IN4080 – 2019 FALL
NATURAL LANGUAGE PROCESSING

Jan Tore Lønning

Logistic regression

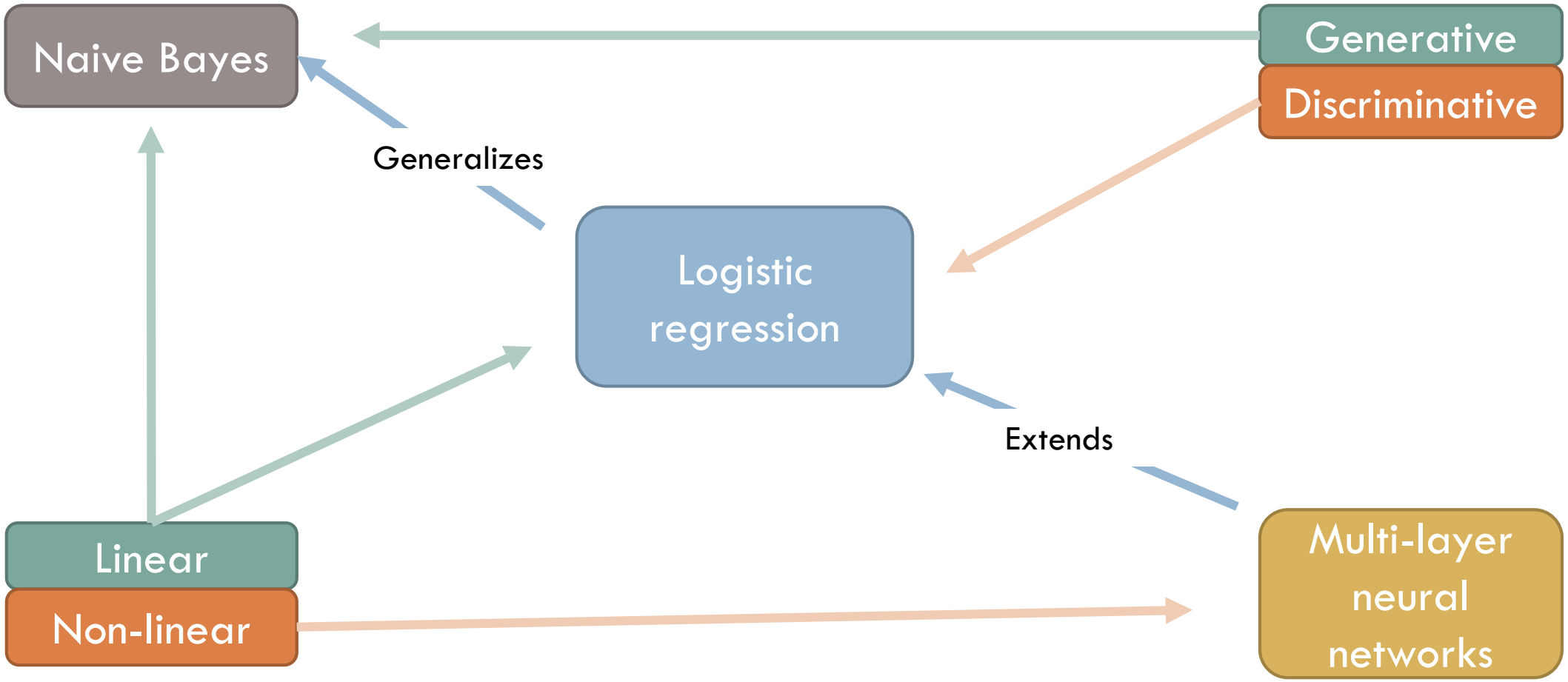
Lecture 6, 26 Sept

Logistic regression

In natural language processing, logistic regression is the baseline supervised machine learning algorithm for classification, and also has a very close relationship with neural networks.

(J&M, 3. ed., Ch. 5)

Relationships

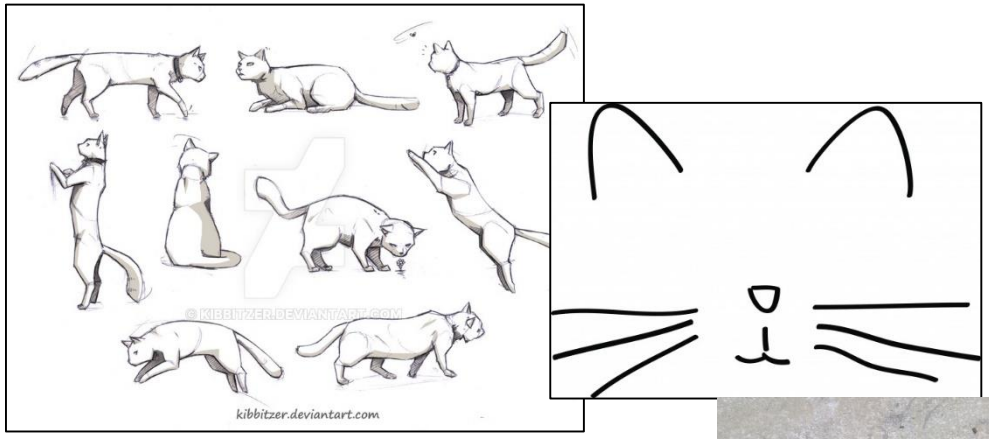


Today

- Generative and discriminative classifiers
- Representing categorical features
- Linear classifiers
- Linear regression
- Logistic regression
- Training the logistic regression classifier
- Multinomial Logistic Regression
- Application to tagging
- Comparing to NB

Drawing a cat – generative model

- Choose silouette
- Choose face
- Choose color, e.g.:
  - P(red) = 0.1, P(black&white) = 0.3, etc
- Choose tail length
  - P(long) = 0.2, P(medium)=0.4, P(no tail)=0.0…

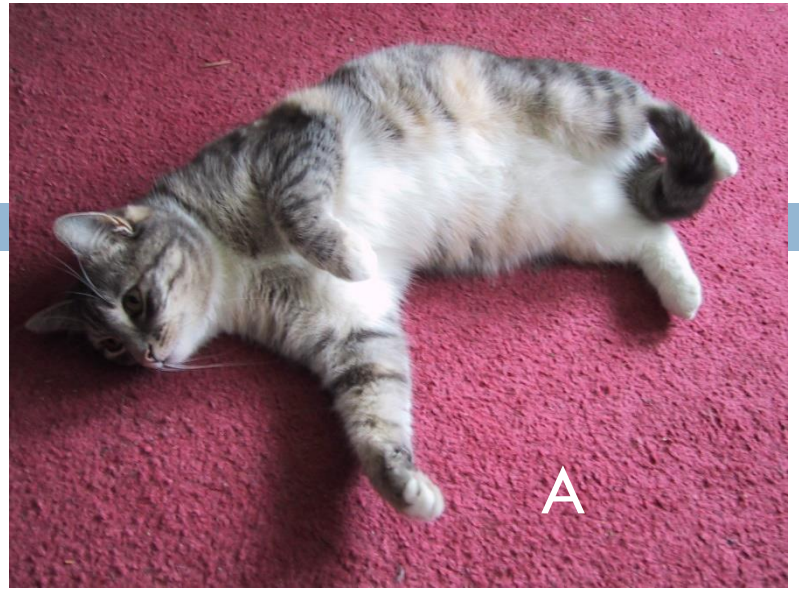






Generating cats and dogs

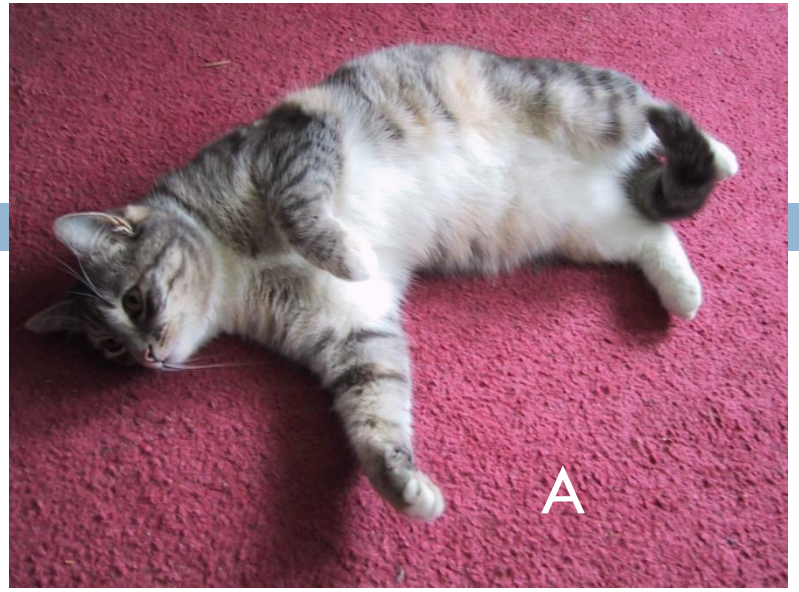
- What is the probability that it will look like A?
- Make a similar "dog"-model
- Decide:
  - What is the probability of (B) in the cat model?
  - What is the probability of (B) in the dog model?
- Choose on the basis of this and the general distribution of dog and cat pictures.





