Freie Universität Berlin

Lecture: Machine Learning for Data Science

Winter semester 2021/22

Lecture 9: Regression

Prof. Dr. Eirini Ntoutsi

Outline

- Intro into regression
- Univariate linear regression
- Multivariate linear regression
- Linear classifiers with a hard/ logistic threshold
- Things you should know from this lecture & reading material

Regression vs Classification

- Both supervised learning tasks
  - In classification, the class attribute is discrete.
  - In regression, the class attribute is continuous.

ID	Age	Car type	Risk
1	23	Familie	high
2	17	Sport	high
3	43	Sport	high
4	68	Familie	low
5	32	LKW	low

What is the predicted risk of a person with a certain Age and Car type?

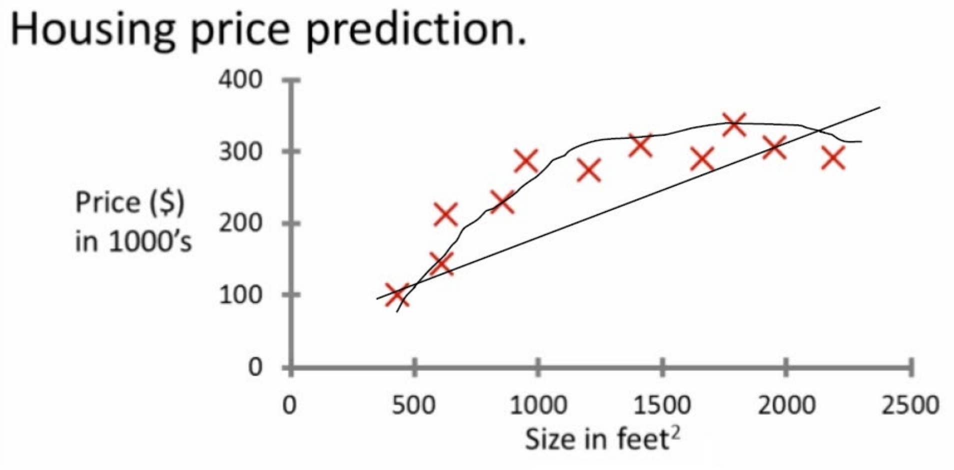
House ID	Size (feet)	Old	Price
1	500	10	100K
2	1000	20	500K
3	2000	50	300K
4	300	15	200K

What is the predicted price for a house of a certain size and age?

Regression example (univariate case)

- Consider the following dataset of different houses and their prices
- Given this data, a friend has a house 750 square feet -how much can they be expected to get?

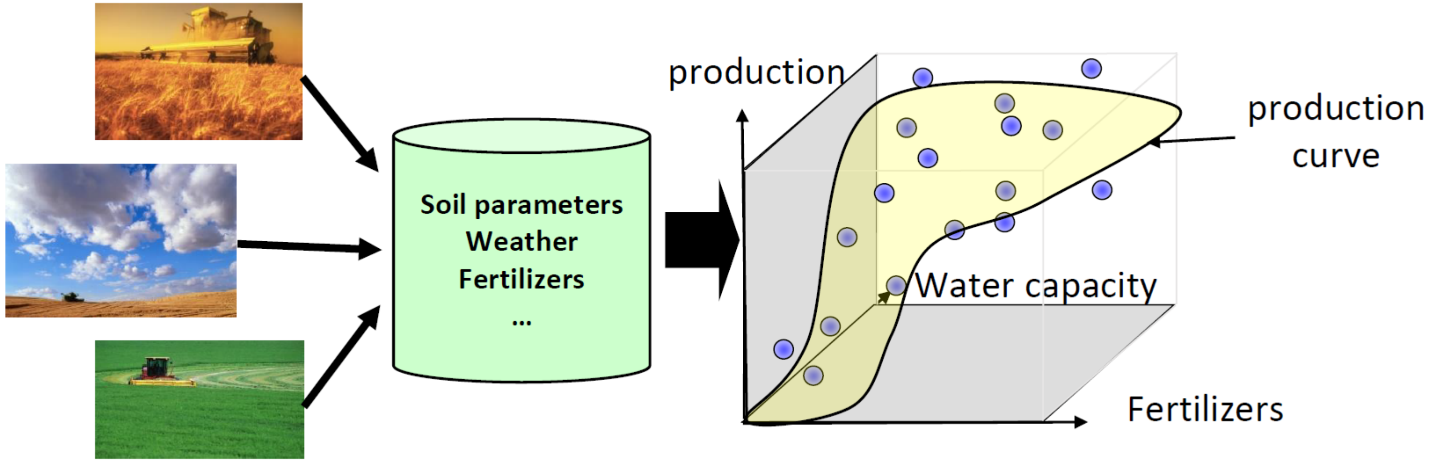
instance	size	price
1	500	100
2	1000	250
3	2000	300



Source: Andrew Ng ML course, Coursera

Regression application: Precision farming (multivariate case)

- Predict the amount of fertilizers based on multiple attributes like soil characteristics, weather, used fertilizers.
  - Only the appropriate amount of fertilizers given the environmental settings (soil, weather) will result in maximum yield.
  - Controlling the effects of over-fertilization on the environment is also important



Problem formulation

- *"A computer program is said to learn from experience E w.r.t. some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."*

  Tom Mitchell, Machine Learning 1997.

- What is the task *T*?
- What is the experience *E*?
- What is the performance/evaluation measure *P*?

Problem formulation

- In classification, the class attribute *Y* is discrete: $d(Y) = \{c1, c2, ..., c_k\}$; *k* is the number of classes.
- The goal is to find a mapping/function/hypothesis $h(): X \rightarrow Y$

- In regression, the prediction aims at a real value $Y \in \mathbb{R}$
- The goals is to learn a function h()/f(): $X \rightarrow \mathbb{R}$

Task T

House ID	Size (feet)	Old	Price
1	500	10	100K
2	1000	20	500K
3	2000	50	300K
4	300	15	200K

- Each training instance has the form ($\vec{x}$,y) where $y \in \mathbb{R}$

Experience E

- The predicted value *f(x)* is also called predicted variable, response variable, or dependant variable and is often denoted by *y*
- Examples
  - Recommended amount of fertilizer for a certain type of soil
  - Prediction of house prices

Problem formulation

- How to evaluate the performance of *h()*?

