UiO : University of Oslo

institutt for informatikk

IN3050/IN4050 - Introduction to Artificial Intelligence and Machine Learning

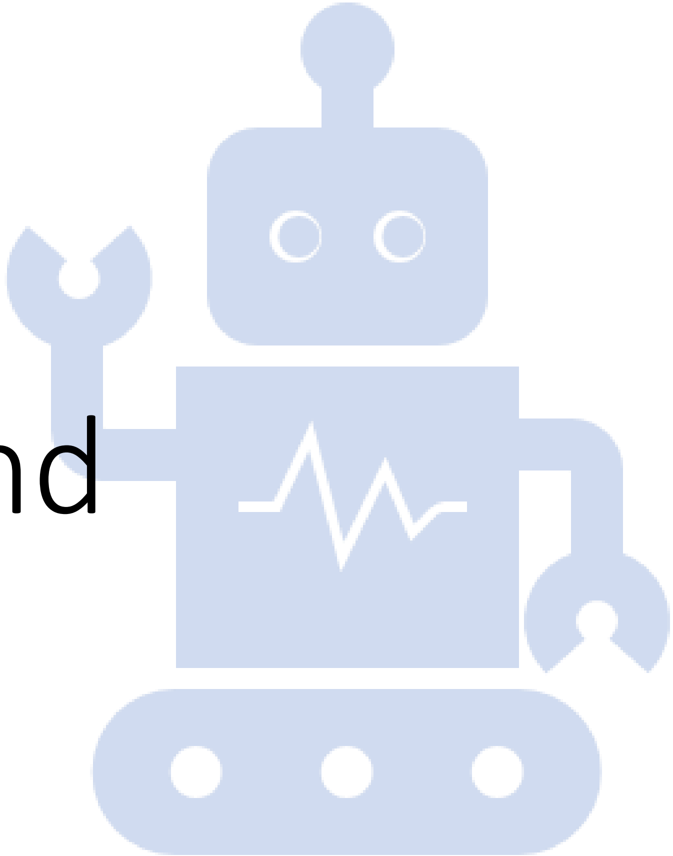
Lecture 7 – 2021

Logistic Regression

Jan Tore Lønning

7.1 Linear Regression and Classification

IN3050/IN4050 Introduction to Artificial Intelligence and Machine Learning



Today

1. Linear Regression and Classification
2. The Logistic Function and its Derivative
3. The Logistic Regression Classifier
4. Cross-Entropy Loss
5. Training the Logistic Regression Classifier
6. Variants of Gradient Descent
7. Multi-Class Classification

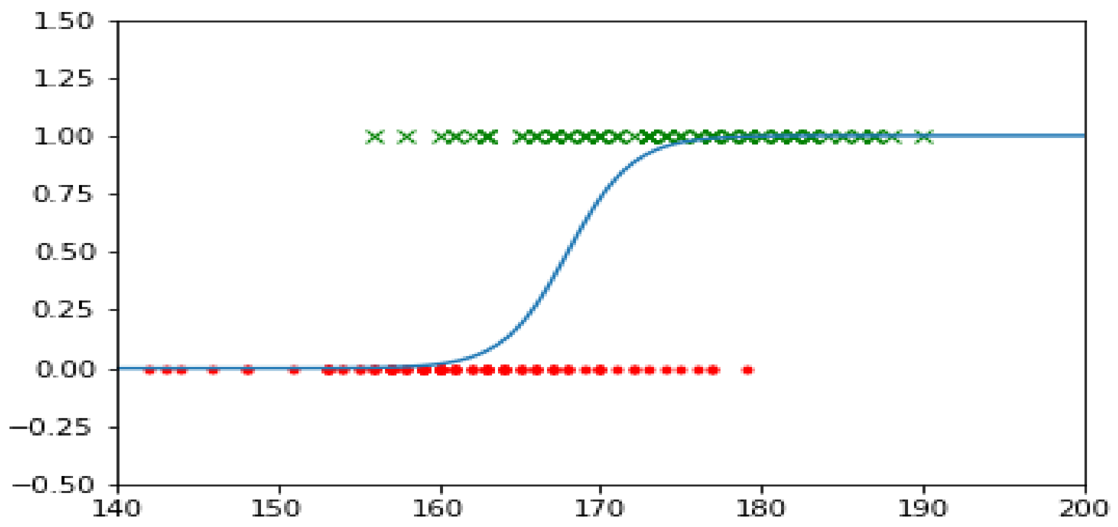
Supervised learning - Where are we?

	Classification	Regression
Decision tree	Lec.1 (simplified form)	
k Nearest Neighbors	Lec.5	
Perceptron	Lec.5	
Linear regression		Lec. 6
Logistic regression		
Neural networks		

Supervised learning - Where are we?

	Classification	Regression
Decision tree	Lec.1 (simplified form)	
k Nearest Neighbors	Lec.5	Possible
Perceptron	Lec.6	
Linear regression	today	Lec. 6
Logistic regression	today!	
Neural networks	Next week	

Logistic regression?

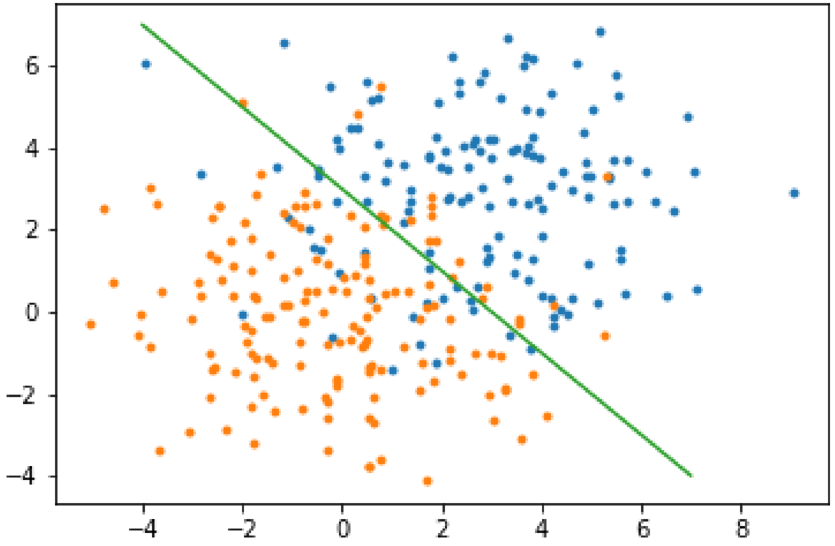


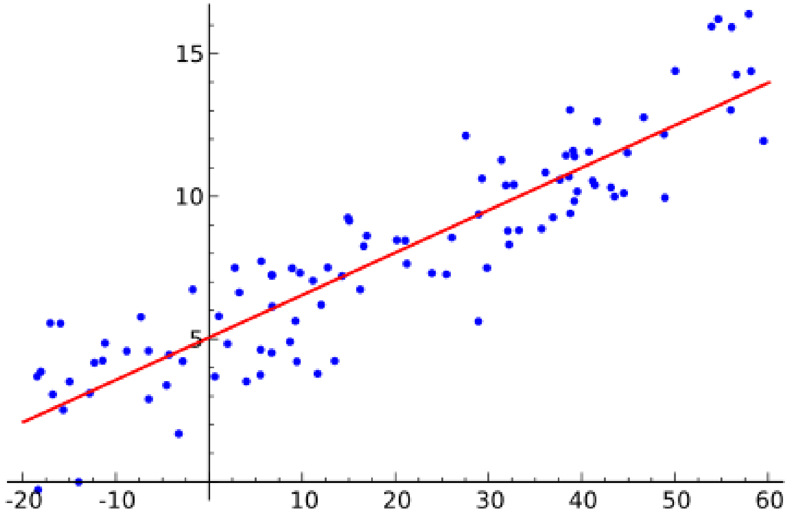
Interesting by itself

- A classifier
  - (not numerical regression)
- "Standard" ("best") purely linear classifier
- (Not in Marsland)

Useful tools for neural networks:

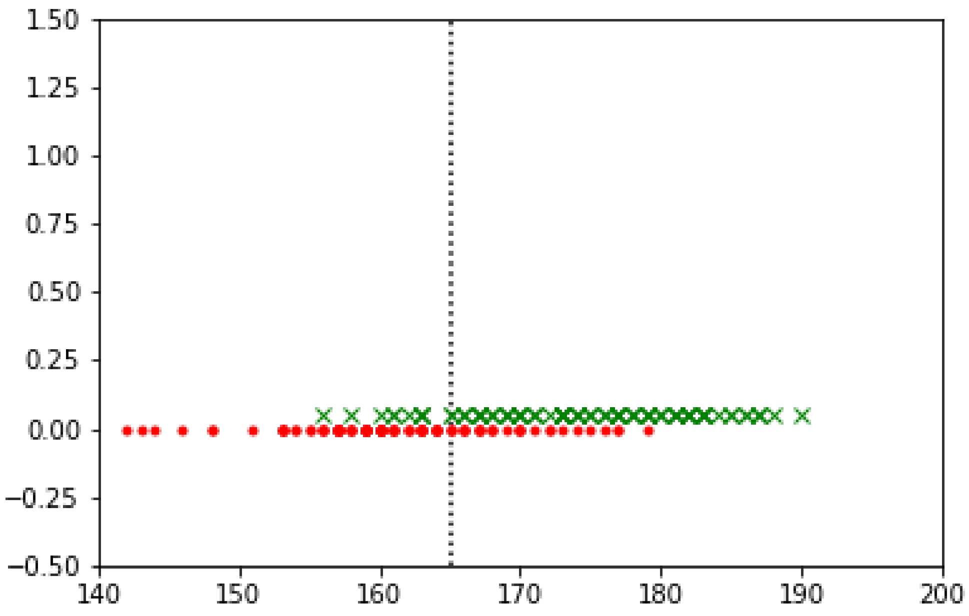
- The logistic function:
  - Its derivative
- Loss function
- Application of the chain rule for derivatives for gradient descent





	Linear Classifier	Linear regression
Number of input variable	Decision boundary	Prediction
One	Point	Line
Two	Line	Plane
Three	Plane	Hyper-plane
>3	Hyper-plane	
Update	Perceptron: $w_i = w_i - \eta(y-t)x_i$	$w_k = w_k - \eta \frac{2}{N}\sum_{j=1}^{N}\left((t_j - y_j)(-x_{j,k})\right)$
Type of y, t	{0,1}	Real numbers

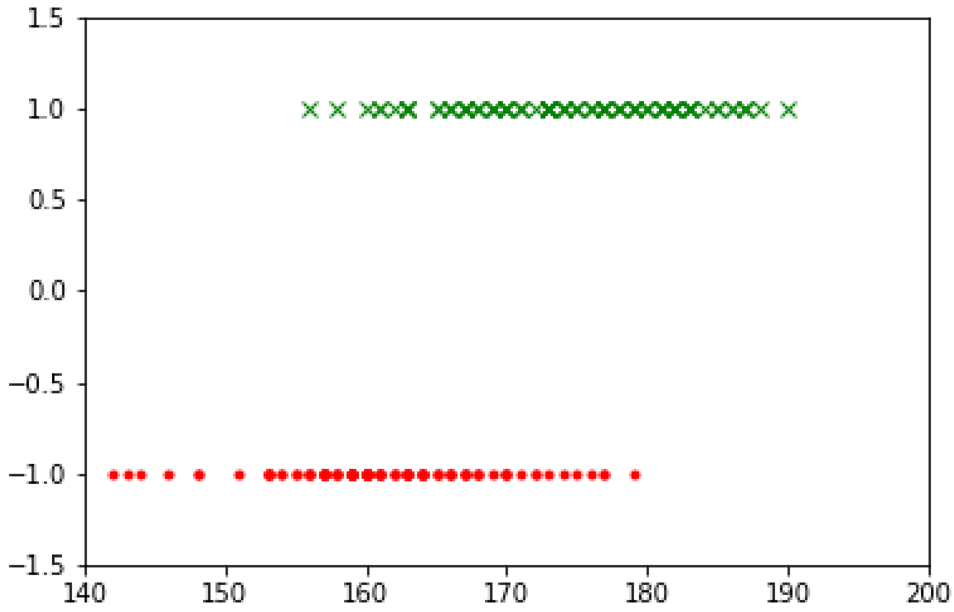
Example: predicting gender from height



- 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*?

