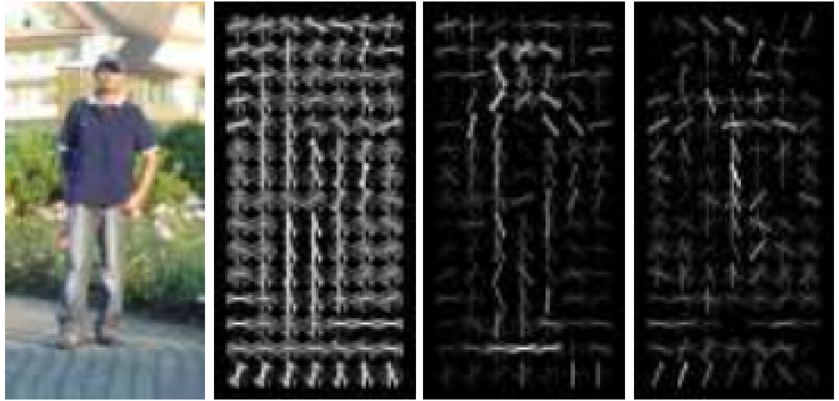
Introduction to Machine Learning

Lecture 08

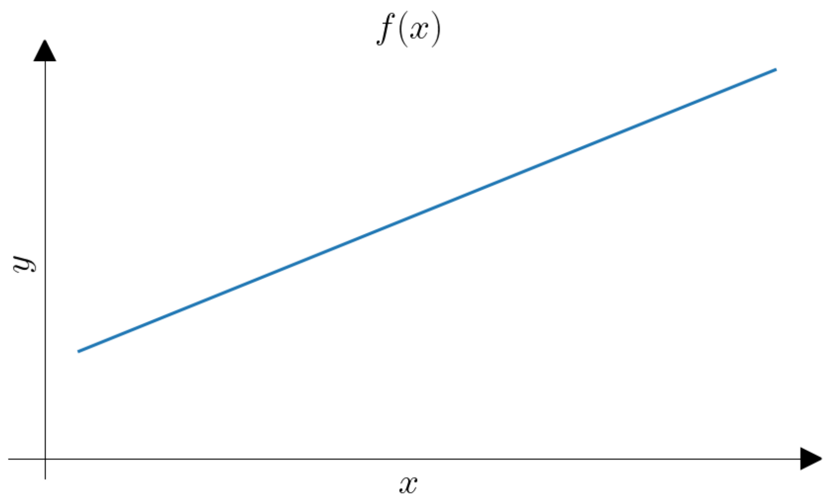
Automatic Image Analysis

May 31, 2021

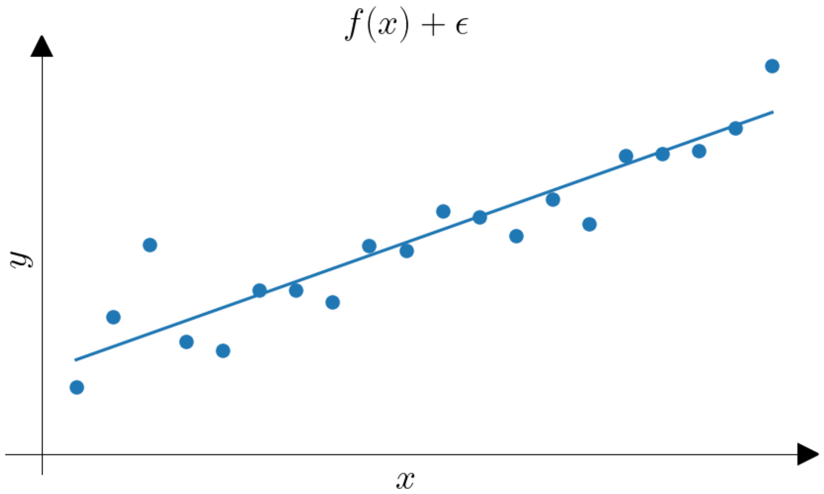
Technische Universität Berlin



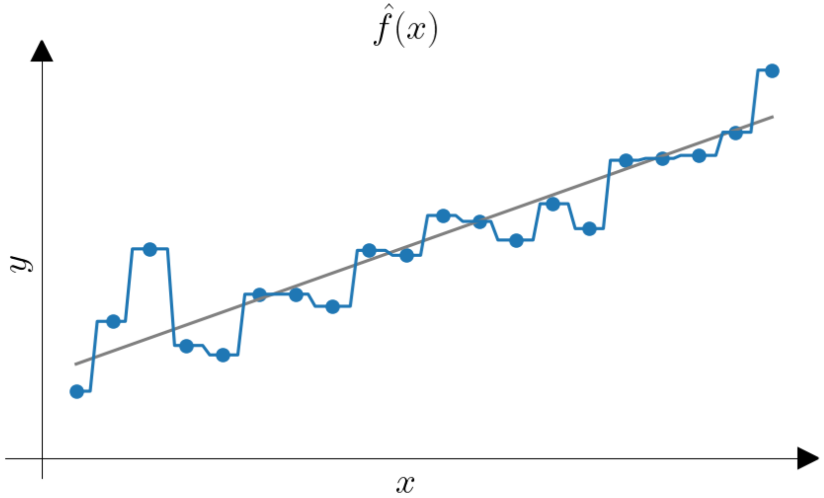
- How to connect our features to actual categories or measurements of image content in human terms?
- It would be hard to write heuristics to describe which HOG/SIFT feature corresponds to a dog or cat.
- There are two reason to make this connection. One is prediction of responses for unseen data. The other to analyze the connection between x and y (in statistics called inference).
- Image from Histograms of oriented gradients for human detection, Dalal & Triggs



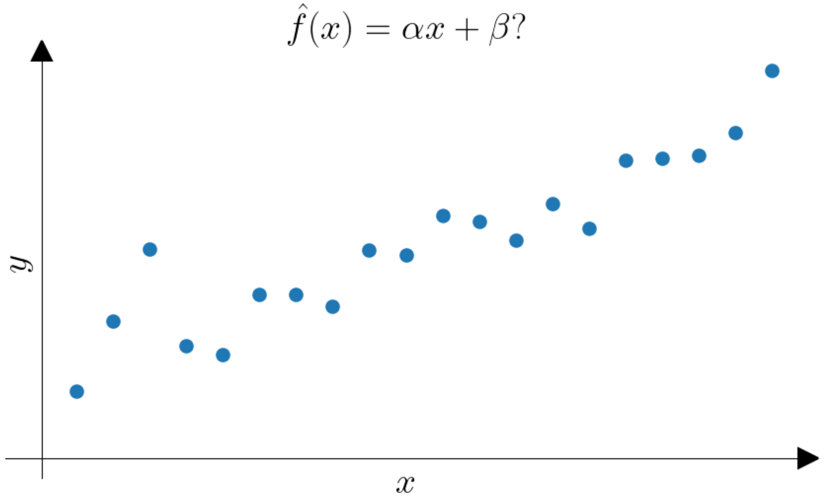
- We assume that there is a true mapping f that maps from the image or feature space (predictor x) to e.g. an object category (response y).
- x is also often called feature, input variable, just variable or independent variable.
- y is also often called ground truth, target, label, output variable or dependent variable
- In the following we will often consider x and y to be multidimensional but visualize them mostly as scalars.



- We want to estimate this function based on data we collected.
- When data is collected, we make an error ϵ.
- This error is almost always of probabilistic nature. Our data is noisy.
- The set of measurements is denoted by (Y, X) with all values collected for X and their corresponding ys in Y.



- Modeling of a wide range of functional forms possible.
- Usually very high number of observations necessary.
- In this case simply $\hat{f}(x) = Y_{argmin(|X-x|)}$



- We make an assumption about the functional form of f.
- In this case we might assume that the f that generated our data is linear.

How can we estimate our parameters?

- We need a criterion that tells us how well the estimation fits our data.
- An often used metric is the mean square error.