Performance measure P

- Using the empirical error or empirical loss of *h()*
  - So errors over the training set
- In classification, we were *mainly* checking whether the predicted class agrees with the real class/ground truth for each instance, i.e., $h(x_i)=y_i$
- For regression, we want to check how close is the predicted value $h(x_i)$ from the real value y_i
- Popular loss functions
  - L_1 loss or absolute-value loss: $L_1(h(x_i),y_i)=|h(x_i)-y_i|$
  - L_2 loss or squared-error loss: $L_2(h(x_i),y_i)=(h(x_i)-y_i)^2$
  - 0/1 loss: $L_{0/1}(h(x_i),y_i)=0$, if $h(x_i)=y_i$; 1 otherwise

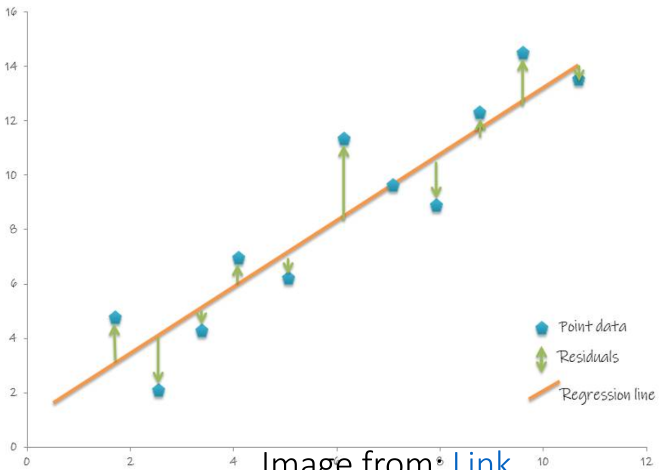


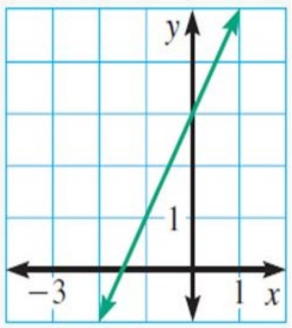
Image from: Link

Outline

- Intro into regression
- Univariate linear regression
- Multivariate linear regression
- Linear classifiers with a hard/ logistic threshold
- Things you should know from this lecture & reading material

Univariate linear regression

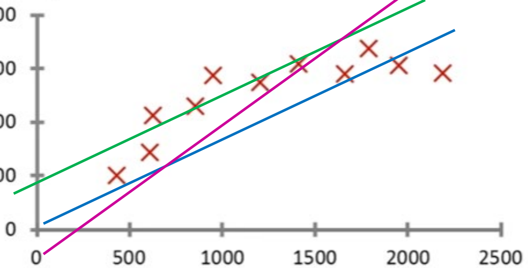
f(x)=2x+3
offset is 3
Slope is 2



- Let a training set of *N* instances: $D=\{(x_i, y_i)\}$
  - each instance is described in the *1*-dimensional feature space: (*X*)
  - The class *Y* is continuous
- We are looking for a univariate linear function (a straight line) with the form

$$f_w(x) = w_1 \cdot x + w_0$$

  - w_1 is the slope (orientation)
  - w_0 is the intercept (offset from origin) (bias)
- $w=<w_1, w_0>$ is the weight vector/the parameters of the model/line (to be learned from data)
- Among the available lines, which one to choose?
  - The one that best fits the data!



Univariate linear regression

- Operational definition: the one ($w*$) that minimizes the empirical loss, i.e., loss over the training data
- Typically L_2 loss is used

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} L_2\left(y_j, h_{\mathbf{w}}\left(x_j\right)\right) = \sum_{j=1}^{N} \left(y_j - h_{\mathbf{w}}\left(x_j\right)\right)^2 = \sum_{j=1}^{N} \left(y_j - \left(w_1 x_j + w_0\right)\right)^2$$

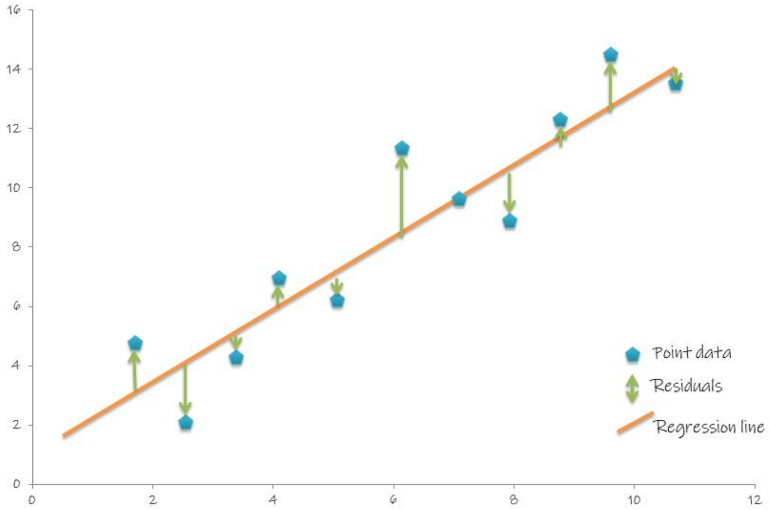
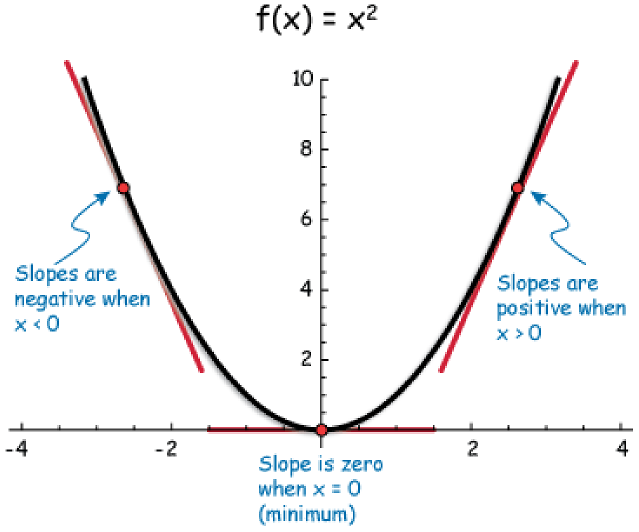


Image from: [Link]

- So, the best line is:

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} Loss(h_{\mathbf{w}})$$

How to find the best line w*



- The goal is to find the line that minimizes the empirical L_2 error

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} Loss(h_{\mathbf{w}})$$

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} L_2\left(y_j, h_{\mathbf{w}}\left(x_j\right)\right) = \sum_{j=1}^{N} \left(y_j - h_{\mathbf{w}}\left(x_j\right)\right)^2 = \sum_{j=1}^{N} \left(y_j - \left(w_1 x_j + w_0\right)\right)^2$$

- Analytical solution: Loss() is minimized when partial derivatives w.r.t. w_1 and w_0 are zero

$$\frac{\partial}{\partial w_0} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0 \text{ and } \frac{\partial}{\partial w_1} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0.$$

