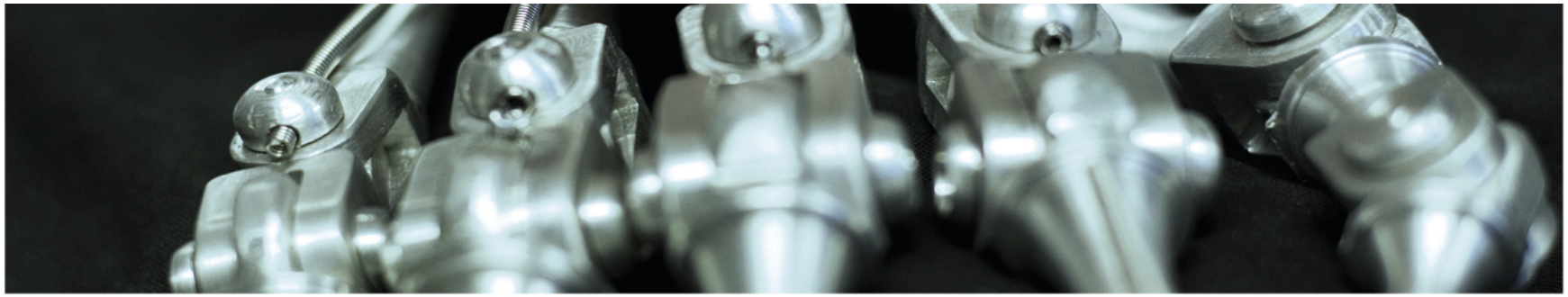


UiO : Department of Informatics
University of Oslo

INF 5860 Machine learning for image classification

Lecture 2 : Image classification and regression

Anne Solberg

January 24, 2017

Today's topics

- KNN (K- nearest neighbor)
- Crossvalidation
- Linear regression
- Loss functions
- Minimize loss functions using gradient descent
- Logistic regression – continued next week

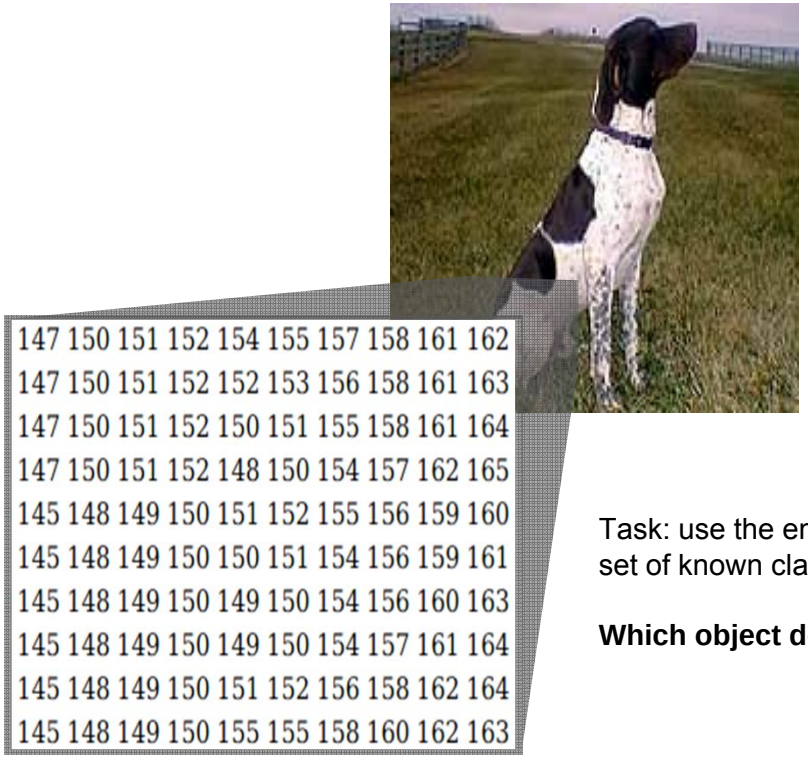
Today's read

- Note on introduction to classification
  - http://cs231n.github.io/classification
- Linear regression
  - Deep Learning Chap 5.1 (brief introduction)
  - More details: read
    - http://cs229.stanford.edu/notes/cs229-notes1.pdf
      (pages 1-7 and 16-19)
- Gradient descent
  - Deep Learning Chap 4.3

Relevant additional video links:

- https://www.youtube.com/playlist?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv
  - Lecture 2 and 3
  - Remark: they do not cover regresion.

Image classification - introduction



DOG

Task: use the entire image to classify the image into one of a set of known classes

Which object does the image contain

Where we currently are

- **KNN (K- nearest neighbor)**
- Crossvalidation
- Linear regression
- Loss functions
- Minimize loss functions using gradient descent
- Logistic regression

From last week: k-Nearest-Neighbor (KNN) classification

- Classification of a new sample $x^{(i)}$ is done as follows:
  - Out of N training images, identify the *k* nearest neighbor images (measured by L1 or L2) in the training set, irrespectively of the class label.
  - Out of these *k* samples, identify the number of images k_j that belong to class ω_j , *j:1,2,....M* (if we have *M* classes)
  - Assign $x^{(i)}$ to the class ω_j with the maximum number of k_j samples.
- *k* should be odd, and must be selected a priori.
- The distance measure and k are hyperparameters.

Measuring similarity between two images

- L1-distance

$$d_1 = \sum_i \sum_j |I_1(i,j) - I_2(i,j)|$$

- L2-distance

$$d_2 = \sqrt{\sum_i \sum_j \left(I_1(i,j) - I_2(i,j)\right)^2}$$

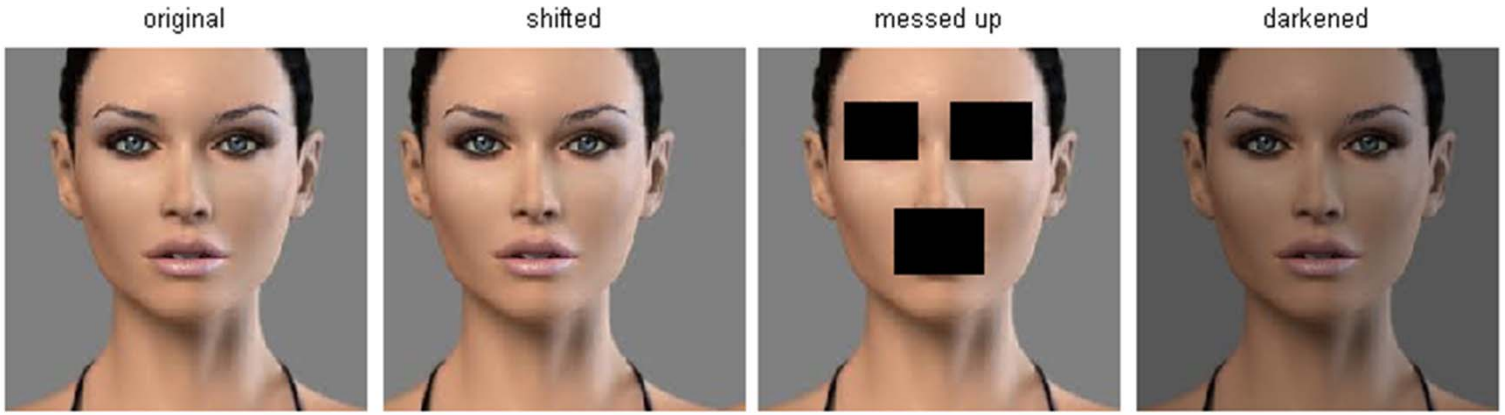
L2 is called Euclidean distance

KNN

- The KNN-classifier is mostly used for small datasets.
- There is no training, but classifying/predicting new data is slow as the size of the training data increases.
- The parameter k (and possibly the distance measure) should be found by cross-validation.
  - IF we have large amounts of data we will have a separate validation set, but we avoid using KNN for large datasets.