Discriminative model

- Does not tell us how probable (B) is in the cat or in the dog model.
- It only tells how probable it is that (B) is a cat compared to a dog.
- The model can concentrate on discriminative features, like:
  - whiskers
  - claws
  - eyes





Naïve Bayes and HMM are generative

- Generative
  - Using Bayes theorem: $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}$
  - Observe that $P(w_1^n|t_1^n)P(t_1^n) = P(w_1^n, t_1^n)$
  - A probability distr. over all tagged sentences is part of the model
- Discriminative
  - Considers $\underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n)$ directly

    without knowing anything about $P(w_1^n, t_1^n)$

Today

- Generative and discriminative classifiers
- Representing categorical features
- Linear classifiers
- Linear regression
- Logistic regression
- Training the logistic regression classifier
- Multinomial Logistic Regression
- Application to tagging
- Comparing to NB

Categories as numbers

- In the naive Bayes model we could handle categorical values directly, e.g., characters:
  - What is the probability that c_*n* = '*z*'
- But many classifier can only handle numerical data
- How can we represent categorical data by numerical data?
- (In general, it is not a good idea to just assign a single number to each category: ~~'a'→1, 'b'→2, 'c'→ 3, ...~~)

Data representation

Assume the following example

	4 different featues				Classes
feature	f1	f2	f3	f4	
type	cat	cat	Bool (num)	num	
Value set	a, b, c	x, y	True, False	0, 1, 2, 3, …	Class1, class2

Representation in NLTK

```
[({'f1': 'a', 'f2': 'y', 'f3': True, 'f4': 5}, 'class_1'),
 ({'f1': 'b', 'f2': 'y', 'f3': False, 'f4': 2}, 'class_2'),
 ({'f1': 'c', 'f2': 'x', 'f3': False, 'f4': 4}, 'class_1')]
```

3 training instances

4 features

class

One-hot encoding

feature 1				feature 2	
a	b	c		x	y
(1,0,0)	(0,1,0)	(0,0,1)		(1,0)	(0,1)

- Represent categorical variables as vectors/arrays of numerical variables

Representation in scikit: "one hot" encoding

NLTK

```
[({'f1': 'a', 'f2': 'y', 'f3': True, 'f4': 5}, 'class_1'),
 ({'f1': 'b', 'f2': 'y', 'f3': False, 'f4': 2}, 'class_2'),
 ({'f1': 'c', 'f2': 'x', 'f3': False, 'f4': 4}, 'class_1')]
```

3 training instances

4 features

class

scikit

```
X_train:
array([[ 1.,  0.,  0.,  0.,  1.,  1.,  5.],
       [ 0.,  1.,  0.,  0.,  1.,  0.,  2.],
       [ 0.,  0.,  1.,  1.,  0.,  0.,  4.]])
```

7 features

```
train_target: ['class_1', 'class_2', 'class_1']
```

3 training instances

3 corresponding classes

One-hot encoding		
a	b	c
[1, 0, 0]	[0, 1,]	[0, 0, 1

Converting a dictionary

- We can construct the data to scikit directly
- Scikit has methods for converting Python-dictionaries/NLTK-format to arrays

```
train_data = [inst[0] for inst in train]
train_target = [inst[1] for inst in train]
v = DictVectorizer()
X_train=v.fit_transform(train_data)

X_test=v.transform(test_data)
```

1. Constructs (=fit) repr. format
2. Transform

Transform
Use same v as for train

Multinomial NB in scikit

- We can construct the data to scikit directly
- Scikit has methods for converting text to bag of words arrays

```
train_data=["en rose er en rose",
                    "anta en rose er en fiol"]
v = CountVectorizer()
X_train=v.fit_transform(train_data)


print(X_train.toarray())
[[0 2 1 0 2]
[1 2 1 1 1]]
```

- Positions corresponds to [anta, en, er, fiol, rose]

Sparse vectors

- One hot encoding uses space
- 26 English characters:
  - Each is represented as a vector with 25 ‘0’-s and a singel ‘1’
- Bernoulli NB text. classifier with 2000 most frequent words
  - Each word represented by a vector with 1999 ‘0’-s and a singel ‘1’.
- scikit-learn uses internally a dictionary-like representation for these vectors, called ”sparse vectors”

Today

- Generative and discriminative classifiers
- Representing categorical features
- Linear classifiers
- Linear regression
- Logistic regression
- Training the logistic regression classifier
- Multinomial Logistic Regression
- Application to tagging
- Comparing to NB

Types of supervised learning

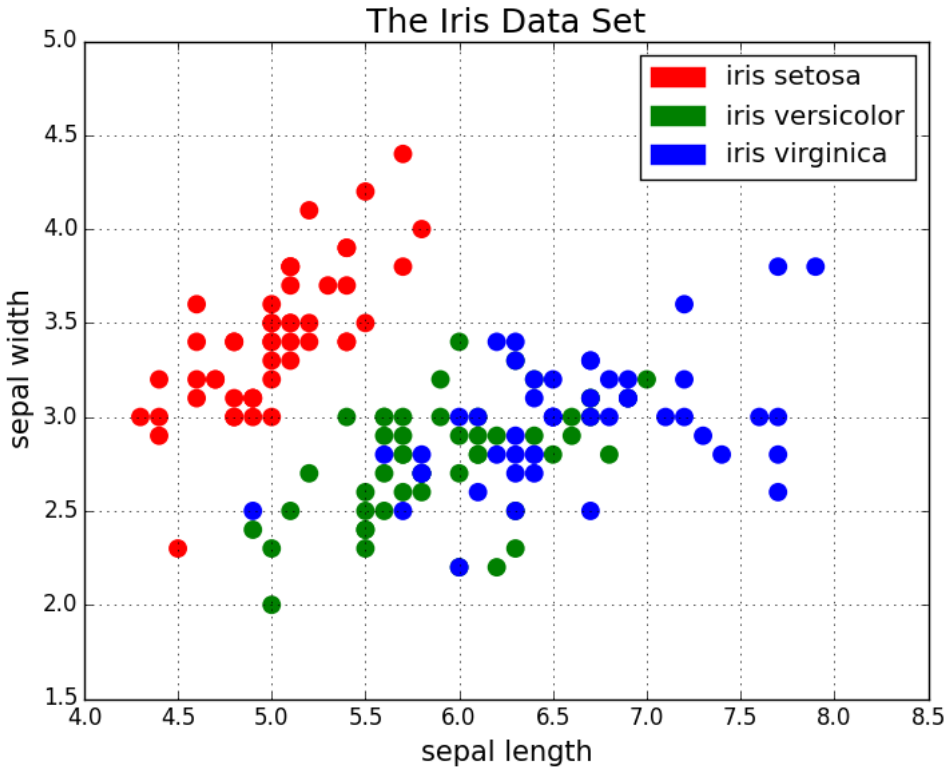
		Methods	
		Linear	Non-linear
Tasks	Regression: Predict a numerical value for an observation	- Linear regression	
	Classification: Assign a class from a set of categorical classes	- Perceptron - Naive Bayes - Logistic regression - Support vector machines (SVM) - …	- (SVM+)kernels - Multi-layer neural nets

Machine learning

- In NLP, we often consider
  - thousands of features (dimension)
  - categorical data
- These are difficult to illustrate by figures
- To understand ML algorithms
  - it easier to use one or two features, 2-3 dimensions, to be able to draw figures
  - and then to use numerical data, to get non-trivial figures