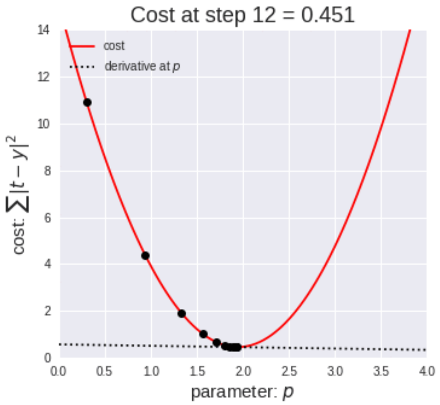
- These equations have a unique solution:

$$w_1 = \frac{N\left(\sum x_j y_j\right) - \left(\sum x_j\right)\left(\sum y_j\right)}{N\left(\sum x_j^2\right) - \left(\sum x_j\right)^2}; \qquad w_0 = \left(\sum y_j - w_1\left(\sum x_j\right)\right)/N.$$

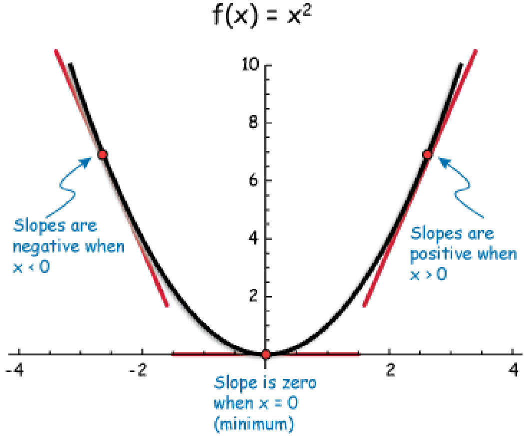
How to find the best line w*

- In this case, we can find a solution analytically because the loss function is convex
- In the general case, for any loss function, we can use gradient-descent (see previous lecture)
  - Start with an arbitrary initial line represented by the weight vector *(w_0, w_1)*
  - Repeatedly modify it in small steps
  - At each step, the weight vector is altered in the direction that produces the steepest descent along the error surface.
  - gradient descent learning rule: At each step, we take a step into the opposite direction of the gradient, and the step size is determined by the value of the learning rate η as well as the slope of the gradient
    - Recall the perceptron lecture with L2 loss

$$\Delta\vec{w} = -\eta\nabla E[\vec{w}] \qquad E[\vec{w}] \equiv \frac{1}{2}\sum_{d\in D}(t_d - o_d)^2$$



Gradient Descent visualization Link



Gradient descent pseudocode

Initialize weight vector w
While not converged
For each $w_i \in w$ do

$$wi \leftarrow wi - \eta\frac{\vartheta}{\vartheta w_i}Loss(w)$$

Univariate regression: an example

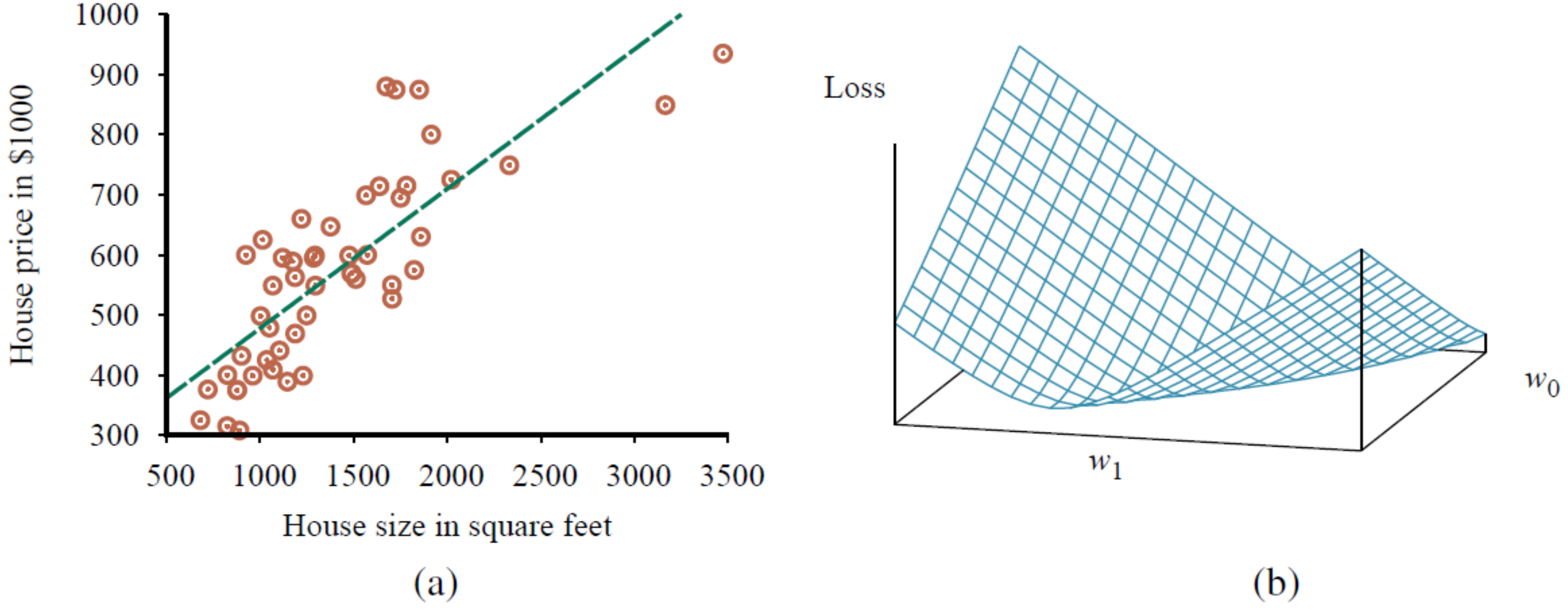


Figure 19.13 (a) Data points of price versus floor space of houses for sale in Berkeley, CA, in July 2009, along with the linear function hypothesis that minimizes squared-error loss: $y = 0.232x + 246$. (b) Plot of the loss function $\sum_j (y_j - w_1 x_j + w_0)^2$ for various values of w_0, w_1. Note that the loss function is convex, with a single global minimum.