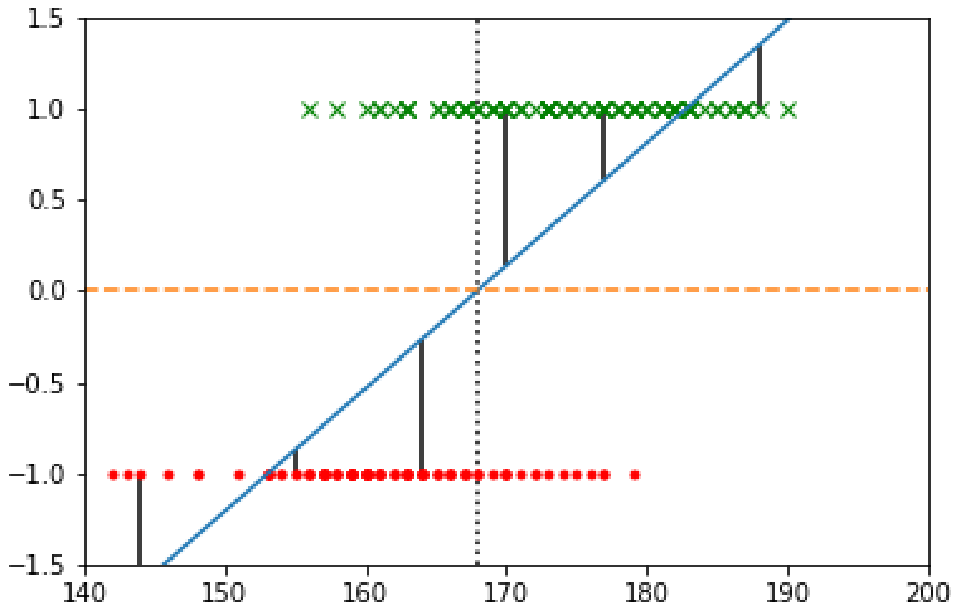
Linear regression as a classifier

1. Consider the prediction of classes as prediction of the two numbers 1, -1, resp.
2. Fit a linear regressor to these data (minimizing) MSE



Linear regression as a classifier

1. Consider the prediction of classes as prediction of the two numbers 1, -1, resp.
2. Fit a linear regressor to these data (minimizing) MSE



Linear regression as a classifier

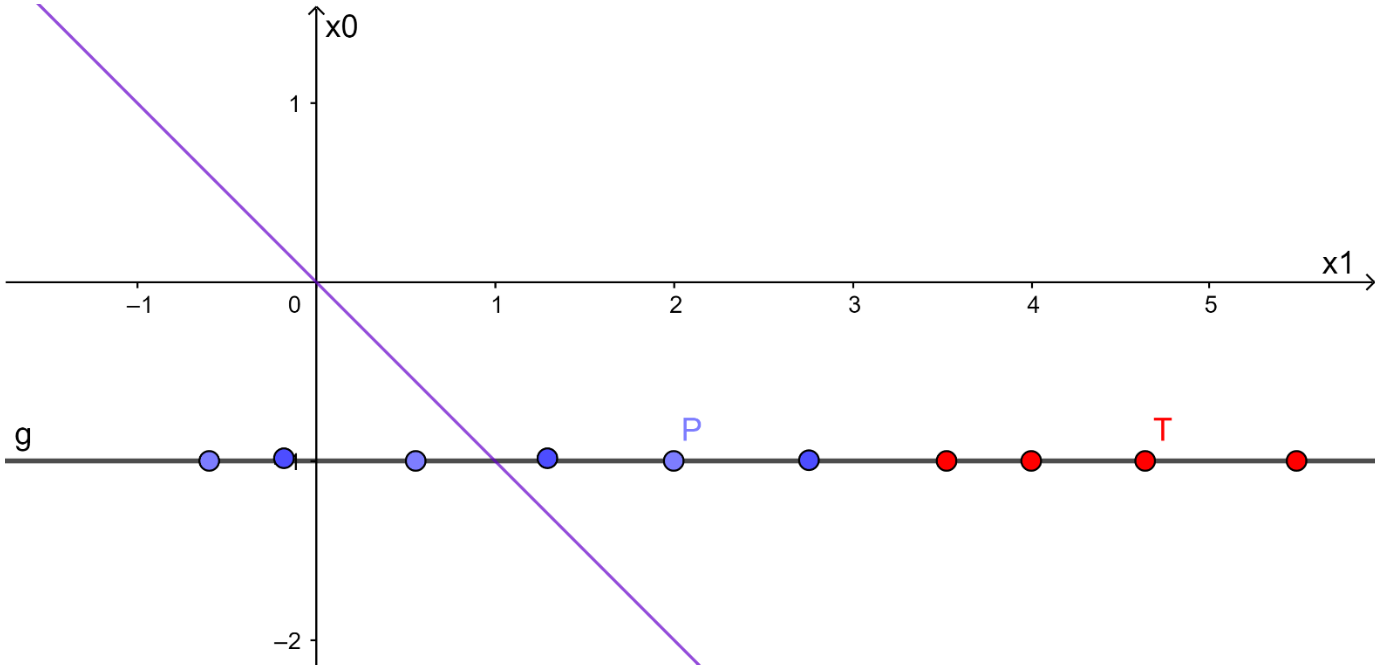
1. Consider the prediction of classes as prediction of the two numbers 1, -1, resp.
2. Fit a linear regressor to these data (minimizing) MSE
3. Predict
   - Positive class if $y > 0$ and
   - Negative class, otherwise otherwise

- Hence, decision boundary is dotted black line

1. Consider the prediction of classes as prediction of the two numbers 1, -1, resp.
2. Fit a linear regressor to these data (minimizing) MSE
3. Predict
   - Positive class if $y > 0$ and
   - Negative class, otherwise otherwise

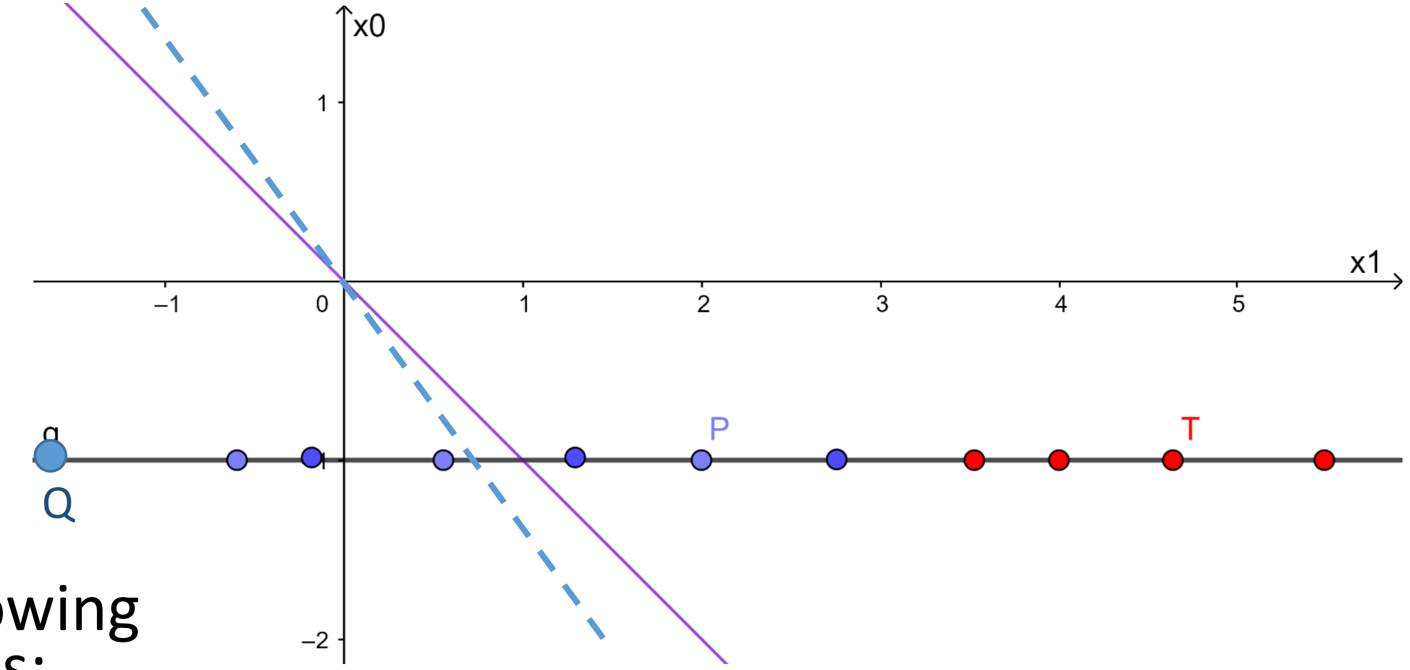
- Hence, decision boundary is dotted black line

Example



- Consider example from last week.
- Compare Lin.reg.-classifier to perceptron
- Assume stochastic gradient descent:
  We update for one datapoint at a time

Example



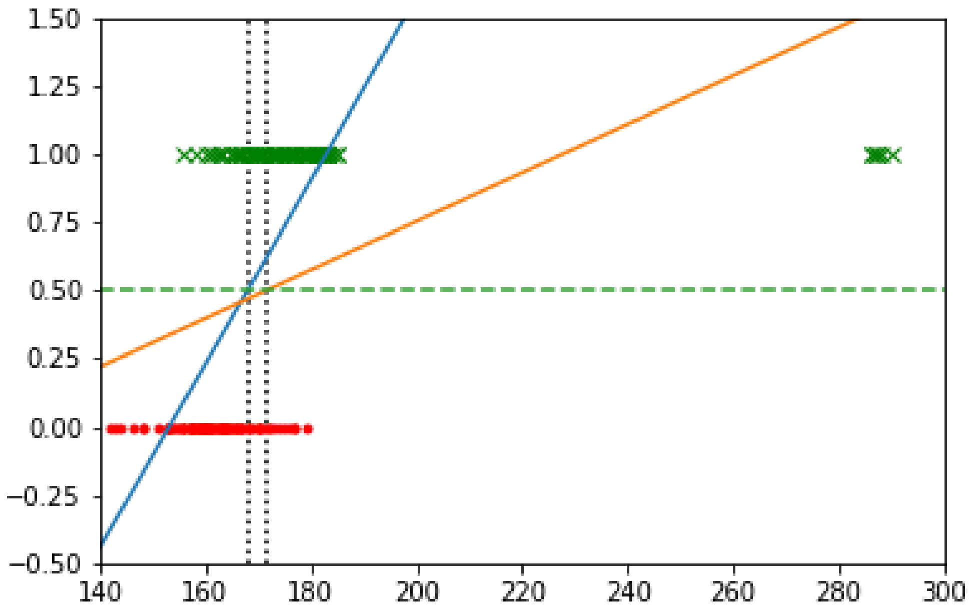
- We are in the middle of training
- Learning rate: $\eta = 0.1$
- We have so far, the following weights for the decisions:
- Positive class provided
  $h = -w_0 + w_1 x_1 = 1 - x_1 > 0$
  - i.e., $w_0 = -1$ and $w_1 = -1$