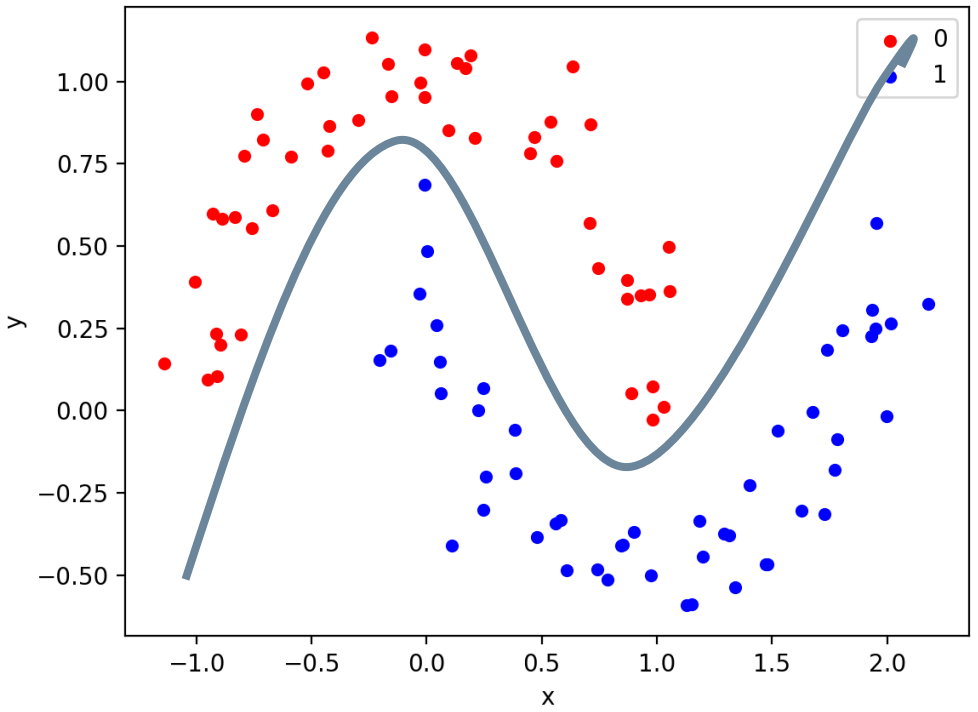
Scatter plot example

- Two numeric features
- Three classes
- We may indicate the classes by colors or symbols



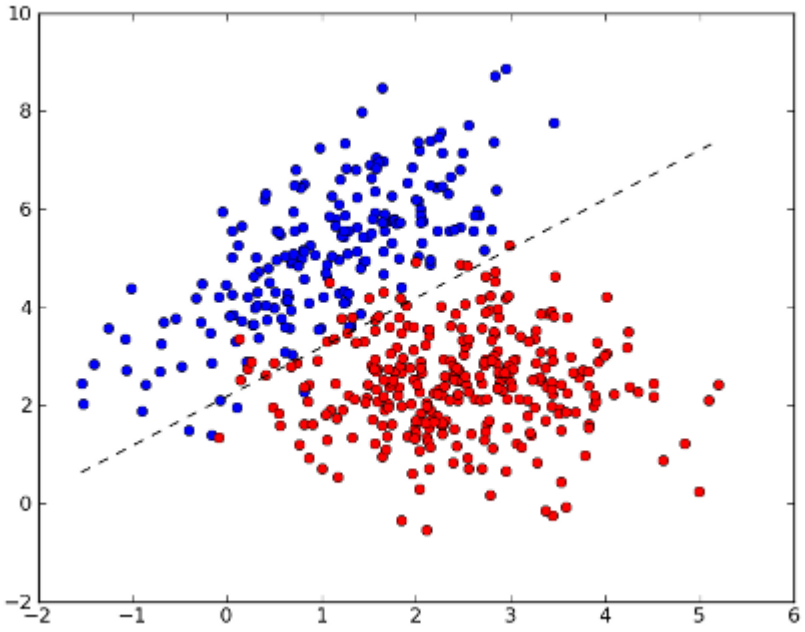
Classifiers – two classes

- Many classification methods are made for two classes
  - And then generalizes to more classes
- The goal is to find a curve that separates the two classes
- With more dimensions: to find a (hyper-)surface



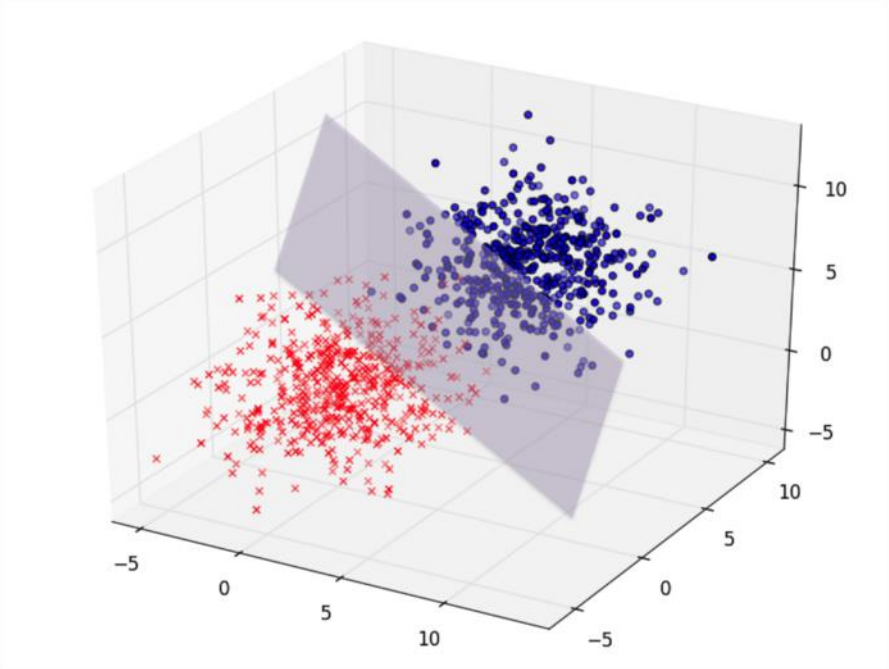
Linear classifiers

- Linear classifiers try to find a straight line that separates the two classes (in 2-dim)
- The two classes are linearly separable if they can be separated by a straight line
- If the data isn't linearly separable, the classifier will make mistakes.
- Then: the goal is to make as few mistakes as possible



More dimensions

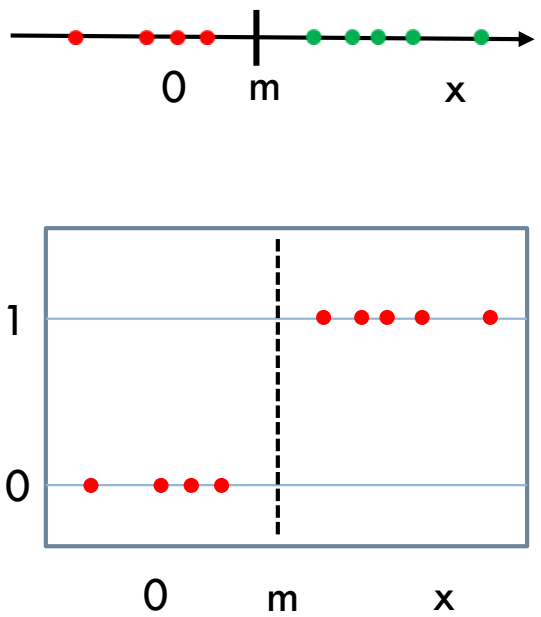
- In a 3 dimensional space (3 features) a linear classifier corresponds to a plane
- In a higher-dimensional space it is called a hyper-plane



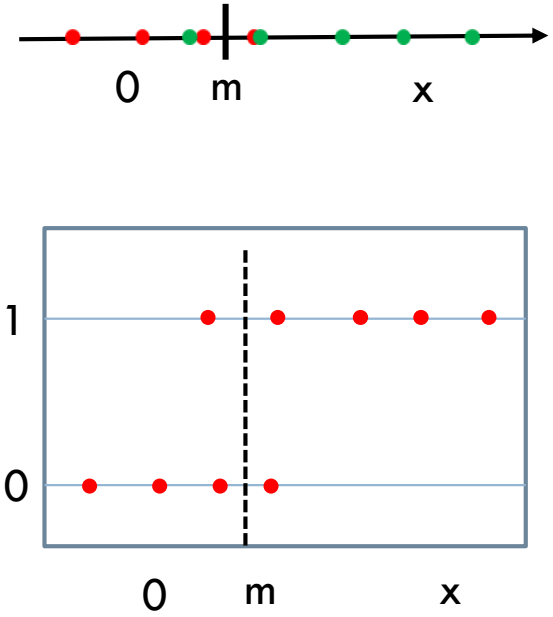
One-dimensional classification

- A linear separator is simply a point
- An observation is classified as
  - class 1 iff x>m
  - Class 0 iff x<m