Is KNN robust to changes in position, color, shape, contrast?

- All these images have the same L2-distance from the original.



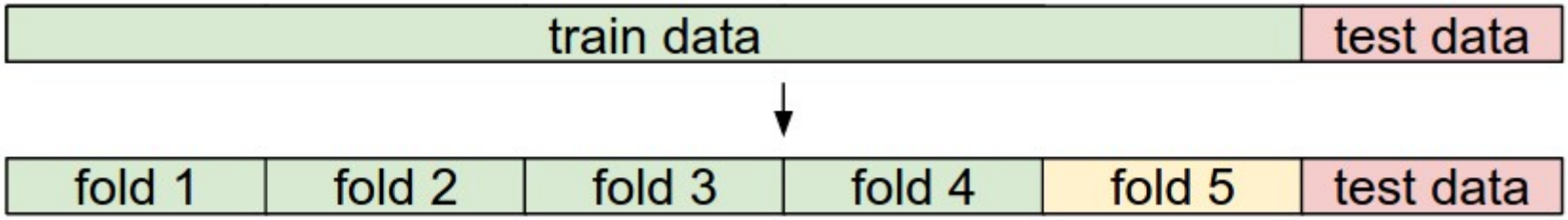
Where are we

- KNN (K- nearest neighbor)
- **Crossvalidation**
- Linear regression
- Loss functions
- Minimize loss functions using gradient descent
- Logistic regression

Selecting k using cross-validation

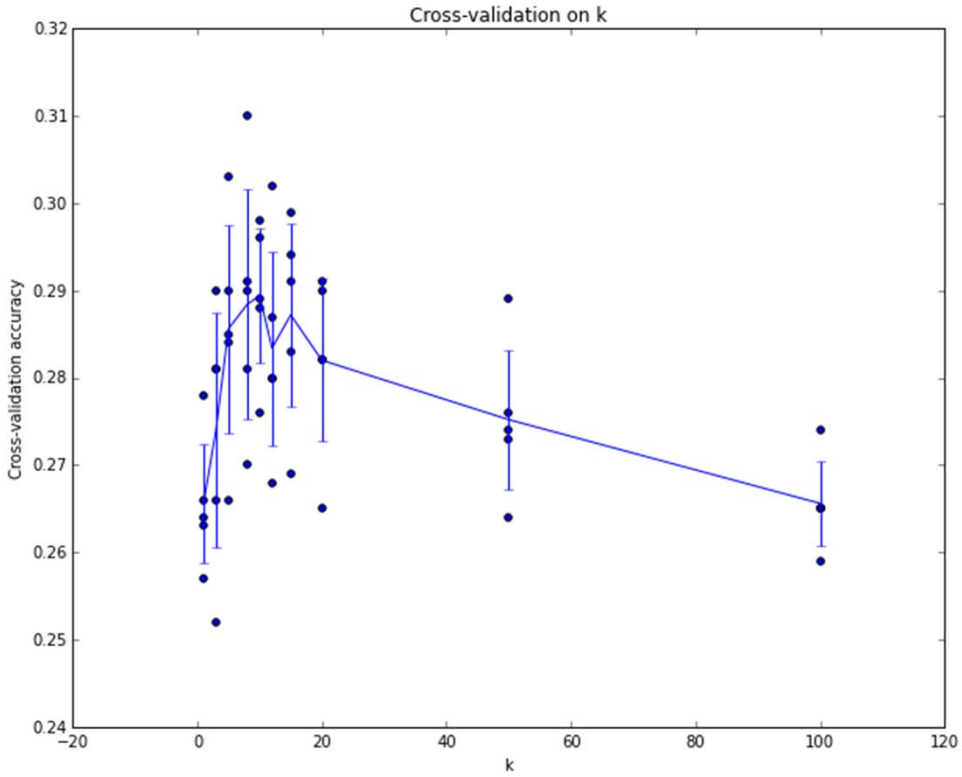
- For each value of k:
  - Cross-validation: split the training data into d subset/folds
  - Train on data from d-1 folds ,
  - Estimate the accuracy/compute the number of correctly classified images on the last fold , store the accuracy.
  - Repeat this *nfold* times and compute for the average of the accuracies.

- Repeat with different values of k, select the value that got the highest accuary.
- Now train on the entire dataset using this value of k, then classify the test data set ONCE to get the accuracy of the classifier.



- Example: 10000 training images, 5000 test images.
- Split training images into 10 folds of 1000 images.
- Train on 9 folds (9000 images), compute accuracy on the last 1000 images during cross-valiation.
- After finding k: train on all 10000, and estimate the reported accuracy on the 5000 test images.

Cross-validation plot



For each value of k, show the mean value and the standard deviation of the classification accuracy

Where are we

- KNN (K- nearest neighbor)
- Crossvalidation
- **Linear regression**
- Loss functions
- Minimize loss functions using gradient descent
- Logistic regression

Notation

- m – number of examples in the training data set
- n_x – input size, dimension of the input data
- n_y – output size: typically number of classes
- Superscript $x^{(i)}$: training sample number i, if x is a single measurement for each sample.
- Later for images: matrix notation for X for a set of m training images.

From linear regression to logistic regression for classification

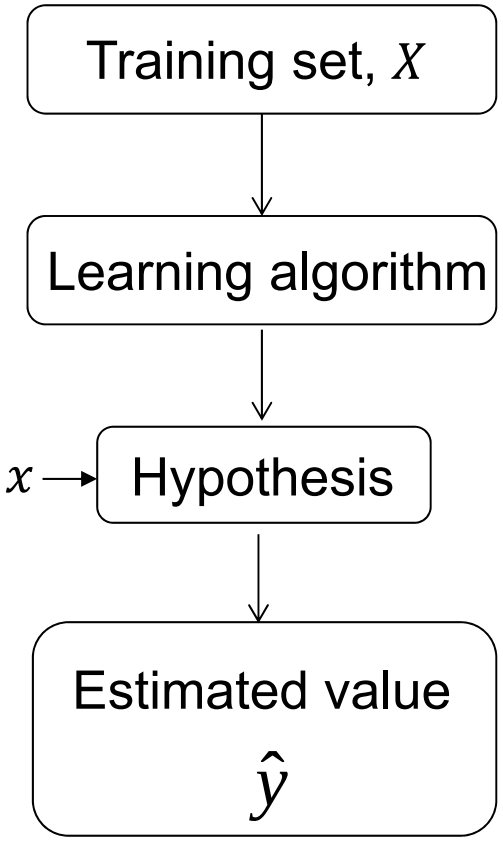
- Logistic classification is done iteratively in a manner similar to neural nets.
- We study this method before proceeding to neural networks.
- First – a brief look at linear regression to introduce loss functions and gradient descent minimization.