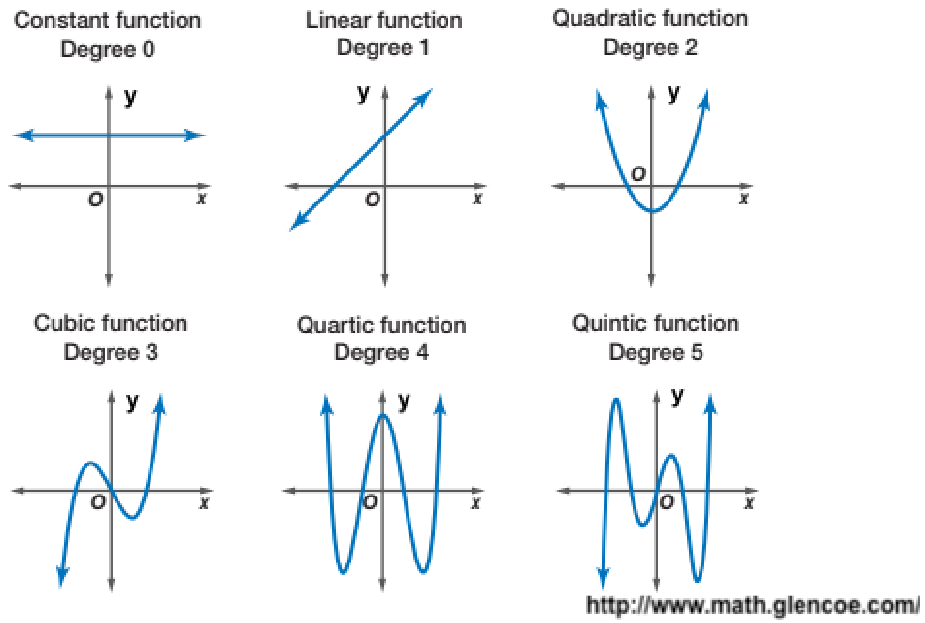
Source: AI book

Polynomial regression

- In the general case, the relationship between X and Y can be approximated using a larger degree polynomial
- Polynomial regression: A form of regression, in which the relationship between the predictive attribute X and the predictive attribute Y is modeled as a n-degree polynomial

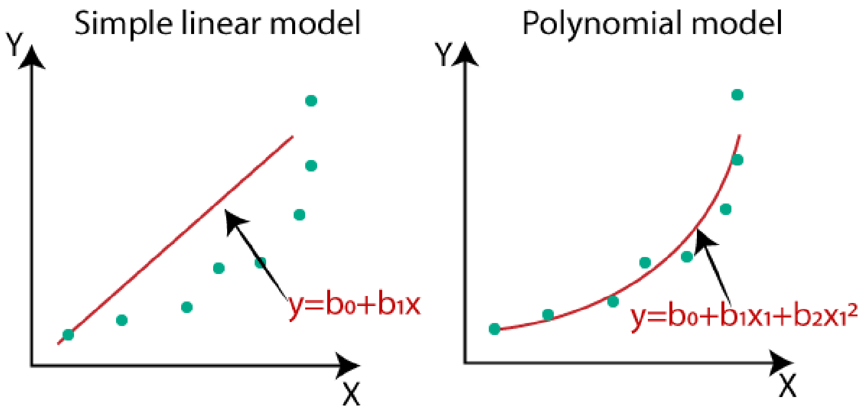
$$f(x) = wo + w_1 x + w_2 x^2 + \cdots + wnxn$$

- For a 1-degree polynomial → linear regression
  $f(x) = wo + w_1 x$
- For a 2-degree polynomial → quadratic
  $f(x) = wo + w_1 x + w_2 x^2$
- For a 3-degree polynomial → cubic
  $f(x) = wo + w_1 x + w_2 x^2 + w_3 x^3$



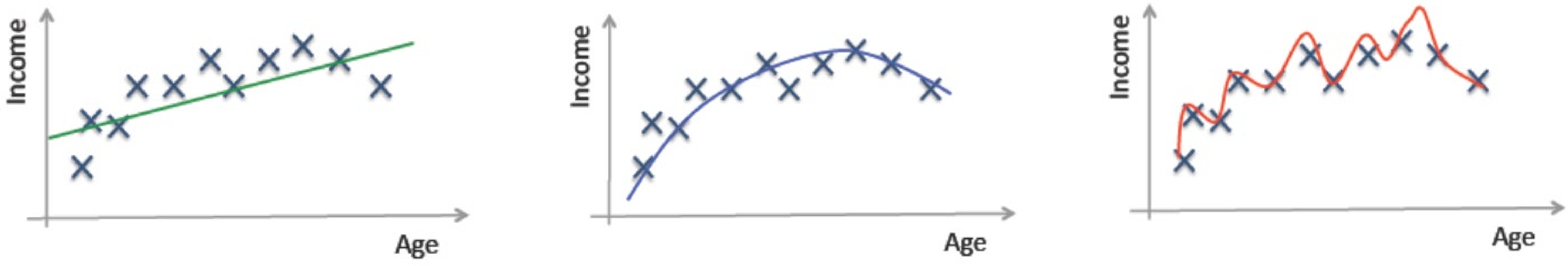
Polynomial regression

- In many situations there exist no linear relationship between *X* and *Y*



Beware of the overfitting

- The more complex the model (higher degree *n*), the higher the overfitting risk



Underfitting

Best model

Overfitting

$$f(x) = \lambda_0 + \lambda_1 x \;\ldots\; (1)$$

$$f(x) = \lambda_0 + \lambda_1 x + \lambda_2 x^2 \;\ldots\; (2)$$

$$f(x) = \lambda_0 + \lambda_1 x + \lambda_2 x^2 + \lambda_3 x^3 + \lambda_4 x^4 \;\ldots\; (3)$$

Overfitting

- Intuition: Both blue and green lines are solutions of squared loss, we prefer green as illustrates a less complex model than the blue.
- How to avoid overfitting→ by regularizing the complexity of the hypothesis
- Complexity of a hypothesis $h_w()$
  - Defined as a function of the weights w

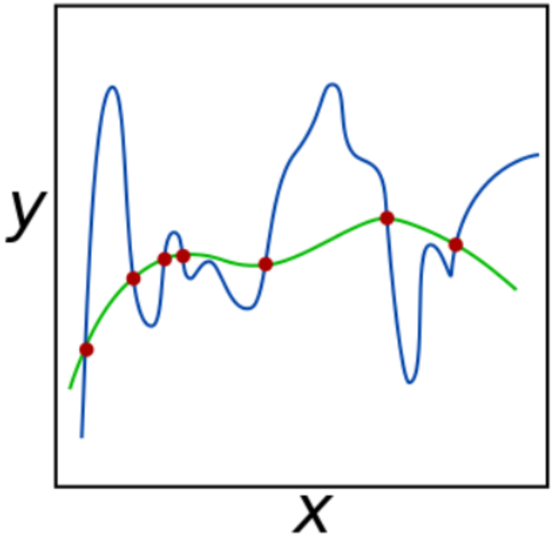
$$Complexity(h_{\mathbf{w}}) = L_q(\mathbf{w}) = \sum_i |w_i|^q$$