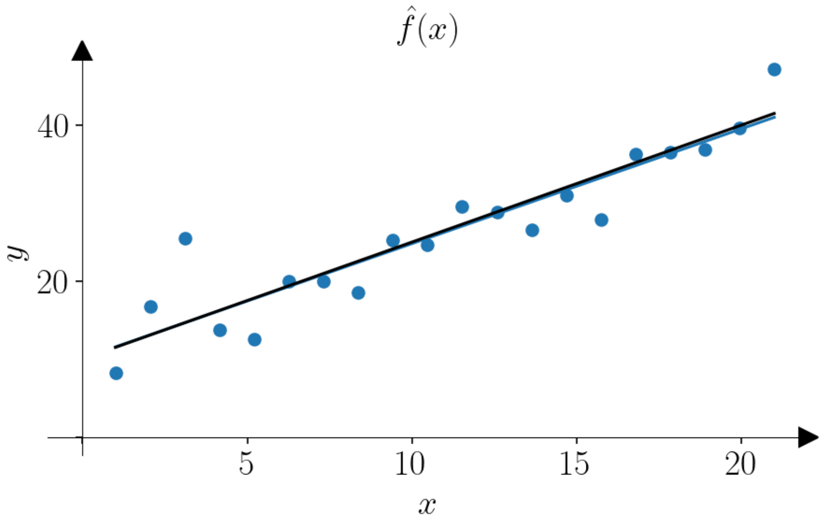
$$E(\alpha, \beta) = \frac{1}{n} \sum_i (y_i - \hat{f}(x_i))^2 = \frac{1}{n} \sum_i (y_i - \alpha x_i + \beta)^2$$

$$\frac{dE}{d\alpha} \stackrel{!}{=} 0 \text{ and } \frac{dE}{d\beta} \stackrel{!}{=} 0$$

$$\hat{\alpha} = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

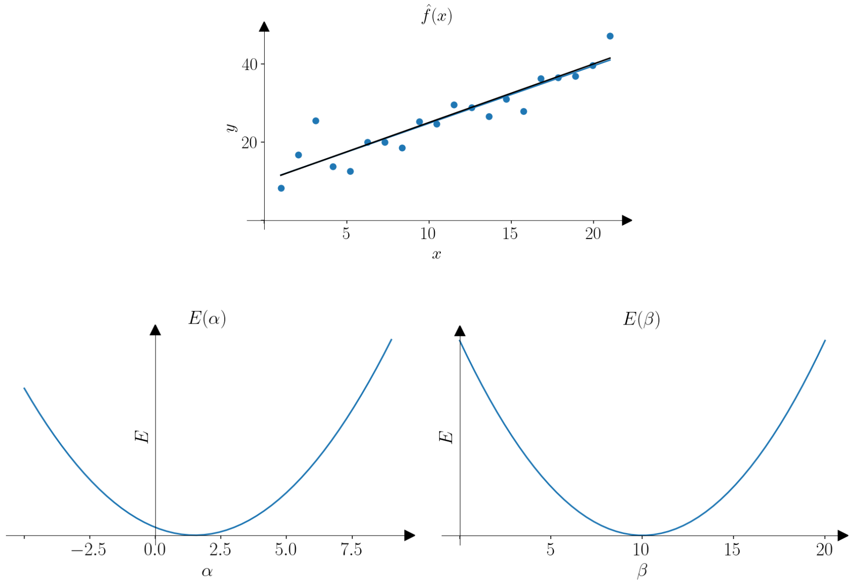
$$\hat{\beta} = \bar{y} - \hat{\alpha}\bar{x}$$

- To minimize the error, the first derivatives have to be zero.
- Using a linear model and mean square error allows for an analytical solution.
- Procedure is known as linear regression, a very simple and very popular method.

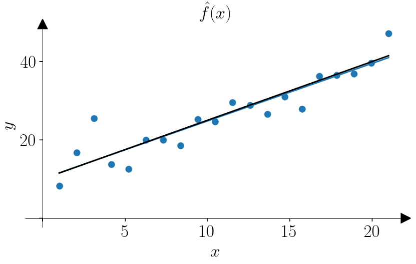


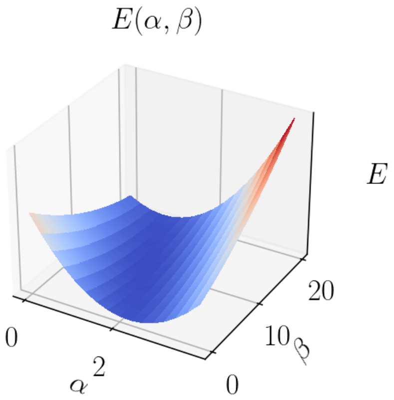
- Black shows the data generating ground truth, blue the estimate based on the measured data.

