Basic Image Processing Algorithms

PPKE-ITK

Lecture 12.

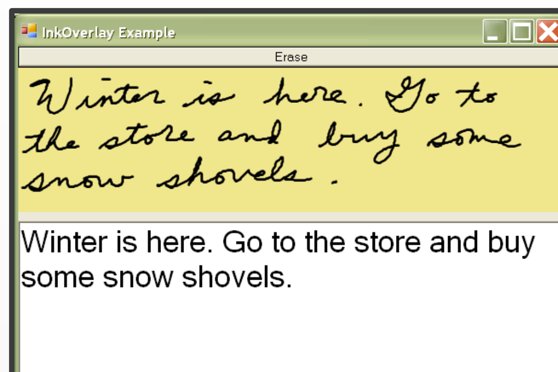
Introduction to Machine Learning

What is Machine Learning?

- Face detection

- E-mail spam filter
- Page ranking in Google search
- Road sign recognition in cars

- Advertisements on web pages and other recommendation systems
- Handwriting recognition



- Credit card fraud detection

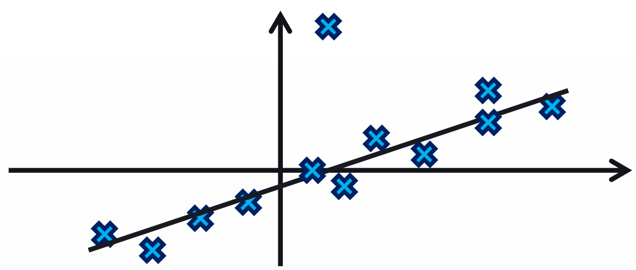
What is Machine Learning?

- Arthur Samuel (1959): "Field of study that gives computers the ability to learn without being explicitly programmed".

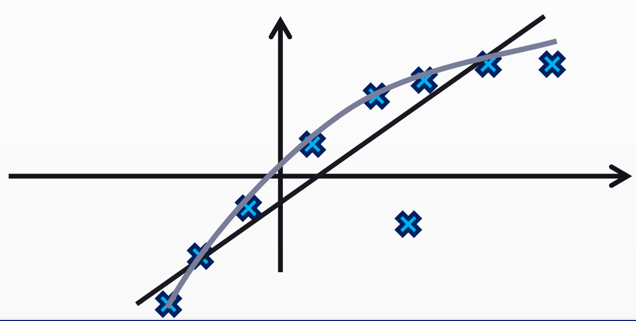
- Tom M. Mitchell: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E".

Machine Learning Algorithms

- Supervised Learning:
  - The supervised algorithms are trained on labeled data, where the desired output is known. The goal is to train a classifier that can work on previously unknown data.
    - **Regression**: prediction of continuous valued output



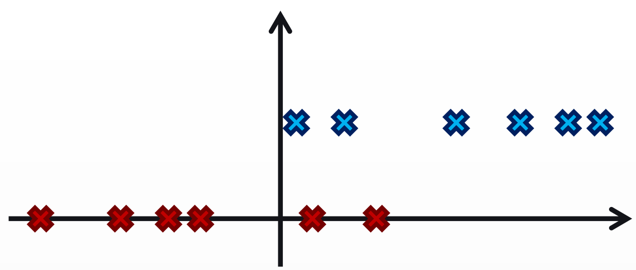
Y: Stock prices
X: Company sells data



What is the right model?

Machine Learning Algorithms

- Supervised Learning:
  - The supervised algorithms are trained on labeled data, where the desired output is known. The goal is to train a classifier that can work on previously unknown data.
    - **Classification**: prediction of discrete valued output



Y: Spam/Not Spam
X: frequency of a certain word
or
Y: Face/Not Face
X: Haar-feature

We can represent the problem in a different way

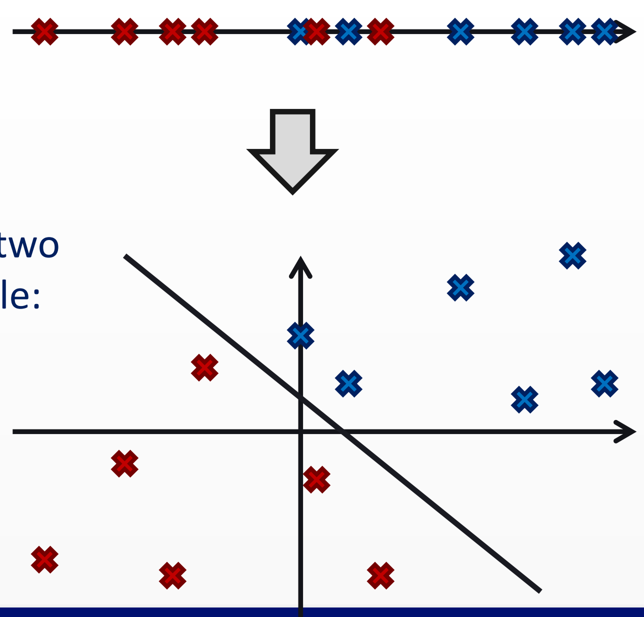
Based on only one feature we cannot make a good decision

Machine Learning Algorithms

- Supervised Learning:
  - The supervised algorithms are trained on labeled data, where the desired output is known. The goal is to train a classifier that can work on previously unknown data.
    - **Classification**: prediction of discrete valued output

In 1D the problem
is not separable:

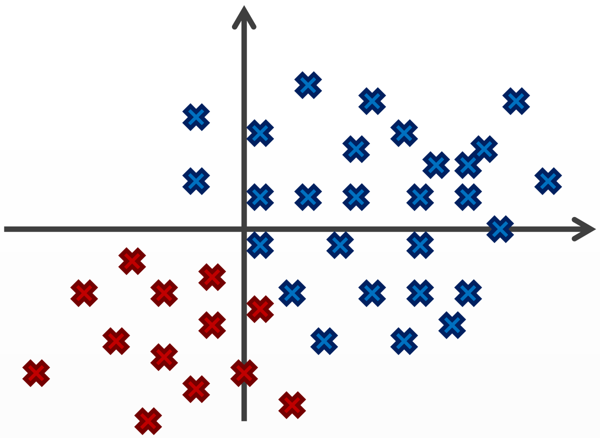
By adding a new feature, the two
classes are becoming separable:



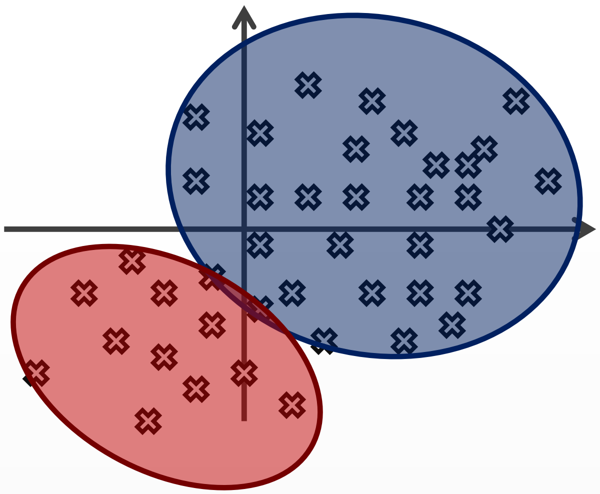
Spam/Not Spam
X: frequency of a certain word
Y: frequency of another word
or
Face/Not Face
X: Haar-feature
Y: Haar-feature II.