Data set 1:
linerarly separable

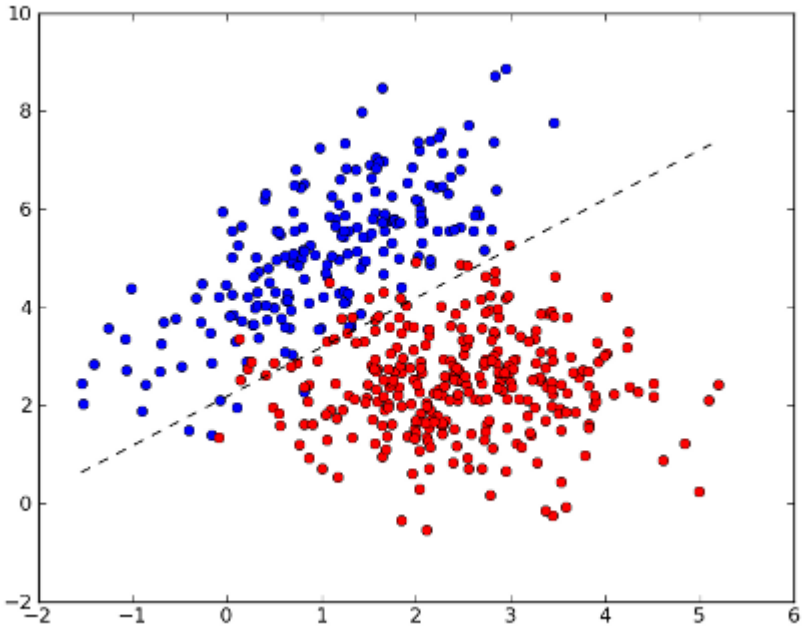


Data set 2:
not linerarly separable



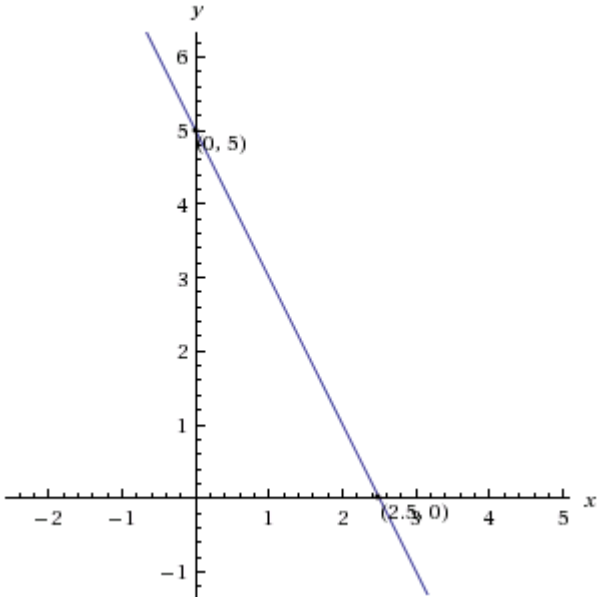
Linear classifiers: two dimensions

- a line has the form $ax+by+c=0$
- $ax + by < -c$ for red points
- $ax + by > -c$ for blue points



Geometry: lines

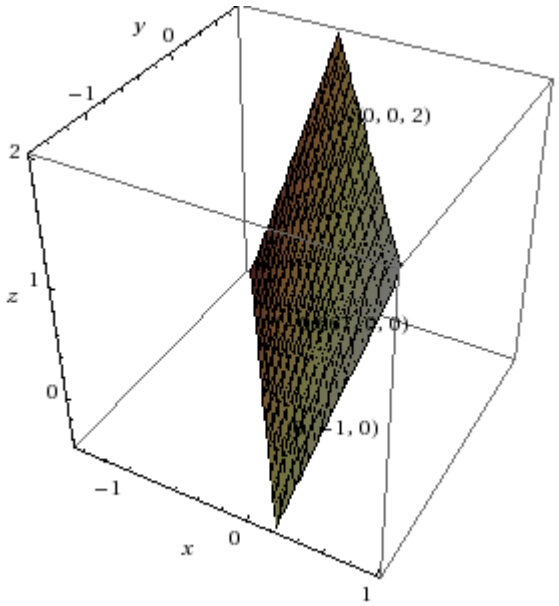
- Descartes
  - (1596-1650)
- Line:
- $ax + by + c = 0$
- If $b \neq 0$:
  - $y = mx + n$
  - $n = - c/b$ is the intercept with the y-axis
  - $m = -a/b$ is the slope
- A point = intersection of two lines

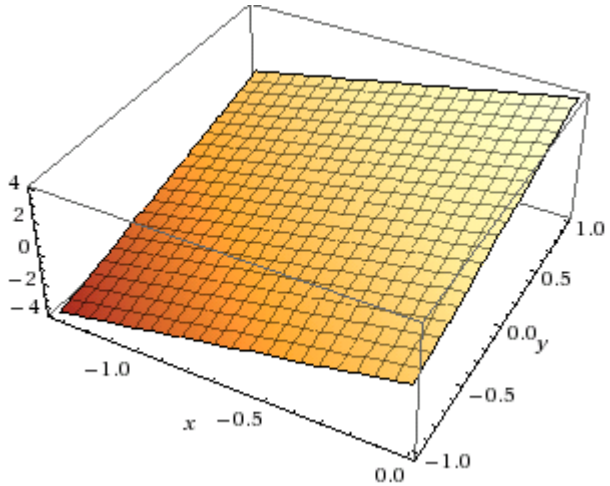


- $y = - 2x + 5$
- $4x + 2y - 10 = 0$

Geometry: planes

- Plane:
- $ax + by + cz + d = 0$
- If $c \neq 0$:
  - $z = mx + ny + n$
- A line is
  the intersection of two planes





- $3x + 2y - z + 2 = 0$
- $z = 3x + 2y + 2$

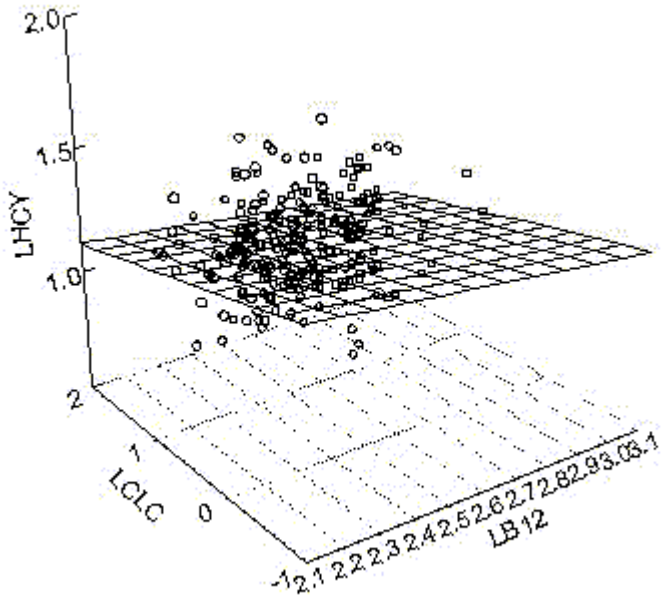
http://www.univie.ac.at/future.media/moe/galerie/geom2/geom2.html#eb

Hyperplanes

- Generalizes to higher dimensions
- In n-dimensional space $(x_1, x_2, \ldots, x_n)$:
  - Points satisfying:
- $w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n = 0$
  - for any choice of $w_0, w_1, w_2, \ldots w_n$
  - where not all of $w_1, w_2, \ldots w_n = 0$
- is called a hyper-plane
- (In machine learning) the same as the intersection of two hyper-planes in n+1 dimensional space:
  - $w_0x_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$
  - $x_0 = 1$

Linear classifiers: n dimensions

- A hyperplane has the form
- $\sum_{i=1}^{n} w_i x_i + w_0 = 0$
- which equals $\sum_{i=0}^{n} w_i x_i = (w_0, w_1, \ldots, w_n) \cdot (x_0, x_1, \ldots, x_n) = \vec{w} \cdot \vec{x} = 0$,
  - assuming $x_0 = 1$