- Consider the point Q=(-1, -2):
  - Correctly classified
  - Perceptron: Do nothing

- Lin.reg.classifier:
- $h(Q) = 1 - 1(-2) = 3$
  - $w_0 = w_0 - \eta(y - t)x_0 =$
    $-1 - 0.1(3 - 1)(-1) = -0.8$
  - $w_1 = w_1 - \eta(y - t)x_1 =$
    $-1 - 0.1(3 - 1)(-2) = -0.6$

Limitations

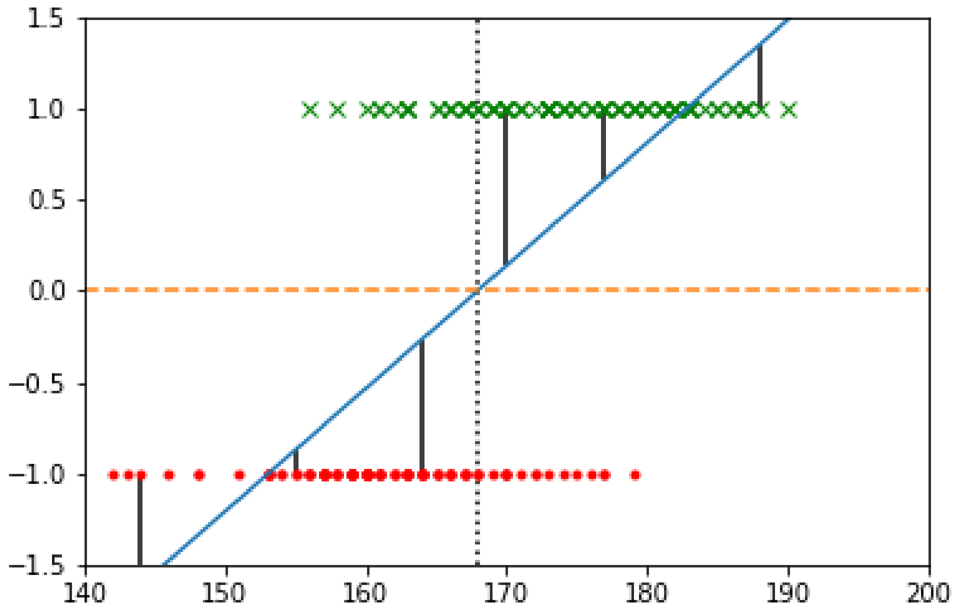
- For example
  - moving 7 (out of) 100 pos 100 steps to the right
  - the decision boundary is moved
    - from 168
    - to 171.5
  - the accuracy (on the 200 training set) goes
    - from 0.81
    - to 0.78
- Should these outliers have such an effect?



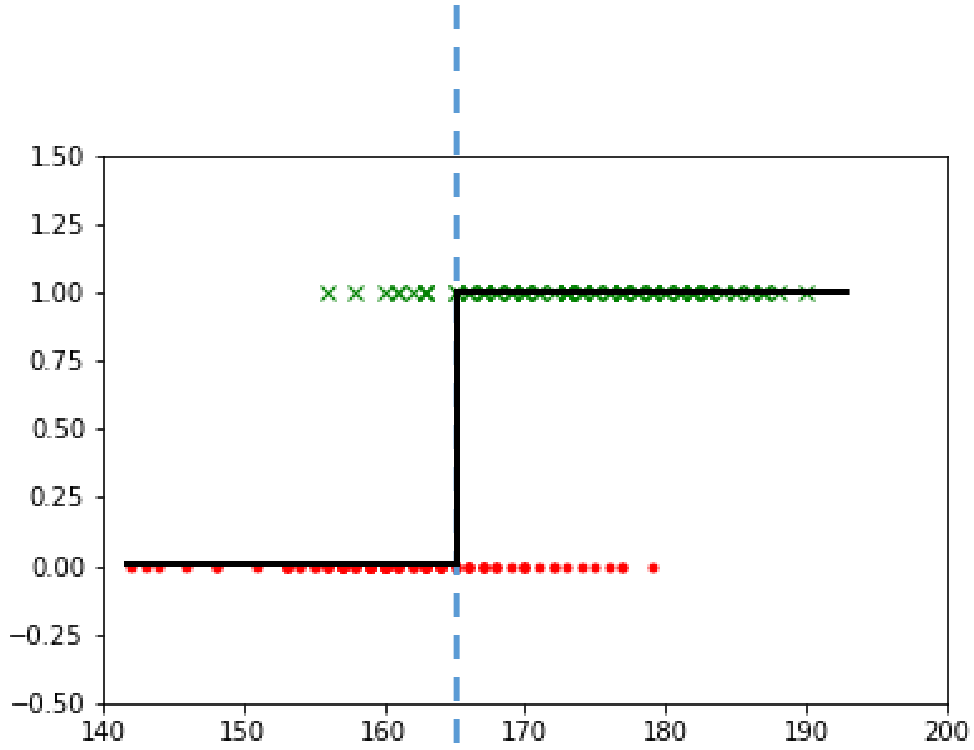
By the way:
We have here used 0 and 1 for the two classes.
This works equally fine.
Prediction: Positive class for $y > 0.5$

Linear regression as a classifier

- The MSE seems to punish correctly classified items too severely.



The ‘’correct’’ decision boundary

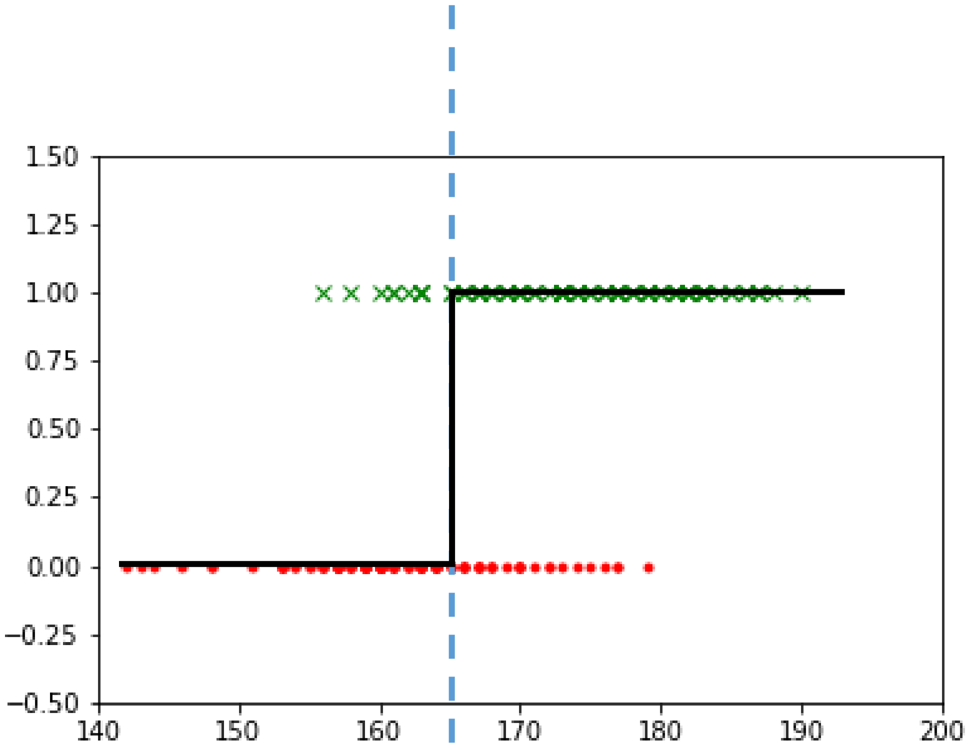


- The (Heaviside) step function
- But:
  - How do we find the best one?
  - Not a differentiable function

7.2 The Logistic Function and its Derivative

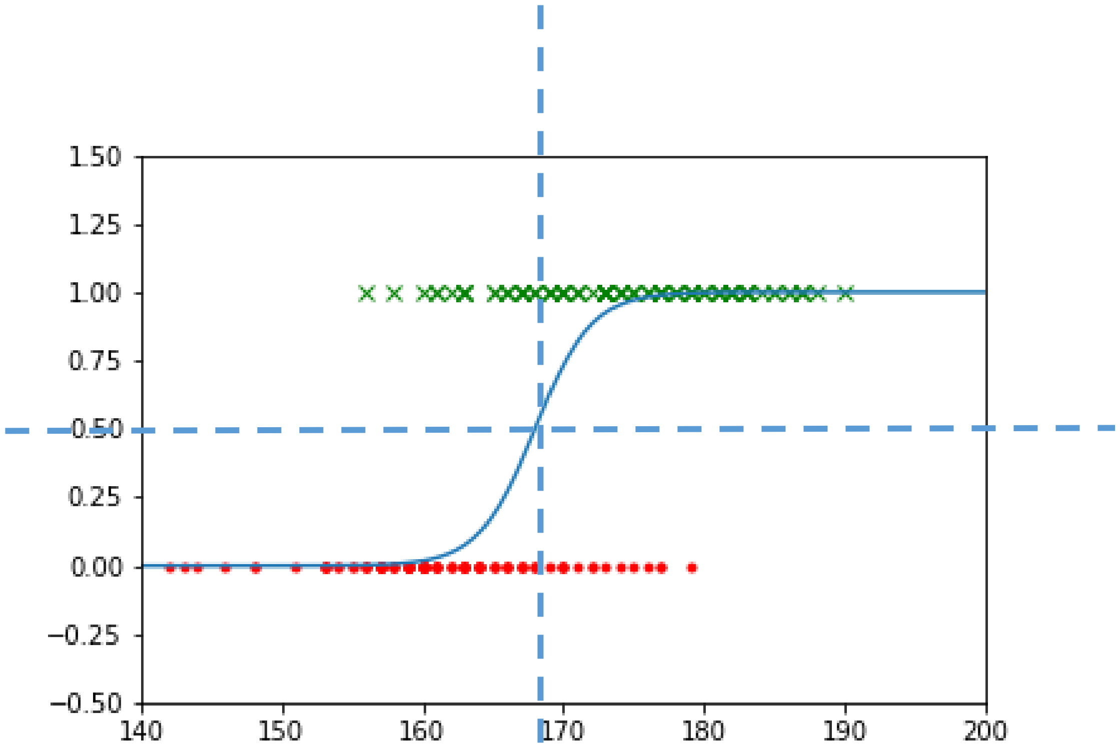
IN3050/IN4050 Introduction to Artificial Intelligence and Machine Learning