Machine Learning Algorithms

- Unsupervised Learning
  - In case of unsupervised learning the training data is not labeled.



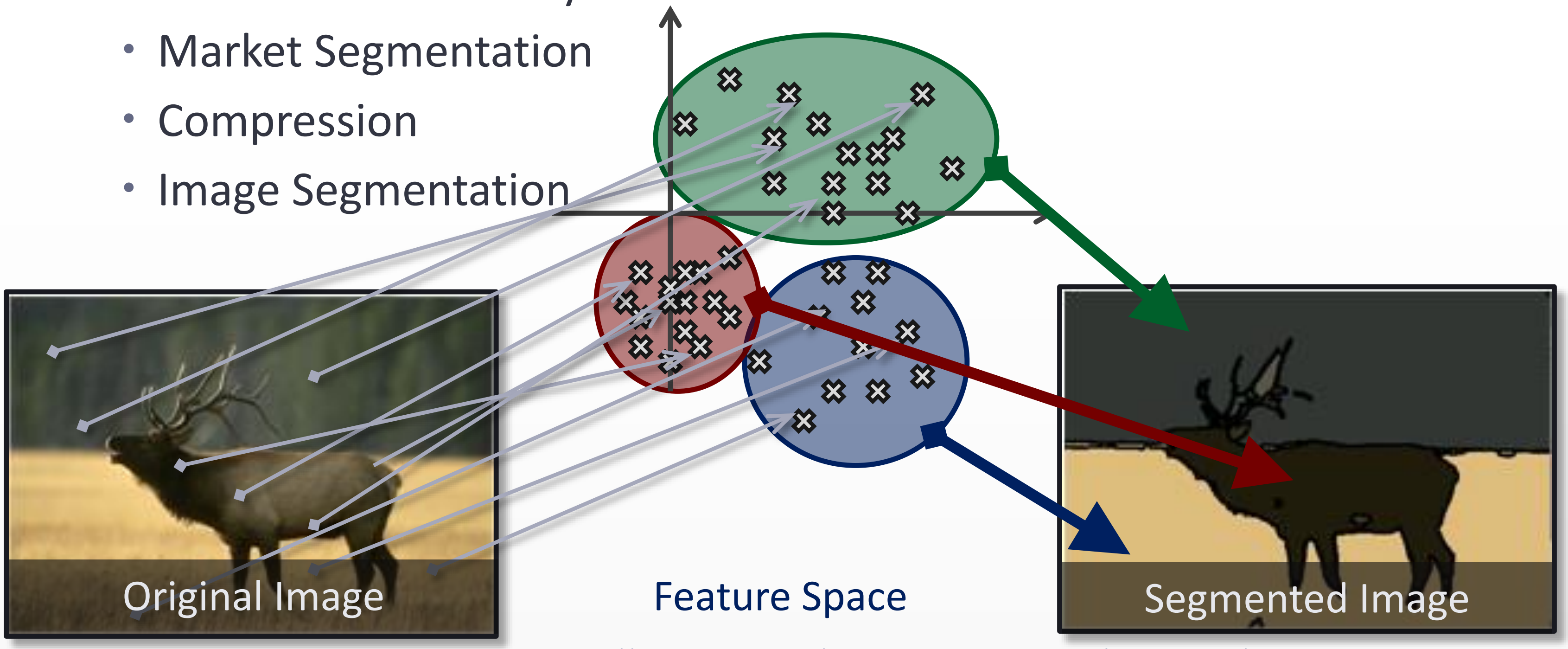
Supervised learning



Unsupervised learning

Machine Learning Algorithms

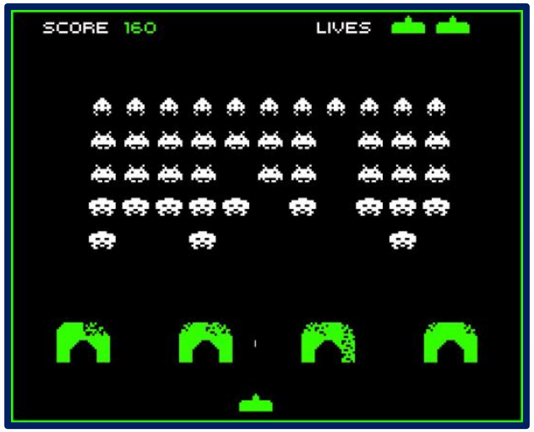
- Unsupervised Learning
  - The goal is to find meaningful structure in the data.
  - Applications:
    - Social Network Analysis
    - Market Segmentation
    - Compression
    - Image Segmentation

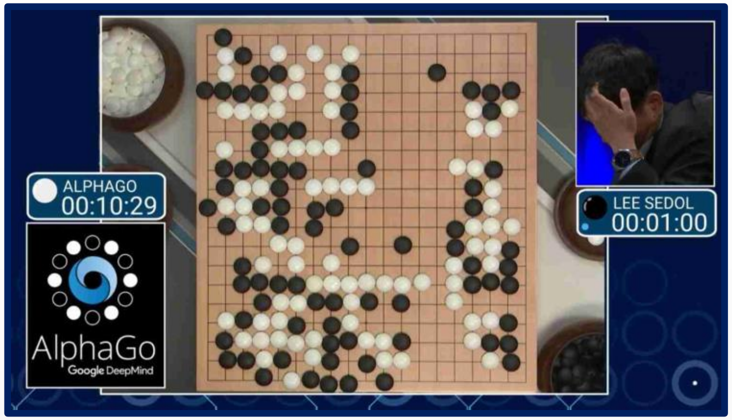


Source of the Images: http://ivrgwww.epfl.ch/supplementary_material/RK_CVPR09/

Machine Learning Algorithms

- Reinforcement Learning
  - The goal is to get an agent to act in the world so as to maximize its rewards.
- Recently very hot topic:
  - Computers can automatically learn to play ATARI games...
  - ...can beat humans in go (AlphaGo)
  - ...can learn to walk