Error surface

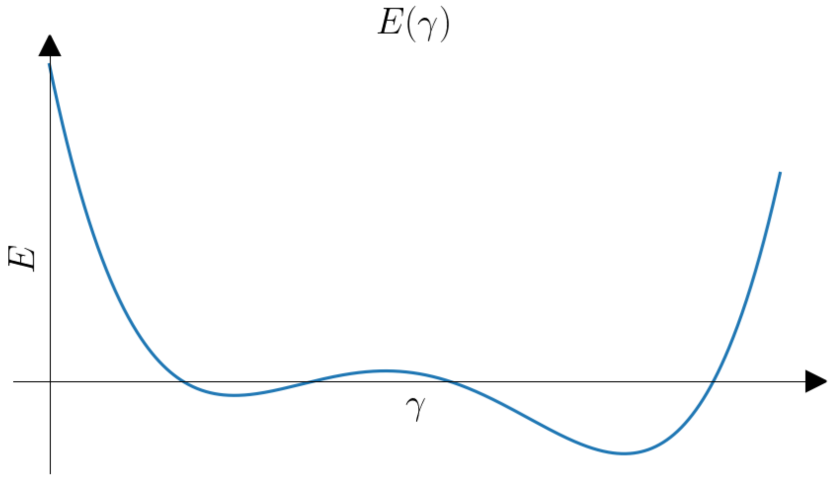


- This slide shows the error surface of the linear model we just fitted to the data.
- On the left for the parameter alpha on the right for beta.
- We were lucky, not only has our problem a analytical solution it also has a convex error surface.



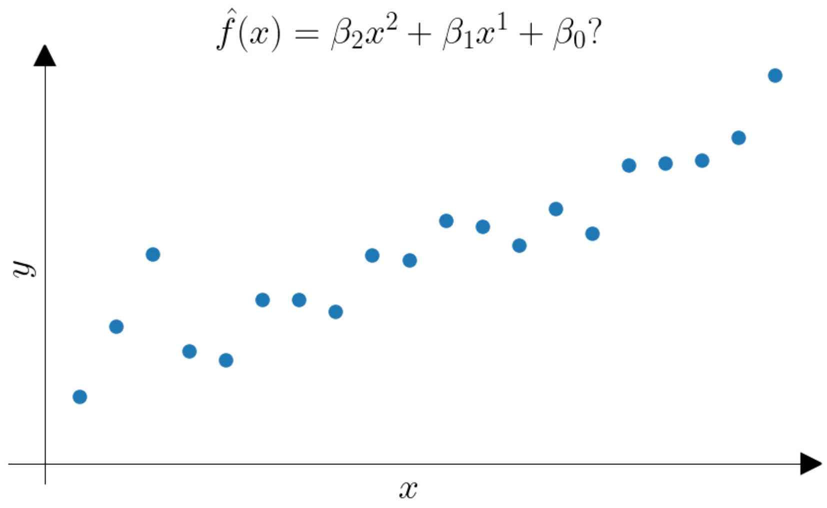


- Same function plotted in 2d.
- We were lucky, not only has our problem a analytical solution it also has a convex error surface.



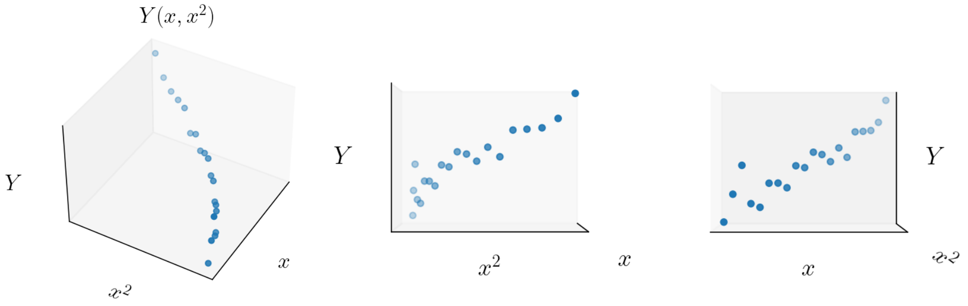
- Unfortunately, for more complex problems, this error surfaces are often non-convex.
- Especially when we can not find analytical solutions, local minima in such non-convex objective function can be problematic.