The "correct" decision boundary



- The (Heaviside) step function
- But:
  - How do we find the best one?
  - Not a differentiable function

The sigmoid curve



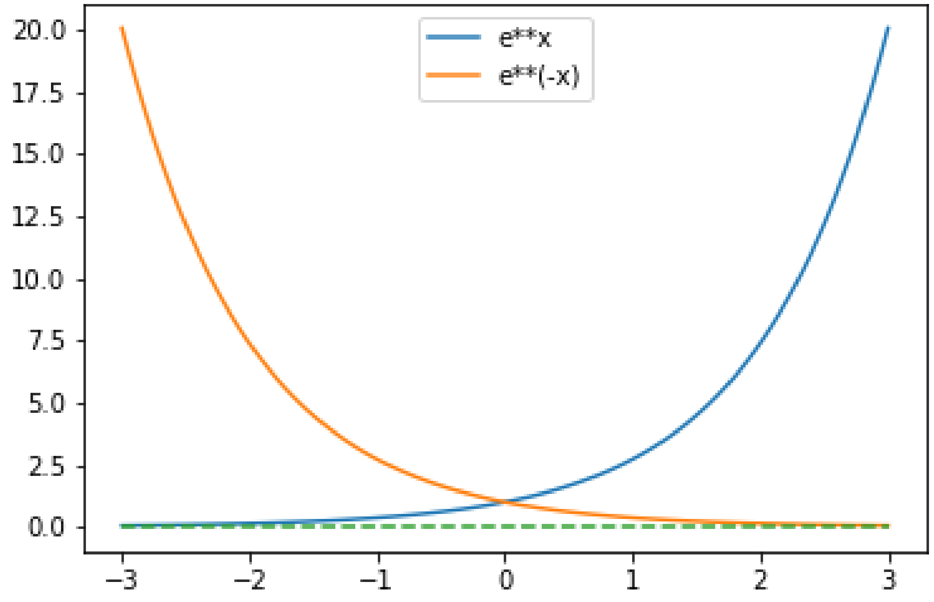
- An approximation to the ideal decision boundary
- Differentiable
  - Gradient descent
- Mistakes further from the decision boundary are punished harder

An observation, *n*, is classified

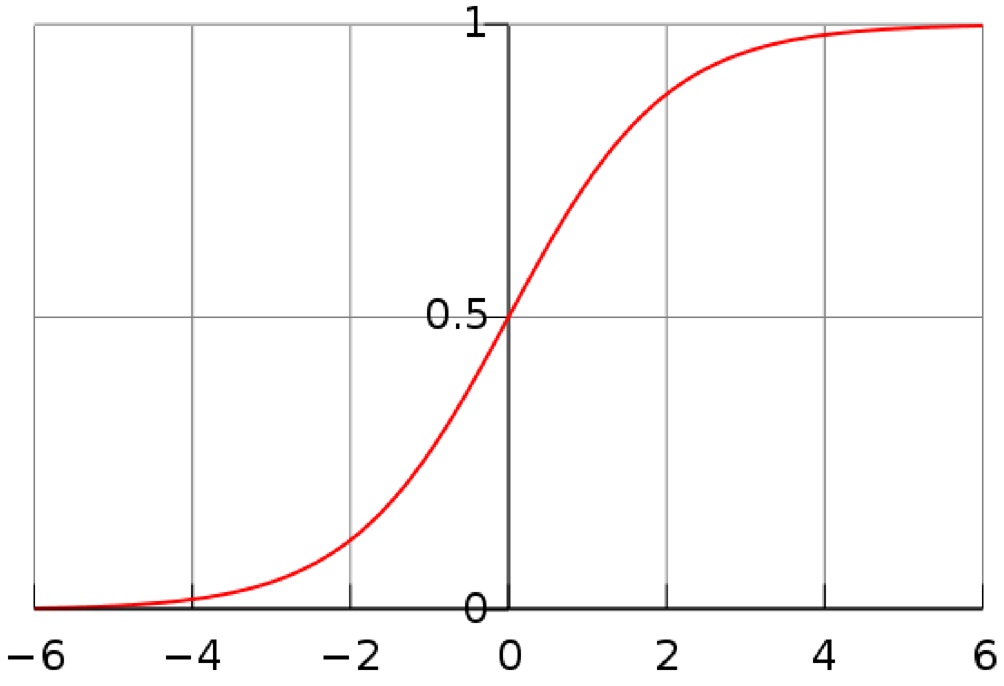
- *male* if f(*height_n) > 0.5*
- *female* otherwise

Exponential function - Logistic function

$$y = e^z$$

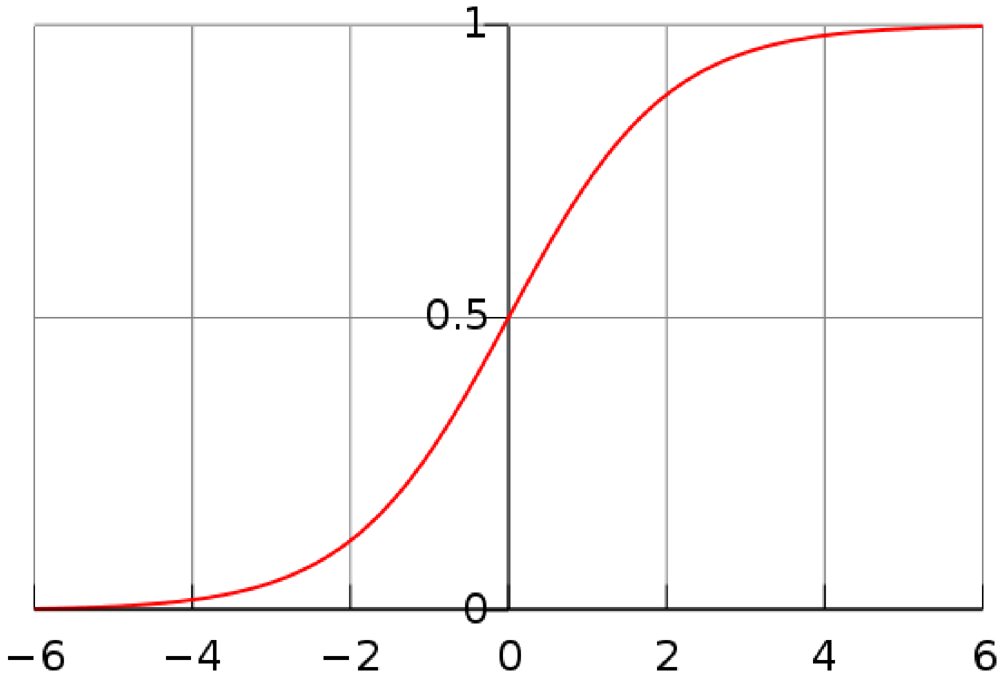


$$y = \frac{1}{1+e^{-z}} = \frac{e^z}{e^z+1}$$



The logistic function

- $y = \frac{1}{1+e^{-z}} = \frac{e^z}{e^z+1}$
- A sigmoid curve
  - Other functions also make sigmoid curves e.g., $y = \tanh(z)$
- Maps $(-\infty, \infty)$ to $(0,1)$
- Monotone
- Can be used for transforming numeric values into probabilities



The derivative of the logistic function

- $y = f(x) = \frac{1}{1+e^{-x}}$
- This has the form $y = g\big(h(x)\big)$
  where $g(z) = \frac{1}{z}$ and $z = h(x) = 1 + e^{-x}$
- Hence $f'(x) = g'(z)h'(x) = \frac{-1}{(1+e^{-x})^2}(-e^{-x}) =$
- $\frac{e^{-x}+1-1}{(1+e^{-x})^2} = \frac{e^{-x}+1}{(1+e^{-x})^2} - \frac{1}{(1+e^{-x})^2} = \mathrm{y} - y^2 = y(1-y)$
- We will use this also in the multi-layer neural networks

7.3 The Logistic Regression Classifier

IN3050/IN4050 Introduction to Artificial Intelligence

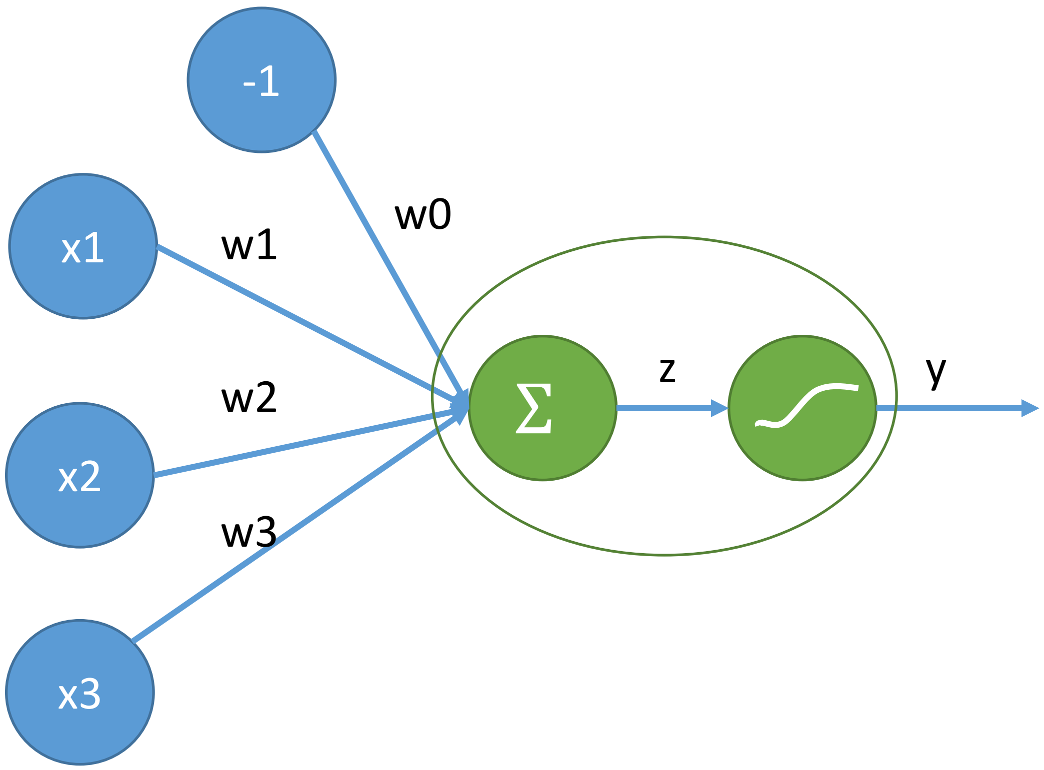
and Machine Learning

Logistic Regression:

- First sum of weighted inputs :
- $z = \sum_{i=0}^{m} w_i x_i = \boldsymbol{w} \cdot \boldsymbol{x}$
- Apply the logistic function σ to this sum