The linear regression problem

- Predict the true values y based on data vector x from a training data set.
- In regression, we want to predict y (a continuous number) based on data x.
  - Example: Predict the population in Norway based on measurements from 1990-2010.
- Linear hypothesis

$$\hat{y} = wx + b$$

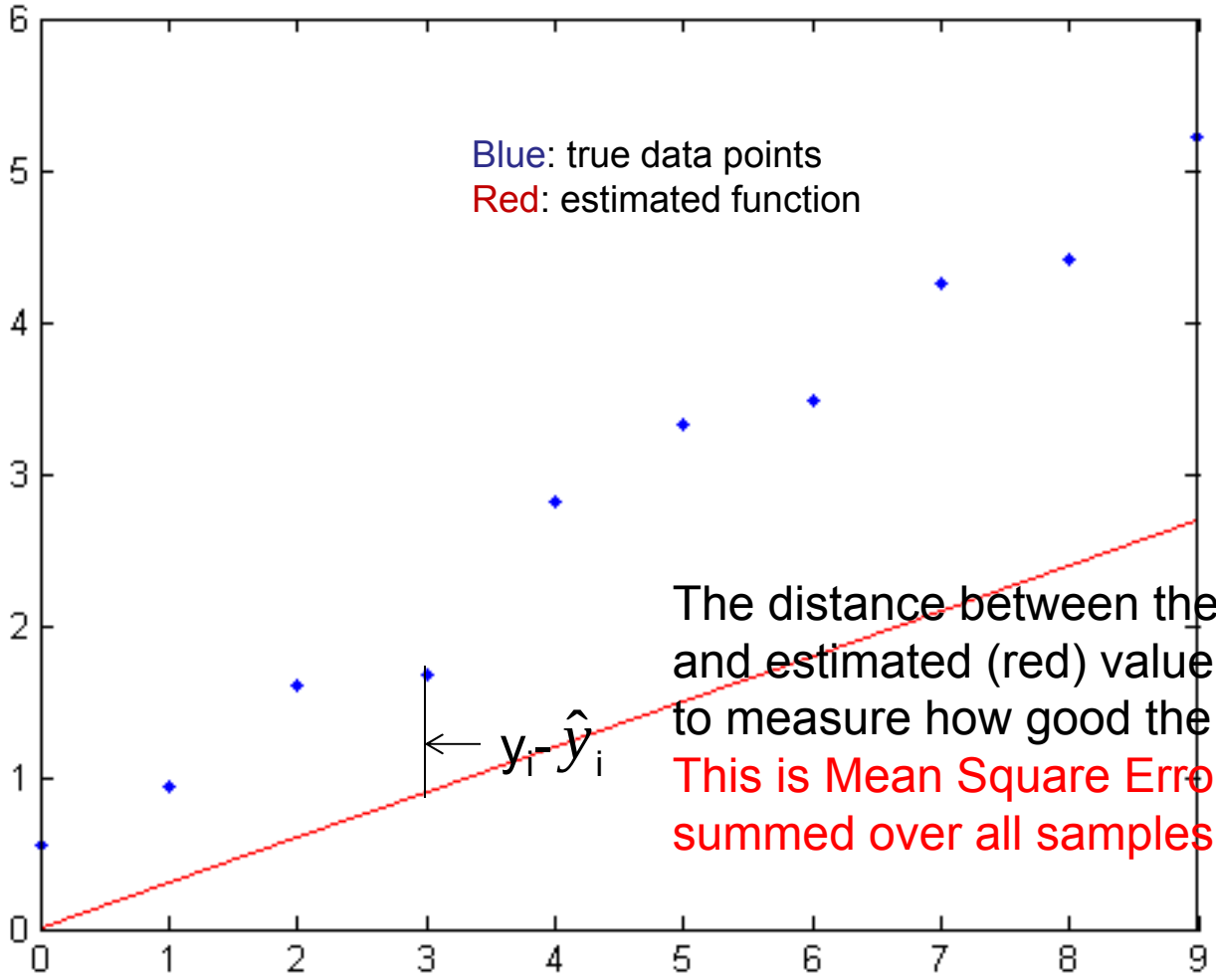
- Learning will be based on comparing y and $\hat{y}$



Linear regression: training data set

$$\mathbf{x} = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \\ \\ x^{(m)} \end{bmatrix} \mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \\ \\ y^{(m)} \end{bmatrix}$$

- Want to estimate $\boldsymbol{y}$ based on data $\mathbf{x}$.
- Given m training samples where x and y are known.
- If x_i has one variable pr. sample (e.g. one gray level), this is called univariate regression.



The distance between the true (blue) and estimated (red) value will be used to measure how good the fit is
This is Mean Square Error (MSE) if summed over all samples.

Where are we

- KNN (K- nearest neighbor)
- Crossvalidation
- Linear regression
- **Loss functions**
- **Minimize loss functions using gradient descent**
- Logistic regression

Error measure for learning linear regression: Mean square error(MSE)

- Mean square error over the training data set
- Training data: a set of m samples **x**={$x^{(i)}$,i,i==1..m}
- $x^{(i)}$ can consist of one of more variables/features, e.g. several measurements.

$$J(w,b) = MSE = \frac{1}{2m}\sum_{i=i}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

In vector form : $\frac{1}{2m}\left\|\hat{\mathbf{y}} - \mathbf{y}\right\|_2^2$ L2 - norm

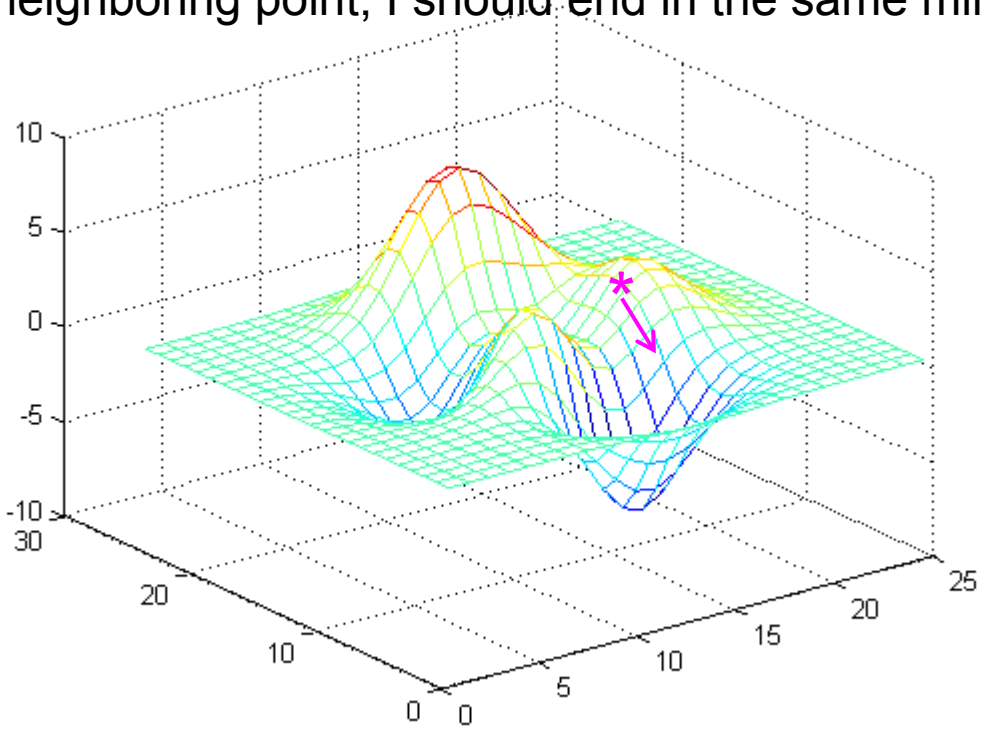
θ is the parameter we want to fit, the parameters of a line

Gradient descent minimization

- Let's see how gradient descent can be used to find w that mimize MSE.
- Read Section 4.3 in Deep Learning.

