )



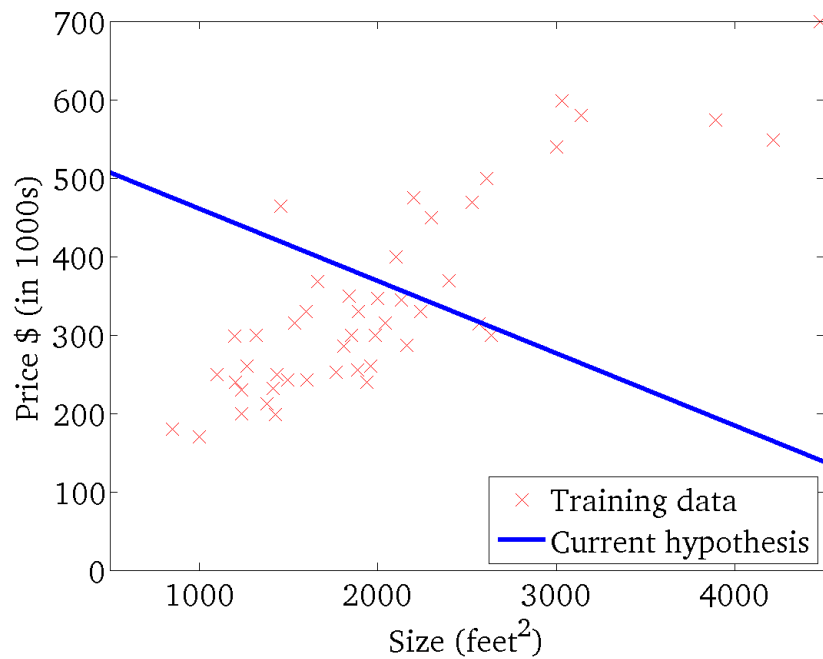
$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )



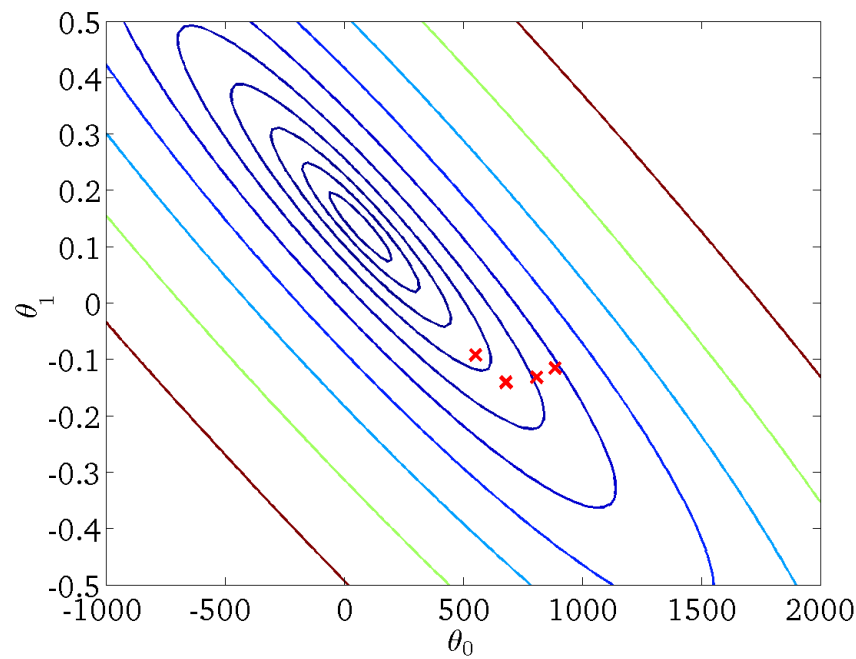
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$  this is a function of  $x$ )