- An object belongs to class C iff

$$\sum_{i=0}^{n} w_i x_i = \vec{w} \cdot \vec{x} > 0$$

- and to not C, otherwise

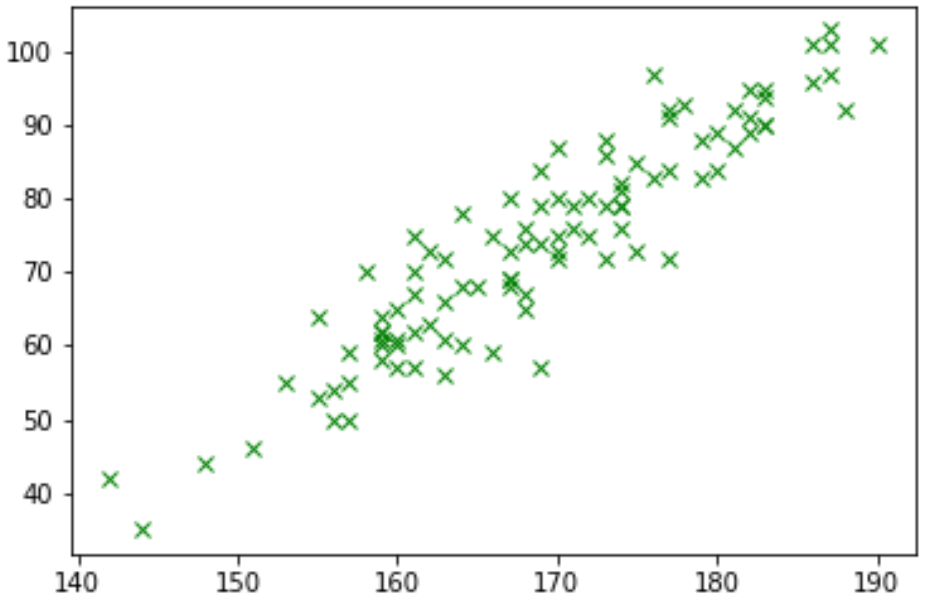
Linear classifiers

- There are many types of classifiers that are basically linear
- They depart from each other with respect to
  - How they see the feature set
  - What they consider the best separator

Today

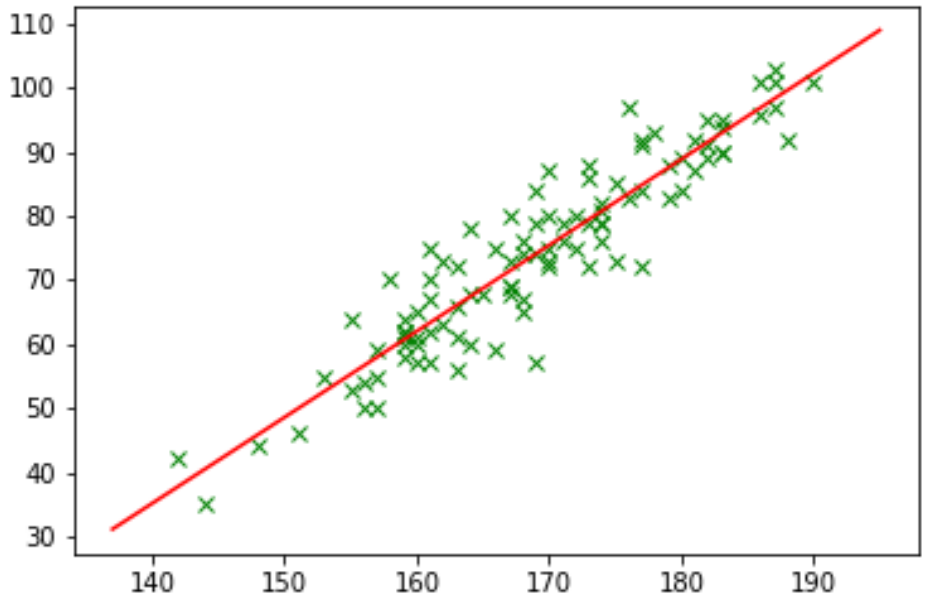
- Generative and discriminative classifiers
- Representing categorical features
- Linear classifiers
- Linear regression
- Logistic regression
- Training the logistic regression classifier
- Multinomial Logistic Regression
- Application to tagging
- Comparing to NB

Linear regression



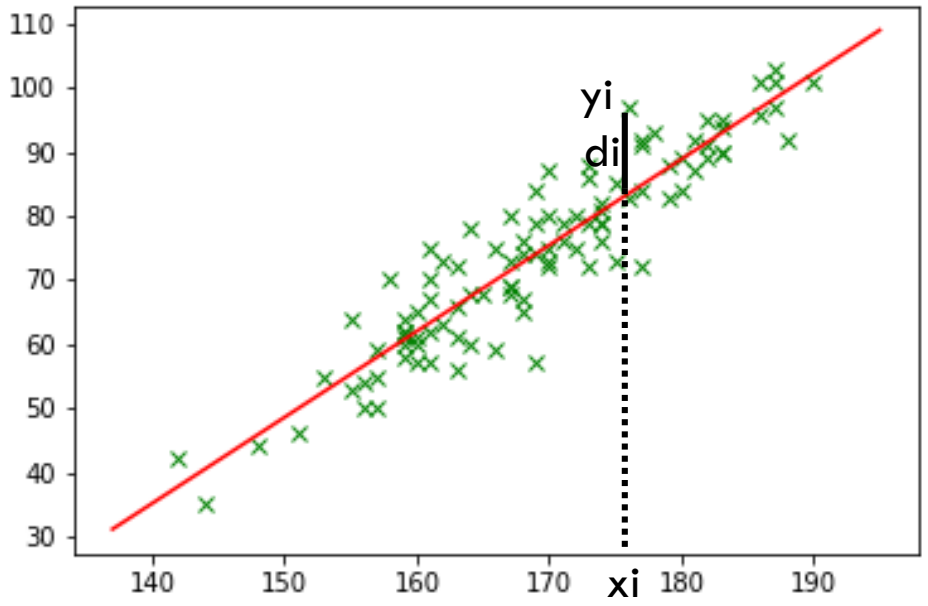
- Data:
  - 100 males: height and weight
- Goal:
  - Guess the weight of other males when you only know the height

Linear regression



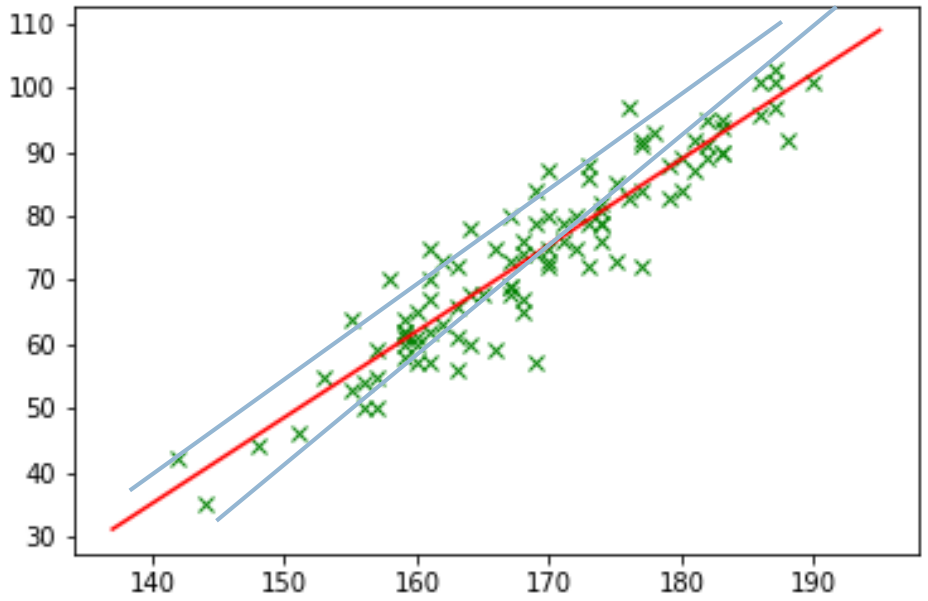
- Method:
  - Try to fit a straight line to the observed data
  - Predict that unseen data are placed on the line
- Questions:
  - What is the best line?
  - How do we find it?

Best fit



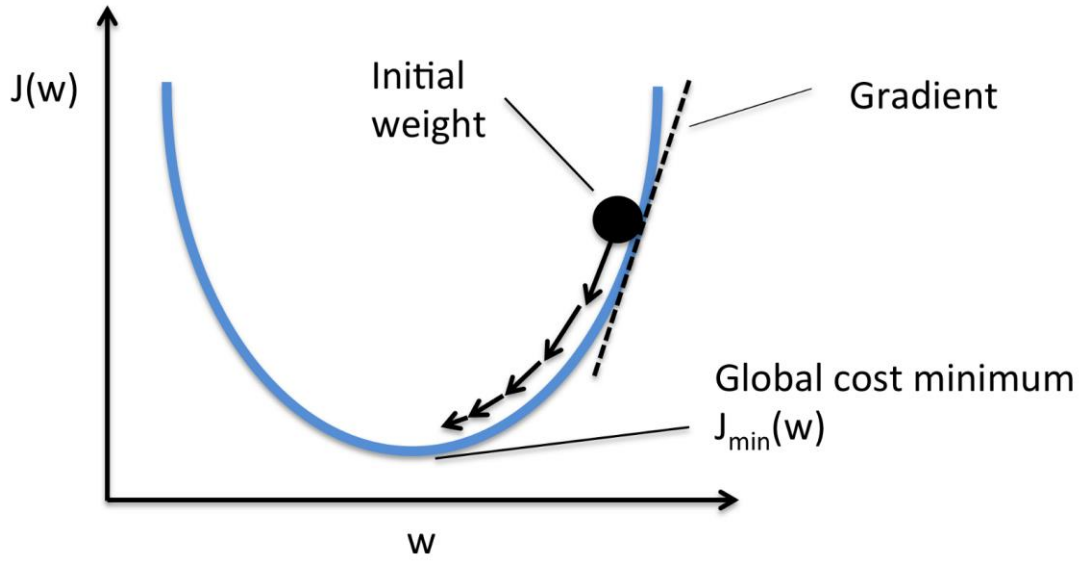
- Mean square error:
  - $d_i = \left(y_i - (ax_i + b)\right)$
  - mse $= \frac{1}{m}\sum_{i=1}^{m} {d_i}^2$
- Squaring:
  - To not get 0
  - Large mistakes are punished more severly
  - Cf. variance in statistics

How do we find this best line?