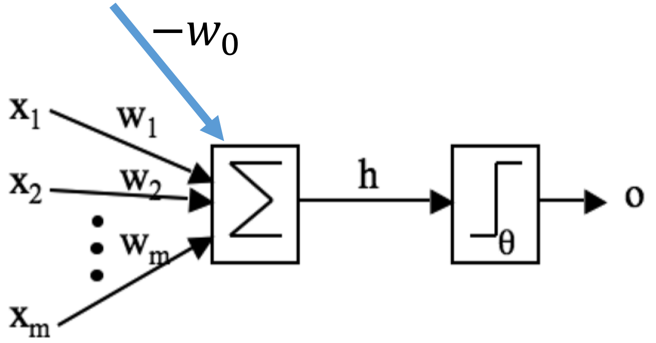
$$y = \sigma(z) = \frac{1}{1 + e^{-\vec{w}\cdot\vec{x}}}$$

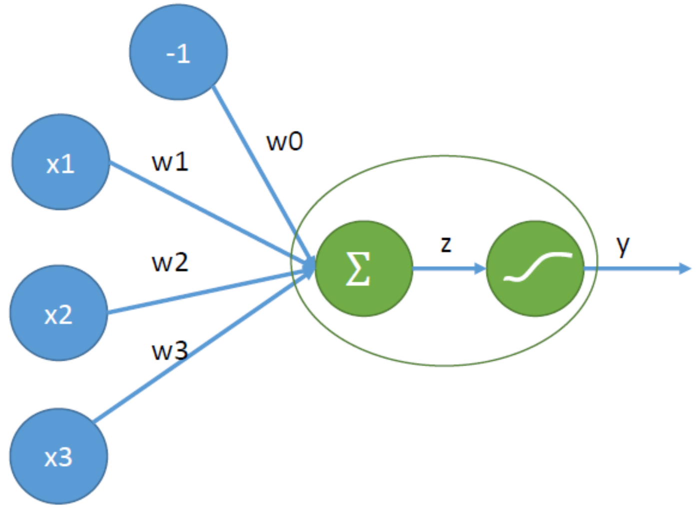
- For $\boldsymbol{x} = \vec{x}$ predict
  - as the positive class if $y > 0.5$,
  - otherwise, the negative class



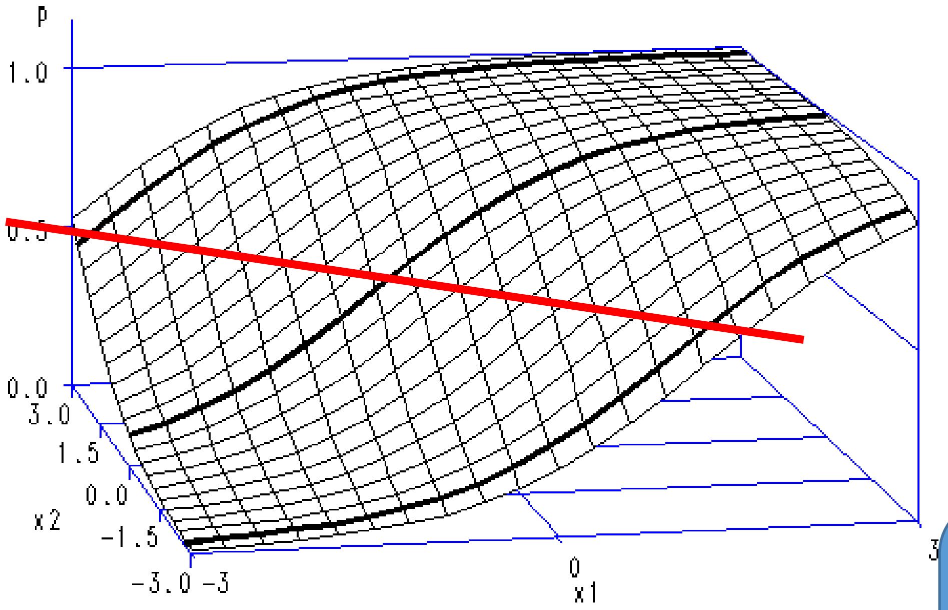
Comparison: activation function

- Perceptron: step function
- Linear regression: identity
- Logistic regression:
  the logistic function





With two features

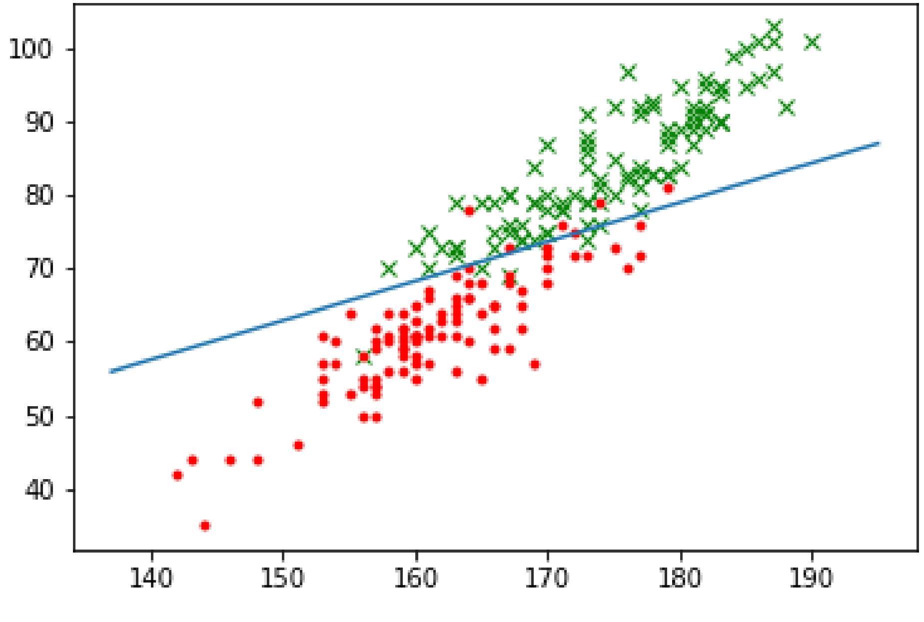


From IDRE, UCLA

- Two features: x_1, x_2
- Apply weights: w_0, w_1, w_2
- Let $y = -w_0 + w_1 x_1 + w_2 x_2$
- Apply the logistic function, σ, and check whether
  - $\sigma(y) = \frac{1}{1+e^{-y}} > 0.5$

Geometrically: Folding a plane along a sigmoid
The decision boundary is the intersection of this surface and the plane $\mathrm{p} = \sigma(y) = 0.5$:
This turns out to be a straight line

Example with two features



- Example:
  - Heights and weights
  - Acc.: = 0.95
- Blue line = decision boundary
  - Points above it gets a value > 0.5

Understanding logistic regression 1

The following 3 slides attempt to give you an understanding of logistic regressions models.

- The model is probability-based
- There are two classes t=1, t=0
- For an observation $\boldsymbol{x} = \vec{x}$, we wonder:
- *How probable is it that this $\vec{x}$ belongs to class 1, and how probable is it that it belongs to class 0?*
- i.e., what are $P(t = 1|\vec{x})$ and $P(t = 0|\vec{x})$? Which is largest?

Understanding logistic regression 2

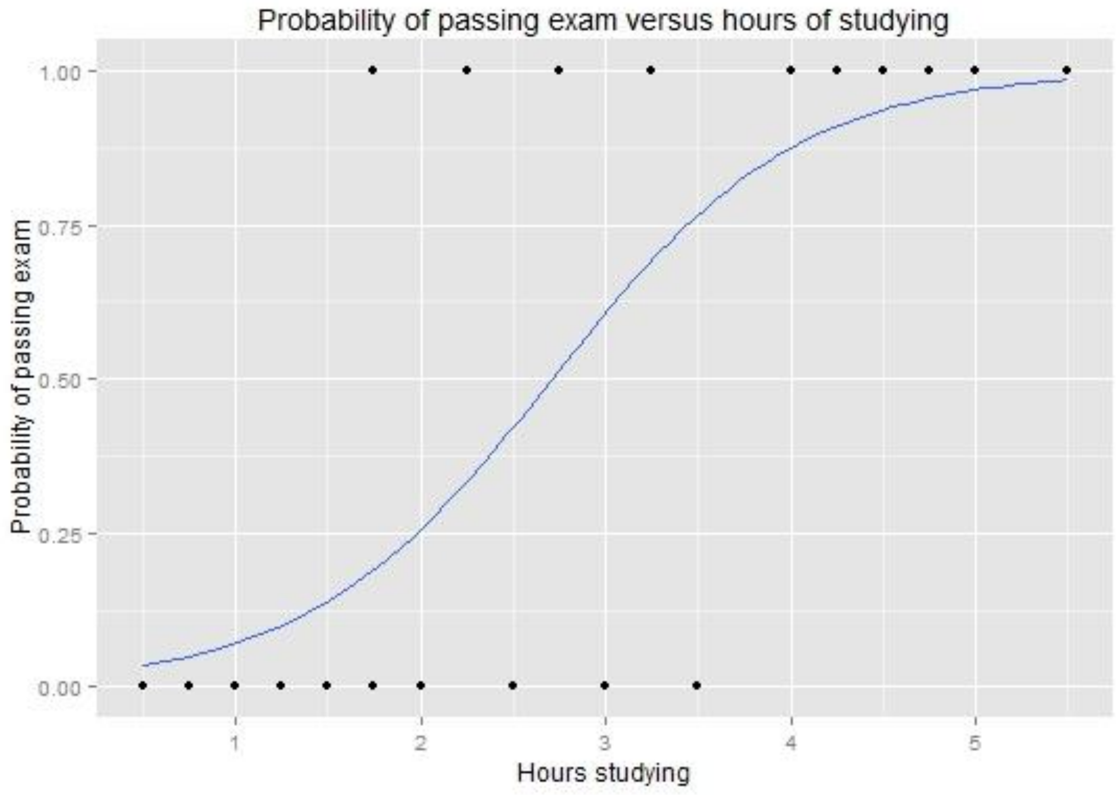
- What are $P(t = 1|\vec{x})$ and $P(t = 0|\vec{x})$? Which is largest?
- Consider the odds: $\frac{P(t=1|\vec{x})}{P(t=0|\vec{x})} = \frac{P(t=1|\vec{x})}{1-P(t=1|\vec{x})}$
  - If this is >1, $\vec{x}$ most probably belongs to t=1, otherwise t=0
  - The odds varies between 0 and infinity
- Take the logarithm of this, $\log \frac{P(t=1|\vec{x})}{1-P(t=1|\vec{x})}$
  - If this is >0, $\vec{x}$ most probably belongs to t=1
  - This varies between minus infinity and plus infinity

Understanding logistic regression 3

- $\log \frac{P(t=1|\vec{x})}{1-P(t=1|\vec{x})} > 0$?
- Try to find a linear expression for this, $\log\left(\frac{P(t=1|\vec{x})}{1-P(t=1|\vec{x})}\right) = \vec{w} \cdot \vec{x} > 0$
- Given such a linear expression, then
    - $\frac{P(t=1|\vec{x})}{1-P(t=1|\vec{x})} = e^{\vec{w}\cdot\vec{x}}$
- Solving this with respect to $P(t = 1|\vec{x})$ yields
    - $P(t = 1|\vec{x}) = \frac{e^{\vec{w}\cdot\vec{x}}}{1+e^{\vec{w}\cdot\vec{x}}} = \frac{1}{1+e^{-\vec{w}\cdot\vec{x}}}$

A probabilistic classifier

- The logistic regression will ascribe a probability to all instances for the class t=1 (and for t=0)
- We turn it into a classifier by ascribing class t=1 if
$$P(t = 1|\vec{x}) > 0.5$$
- We could also choose other cut-offs, e.g., if the classes are not equally important.