Supervised Learning

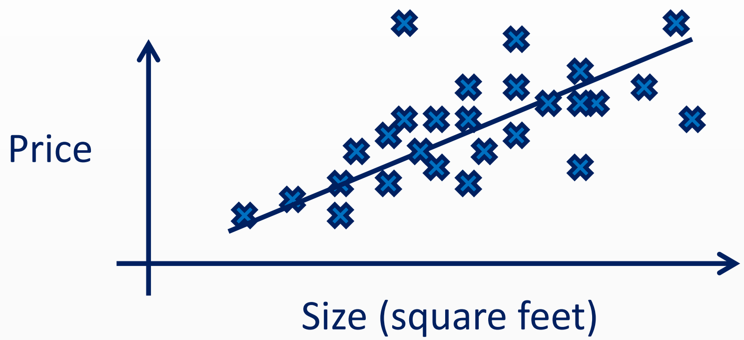
Linear Regression

- Example: Housing Prices
  - It is a supervised learning problem: we have data with ground truth.
  - We know the size of the houses and the price they were sold for:

Training data:

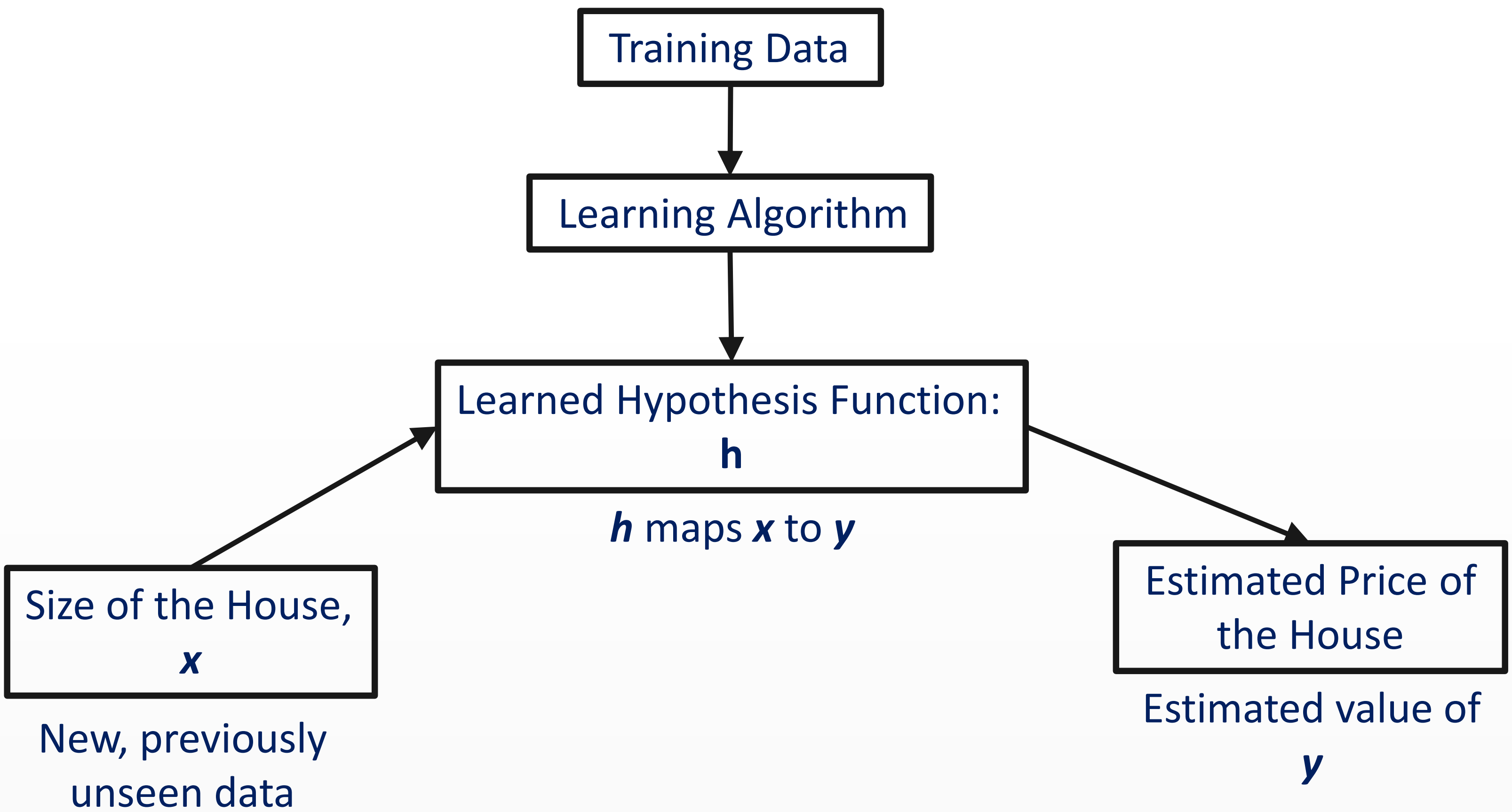
Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

First training example:
$(x^{(1)}, y^{(1)})$



Source: Andrew Ng Machine Learning Course on Coursera https://www.coursera.org/course/ml

Summarization of a Learning Algorithm

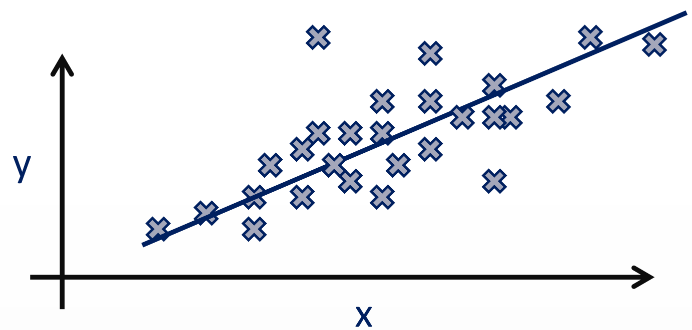


Source: Andrew Ng Machine Learning Course on Coursera https://www.coursera.org/course/ml

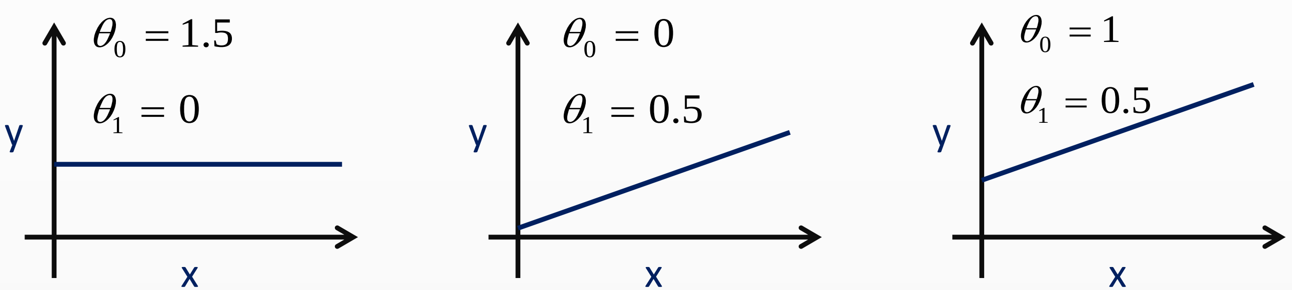
Supervised Learning
Linear Regression

- Representation of ***h*** for linear regression with one variable:

$$h_\theta = \theta_0 + \theta_1 x \longrightarrow \theta_i 's \text{ are the parameters}$$



- How to find the best values for the parameter θ?