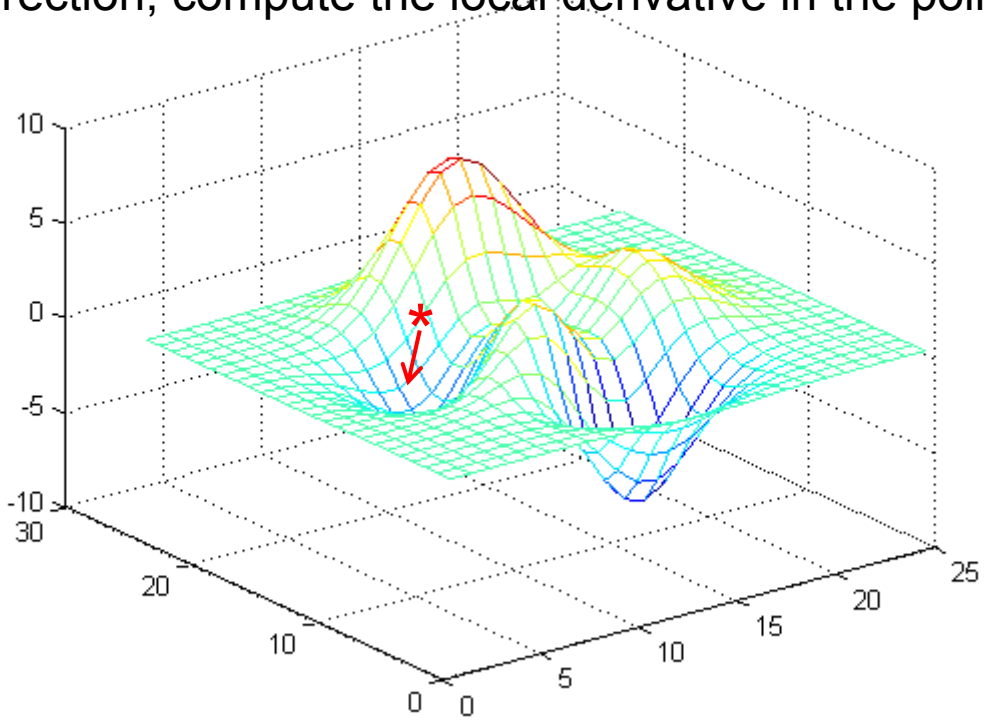
Gradient descent intuition

Start from a point and take a step downhill in the steepest possible direction
Repeat this until we end up in a local minimum
If I start from a neighboring point, I should end in the same minimum



Gradient descent intuition

If we start from a different point we might end up in another local minimum
For finding the direction, compute the local derivative in the point



Iterative minimization outline

- Have a function J(w,b) (can be generalized to more than two parameters)
- Want to find w,b that minimize J(w,b)
- Outline
  1. Start with some value of w,b (e.g. w=0,b =0)
  2. Compute J(w,b) for the given value of w,b and change w,b in a manner that will decrease J(w,b)
  3. Repeat step 2 until we hopefully end up in a minimum

Gradient descent principle

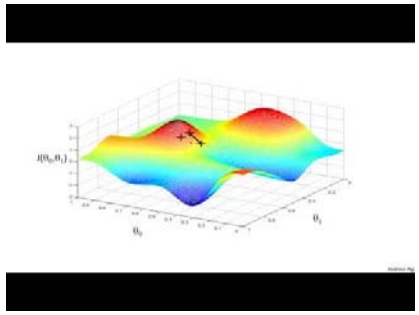
- Given a function J(w,b) of several variables.
- The directional derivative in a given direction is the slope of J(w,b) in that direction.
- To iteratively minimize f, we want to find the direction in which f decreases the fastest.
- This can be shown to be in **the opposite direction** as the gradient.

- **So we can minimize f by taking a step in the direction of the negative gradient**.
- The gradient descent propose a new point $x' = x - \varepsilon \nabla_x f(x)$ where ε is the **learning rate**.
- This is done iteratively in a number of iterations.
- If ε is too small, the algorithm converges too slow.
- It ε is too large, it may fail to converge, or diverge.

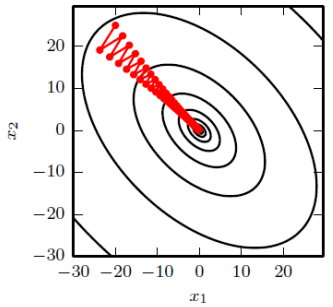
Back to linear regression

Model hypothesis: $\hat{y} = wx + b$

Example function



Countours of the function



Gradient descent for linear regression

- Let w and b be the unknown two parameters in the linear model:

$$\hat{y} = wx + b$$

- We want the minimize the criterion function, $J(w, b)$, by computing the derivate with respect to w and b, and set the derivative to 0.

$$J(w,b) = \frac{1}{2m}\sum_i (\hat{y}_i - y_i)^2 = \frac{1}{2m}\sum_i (wx_i + b\ - y_i)^2$$

- This is a quadratic (convex) function and we can find the global minima.
- Remark: in the literature you will see θ used as a vector of all parameters.

Gradient descent for linear regression Univariate x – a single feature/gray level

$$\frac{\partial}{\partial w} J(w,b) = \frac{\partial}{\partial w} \frac{1}{2m} \sum_i \left(w\ x_i + b - y_i \right)^2$$

$$= \frac{2}{2m} \sum_i \left(w\ x_i + b - y_i \right) x_i$$

Here we use the chain rule

$$\frac{\partial}{\partial b} J(w,b) = \frac{\partial}{\partial w} \frac{1}{2m} \sum_i \left(w\ x_i + b - y_i \right)^2$$

$$= \frac{2}{2m} \sum_i \left(w\ x_i + b - y_i \right)$$

- Here we sum the gradient over all x_i in the training data set.
- This is called **batch gradient descent.**

Gradient descent algorithm for one variable x

Gradient descent

repeat until convergence

Linear regression model

y=wx+b

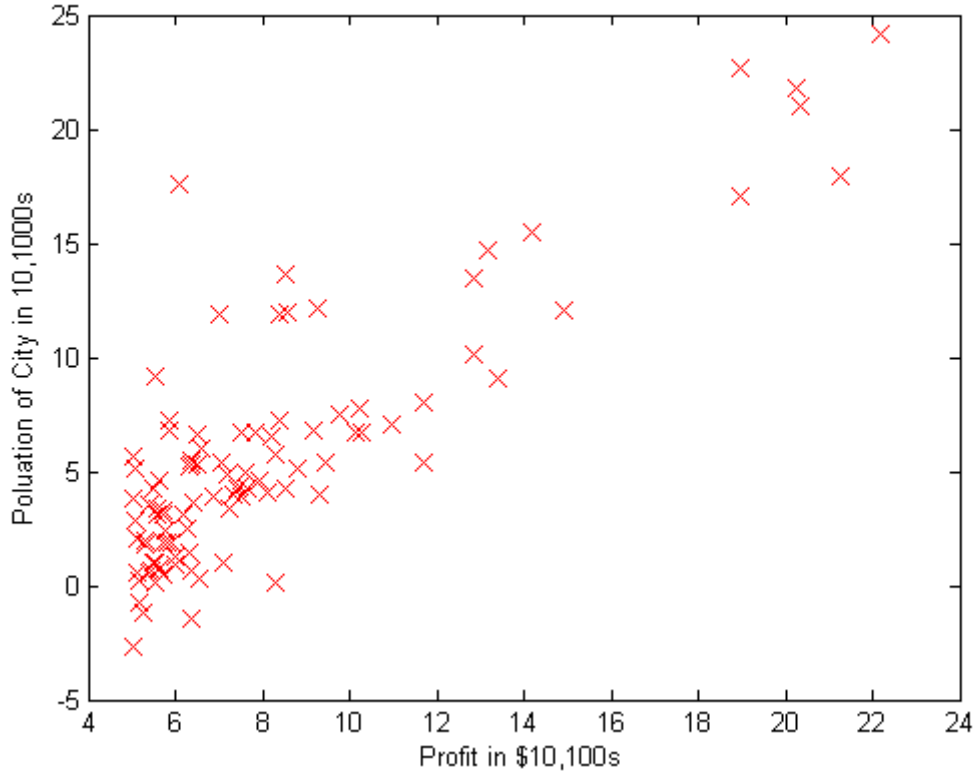
$$w = w - \varepsilon \frac{\delta J}{\delta w} =$$