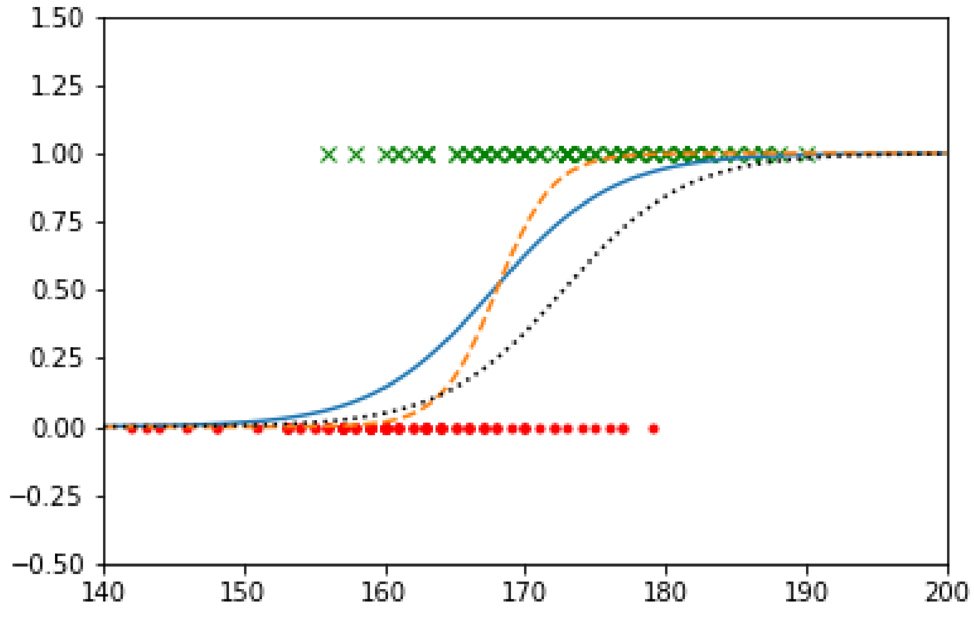


source: Wikipedia

7.4 Cross-Entropy Loss

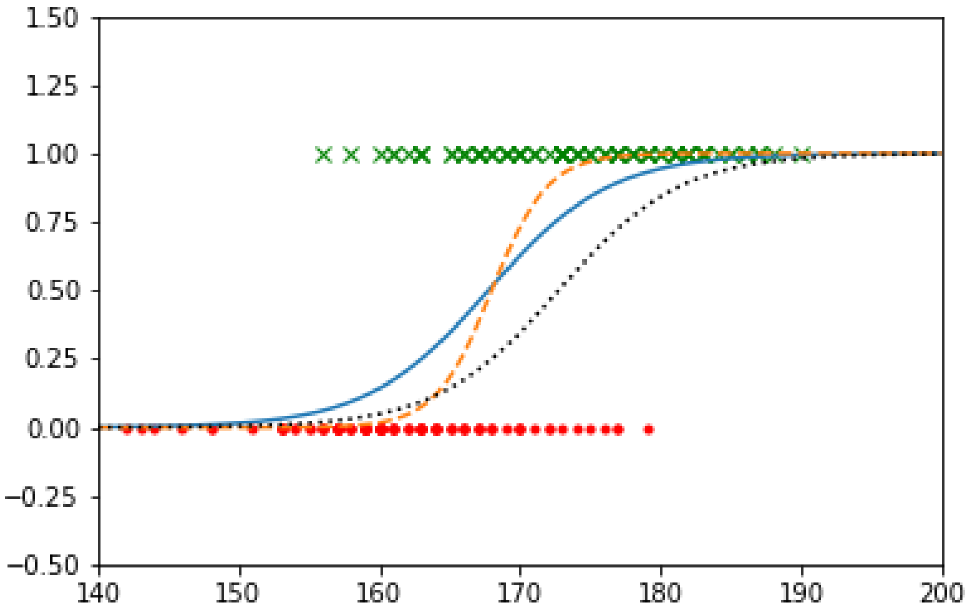
IN3050/IN4050 Introduction to Artificial Intelligence and Machine Learning

How to find the best curve?



- What are the best choices of a and b in $\frac{1}{1+e^{-(ax+b)}}$?
- Geometrically a and b determine the
  - Midpoint (b)
  - Steepness (a)
- of the curve
- What are the best choices of $\vec{w}$ $y = P(t = 1|\vec{x}) = \frac{1}{1+e^{-\vec{w}\cdot\vec{x}}}$

Learning in the logistic regression model



- A training instance consists of
  - a feature vector $\vec{x}$
  - a label (class), t, which is 1 or 0.
- With a set of weights, $\vec{w}$, the classifier will assign
  - $y = P(t = 1|\vec{x}) = \frac{1}{1+e^{-\vec{w}\cdot\vec{x}}}$ to this training instance $\vec{x}$
  - where $P(t = 0|\vec{x}) = 1 - y$
- Goal: find $\vec{w}$ that maximize $P(t|\vec{x})$ of all training inst.s $(\vec{x}, t)$

Loss function

- In machine learning we decide on an objective for the training.
- We can do that in terms of a loss function.
- The goal of the training is to minimize the loss function.
- Example: linear regression
  - Loss: Mean Square Error
- We can choose between various loss functions.
- The choice is partly determined by the learner.
- For logistic regression we choose (simplified) cross-entropy loss

Footnote: Notation

- I observe that I haven’t been consequent in notation
- I fluctuate between boldface $\boldsymbol{x}$ and non-bold with an arrow $\vec{x}$. There are no (intended) differences between the two, $\boldsymbol{x} = \vec{x}$
- I have also fluctuated between $\boldsymbol{x}_j$ and $\vec{x}^{(j)}$ for vector number j in the input set. Again, the two ways of writing amount to the same.

The money game

- I will give you 10 multiple-choice questions. You must answer all.
- I give you a million NOK before the game.
- In each round, you must bet your remaining money on the alternatives. Say there are 3 answers in the first round. You could bet any of the following, e.g.

	Your bet			You keep		
	Answer A	Answer B	Answer C	If A correct	If B correct	If C correct
Strategy 1	1,000,000	0	0	1,000,000	0	0
Strategy 2	400,000	300,000	300,000	400,000	300,000	300,000
Strategy 3	800,000	150,000	50,000	800,000	150,000	50,000

- You proceed to the nest round with the money you keep.
- What would be the best strategy?

Cross-entropy loss

- The underlying idea is that we want to maximize the joint probability of all the predictions we make
  - $\prod_{i=1}^{N} P\left(t^{(i)} \mid \vec{x}^{(i)}\right)$, over all the training data $i = 1,2, \dots , N$
    - (since the training data are independent)
- This is the same as maximizing
  - $\log \prod_{i=1}^{N} P\left(t^{(i)} \middle| \vec{x}^{(i)}\right) = \sum_{i=1}^{N} \log P\left(t^{(i)} \middle| \vec{x}^{(i)}\right)$
- This is the same as minimizing
  - $L_{CE}(\vec{w}) = -\log \prod_{i=1}^{N} P(t^{(i)} | \vec{x}^{(i)}) = \sum_{i=1}^{N} -\log P(t^{(i)} | \vec{x}^{(i)})$
  - Which is an instance of what is called the cross-entropy loss

More on cross-entropy loss

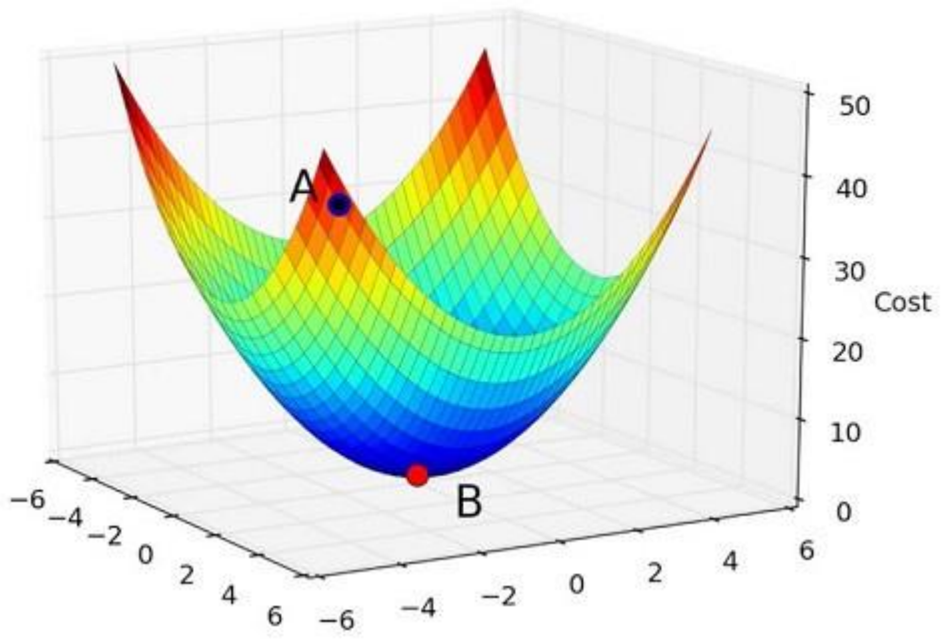
- When t $= 1$, $P(t \mid \vec{x}) = y = \frac{1}{1+e^{-\vec{w}\cdot\vec{x}}}$
- When t$= 0$, $P(t \mid \vec{x}) = 1 - y$
- Since
  - $y^t = y$ when t $= 1$
  - $y^t = 1$ when t $= 0$
  - $(1-y)^{(1-t)} = 1$ when t $= 1$
  - $(1-y)^{(1-t)} = (1-y)$ when t $= 0$
- $P(t|\vec{x}) = y^t(1-y)^{(1-t)}$, whether t $= 1$ or t $= 0$

7.5 Training the Logistic Regression Classifier

IN3050/IN4050 Introduction to Artificial Intelligence and Machine Learning

Gradient descent

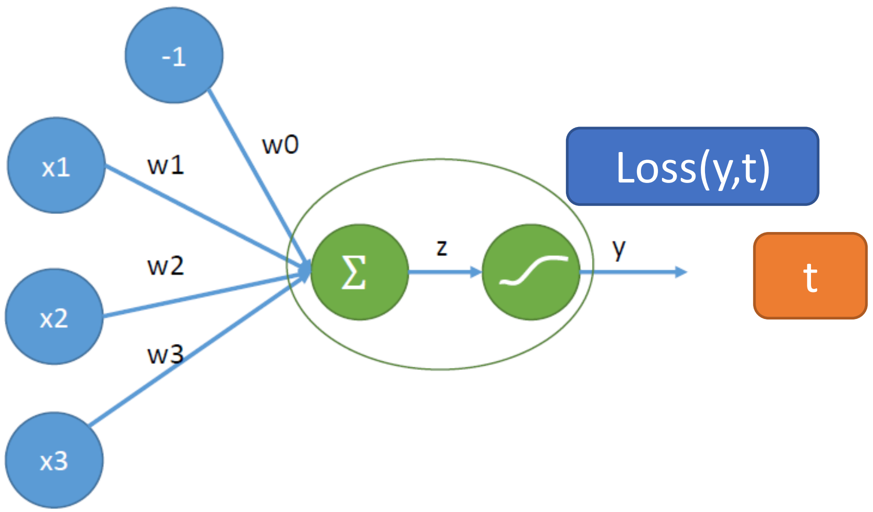
- The loss function tells us which model is best.
- How do we find it?
- No closed-form solution, i.e., formula as there are for linear regression,
- Good news:
  - The log-loss function is convex: you are not stuck in local minima
  - We know which way to go