Source: Andrew Ng Machine Learning Course on Coursera https://www.coursera.org/course/ml

Linear Regression

- How to find the best values for the parameter θ?
- Idea: Find parameters (θ_0, θ_1), so that $h_\theta(x)$ is close to y for the training examples.

$$\min_{\theta_0\theta_1} \frac{1}{2m} \sum_{i=1}^{m} \underbrace{\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)}^{2}$$

where ***m*** is the number of training examples

$$h_\theta\left(x^{(i)}\right) = \theta_0 + \theta_1 x^{(i)}$$

- The conventional notation of the above expression:

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

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

$J(\theta_0, \theta_1)$ is called the **cost function**

Source: Andrew Ng Machine Learning Course on Coursera https://www.coursera.org/course/ml

Linear Regression

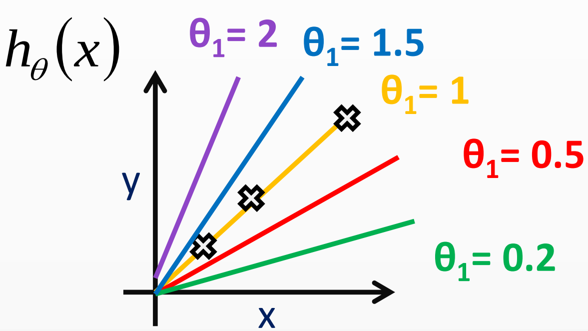
- We have a hypothesis function to map the features to the labels: x to y, house size to price, etc.

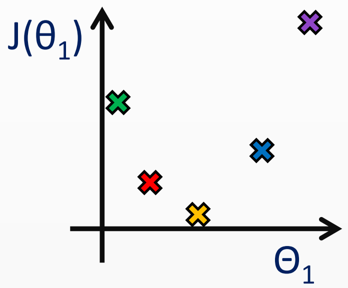
$$h_\theta = \theta_0 + \theta_1 x$$

- The hypothesis function has parameters (θ_0, θ_1), which are optimized during the training by the minimization of the cost function:

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

- Simplified example (with $\theta_0 = 0$):



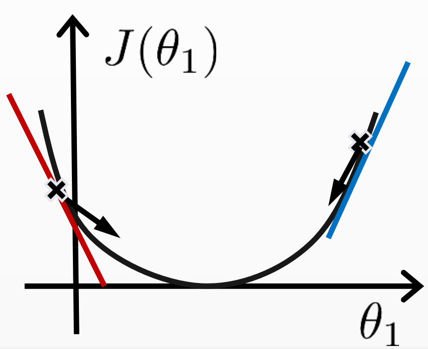


Gradient Descent

- Gradient Descent method will be used to find the minimum of $J(\theta_0, \theta_1)$:
  - Start with ***arbitrary initial values*** (e.g. $\theta_0 = 0,\ \theta_1 = 0$)
  - In each iteration change θ_0 and θ_1 so that J is reduced, until it reaches its minimum value. To achieve this the following ***update rule*** is used:

$$\theta_j = \theta_j - \underbrace{\alpha}_{\text{learning rate}} \underbrace{\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)}_{\text{derivative of J}} \qquad \text{for } j\text{=0,1}$$

  - The update is done simultaneously for all the θ_j.
- Intuition in 1D:

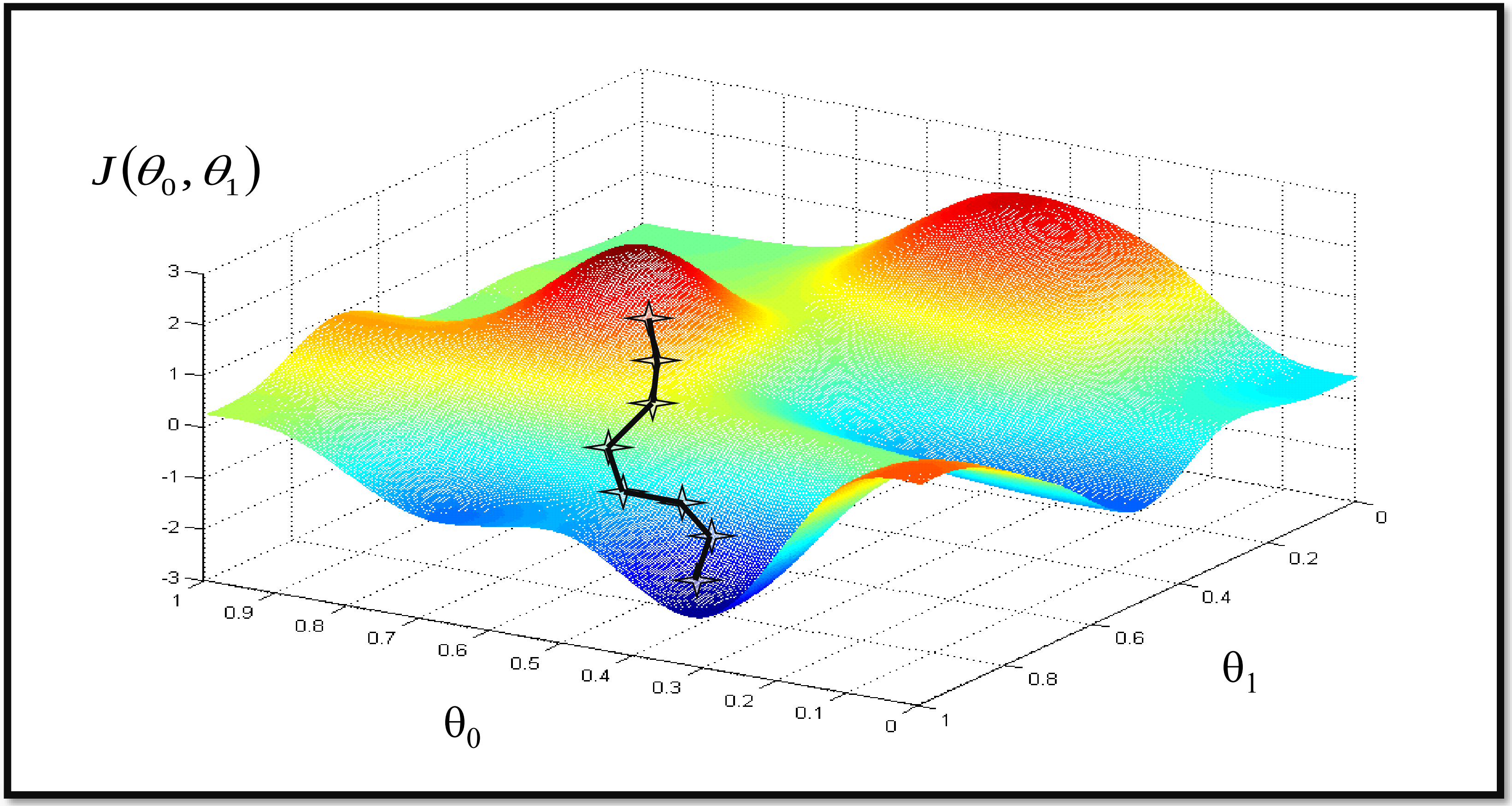


The tangential has a positive slope, the derivative is positive, θ will be decreased.

The tangential has a negative slope, the derivative is negative, θ will be increased.

Gradient Descent

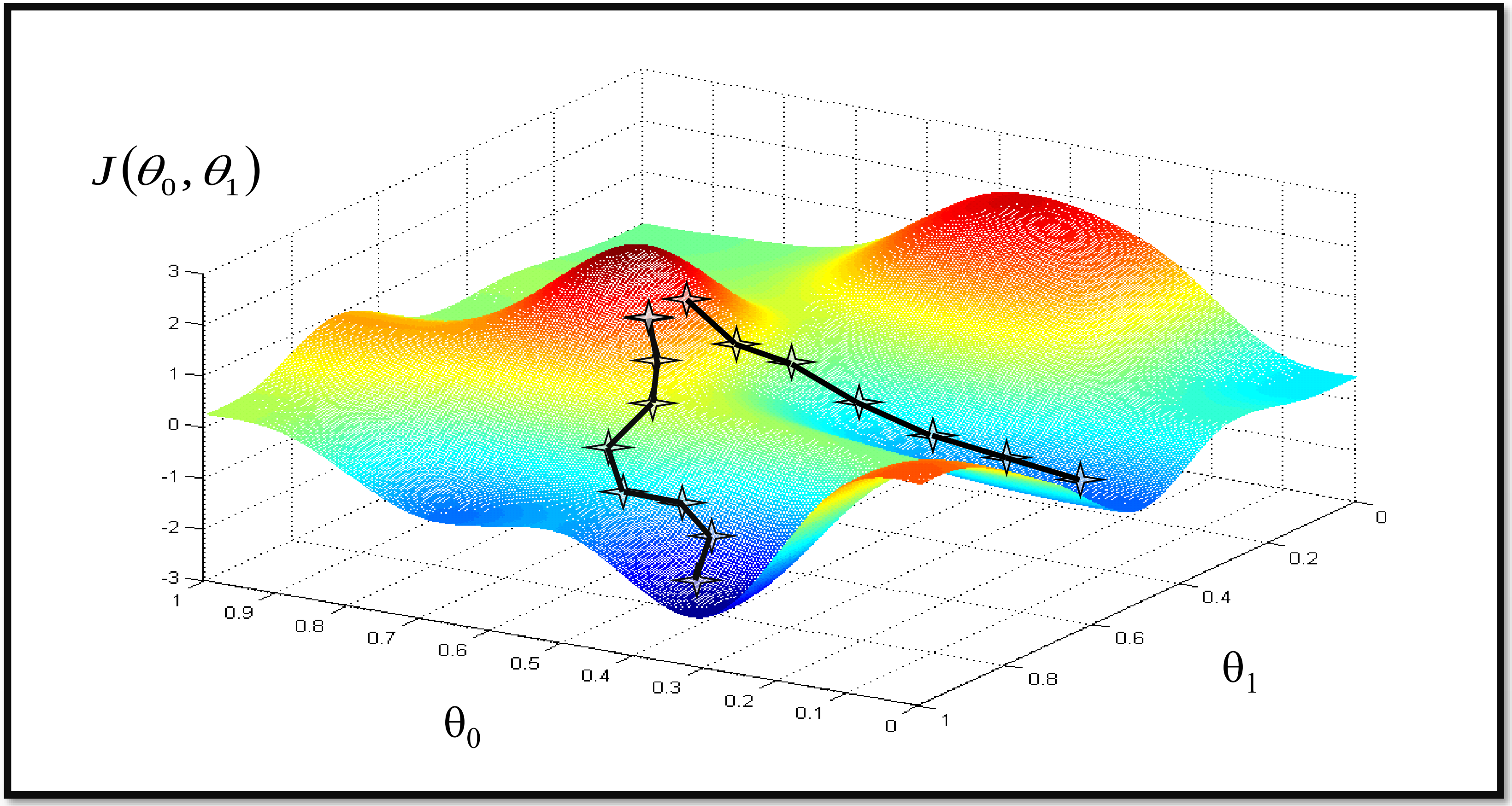
- In each iteration we move toward the minimum:



Source: Andrew Ng Machine Learning Course on Coursera https://www.coursera.org/course/ml