- For lin. regr. there is a formula
  - (this is called an analytic solution)
  - But slow with many (millions) of features
- Alternative:
  - Start with one candidate line
  - Try to find better weights
  - Use Gradient Descent
  - A kind of search problem

Gradient descent

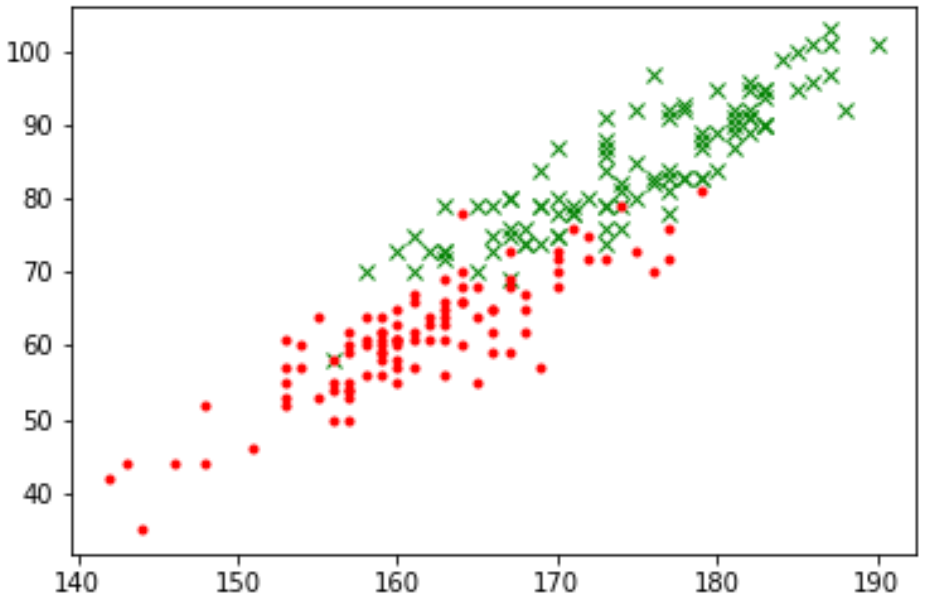


- We use the derivative of the (MSE) error function to point in which direction to move
- We are approaching a unique global minimum
- For details:
  - IN3050/4050 (spring 2020)

Today

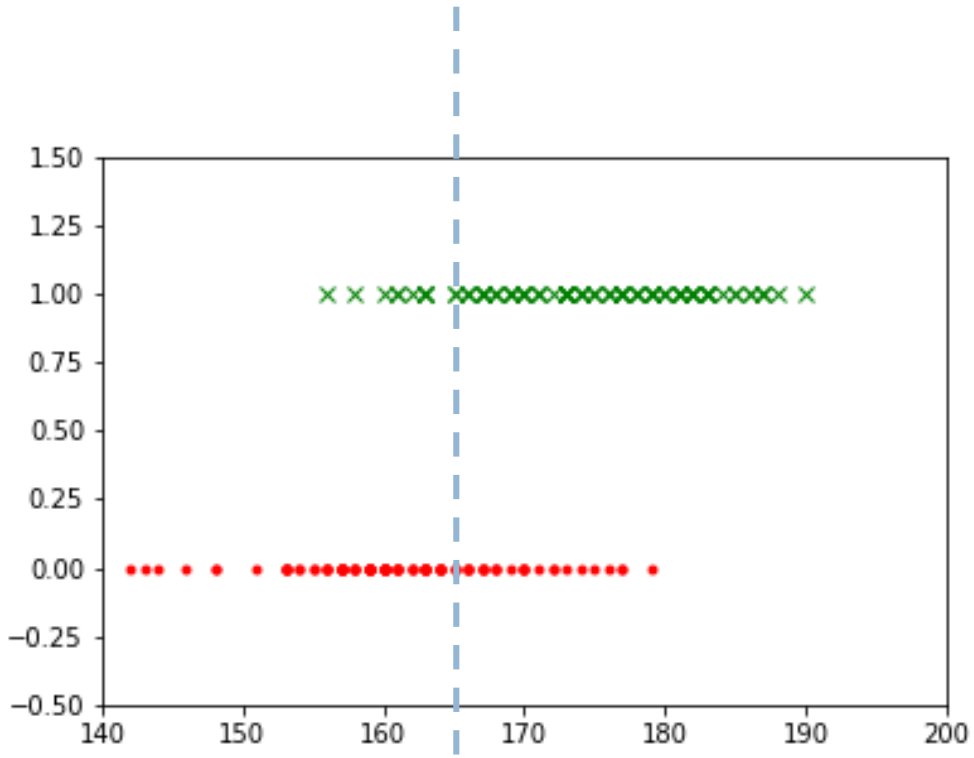
- Generative and discriminative classifiers
- Representing categorical features
- Linear classifiers
- Linear regression
- Logistic regression
- Training the logistic regression classifier
- Multinomial Logistic Regression
- Application to tagging
- Comparing to NB

Example: heights-weights-male-female



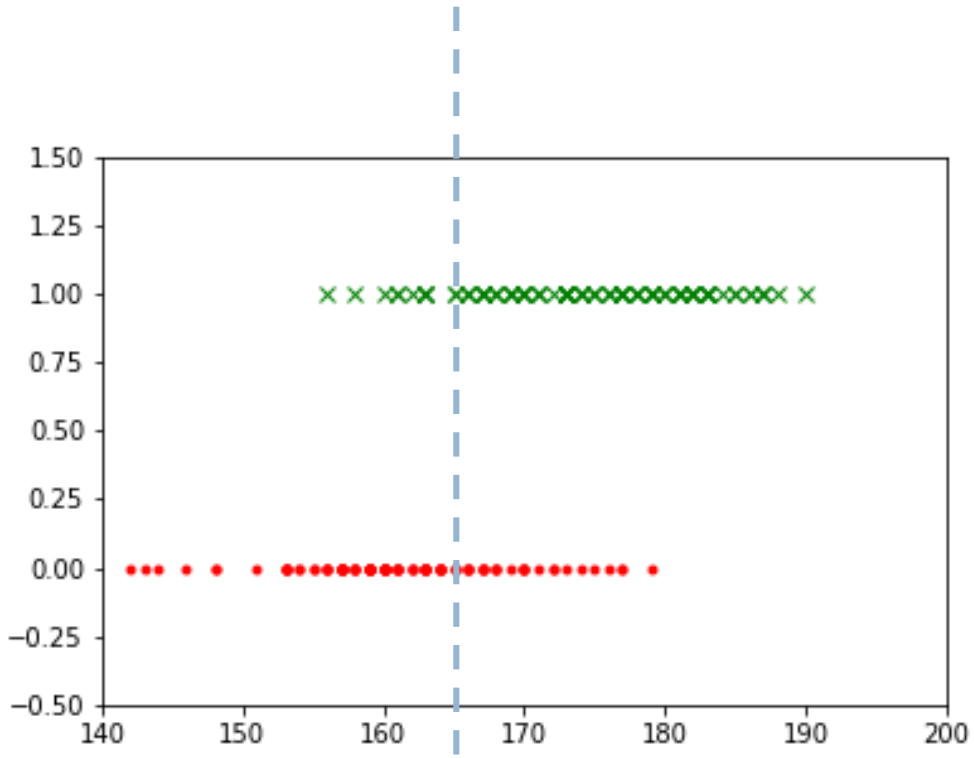
- Goal: predict gender
- First: try to predict from height only