$$= w - \varepsilon \frac{1}{m} \sum_i \left(w x^{(i)} + b \right) x^{(i)}$$

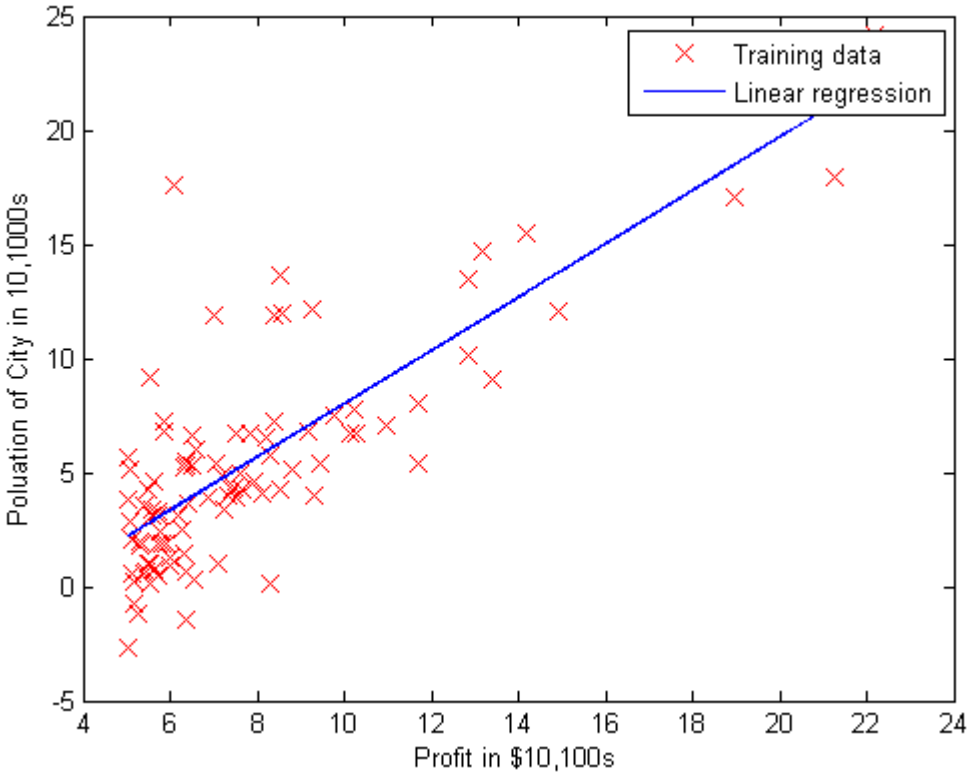
$$b = b - \varepsilon \frac{\delta J}{\delta b}$$

$$= \text{b} - \varepsilon \frac{1}{m} \sum_i \left(w x^{(i)} + b \right)$$

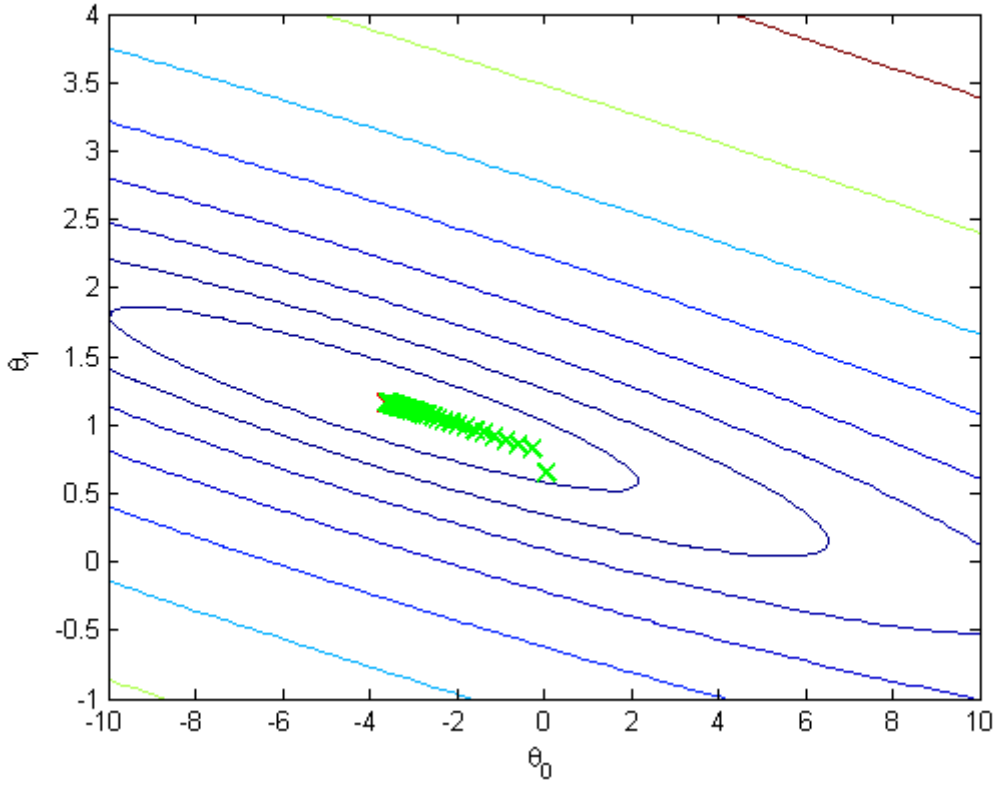
Example of J(w,b) for a general line (y= w x+ b)



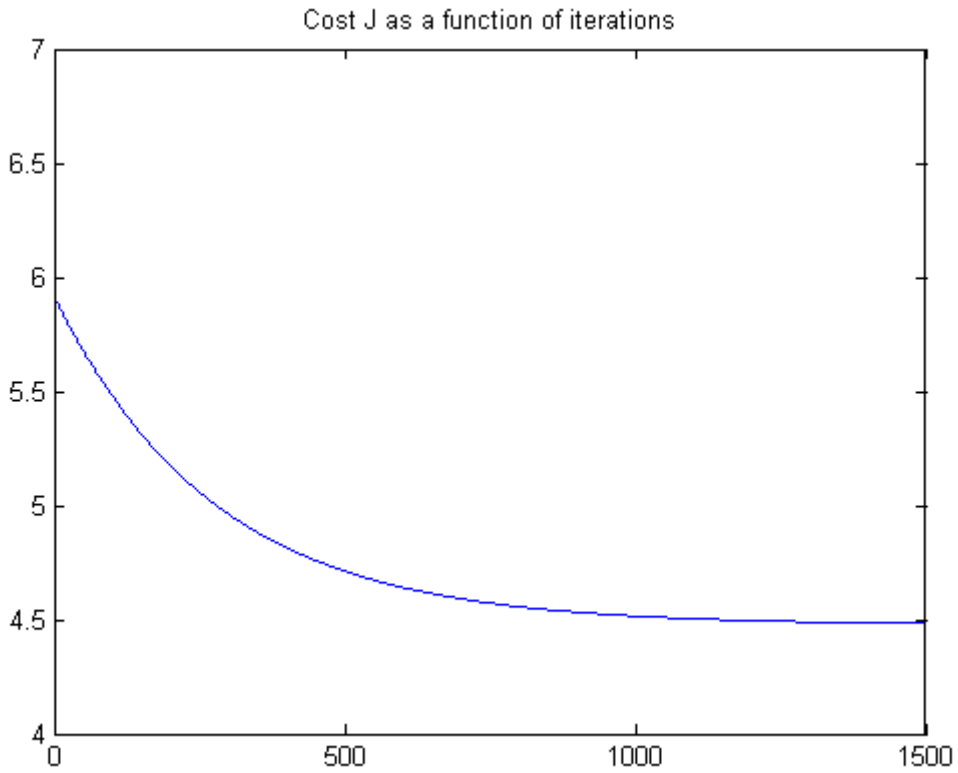
The result from gradient descent



The value of J overlaid the values of w,b after every 50th iteration



J as a function of iterations



Implementing gradient descent

$$\mathrm{w} = w - \varepsilon \frac{1}{m} \sum_i \left(w x^{(i)} + b \right) x^{(i)}$$

$$\mathrm{b} = \mathrm{b} - \varepsilon \frac{1}{m} \sum_i \left(w x^{(i)} + b \right)$$

- The sum over all samples $x^{(i)}$ can be done on vectors using np.sum() and other vector operations.