The gradient

- We have
  - $L_{CE}(\vec{w}) = -\log \prod_{i=1}^{N} P(t^{(i)}|\vec{x}^{(i)}) = \sum_{i=1}^{N} -\log P(t^{(i)}|\vec{x}^{(i)})$
  - $P(t|\vec{x}) = y^t(1-y)^{(1-t)}$
  - $y = \sigma(\vec{w}\cdot\vec{x}) = \frac{1}{1+e^{-\vec{w}\cdot\vec{x}}}$
- We shall find
  - $\frac{\partial}{\partial w_j} L_{CE}(\vec{w})$ for each w_j
  - since $\frac{\partial}{\partial w_j} L_{CE}(\vec{w}) = \sum_{i=1}^{N} -\frac{\partial}{\partial w_j} \log P(t^{(i)}|\vec{x}^{(i)})$
  - we can consider what this looks like for one pair $\left(\vec{x}^{(i)}, t^{(i)}\right)$ at a time
  - $-\frac{\partial}{\partial w_i} \log P(t|\vec{x}) = -\frac{\partial}{\partial w_i}\left(\log\left(y^t(1-y)^{(1-t)}\right)\right) =$

$$-\frac{\partial}{\partial w_i}(\mathrm{t}\log(\mathrm{y}) + (1-\mathrm{t})\log(1-\mathrm{y}))$$



Derivative: the chain rule

- We shall find
- $-\frac{\partial}{\partial w_i} \log P(t|\vec{x}) = -\frac{\partial}{\partial w_i}\left(\mathrm{t}\log(\mathrm{y}) + (1-\mathrm{t})\log(1-\mathrm{y})\right)$
- $= -\frac{\partial}{\partial y}\left(\mathrm{t}\log(\mathrm{y}) + (1-\mathrm{t})\log(1-\mathrm{y})\right)\left(\frac{\partial}{\partial w_i} y\right)$ by the chain rule for derivatives
- $\frac{\partial}{\partial y}\left(\mathrm{t}\log(\mathrm{y}) + (1-\mathrm{t})\log(1-\mathrm{y})\right) = \frac{t}{y} - \frac{(1-t)}{(1-y)} = \frac{t(1-y)-y(1-t)}{y(1-y)} = \frac{(t-y)}{y(1-y)}$

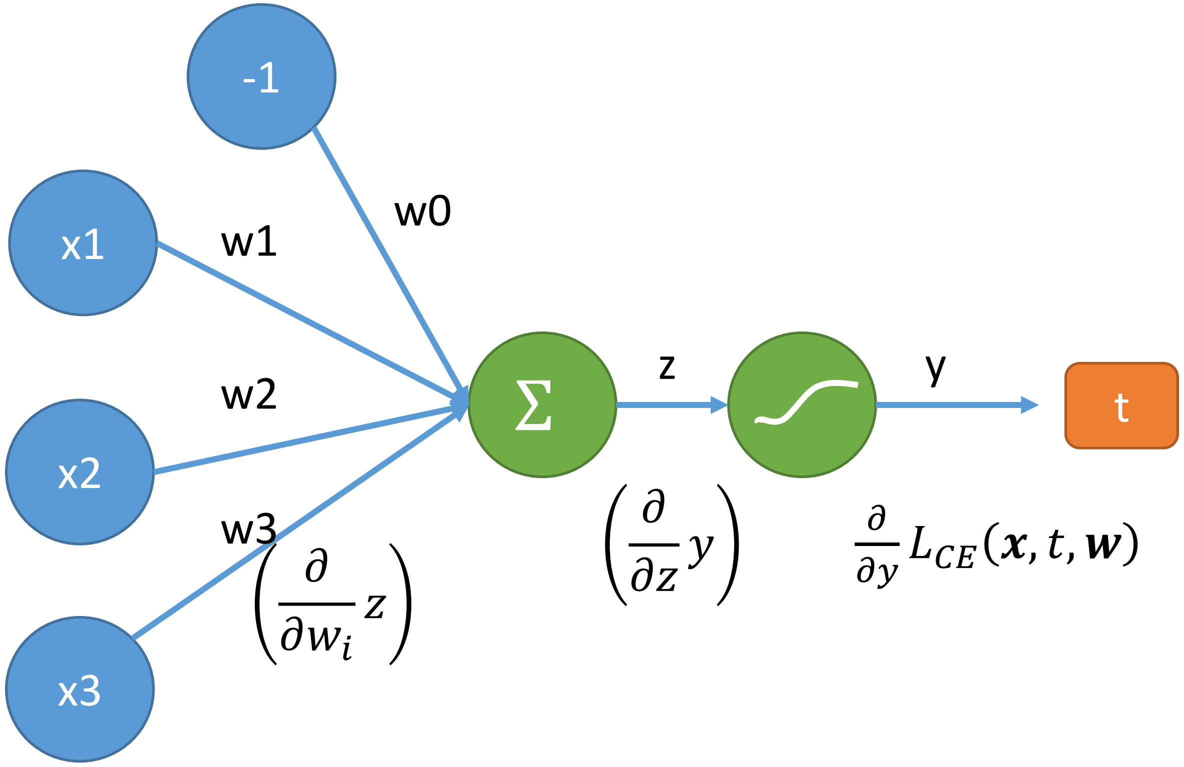
The derivative of the logistic function

- $y = \sigma(\vec{w} \cdot \vec{x}) = \frac{1}{1+e^{-\vec{w}\cdot\vec{x}}} = \frac{1}{1+e^{-z}}$, where $z = \vec{w} \cdot \vec{x}$

- $\frac{\partial}{\partial w_i} y = \left(\frac{\partial}{\partial z} y\right)\left(\frac{\partial}{\partial w_i} z\right)$
- $\frac{\partial}{\partial z} y = y(1 - y)$ (the logistic function)
- $\frac{\partial}{\partial w_i} z = x_i$
- $\frac{\partial}{\partial w_i} y = y(1 - y)x_i$

Putting it together graphically

- $\frac{\partial}{\partial w_i} L_{CE}(\boldsymbol{x}, t, \boldsymbol{w})$=
- $\frac{\partial}{\partial y} L_{CE}(\boldsymbol{x}, t, \boldsymbol{w}) \left(\frac{\partial}{\partial z} y\right) \left(\frac{\partial}{\partial w_i} z\right)$



Putting it all together

- $\frac{\partial}{\partial w_i} L_{CE}(\boldsymbol{x}, t, \boldsymbol{w}) = -\frac{\partial}{\partial w_i} \log P(t|\vec{x}) = -\frac{\partial}{\partial w_i}\big(\mathrm{t}\log(\mathrm{y}) + (1-\mathrm{t})\log(1-\mathrm{y})\big)$

- = $-\frac{\partial}{\partial y}\big(\mathrm{t}\log(\mathrm{y}) + (1-\mathrm{t})\log(1-\mathrm{y})\big)\left(\frac{\partial}{\partial w_i} y\right)$
- $= -\frac{(t-y)}{y(1-y)} y(1-y)x_i = -(\mathrm{t}-\mathrm{y})x_i$

- A long journey – but the result is simple
- Adding together (matrix multiplication) for all the training data yields the gradient
- $(\nabla f)_i = \frac{\partial}{\partial w_i} L_{CE}(X, T, \boldsymbol{w}) = \sum_{j=1}^{N} -(t_j - y_j)x_{j,i}$

7.6 Variants of Gradient Descent

IN3050/IN4050 Introduction to Artificial Intelligence and Machine Learning

Variants of gradient descent

Batch training:

- Calculate the loss for the whole training set, and the gradient for this
- Make one move in the correct direction
- Repeat (an epoch)

- Can be slow

Stochastic gradient descent:

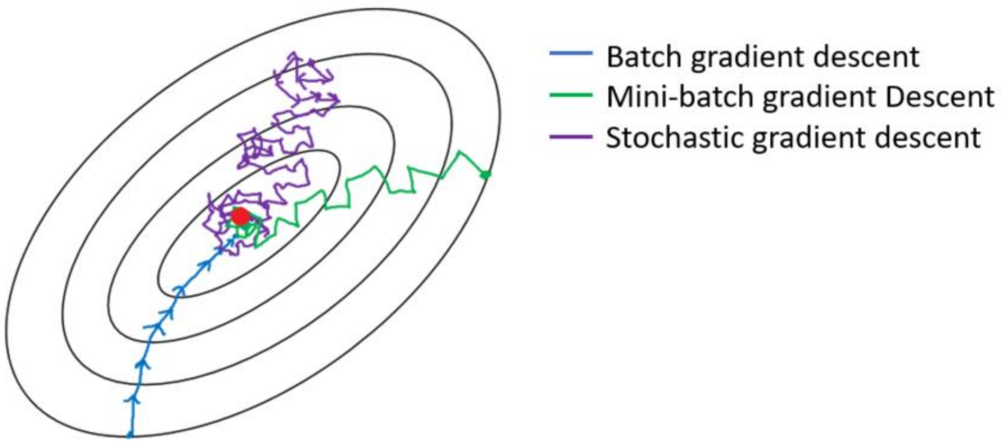
- Pick one item
- Calculate the loss for this item
- Calculate the gradient for this item and move in the opposite direction

- Each move does not have to be in towards the direction of the gradient for the whole set.
- But the overall effect may be good
- Can be faster

Variants of gradient descent

Mini-batch training:

- Pick a subset of the training set of a certain size
- Calculate the loss for this subset
- Make one move in the direction opposite of this gradient
- Repeat (an epoch)



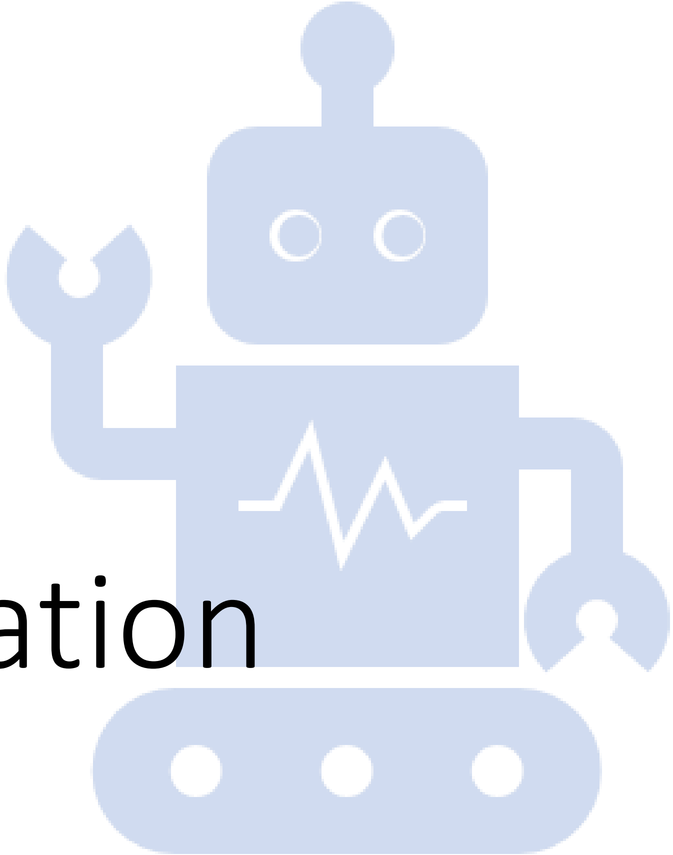
- A good compromise between the two extremes
- (The other two are subcases of this)

https://suniljangirblog.wordpress.com/2018/12/13/variants-of-gradient-descent/

7.7 Multi-Class Classification

IN3050/IN4050 Introduction to Artificial Intelligence

and Machine Learning



Multi-class classification

Classification

- Assign a label (class) from a finite set of labels to an observation

 ?