What if linear is not good enough?



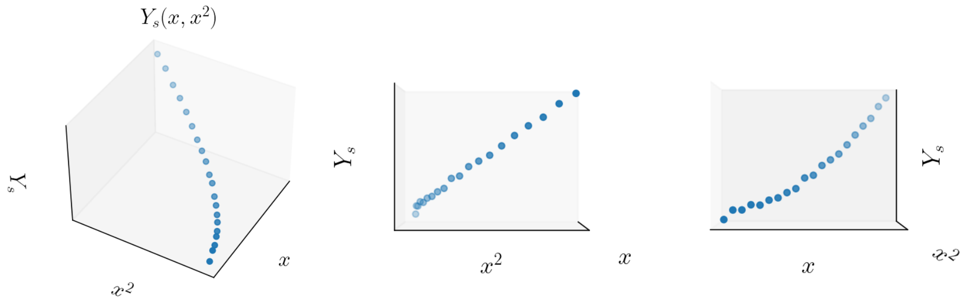
- A common pattern in machine learning is to apply linear methods trained on non-linear functions of the data.
- We map in a non-linear way to a higher dimensional features space and do linear regression.

What if linear is not good enough?



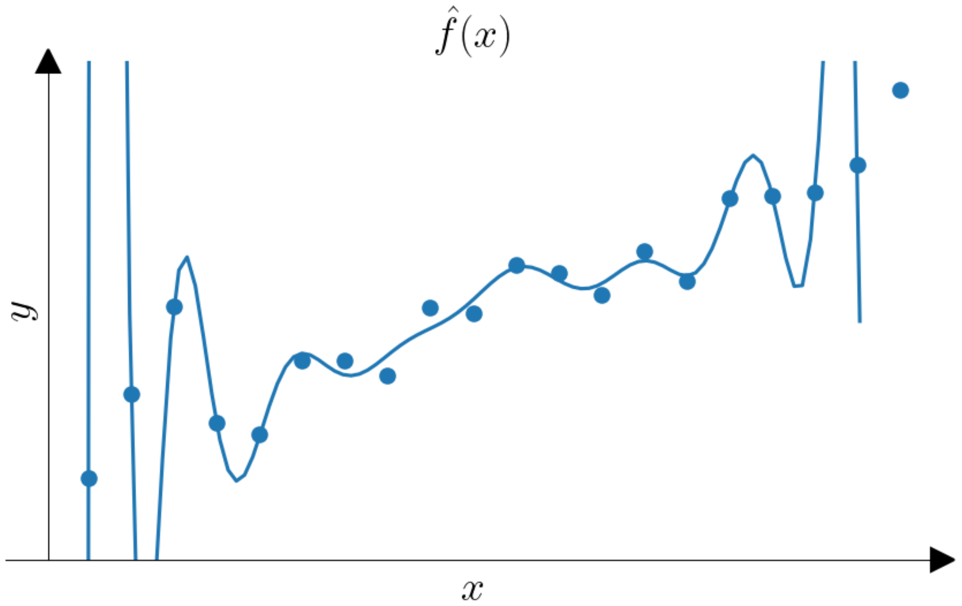
- In our case we can map from our scalar feature space to $x' = (x, x^2)$
- We see that relation between x and y stays the same while the x^2 dimension shows square root characteristics.

What if linear is not good enough?

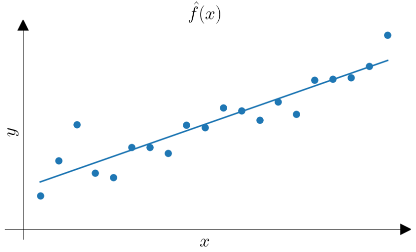


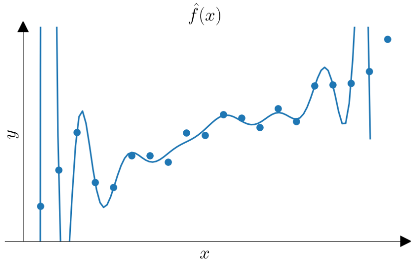
- On this slide the underlying f from which the data is generated is $f(x) = x^2$.
- We see that relation between x and y is polynomial, while the x^2 dimension now is linear.