Predicting gender from height



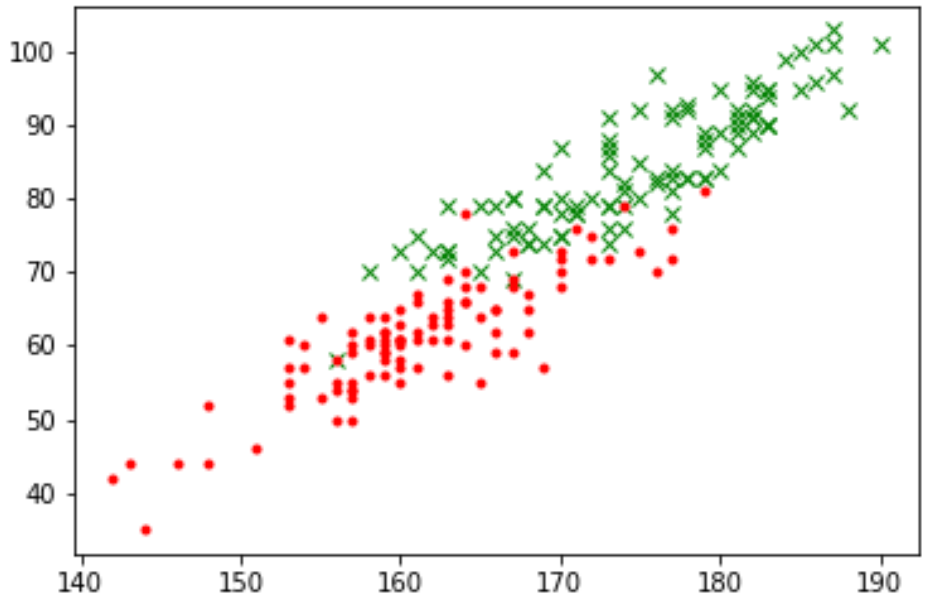
- The decision boundary should be a number: c
- An observation, *n*, is classified
  - *male* if *height_n* > c
  - *female* otherwise
- How do we determine c?

Digression



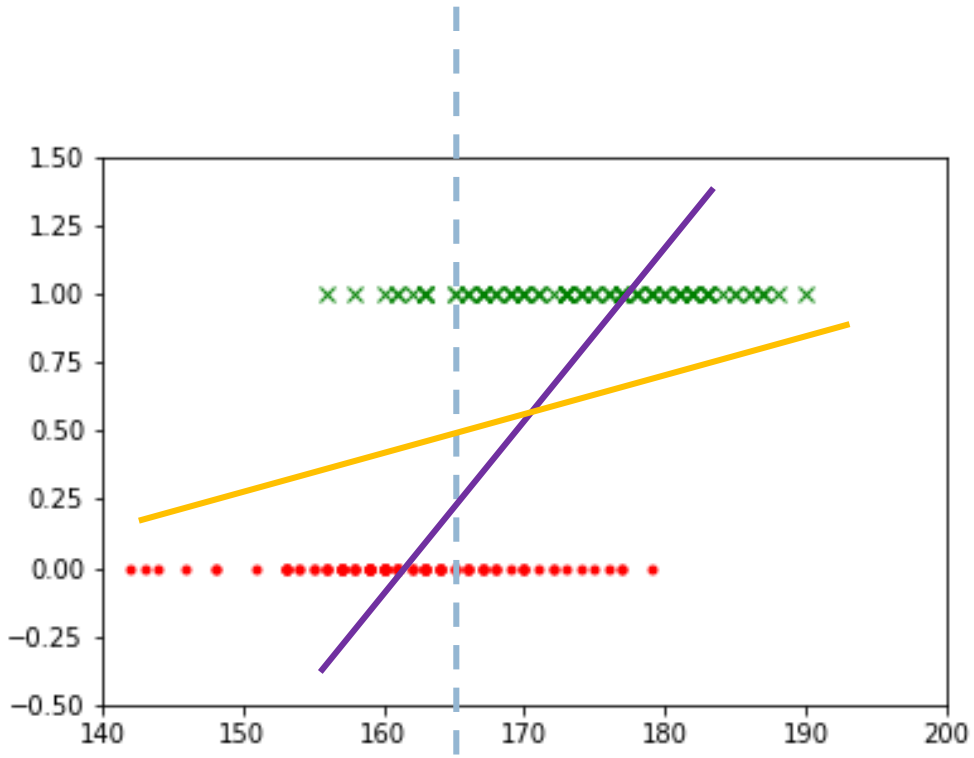
- By the way
- How good are the best predictions og gender given height?
  - …
- Given weight?
  - …
- Given height+weight?
  - …

Digression



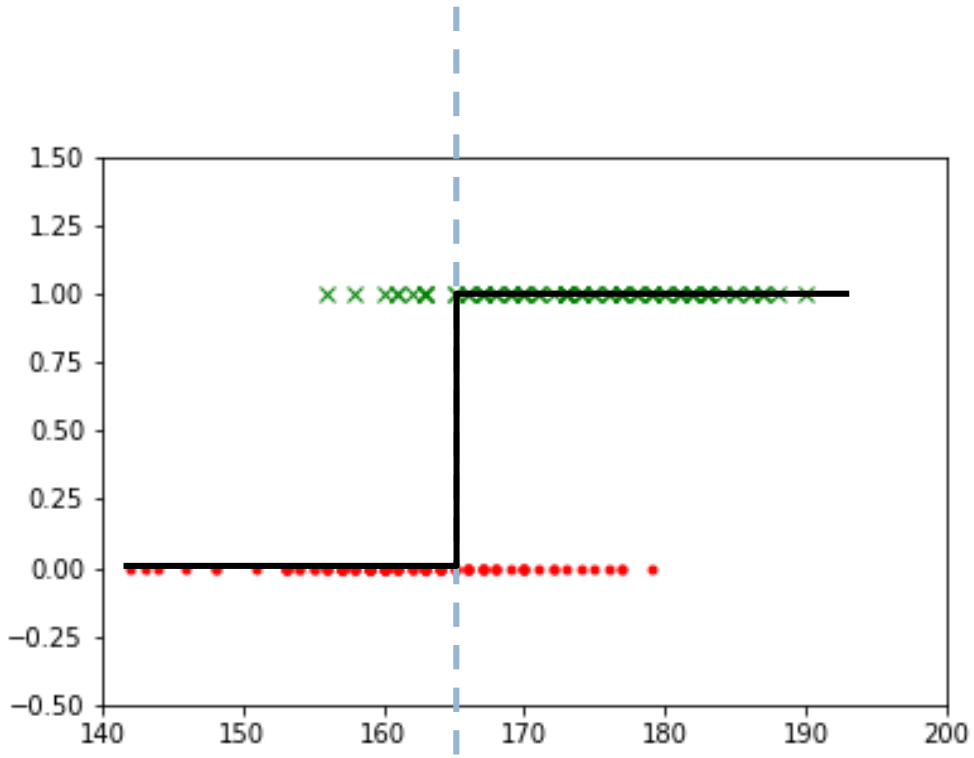
- By the way
- How good are the best predictions og gender given height?
  - …
- Given weight?
  - …
- Given height+weight?
  - …

Linear regression is a bad choice



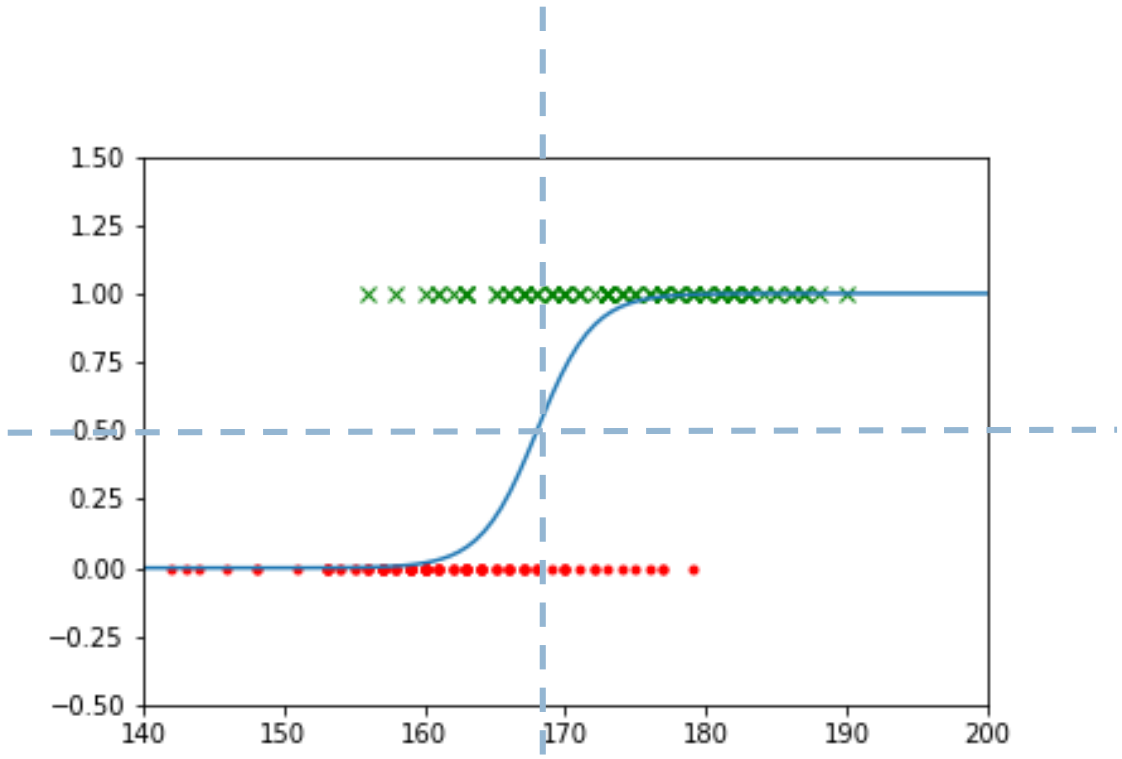
- How do we determine c?
- Linear regression:
  - Bad fit
  - Correctly classified objects contribute to the error (wrongly!)