  - For $q=1$, minimize the sum of absolute weight values → L_1 regularization
    - E.g., in Lasso regression
  - For $q=2$, minimize the sum of squared values → L_2 regularization
    - E.g., in Ridge regression

- The goal then is to find the hypothesis $h*$, with the minimum total cost

$$Cost(h) = Loss(h) + \lambda Complexity(h)$$

- By adding a penalty to the loss function the overfitting is reduced



Source: https://en.wikipedia.org/wiki/Regularization_(mathematics)

L2 vs L1 regularization

- Which regularization to choose depends on the problem at hand
- L_1 regularization tends to produce sparse models
  - Larger penalties result in weights closer to zero, which is the ideal for producing simpler models
  - In practice, this means setting many feature weights to zero
    - Effectively, the corresponding features do not count for the prediction
- Important property
  - for intepretability
  - Less likely for the model to overfit

Outline

- Intro into regression
- Univariate linear regression
- Multivariate linear regression
- Linear classifiers with a hard/ logistic threshold
- Things you should know from this lecture & reading material

Multivariate linear regression

- Let a training set of *N* instances: $D=\{(\vec{x}_i, y_i)\}$
  - each instance is described in the *n*-dimensional feature space: $(X_1, X_2, \ldots, X_n)$
  - The class *Y* is continuous
- We are looking for a multivariate linear function (not a line anymore but a hyperplane) with the form

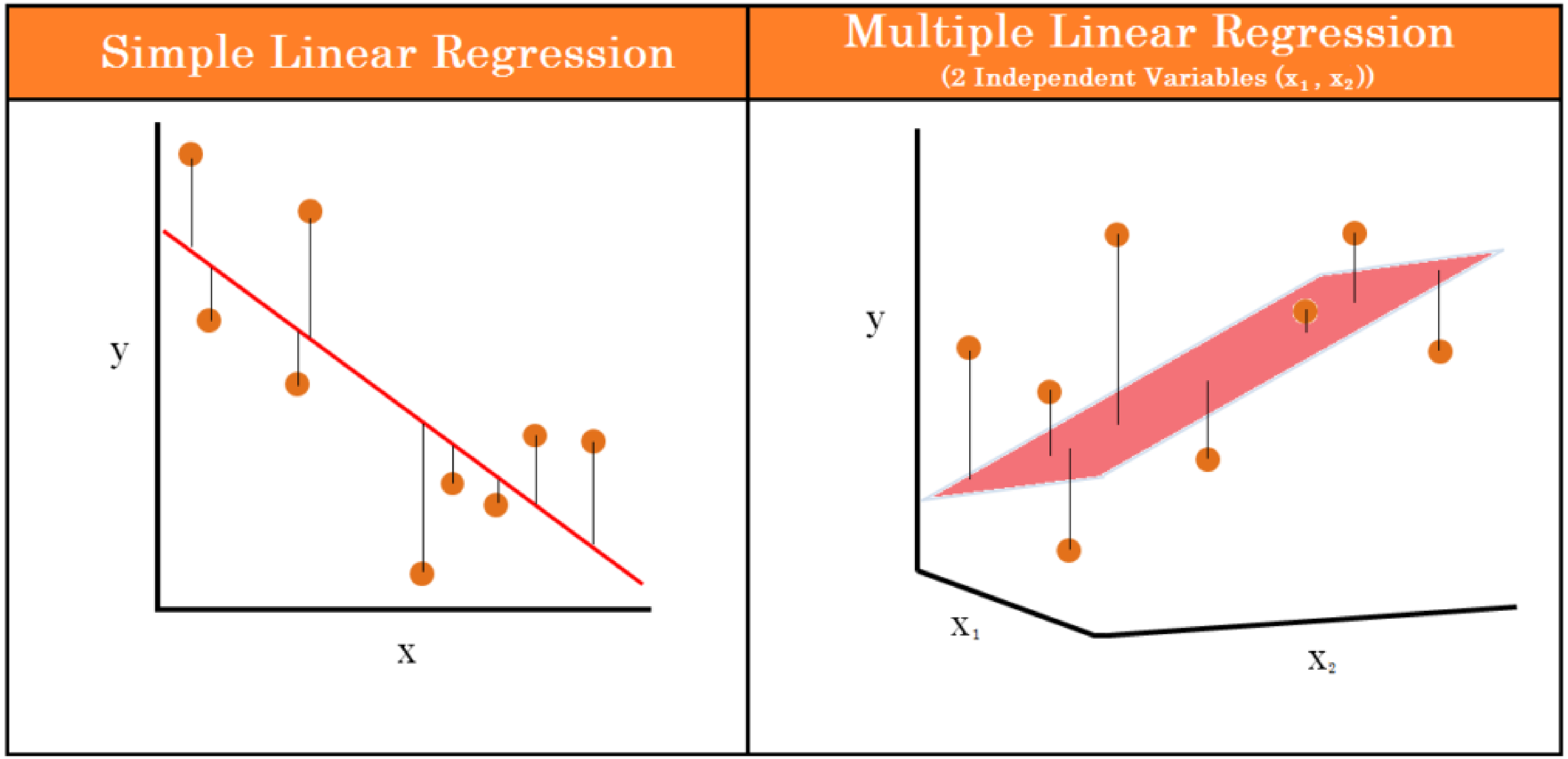
$$f_w(x) = w_1 \cdot x_1 + w_2 x_2 + \cdots + wnxn + w_0 = \sum_{i=0}^{n} w_i x_i$$

  - Weight vector *w* consists of: the weights of the features $w_1, \ldots, w_n$ and the bias term w_0
- The weight vector are the parameters of the model/line (to be learned from data)
- Among the available models, which one to choose?
  - The one that best fits the data → minimize L_2 error over the training data

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_j L_2(y_j, \mathbf{w} \cdot \mathbf{x}_j)$$