What if linear is not good enough?



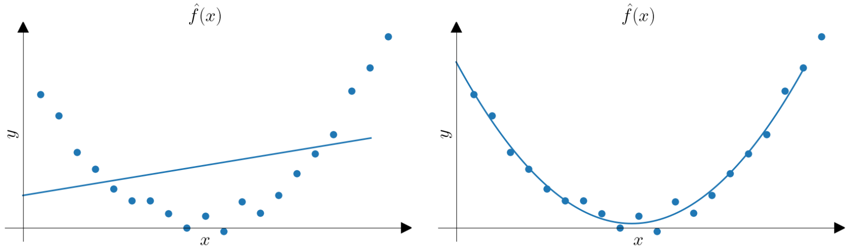
- This slide shows a 17th degree polynomial fitted to our data from before.
- $y = f(x) + \epsilon = \frac{3}{2}x + 10 + \mathcal{N}(0, 4)$



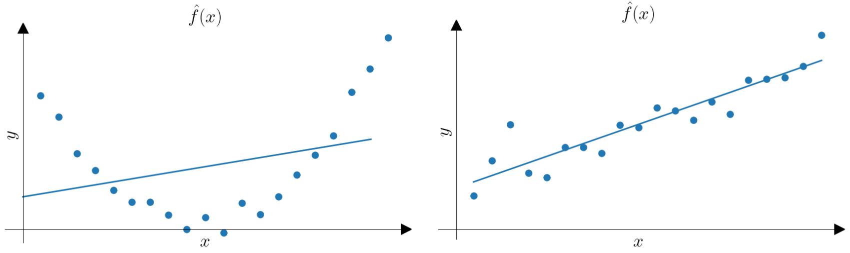


- Which of the two estimates of f is better?
- $y = f(x) + \epsilon = \frac{3}{2}x + 10 + \mathcal{N}(0, 4)$

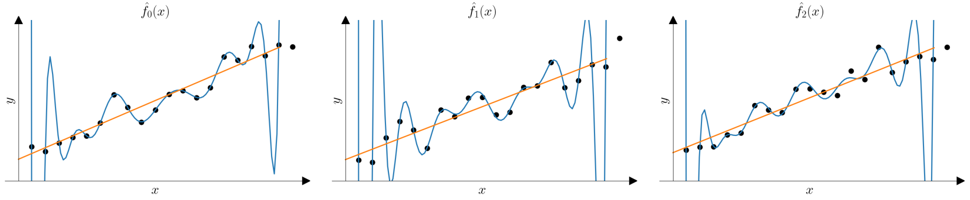
Underfitting



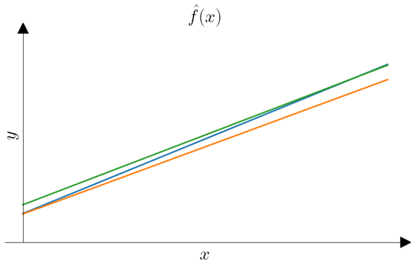
- Which of the two estimates of f is better?
- $y = f(x) + \epsilon = 4(x - 10)^2 + \mathcal{N}(0, 4)$

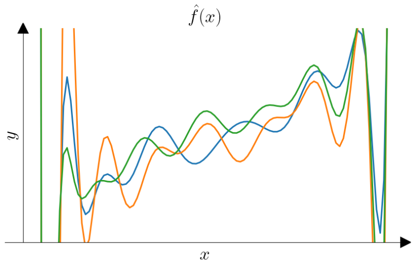


- If we restrict our model e.g. by limiting the complexity we call that bias.
- In this case the model is limited to learn linear mappings (high bias).