Gradient Descent

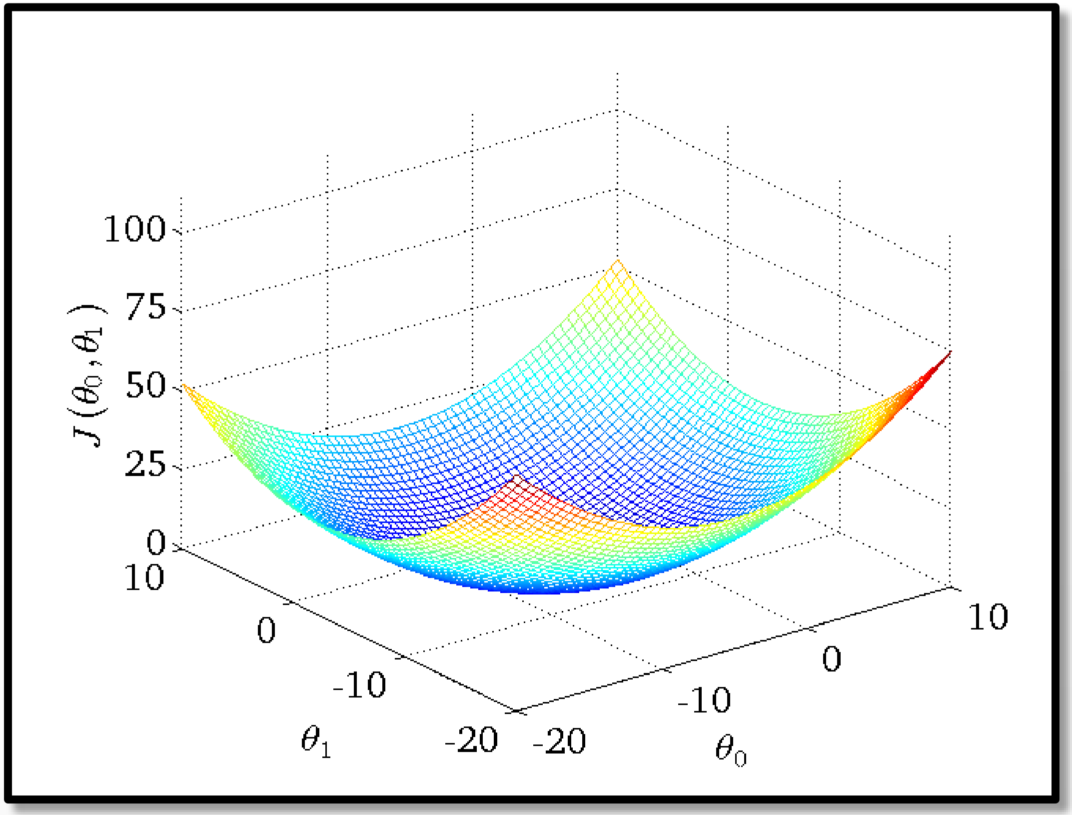
- In each iteration we move toward the (local) minimum:



Source: Andrew Ng Machine Learning Course on Coursera https://www.coursera.org/course/ml

Linear Regression

- With a convex *J* function, there is only one minimum, the global minimum:



Source: Andrew Ng Machine Learning Course on Coursera https://www.coursera.org/course/ml

Linear Regression

- Linear regression can be more powerful with **multiple variables**:
  - Size of the house, # bedrooms, age, # floors, …
  - The new hypothesis function

$$h_\theta(x_1, x_2, x_3, \dots, x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n$$

  - More convenient to write it in a matrix-vector form:

$$h_\theta(x) = \theta^T \cdot x = \begin{bmatrix} \theta_0 & \theta_1 & \dots & \theta_n \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

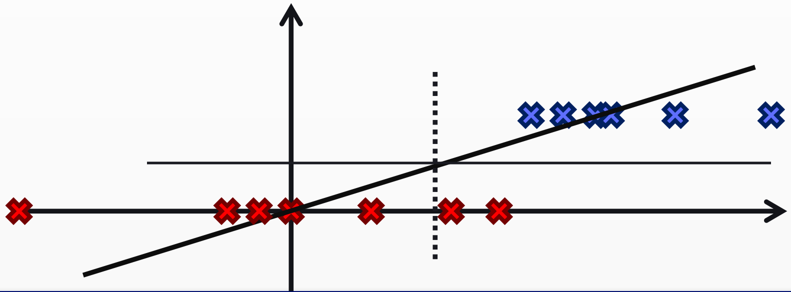
$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)^2$$

  - Features may have different scale (#bedrooms: 1-5, size of the house: 500-2000). Scaling the features to the same range + normalizing the mean often helps the learning algorithm to perform better.

Source: Andrew Ng Machine Learning Course on Coursera https://www.coursera.org/course/ml

Logistic Regression

- Supervised classification algorithm:
  - From the input features (x) it predicts a discrete output (y):
    - Face/Not Face, Spam/Not Spam, …
  - In the training data y is a vector of 0's and 1's:
    - 0 denotes that the samples belongs to the negative class: not face, not spam, …
    - 1 denotes that the samples belongs to the positive class: face, spam, …
  - There are multiclass classification problems with N different classes, where y = 1,2,3,..N. (e.g. car recognition: Opel, Honda, Peugeot,…)
  - Can we use linear regression for this problem?



$$h_\theta(x) \geq 0.5 \rightarrow y = 1$$
$$h_\theta(x) < 0.5 \rightarrow y = 0$$

Logistic Regression

- Logistic regression produces answers between [0,1]:

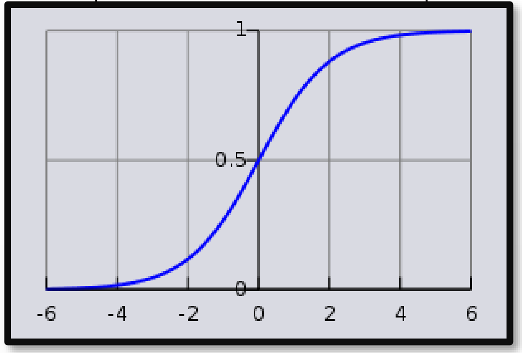
$$0 \leq h_\theta(x) \leq 1$$

- To achieve this we take the logistic function of $\theta^T x$:

$$h_\theta(x) = g(\theta^T \cdot x) \quad \text{where} \quad g(z) = \frac{1}{1+e^{-z}}$$

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T \cdot x}}$$

Logistic or sigmoid function



- Interpretation of the hypothesis:
  - If for some x, $h_\theta(x) = 0.8$, it means that x, has 80% probability to belong to the positive class:

$$h_\theta(x) = P(y=1 \mid x;\theta) = 0.8$$

$$\rightarrow P(y=0 \mid x;\theta) = 1 - P(y=1 \mid x;\theta) = 0.2$$

Decision Boundary

- Interpretation of the hypothesis:
  - To predict binary class labels we use a threshold 0.5:

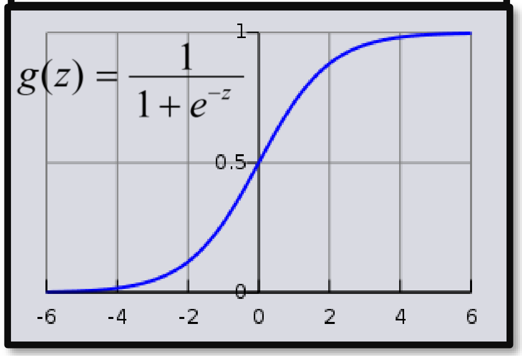
$$P(y=1 \mid x;\theta) \geq 0.5 \rightarrow y=1$$
$$P(y=1 \mid x;\theta) < 0.5 \rightarrow y=0$$

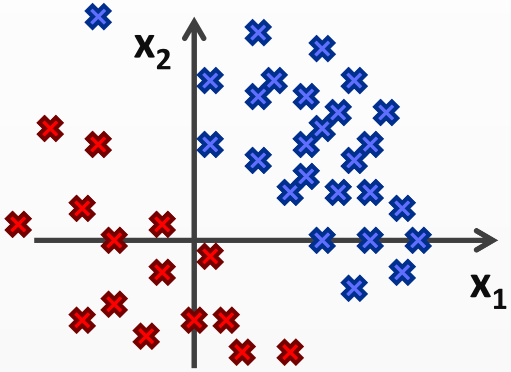
  - Using a sigmoid function this mean:

$$g(z) \geq 0.5 \quad \text{when} \quad z \geq 0 \quad \Rightarrow \quad \theta^T \cdot x \geq 0$$
$$g(z) < 0.5 \quad \text{when} \quad z < 0 \quad \Rightarrow \quad \theta^T \cdot x < 0$$

  - How can we classify our dataset, assuming we have the trained parameters θ?