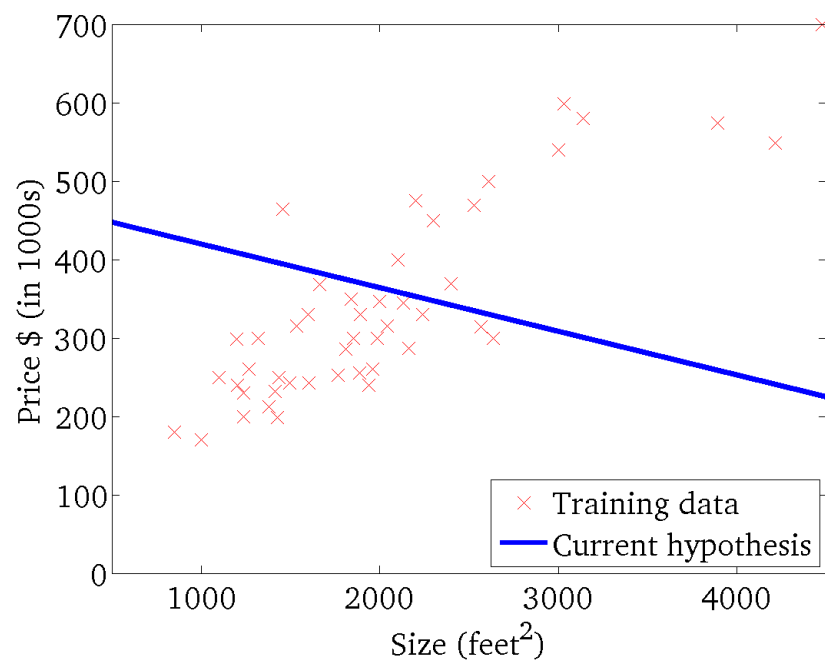
$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )



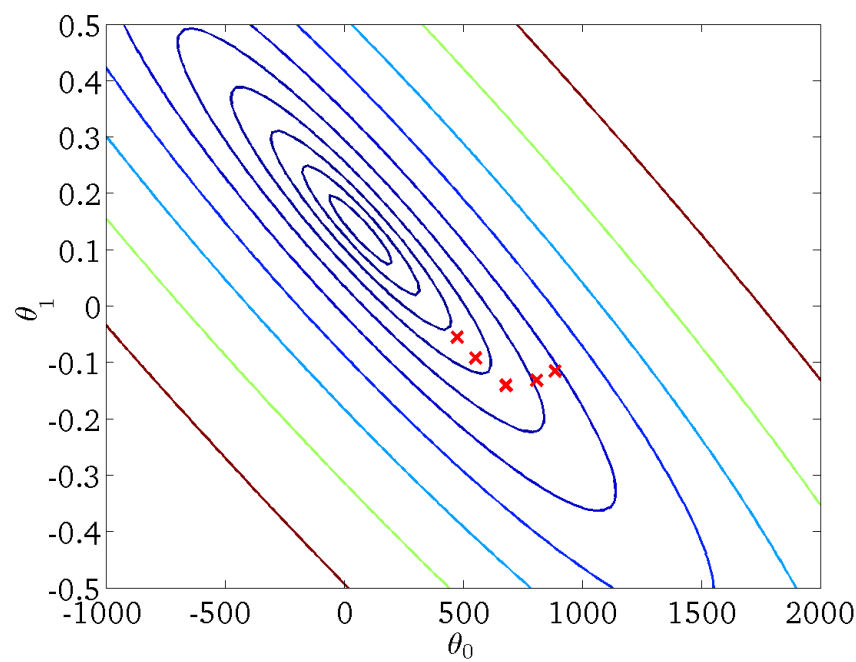
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$  this is a function of  $x$ )



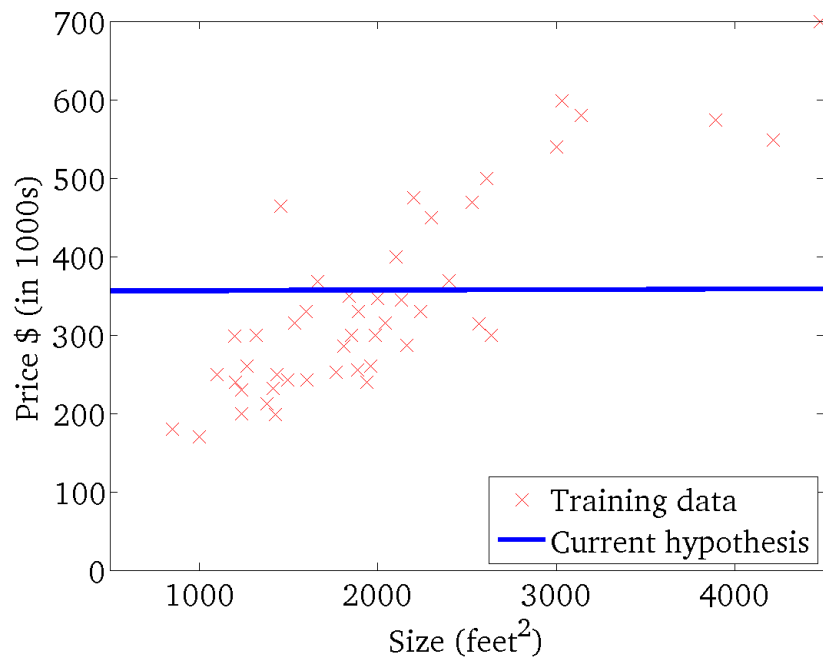
$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )



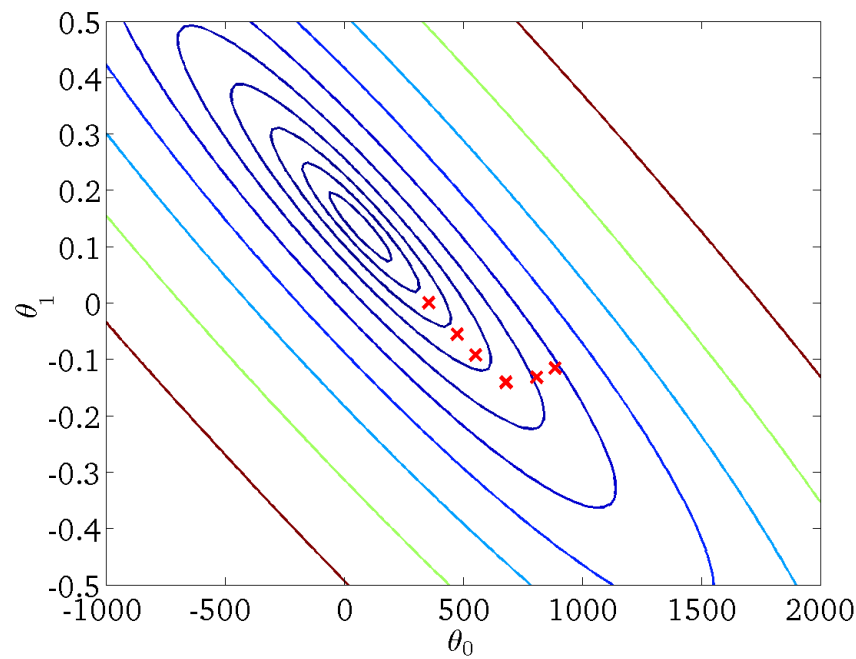
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$  this is a function of  $x$ )



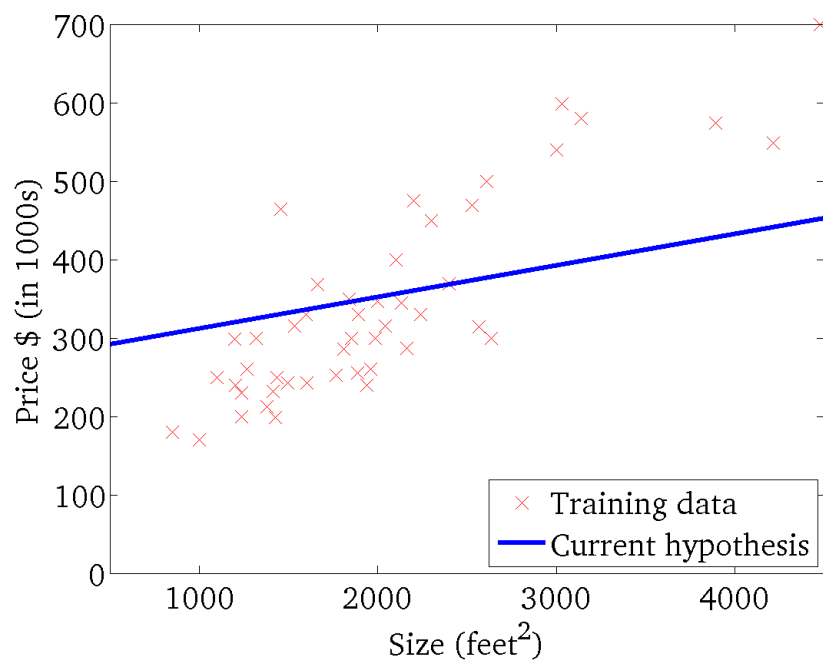
$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )



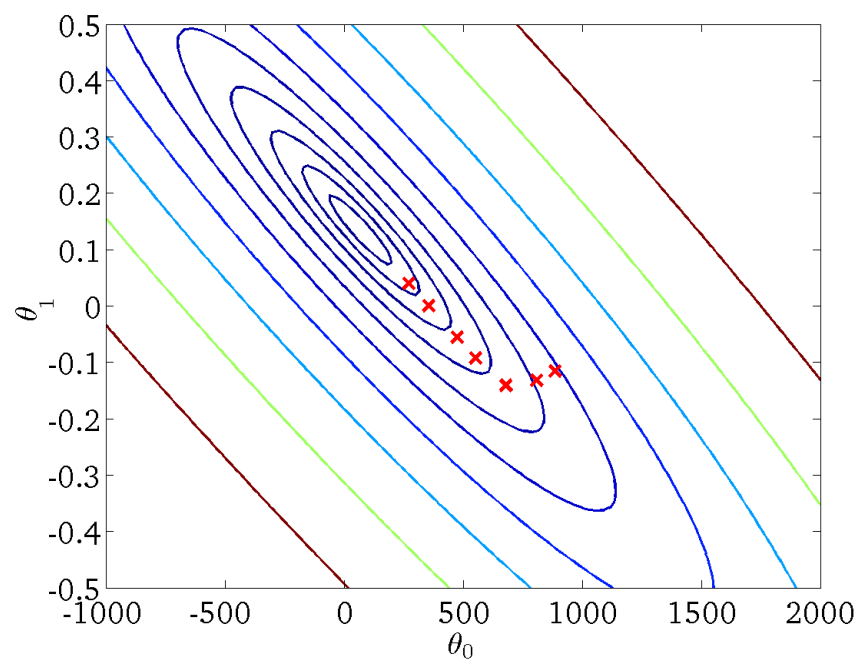
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$  this is a function of  $x$ )



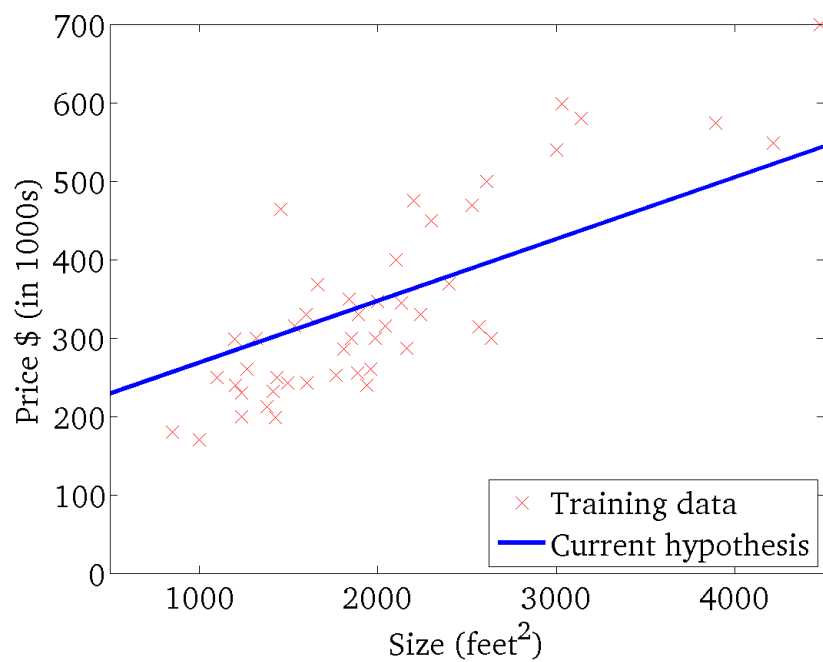
$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )



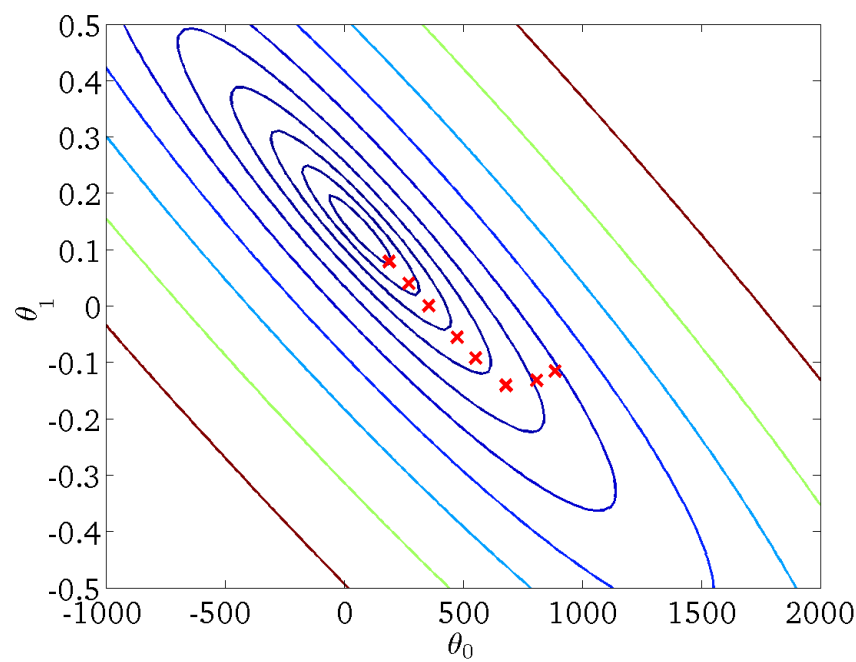
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$  this is a function of  $x$ )



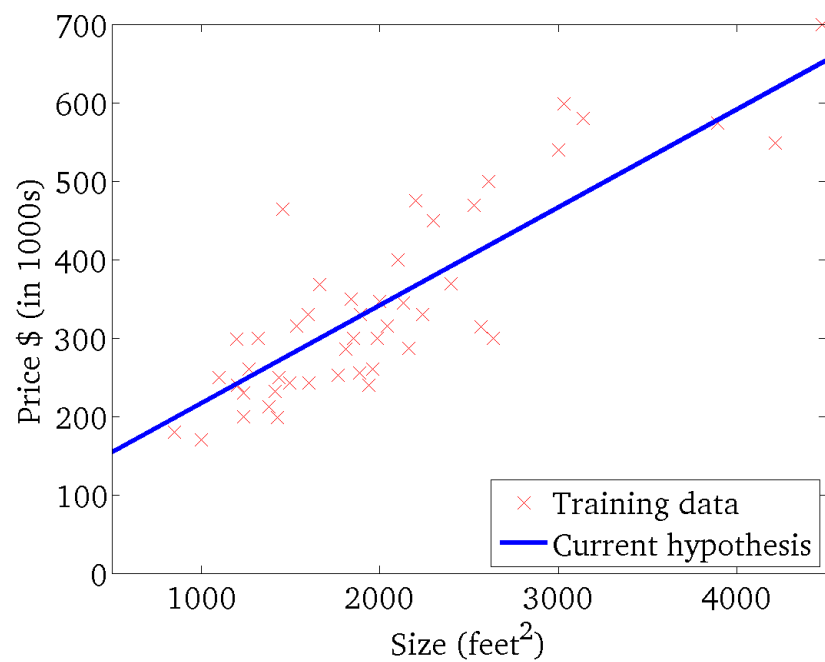
$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )



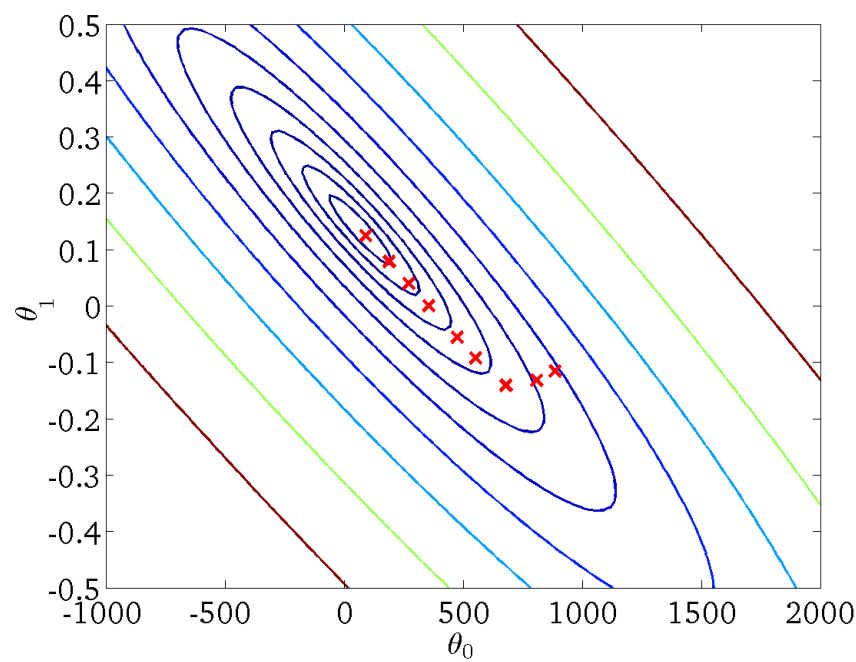
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$  this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