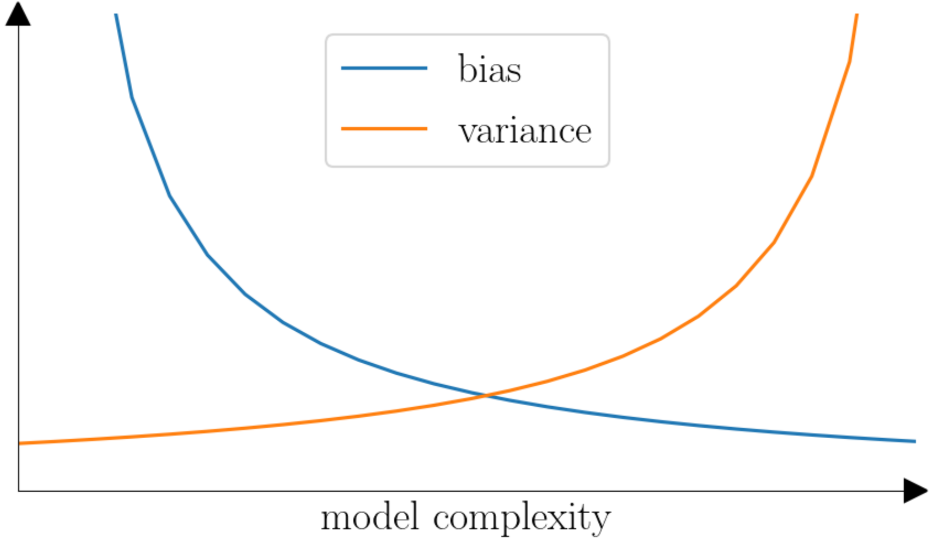
nnnn 

- Three different datasets. Each generated with the linear f we used above.
- A 17th degree polynomial is fitted to each of them.
- We observe a high variance in the resulting polynomials.
- $y = f(x) + \epsilon = \frac{3}{2}x + 10 + \mathcal{N}(0, 4)$

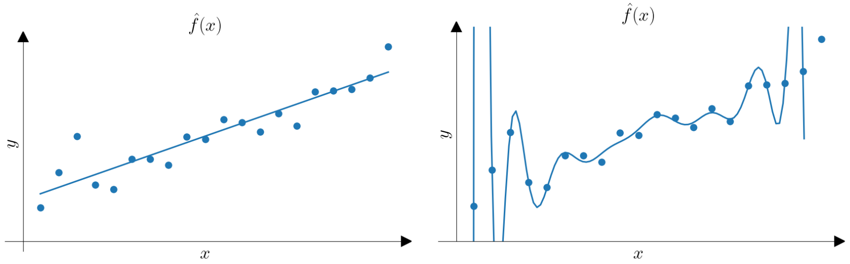




- Three different datasets. Each generated with the linear f we used above.
- Comparison on linear models versus polynomial models fitted to the same data.

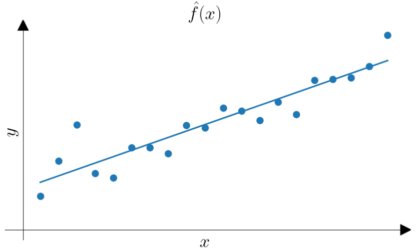


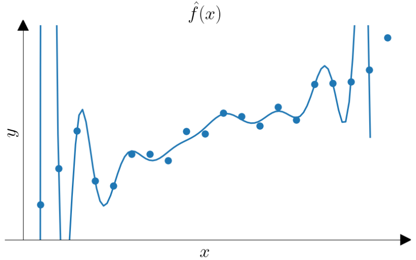
- Higher model complexity leads to higher variance and lower bias.



- How can we measure the quality of our model?
- Which of the two is the better fit?

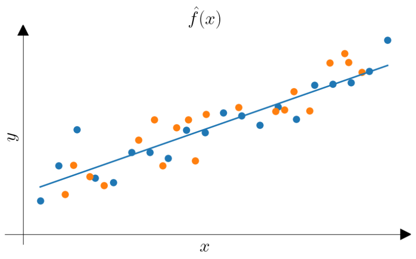
$$E = \frac{1}{n}\sum_{i}(y_i - \hat{f}(x_i))^2 \quad (1)$$