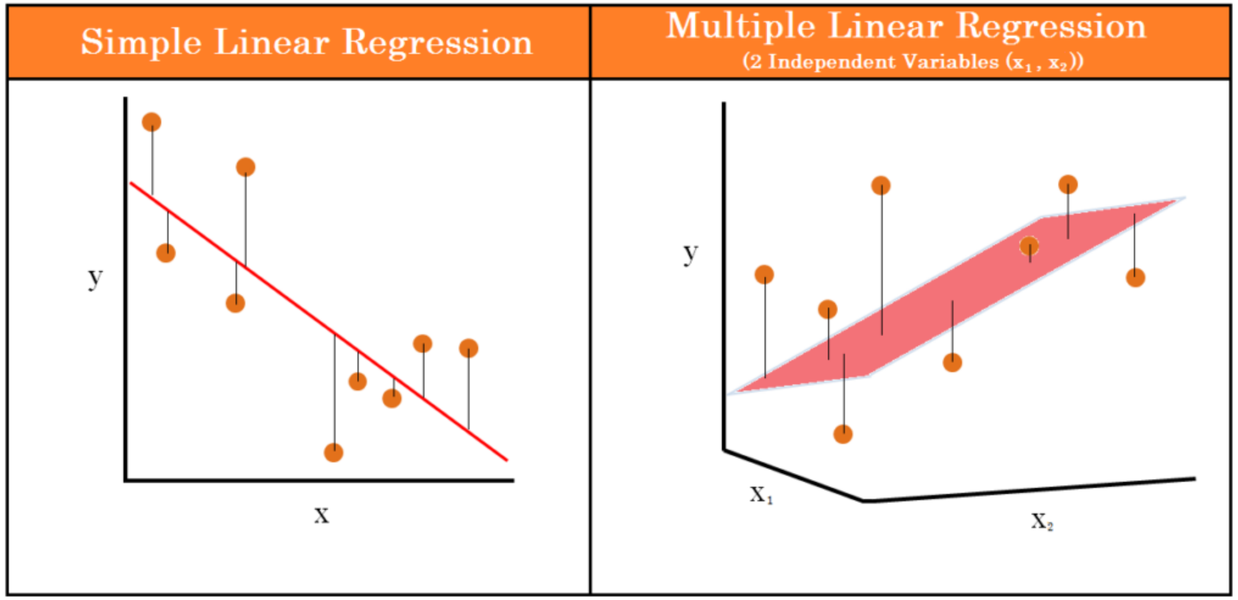
- Can be solved analytically or via gradient descent

Regression line vs regression (hyper)plane

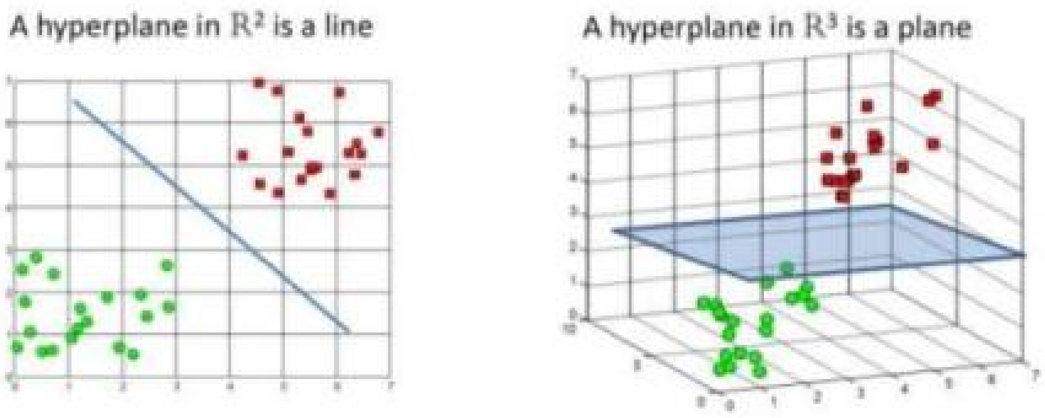


[Link](#)

Regression line/ regression (hyper)plane vs classification line/(hyper)plane



[Link]



A hyperplane in $\mathbb{R}^n$ is an n-1 dimensional subspace

Overfitting (discussed already)

- In multivariate regression, some attribute/dimension that is irrelevant might by chance appear to be useful → overfitting
- How to avoid overfitting→ by regularizing the complexity of the hypothesis (already discussed)
- Complexity of a hypothesis $h_w()$
  - Defined as a function of the weights *w*

$$Complexity(h_{\mathbf{w}}) = L_q(\mathbf{w}) = \sum_i |w_i|^q$$

  - For $q=1$, minimize the sum of absolute weight values → L_1 regularization
  - For $q=2$, minimize the sum of squared values → L_2 regularization

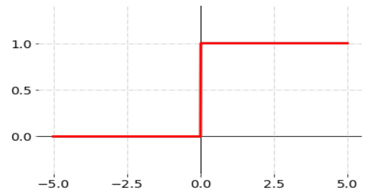
- The goal then is to find the hypothesis $h*$, with the minimum total cost

$$Cost(h) = Loss(h) + \lambda Complexity(h)$$

Outline

- Intro into regression
- Univariate linear regression
- Multivariate linear regression
- Linear classifiers with a hard/ logistic threshold
- Things you should know from this lecture & reading material

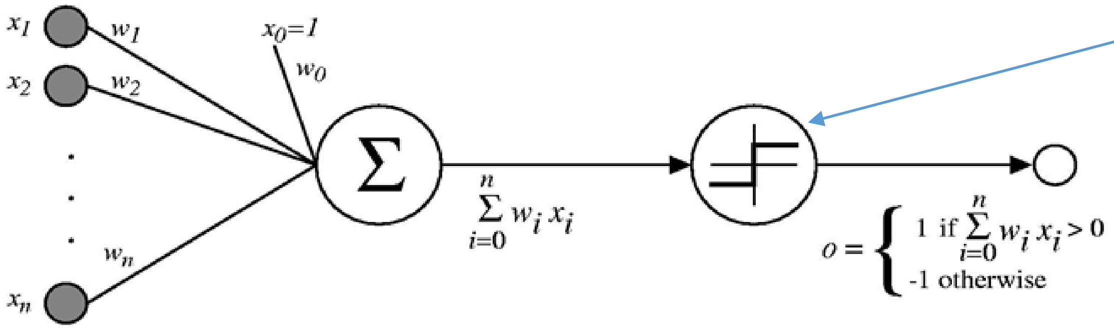
Linear classifiers with a hard threshold



Step function (threshold=0)

- We have already seen that linear functions can be also used for classification
  - The decision boundary is a line or hyperplane in higher dimensional spaces that separates the classes
- Recall the simple perceptron classifier