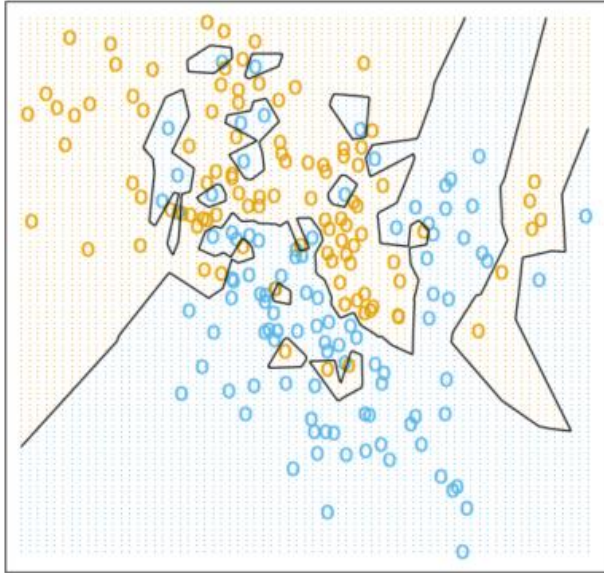
(function of the parameters  $\theta_0, \theta_1$ )



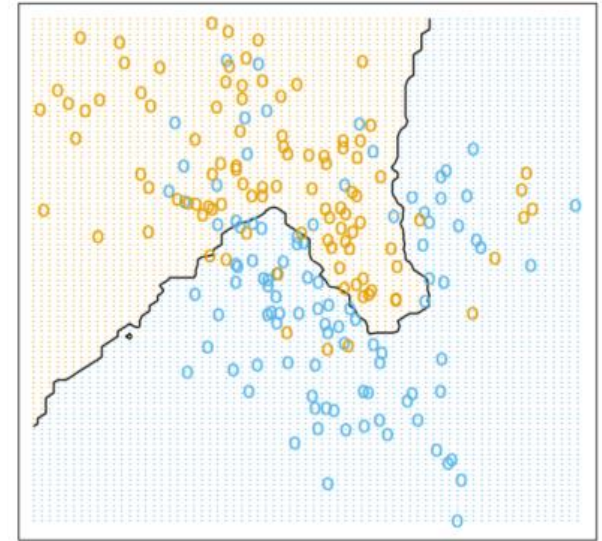


# Linear Regression vs. k-Nearest Neighbours

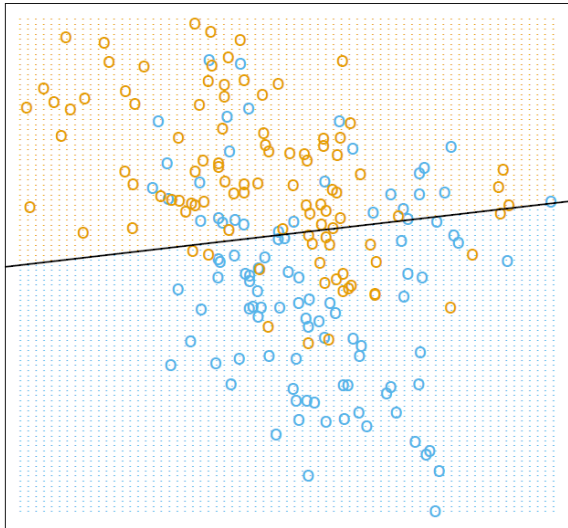
1-Nearest Neighbor Classifier



15-Nearest Neighbor Classifier



Linear Regression of 0/1 Response



Orange:  $y = 1$   
Blue:  $y = 0$

# Linear Regression vs. k-Nearest Neighbours

- Linear Regression: the boundary can only be linear
- Nearest Neighbours: the boundary can more complex
- Which is better?
  - Depends on what the *actual boundary* looks like
  - Depends on whether we have enough data to figure out the *correct* complex boundary