The "correct" decision border



- The (Heaviside) step function
- But:
  - Not easy to use
  - Not a differentiable function

The sigmoid curve



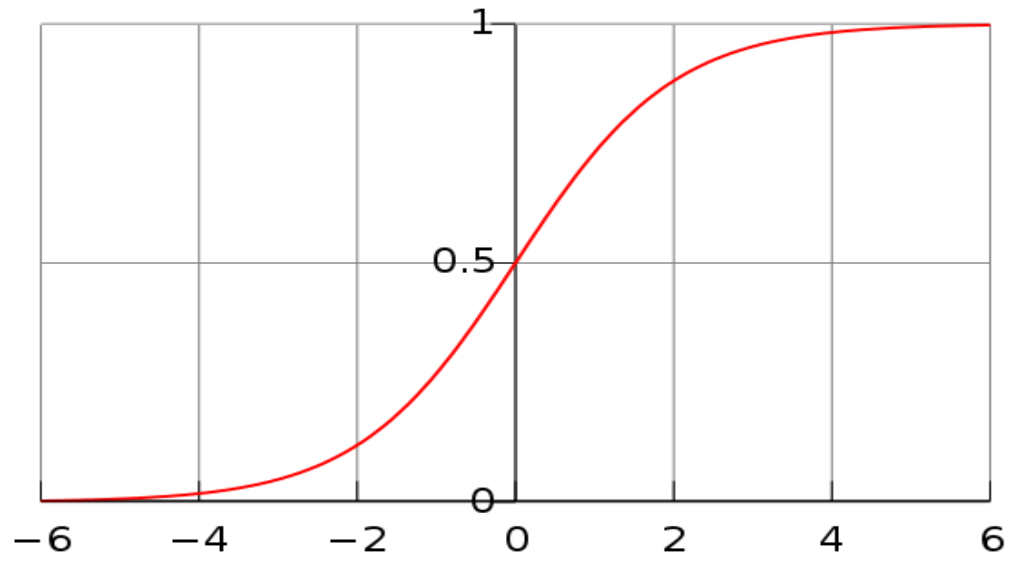
- An approximation to the ideal deciscion border
- Differentiable
  - Gradient descent
- Mistakes further from the deciscion border are punished harder

An observation, *n*, is classified
- *male* if f(*height_n*) > *0.5*
- *female* otherwise

Logistic function

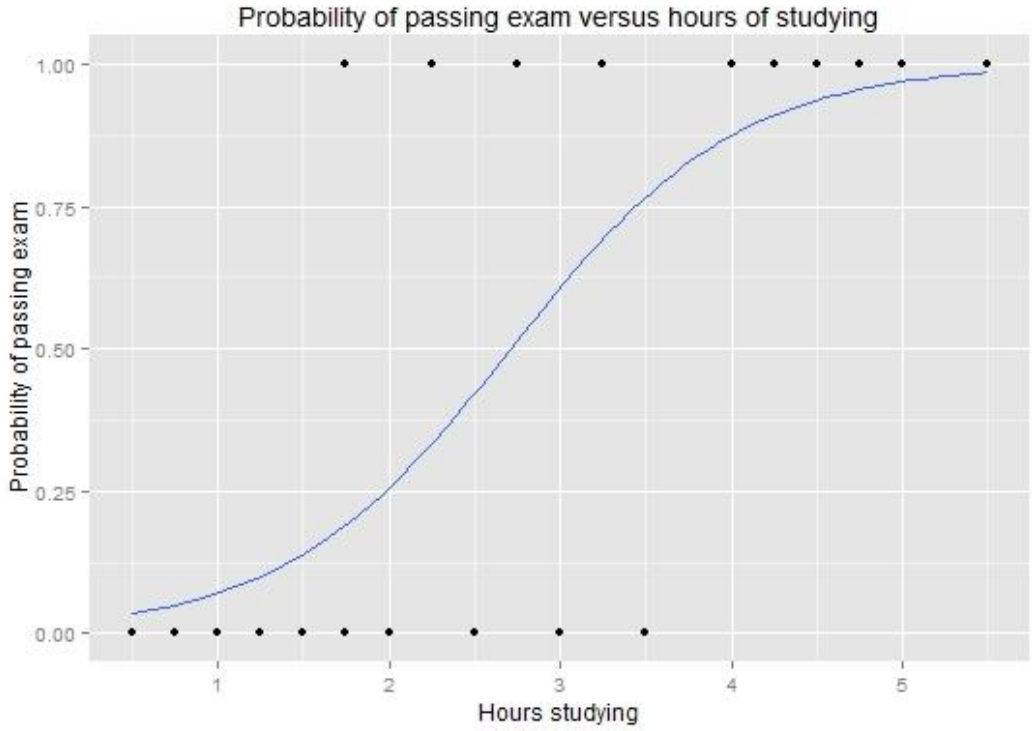
- Takes values between 0 and 1
- Can express probabilities.
- Useful for transforming discrete values to probs.
- Used e.g. in "deep learning"
- Mathematically "well-behaved"
- Used to model population growth, disease spreading etc.

$$P(c_1 \mid \vec{f}) = \frac{e^{\vec{w}\bullet\vec{f}}}{1+e^{\vec{w}\bullet\vec{f}}} \qquad P(c_1 \mid \vec{f}) = \frac{1}{1+e^{-\vec{w}\bullet\vec{f}}}$$



The effect

- Instead of a linear classifier which will classify some instances incorrectly
- The logistic regression will ascribe a probability to all instances for the class C (and for notC)
- We can turn it into a classifier by ascribing class C if $P(C|\vec{x}) > 0.5$
- We could also choose other cut-offs, e.g. if the classes are not equally important



Logistic regression

- Logistic regression is probability-based
- Given two classes C, not-C,
  start with $P(C|\vec{x})$ and $P(notC|\vec{x})$ given a feature vector $\vec{x}$
- Consider the odds $\frac{P(C|\vec{x})}{P(notC|\vec{x})} = \frac{P(C|\vec{x})}{1-P(C|\vec{x})}$
  - If this is >1, $\vec{x}$ most probably belongs to C
  - It varies between 0 and infinity
- Take the logarithm of this $\log \frac{P(C|\vec{x})}{1-P(C|\vec{x})}$
  - If this is >0, $\vec{x}$ most probably belongs to C
  - It varies between minus infinity and pluss infinity

Logistic regression

- $\log \frac{P(C|\vec{x})}{1-P(C|\vec{x})} > 0$?
- Try to find a linear expression for this $\log \frac{P(C|\vec{x})}{1-P(C|\vec{x})} = \vec{w} \cdot \vec{x} > 0$
- Given such a linear expression
  - $\frac{P(C|\vec{x})}{1-P(C|\vec{x})} = e^{\vec{w}\cdot\vec{x}}$
  - $P(C|\vec{x}) = \frac{e^{\vec{w}\cdot\vec{x}}}{1+e^{\vec{w}\cdot\vec{x}}} = \frac{1}{1+e^{-\vec{w}\cdot\vec{x}}}$

Today

- Generative and discriminative classifiers
- Representing categorical features
- Linear classifiers
- Linear regression
- Logistic regression
- Training the logistic regression classifier
- Multinomial Logistic Regression
- Application to tagging
- Comparing to NB