Activation function is the step function



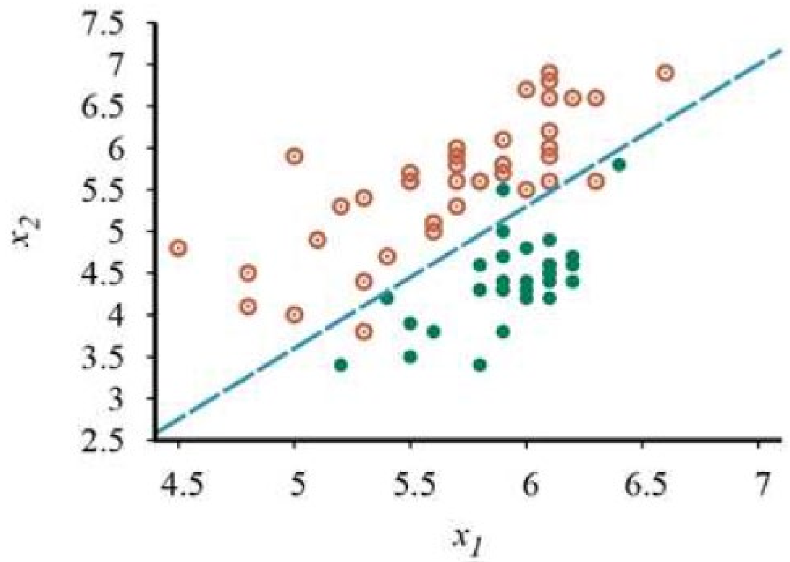
$$o(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ -1 & \text{otherwise.} \end{cases}$$

- In case of correct predictions, weights do not change
- In case of wrong predictions, weights change following the update rule
  - $W_j \leftarrow w_j + \eta(y^i - o^i)x_j^i$

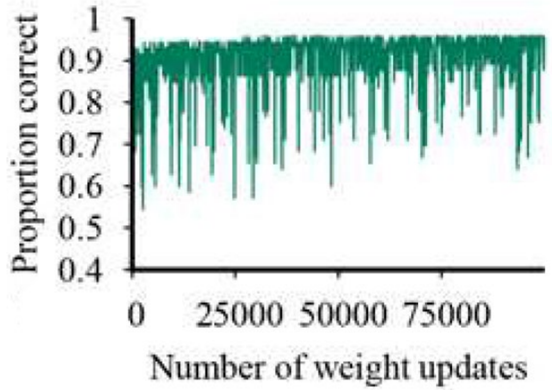
0/1 loss:
$L_{0/1}(h(x_i), y_i) = 0$, if $h(x_i) = y_i$; 1 otherwise

Convergence problems

- Although the perceptron learning rule converges if the two classes can be separated by linear hyperplane, problems arise if the classes cannot be separated perfectly by a linear classifier.



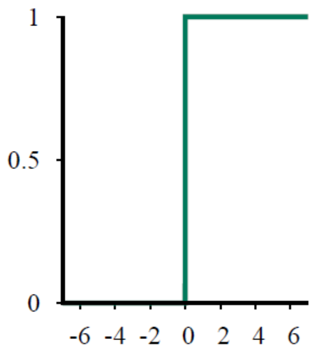
Plot of two seismic data parameters, body wave magnitude (x_1) and surface wave magnitude (x_2) for earthquakes (open orange circles) and nuclear explosions (green circles) occurring between 1982 and 1990 in Asia and the Middle East (Kebeasy *et al.*, 1998).
The earthquakes and explosions are not linearly separable.



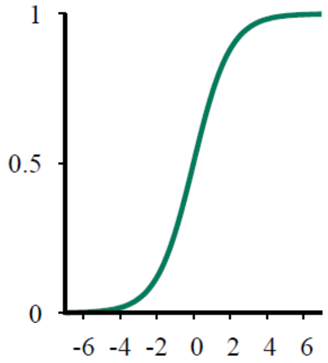
The simple perceptron learning rule fails to converge even after 10000 steps

From hard thresholds to soft thresholds

- The hard nature of the threshold of the simple perceptron causes problems with convergence
  - The hypothesis $h_w(x)$ is not differentiable
  - The classifier makes completely confident class predictions, even for instances close to the boundary
- Such problems can be solved using a soft threshold approach, so approximating the hard threshold with a continuous, differentiable function
- A popular choice is the logistic function, also known as sigmoid function



Hard- threshold/ Step function with 0/1 output

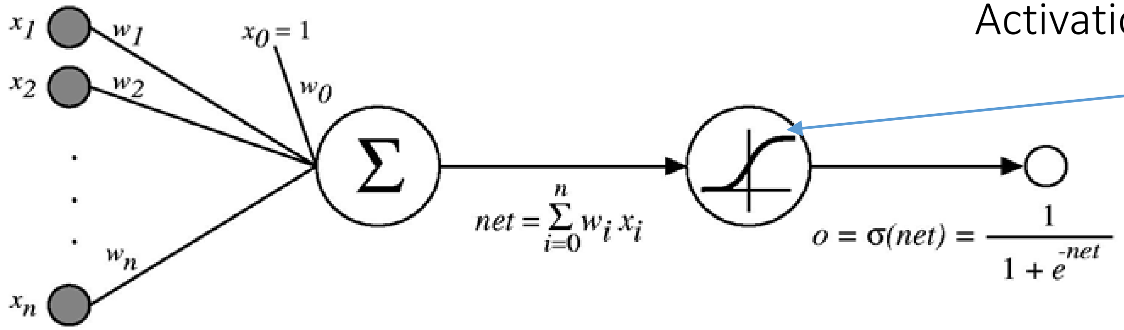


Logistic function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Sigmoid unit

- Very much like the perceptron unit but based on a smoothed, differentiable threshold function