- So far, many algorithms and examples have been binary: *yes-no, 1-0*
- But many classification tasks are multi-class:
    - To each observation $\boldsymbol{x}$ choose one label from a finite set **T**
- What is different?

1-of-N or "one hot encoding"

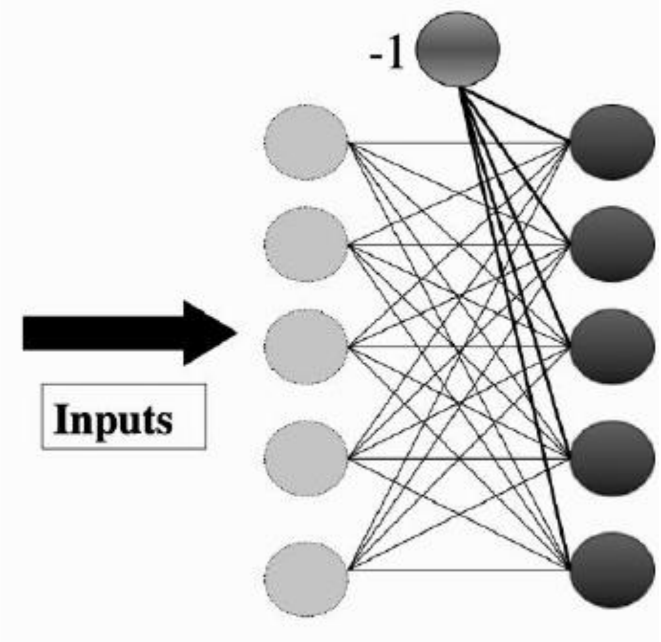
- The labels might be categorical:
  - 'apple', tomato', 'dog', 'horse'
- The algorithms demand numerical attributes.
- First attempt
  - 'apple' = 1
  - 'tomato' = 2
  - 'dog' = 3
  - etc.
- Why isn't this a good idea?
- Better:
  - 'apple' = (1, 0, 0, 0, 0, 0)
  - 'tomato' = (0, 1, 0, 0, 0, 0)
  - 'dog' = (0, 0, 1, 0, 0, 0)
  - etc.
- Both the target and the predicted value are vectors.

One vs. rest

class 2

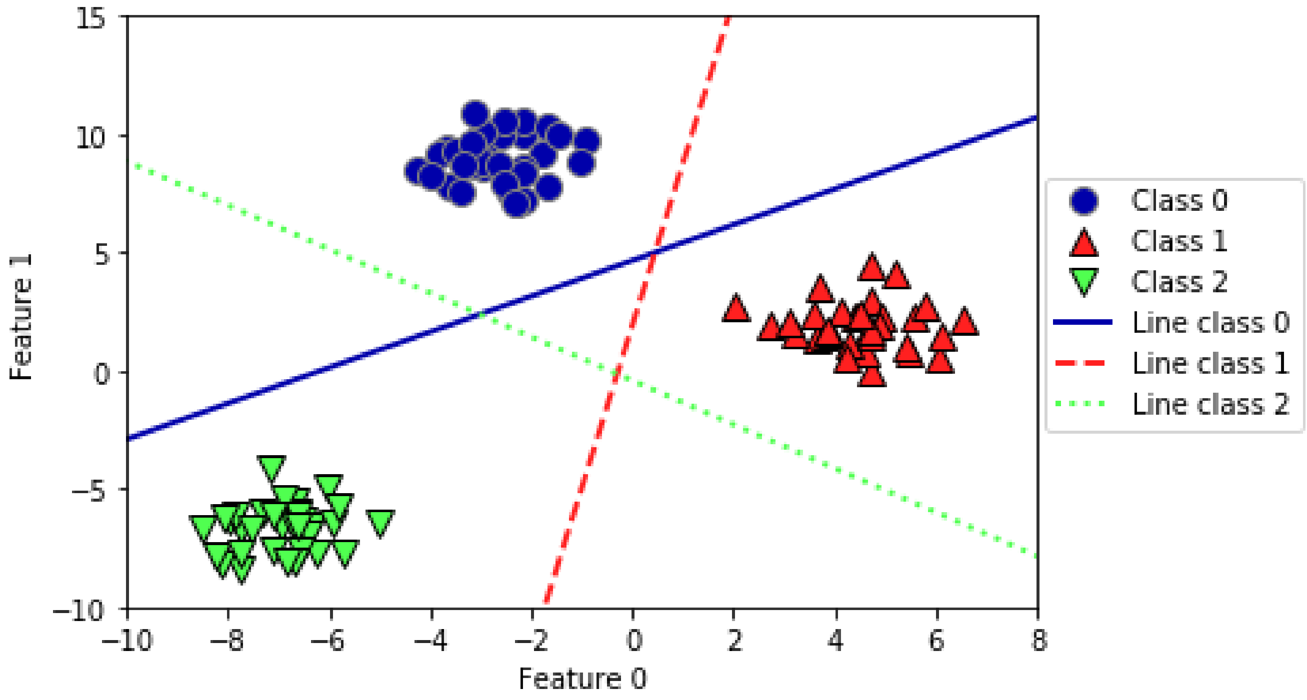
$$\begin{bmatrix} x_{1,0} & x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ x_{2,0} & x_{2,1} & x_{2,2} & \cdots & x_{2,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{N,0} & x_{N,1} & x_{N,2} & \cdots & x_{N,m} \end{bmatrix} \begin{bmatrix} w_{0,1} & w_{0,2} & \cdots & w_{0,n} \\ w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix} = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,n} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N,1} & y_{N,2} & \cdots & y_{N,n} \end{bmatrix}$$

- Train one classifier for each class
  - (cf. Marsland's ch.3)
- Works for multi-label classification
- But the multi-class classifiers should only propose one class
- How to choose?



One vs. rest contd.

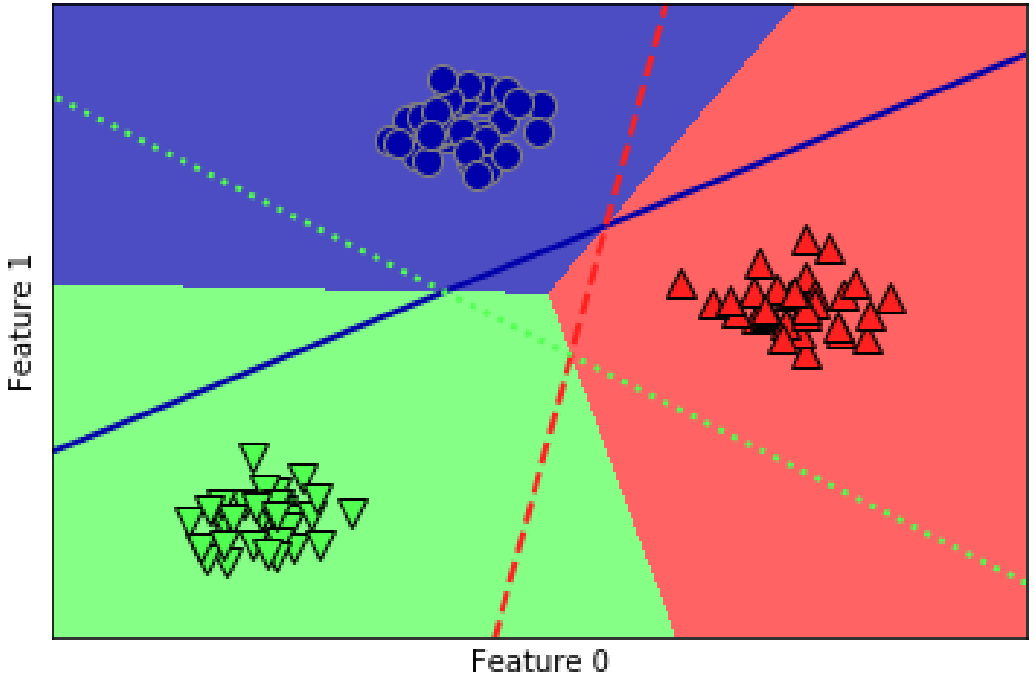
- It is easy to decide for items which fall into exactly one class
- But what if they fall into
  - More than one class?
  - No classes?



https://github.com/amueller/introduction_to_ml_with_python

One vs. rest

- If each classifier predicts a score, compare the scores for the classes
- Choose the class with the highest score.
- E.g., log. reg.:
  - Probability of being red: 0.8
  - Probability of being blue: 0.7
  - Choose red



https://github.com/amueller/introduction_to_ml_with_python

One vs. rest

$$\begin{bmatrix} x_{1,0} & x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ x_{2,0} & x_{2,1} & x_{2,2} & \cdots & x_{2,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{N,0} & x_{N,1} & x_{N,2} & \cdots & x_{N,m} \end{bmatrix} \begin{bmatrix} w_{0,1} & w_{0,2} & \cdots & w_{0,n} \\ w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix} = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,n} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N,1} & y_{N,2} & \cdots & y_{N,n} \end{bmatrix}$$

- For $\boldsymbol{x}_i$ choose the j for which $y_{i,j}$ is the max of $\{y_{i,1}, y_{i,2}, \ldots, y_{i,n}\}$

Multinomial Logistic Regression

- Logistic regression gives another option for making a multi-class classifier
- Called multinomial logistic regression, or softmax regression
  - also called maximum entropy (maxent) classifier,
- With one class we considered $P(t=1|\vec{x}) = \frac{e^{\vec{w}\cdot\vec{x}}}{1+e^{\vec{w}\cdot\vec{x}}} = \frac{1}{1+e^{-\vec{w}\cdot\vec{x}}}$
- and implicitly $P(t=0|\vec{x}) = 1 - \frac{e^{\vec{w}\cdot\vec{x}}}{1+e^{\vec{w}\cdot\vec{x}}} = \frac{1}{1+e^{\vec{w}\cdot\vec{x}}}$
- We now consider a linear expression $\vec{w}_i$, for each class $C_i, i = 1, \ldots, k$
- The probability for each class is then given by the softmax function

$$P(C_j|\vec{x}) = \frac{e^{\vec{w_j}\cdot\vec{x}}}{\sum_{i=1}^{k} e^{\vec{w_i}\cdot\vec{x}}}$$

Training softmax

- It is possible to train towards softmax.
- In each step, we have to calculate y_j for each class j.
- In updating the weights for one class i, the predicted values for all the classes are considered and contribute to the update.
- We skip the details.