Sigmoid function



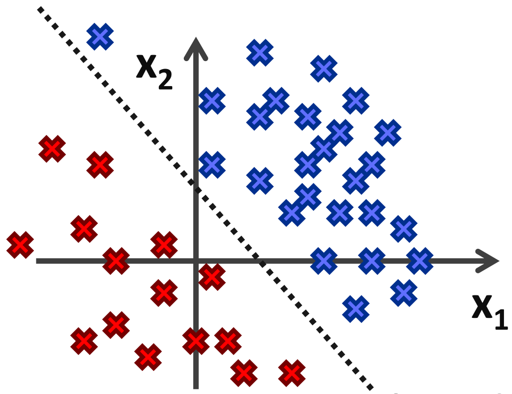


Decision Boundary

- Interpretation of the hypothesis:
  - Example I:
    - We have the following hypothesis function:

$$h_\theta(x) = g\left(\theta^T x\right) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

    - With parameters: $\theta_0 = -3$, $\theta_1 = 1$, $\theta_2 = 1$
    - We predict „1" if $-3 + x_1 + x_2 \geq 0 \;\Rightarrow\; x_1 + x_2 \geq 3$
    - We predict „0" if $-3 + x_1 + x_2 < 0 \;\Rightarrow\; x_1 + x_2 < 3$

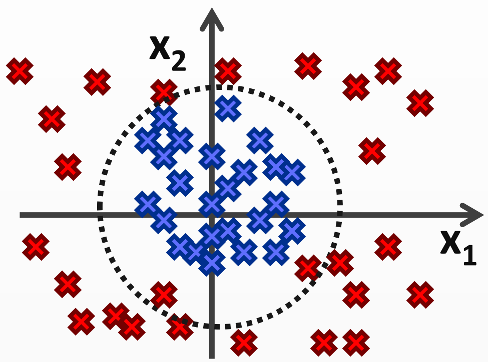


Decision boundary: $x_1 + x_2 = 3$

  - Example II:
    - The decision boundary can be nonlinear

$$h_\theta(x) = g\left(\theta^T x\right) = g\left(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2\right)$$

    - With parameters: $\theta_0 = -1$, $\theta_1 = 0$, $\theta_2 = 0$, $\theta_3 = 1$, $\theta_4 = 1$



Decision boundary: $x_1^2 + x_2^2 = 1$

Logistic Regression

- How to chose parameter θ?
  - We have *m* training examples: $(x^{(1)}, y^{(1)})$, $(x^{(2)}, y^{(2)})$, …, $(x^{(m)}, y^{(m)})$
  - Each training example has an *n* dimensional feature vector *x* and a label *y*.
  - The hypothesis function is $h_\theta(x) = \frac{1}{1 + e^{-\theta^T \cdot x}}$
  - What cost function should we use?
    - In linear regression the cost function was the following:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)^2 = \frac{1}{m} \sum_{i=1}^{m} \mathrm{cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$$

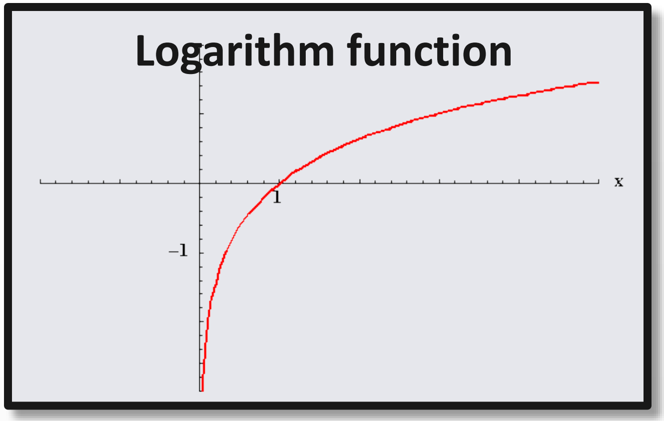
    - The problem is now we have a nonlinear hypothesis function and if we plug it into J(θ) the result will be a non-convex cost function.
    - We need to replace the $\mathrm{cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$

Logistic Regression

- How to chose parameter θ?
  - We will use the following cost term:

$$\mathrm{cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if} \quad y = 1 \\ -\log(1 - h_\theta(x)) & \text{if} \quad y = 0 \end{cases}$$

  - If **y=1**:
    - The cost is equal to zero if $h_\theta(x) = 1$, and as $h_\theta(x)$ goes to 0, the cost goes to infinity.
  - If **y=0**:
    - The cost is equal to zero if $h_\theta(x) = 0$, and as $h_\theta(x)$ goes to 1, the cost goes to infinity.



- The unified cost function of logistic regression is as follows:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right)$$

Softmax Regression

- If the classification problem is not binary, e.g.:
  - Cat, Dog, Car, Airplain, Boat, etc.
  - Handwritten digits/characters
  - Facial expressions

Mutually exclusive categories

- The training set is $\{(x^{(1)},y^{(1)}), \ldots, (x^{(n)},y^{(n)})\}$, where $y^{(i)}$ is in $\{1,\ldots,K\}$.
- For multiclass classification there are different strategies:
  - Transformation to Binary:
    - 1 vs. All (need K classifiers)
    - 1 vs. 1 (need K*(K-1)/2 classifiers)
  - Extension from Binary:
    - Softmax
    - ...
  - Hierarchical