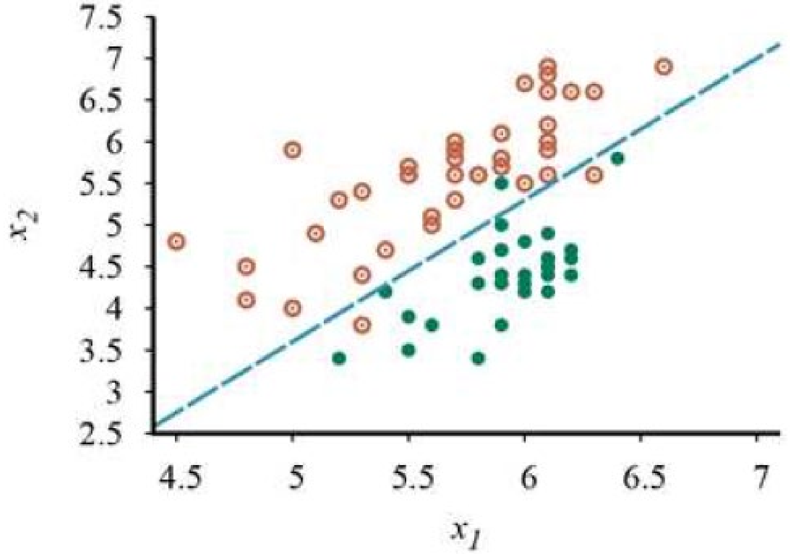


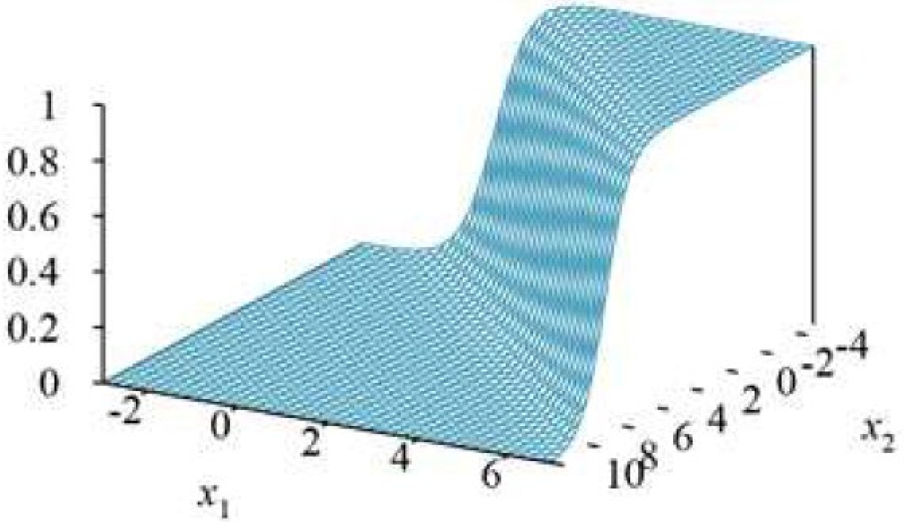
Activation function is the sigmoid or logistic function

- The sigmoid output *σ()* is a continuous function of its input
  - takes real values between -1 and +1
  - The sigmoid is in effect an approximation to the threshold function above, but has a gradient that we can use for learning

$$\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$$

An example





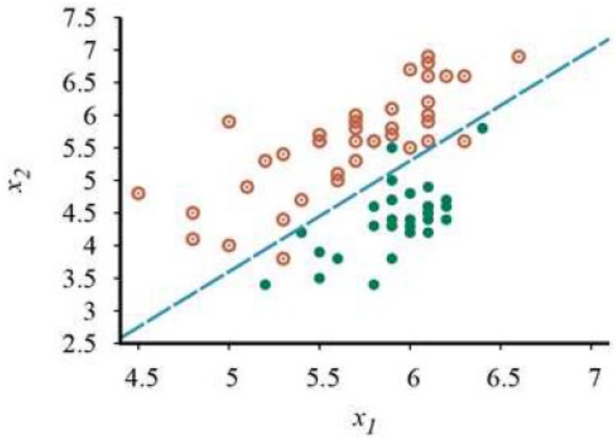
Plot of two seismic data parameters, body wave magnitude (x_1) and surface wave magnitude (x_2) for earthquakes (open orange circles) and nuclear explosions (green circles) occurring between 1982 and 1990 in Asia and the Middle East (Kebeasy *et al.*, 1998).
The earthquakes and explosions are not linearly separable.

A logistic regression hypothesis.
The output is continuous in the [0-1] range

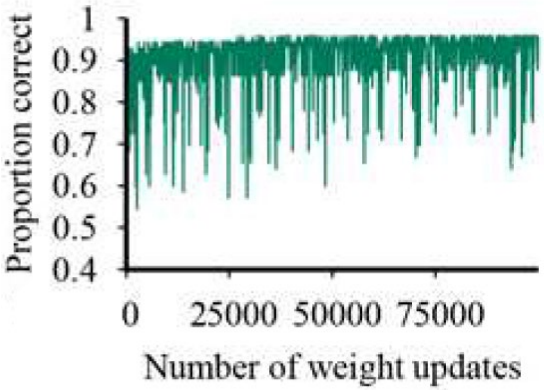
The output can be interpreted as *probability* of belonging to the class labeled 1.

Logistic regression

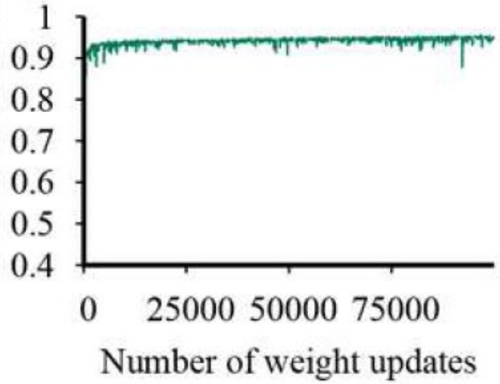
- The process of fitting the weights of this model to minimize loss on a data set is called logistic regression
- The weights are computed using gradient descent (with L2 loss function)
  - (Weight update formula not shown)
- Convergence example



Plot of two seismic data parameters, body wave magnitude (x_1) and surface wave magnitude (x_2) for earthquakes (open orange circles) and nuclear explosions (green circles) occurring between 1982 and 1990 in Asia and the Middle East (Kebeasy *et al.*, 1998).
The earthquakes and explosions are not linearly separable.



The simple perceptron learning rule fails to converge even after 10000 steps



Logistic regression converges fast

Outline

- Intro into regression
- Univariate linear regression
- Multivariate linear regression
- Linear classifiers with a hard/ logistic threshold
- Things you should know from this lecture & reading material

Overview and Reading

- Overview
  - Univariate linear regression
  - Multivariate linear regression
  - Hard threshold linear classifiers
  - Logistic regression

- Reading
  - Artificial Intelligence, A Modern Approach. Stuart Russell and Peter Norvig (Chapter 19)
  - Chapter 9: Linear predictors, Understanding Machine Learning book by Shai Shalev-Schwartz and Shai Ben-David

Hands on experience

- Try regression on the crime-prediction dataset (128 attributes) Link
- Many more datasets available on UCI for regression tasks

Acknowledgements

- The slides are based on
  - KDD I lecture at LMU Munich (Johannes Aßfalg, Christian Böhm, Karsten Borgwardt, Martin Ester, Eshref Januzaj, Karin Kailing, Peer Kröger, Eirini Ntoutsi, Jörg Sander, Matthias Schubert, Arthur Zimek, Andreas Züfle)
  - Introduction to Data Mining book slides at http://www-users.cs.umn.edu/~kumar/dmbook/
  - Pedro Domingos Machine Lecture course slides at the University of Washington
  - Machine Learning book by T. Mitchel slides at http://www.cs.cmu.edu/~tom/mlbook-chapter-slides.html
  - C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, DMKD 1998
  - For regression, images come mainly from Artificial Intelligence, A Modern Approach. Stuart Russell and Peter Norvig (Chapter 19) book
  - Thank you to all TAs contributing to their improvement, namely Vasileios Iosifidis, Damianos Melidis, Tai Le Quy, Han Tran.

Thank you

Questions/Feedback/Wishes?