Gradient descent in practice: finding the learning rate

- How do we make sure that the optimization runs correctly?
  - Make sure J decreases! Plot J as a function of the number of iterations
    - Computation of J should also be vectorized
    - $\mathrm{J}(\mathrm{w},\mathrm{b}) = \frac{1}{m}\sum_i \left(wx^{(i)} + b - y^{(i)}\right)^2$
  - If ε is too small: slow convergence
  - If ε is too large: may not decrease, may not converge
  - ε is a number between 0 and 1, often close to 0 (try 0.001,…0.01,….0.1,….1)

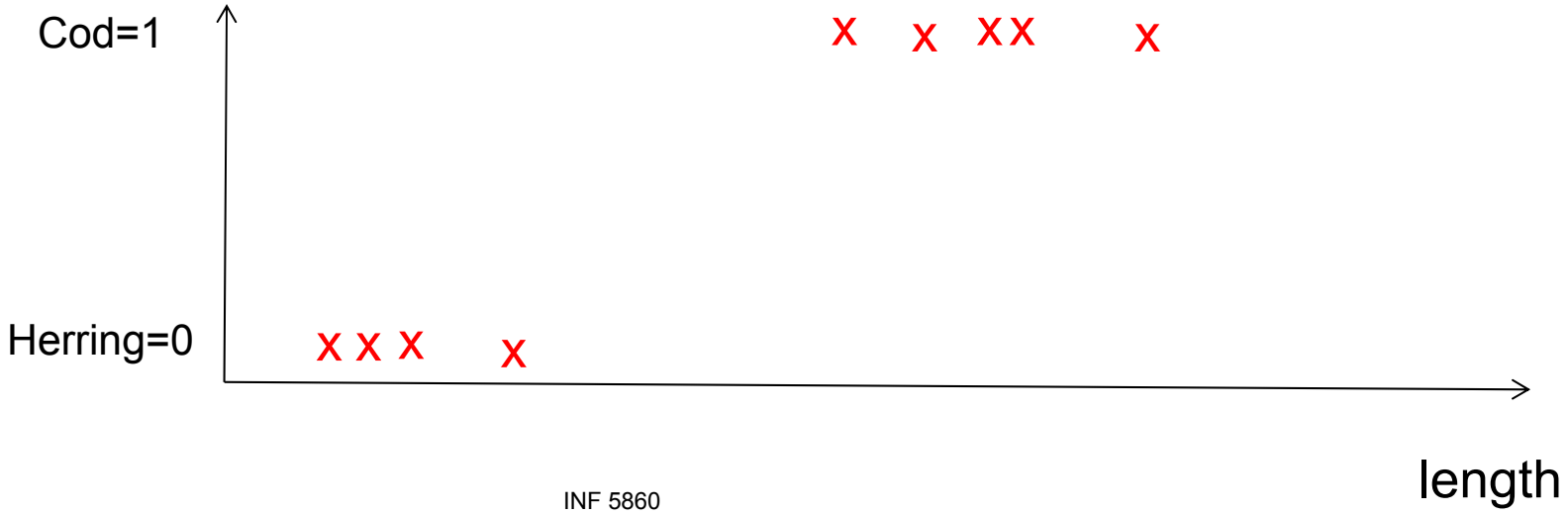
Where are we

- KNN (K- nearest neighbor)
- Crossvalidation
- Linear regression
- Loss functions
- Minimize loss functions using gradient descent
- **Logistic regression**

Introduction to logistic regression

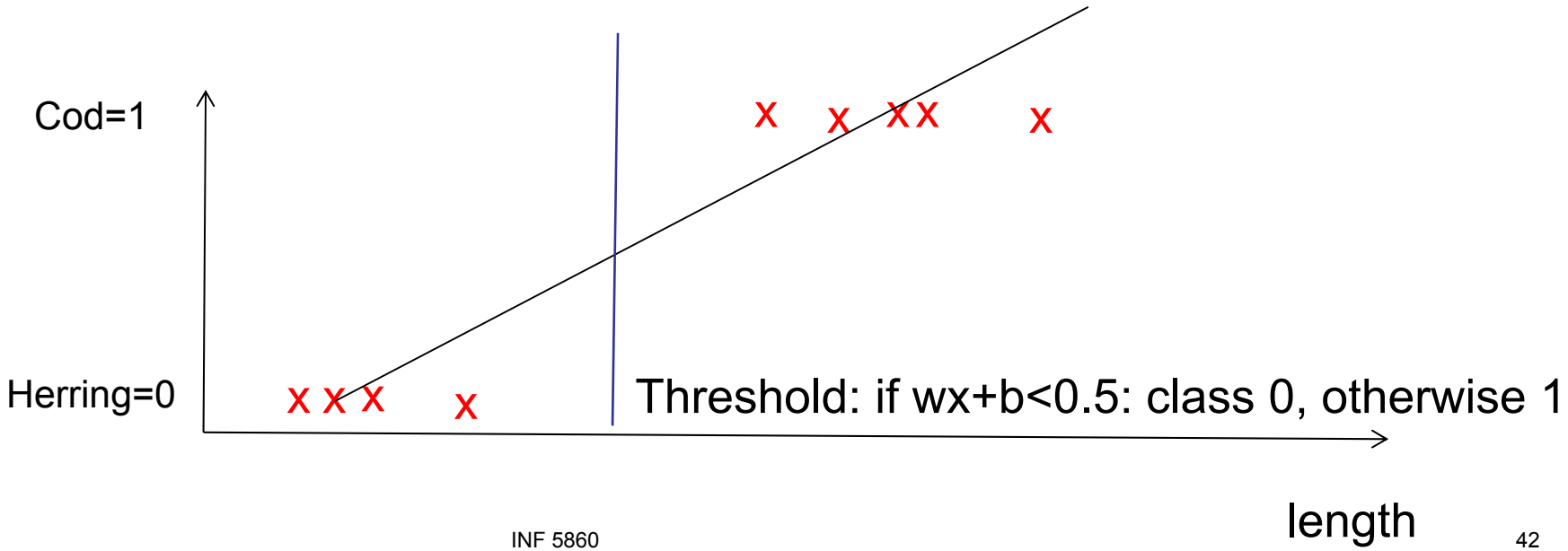
- Let us show how a regression problem can be transformed into a binary (2-class) classification problem using a nonlinear loss function.
- Then generalize to multiple classes (next week).