Onehot encoding:

$$
\begin{array}{lcl}
y = 1 & \Rightarrow & [1 \quad 0 \quad \cdots \quad 0] \\
y = 2 & \Rightarrow & [0 \quad 1 \quad \cdots \quad 0] \\
 & \vdots & \quad \vdots \\
y = K & \Rightarrow & [0 \quad 0 \quad \cdots \quad 1]
\end{array}
$$

http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/

Softmax Regression

- For K exclusive categories we can use Softmax classifier, where the hypothesis function maps the input x to the following a K-dimensional hypothesis vector:

$$h_\theta(x) = \frac{1}{\sum_{j=1}^{K} e^{\boldsymbol{\theta}^{(j)T} \cdot x}} \begin{bmatrix} e^{\boldsymbol{\theta}^{(1)T} x} \\ e^{\boldsymbol{\theta}^{(2)T} x} \\ \vdots \\ e^{\boldsymbol{\theta}^{(K)T} x} \end{bmatrix}$$

Now θ is a matrix and each of its columns is the parameterization of the x feature vector for one class.

$$\boldsymbol{\theta} = \begin{bmatrix} | & | & & | \\ \boldsymbol{\theta}^{(1)} & \boldsymbol{\theta}^{(2)} & \cdots & \boldsymbol{\theta}^{(K)} \\ | & | & & | \end{bmatrix}$$

- The kth element the hypothesis vector can be interpreted as probability of membership of the kth category:

$$P(y_i = k \mid x_i, \theta) = \frac{e^{\theta^{(k)T} x_i}}{\sum_{j=1}^{K} e^{\theta^{(j)T} \cdot x_i}}$$

- Logistic regression can be regarded as a special case (when $K = 2$) of the Softmax classifier.

Softmax Regression

- The cost function (cross-entropy loss) is the following:

$$J(\theta) = -\sum_{i=1}^{n}\sum_{k=1}^{K} 1\{y_i = k\} \log \frac{e^{\boldsymbol{\theta}^{(k)T} x}}{\sum_{j=1}^{K} e^{\boldsymbol{\theta}^{(j)T} \cdot x}} = -\sum_{i=1}^{n}\sum_{k=1}^{K} 1\{y_i = k\} \log P(y_i = k \mid x_i, \boldsymbol{\theta})$$

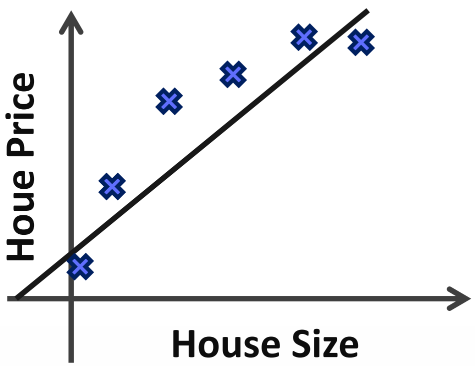
- We cannot solve for the minimum of $J(\theta)$ analytically, and thus as usual we'll resort to an iterative optimization algorithm. Taking the derivatives, one can show that the gradient is:

$$\nabla_{\theta^{(k)}} J(\theta) = -\sum_{i=1}^{m} \left[x_i \left(1\{y_i = k\} - P(y_i = k \mid x_i, \theta) \right) \right]$$

  - where $\nabla_{\theta^{(k)}} J(\theta)$ is itself a vector, so that its j-th element is $\frac{\partial J(\theta)}{\partial \theta_{lk}}$ the partial derivative of $J(\theta)$ with respect to the j-th element of $\theta^{(k)}$.
  - Armed with this formula for the derivative, one can then plug it into a standard optimization package and have it minimize $J(\theta)$.

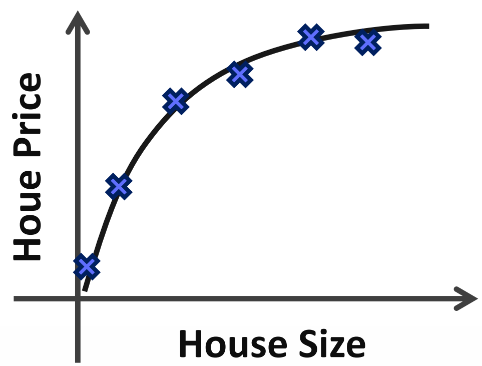
Regularization

- Example: Linear regression



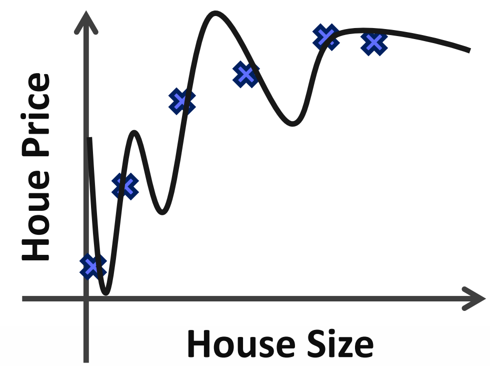
$$h_\theta(x) = \theta_0 + \theta_1 x$$

Underfit or High bias



$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

Right



$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \\ + \theta_3 x^3 + \ldots + \theta_n x^n$$

Overfit or High variance

- If we have too many features we can learn a hypothesis that fits the training data very well, but fails on new samples (= does not generalize well)

Source: Andrew Ng Machine Learning Course on Coursera https://www.coursera.org/course/ml

Regularization

- To handle underfitting we can introduce new features.
- To handle overfitting:
  - We can **reduce the number of features** (but this might mean we lose useful information):
    - We can select manually which features to keep.
    - Use a model selection algorithm.
  - We can apply **regularization**:
    - We can keep all the features but we reduce their magnitude (the value of the θ parameters).
    - Works well if we have a lot of features and each contributes a little bit to predict *y*.
    - The idea is to keep the parameters low, to get a simpler hypothesis function, which is less prone to overfitting.

Regularization

- How can we keep the parameters low?
  - The cost function for linear/logistic regression with regularization:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\operatorname{cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) + \boxed{\lambda\sum_{j=1}^{n}\theta_j^2}$$

Note: θ_0 is not regularized

L2 regularization term

  - The regularization parameter λ controls the trade-off between two goals:
    - Fitting the data well
    - Keeping the parameters low, to avoid overfitting

  - If λ is too large all the parameters (except θ_0) will be close to 0, the model won't fit the data, we will see underfitting.

Main Sources

- Andrew Ng Machine Learning Course
  - On Coursera: https://www.coursera.org/course/ml
  - At Stanford: http://cs229.stanford.edu/

- Further reading:
  - Lectures by Nando de Freitas:
    - Undergraduate Machine Learning at UBC 2012: https://www.youtube.com/playlist?list=PLE6Wd9FR--Ecf_5nCbnSQMHqORpiChfJf&feature=view_all
    - Machine Learning at UBC 2013 http://www.cs.ubc.ca/~nando/540-2013/lectures.html
  - A Few Useful Things to Know about Machine Learning: http://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf
  - Linear classification Loss Visualization: http://vision.stanford.edu/teaching/cs231n/linear-classify-demo/