Introduction

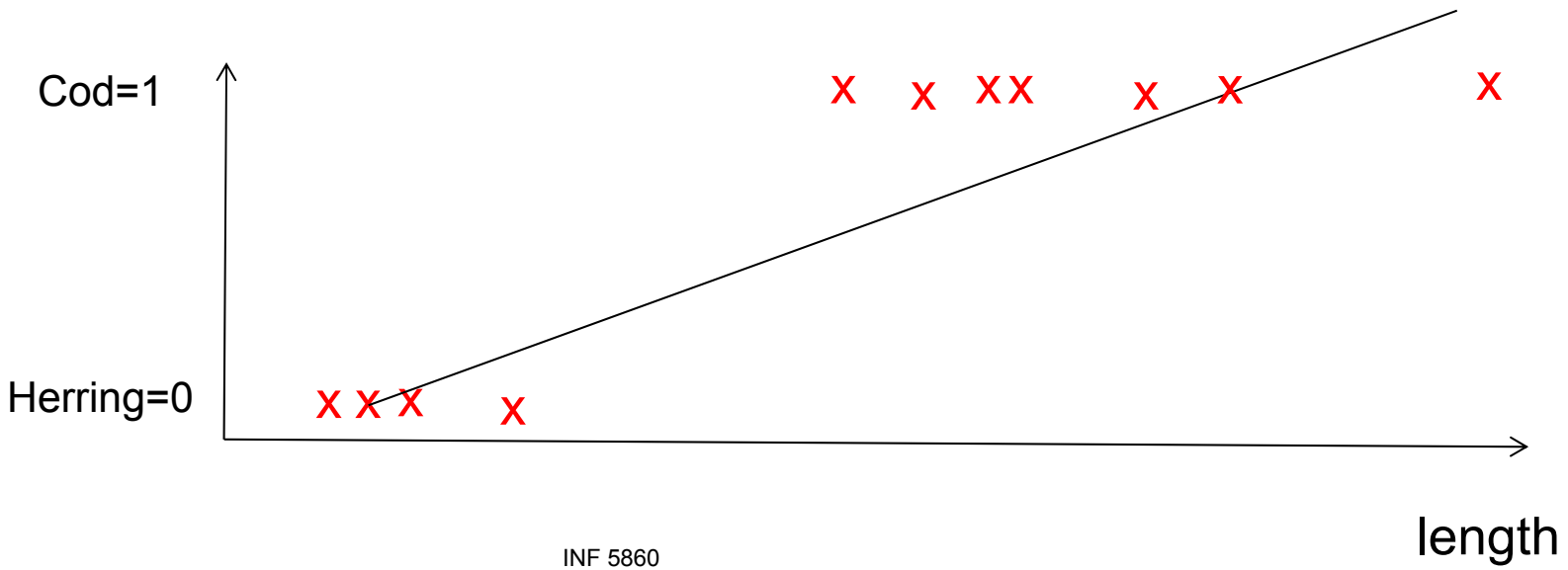


What would linear regression give?

- Maybe we would threshold this?

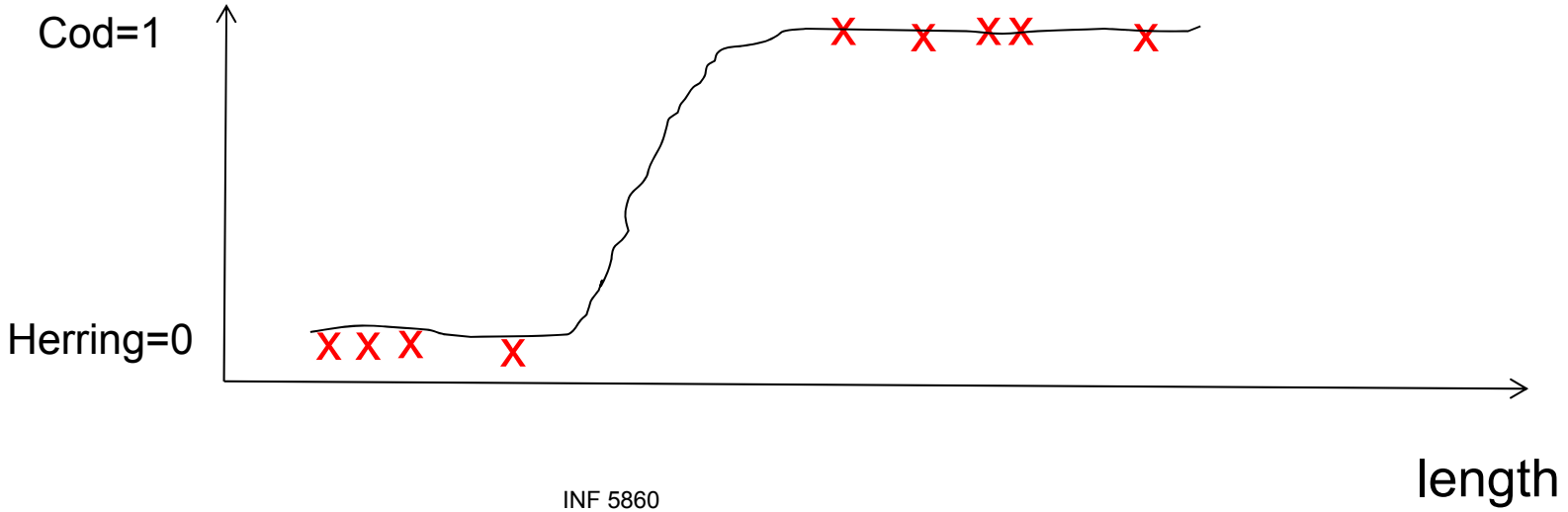


What would linear regression give?



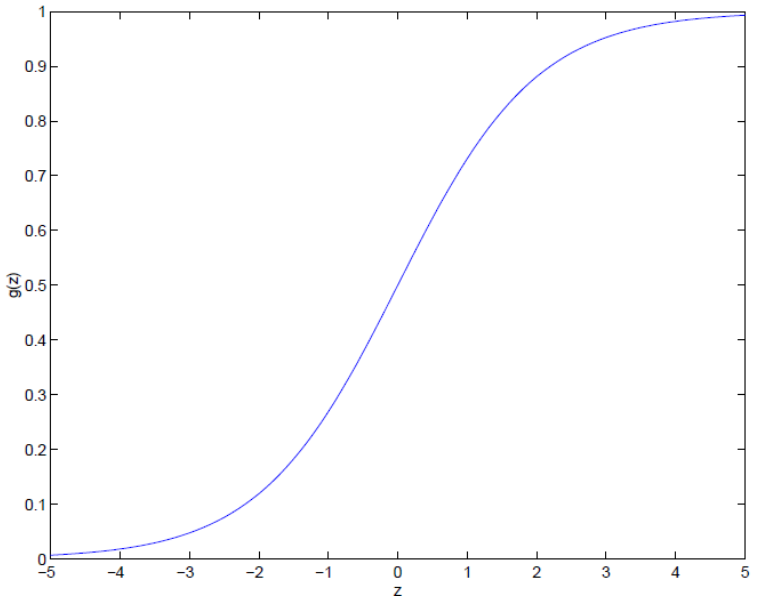
What if we fitted it to a function f(x) that is close to either 0 or 1?

- Hypothesis $h_\theta(x)$ is now a non-linear function of x
  Classification: y=0 or 1
  Threshold $h_\theta(x)$: if $h_\theta(x)>0.5$: set y=1, otherwise set y=0
- Desirable to have $h_\theta(x)\leq 1$



Logistic regression model

- Want 0≤ $h_\theta(x)$≤1 (binary problem)
- Let
- $h_\theta(x) = g(wx + b)$
- $g(z) = \frac{1}{1+e^{-z}}$
- $h_\theta(x) = \frac{1}{1+e^{-(wx+b)}}$
- g(z) is called the sigmoid function



Decisions for logistic regression

- Decide y=1 if $h_\theta(x)$> 0.5, and y=0 otherwise

$$h_\theta(x) = g(wx + b)$$

$$h_\theta(X) = g\left(\theta^T x\right)$$

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

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

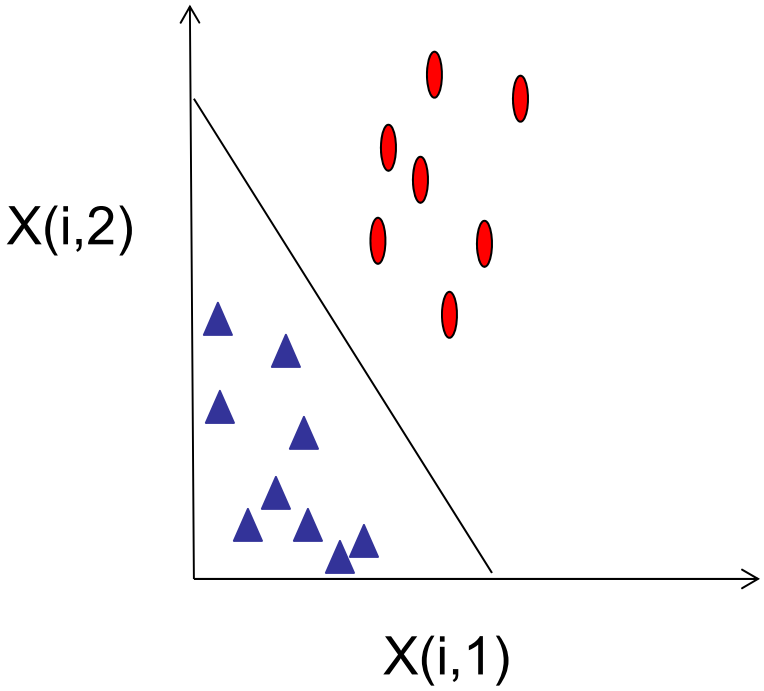
- g(z)>0.5 if z>0
  - wx+b>0

g(z)<0 if z<0

wx+b<0

Here the compact notation θ means the vector of parameters [w,b]

An example with 2 features



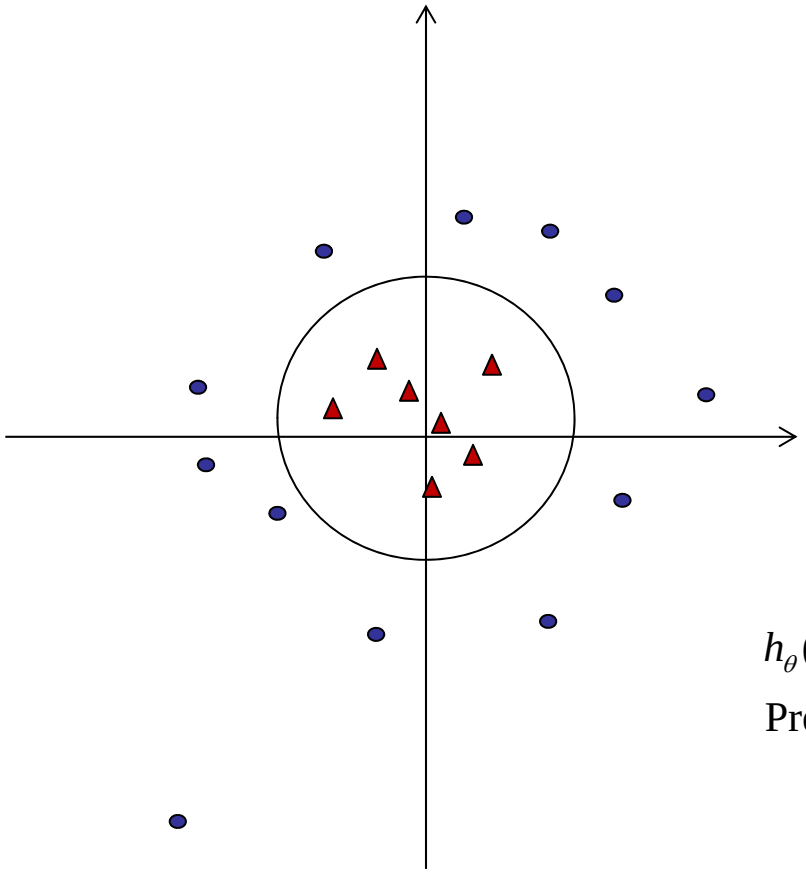
$h_\theta(X)$=g(b+w1X(i,1)+w2X(i,2))

Predict y=1 if X(i,1)+2X(i,2)-4≥0

Decision boundary X(i,1)+2X(i,2)-4=

If we KNOW b, w1 and w2, classification is based on which **side** of the boundary we are on.

Nonlinear boundary possible?



- The basic model gives **a linear boundary.**
- To get a non-linear boundary needed for this example, higher order terms must be added

$$h_\theta(x) = g(\theta_0 + \theta_1 X(i,1) + \theta_2 X(i,2)) + \theta_3 X(i,1)^2 + \theta_4 X(i,2)^2$$

$$\text{Predict y} = 1 \text{ if } -1 + X(i,1)^2 + X(i,2)^2 \geq 0$$

Introducing a logistic cost function

- Training set

$$X = \begin{bmatrix} X(0,0) & X(0,1) & .. & X(0,n) \\ X(1,0) & X(1,1) & & X(1,n) \\ & & & \\ X(m,0) & X(m,1) & & X(m,n) \end{bmatrix} \qquad y = \begin{bmatrix} y(1) \\ y(2) \\ \\ y(m) \end{bmatrix}$$

$X(:, j)$ Column j : all samples for feature j

X(i,:) Row i : all features for sample i

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

- How do we set the parameter vector to have high classification accuracy?

Logistic regression cost

Minimize

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

Due to the sigmoid function g(z), this is a non-quadratic function, and non-convex.

Set

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

Check that this function does what we want

$\text{Cost} = 0 \text{ if } y = 1 \text{ and } h_\theta(x) = 1$ Correct classification, no cost

$\text{If } y = 1 \text{ and } h_\theta(x) \to 0 : \text{Cost} \to \infty$ Wrong classification, high cost

$\text{Cost} = 0 \text{ if } y = 0 \text{ and } h_\theta(x) = 0$ Correct classification, no cost

$\text{If } y = 0 \text{ and } h_\theta(x) \to 1 : \text{Cost} \to \infty$ Correct classification, no cost

Mimick a probability

Cost function for logistic regreesion

- We have two classes, 1 and 0.
- Let us use a probabilistic model
  Let the parameters be $\theta=[w_1,\ldots.w_{nk},b]$ if we have nk features.
- $P(y=1|x,) = h_\theta(x)$
- $P(y=0,x)= (1- h_\theta(x))$
- This can be written more compactly as

  $p(y|x, \theta) = h_\theta(x)^y(1- h_\theta(x))^{1-y}$

Cost function for logistic regreesion

- The likelihood of the parameter values is

$$
\begin{aligned}
L(\theta) &= p(\vec{y} \mid X; \theta) \\
&= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) \\
&= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}}
\end{aligned}
$$

- It is easier to maximize the log-likelihood

$$
\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))
\end{aligned}
$$

- We will use gradient descent to maximize this, taking a step in the positive direction since we are maximizing, not minimizing

Computing the gradient of the likelihood function

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j}\ell(\theta) &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial}{\partial \theta_j} g(\theta^T x) \\
&= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right) g(\theta^T x)(1-g(\theta^T x)\frac{\partial}{\partial \theta_j}\theta^T x \\
&= \left(y(1-g(\theta^T x)) - (1-y)g(\theta^T x)\right) x_j \\
&= \left(y - h_\theta(x)\right) x_j
\end{aligned}
$$

Here, we used the fact the g'(z)=g(z)(1-g(z))

Gradient descent of J(θ)=-L(θ)

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y(i)\log h_\theta(X(i,:) + (1-y(i))\log(1-h_\theta(X(i,:))\right]$$

To find θ: find θ that minimize J(θ) using gradient descent

Repeat :

$$\theta_j = \theta_j - \varepsilon\frac{\partial}{\partial\theta_j}$$

$$\theta_j - \varepsilon\frac{1}{m}\sum_{i=1}^{m}((h_\theta(X(i,:)) - y(i))X(i,j))$$

This algorithm looks similar to linear regression, but now

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

Mandatory exercise 1

- Available in few days.
- Complete two notebooks
  - knn.ipynb and additional functions
    - Can start as soon the the exercise is available
  - softmax.ipynb and additional functions

Next week

- Generalizing logistic regression to multiple classes
- Adding regularization
- Using it to classify images