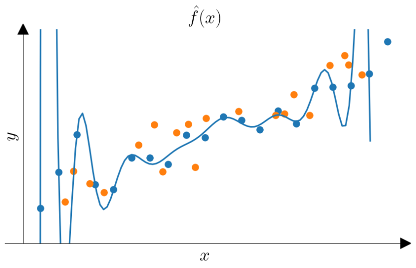




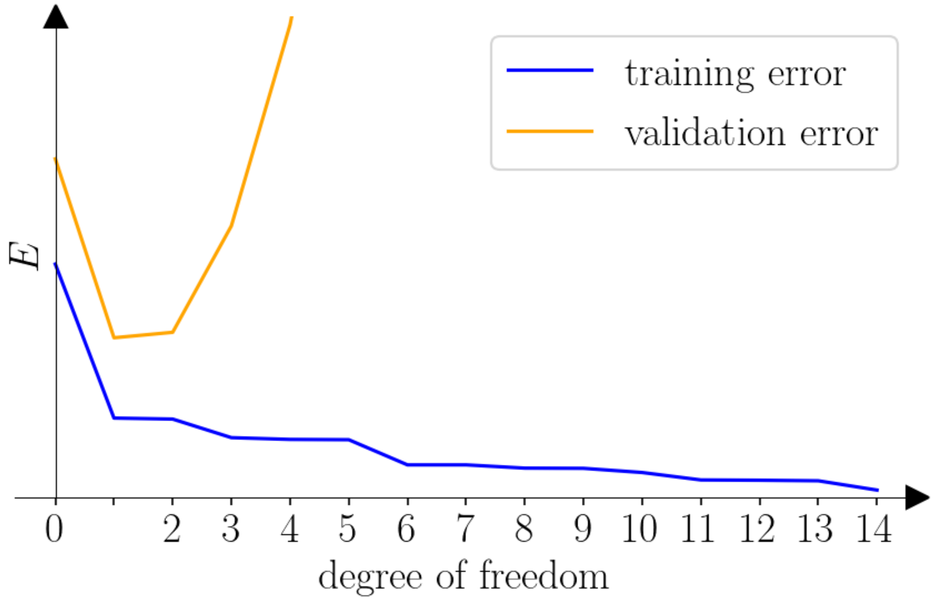
- Which of the two has the smaller error (does minimize our objective)? *rightarrow* Error becomes smaller with increasing model complexity.

$$E = \frac{1}{n}\sum_i (y_i - \hat{f}(x_i))^2 \qquad (2)$$

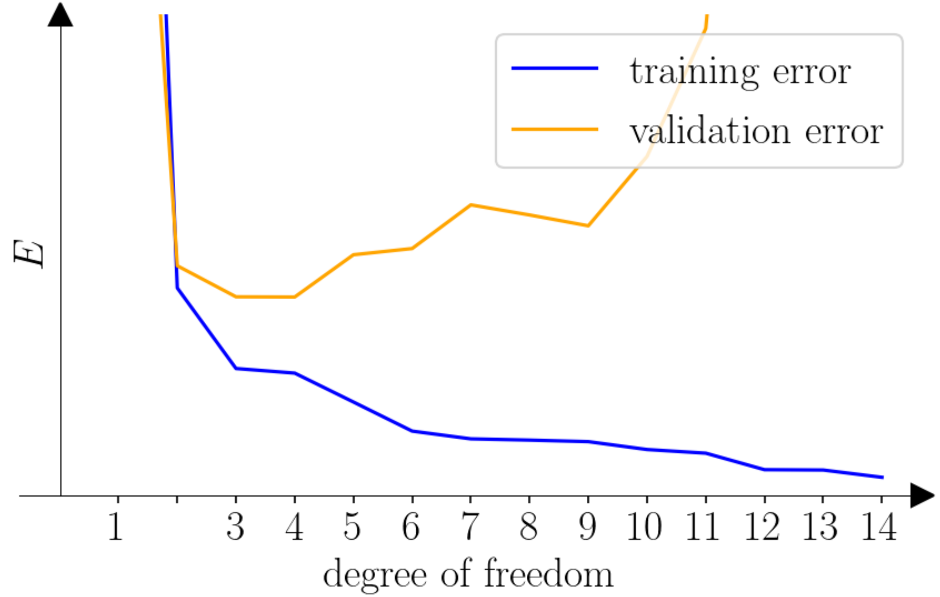




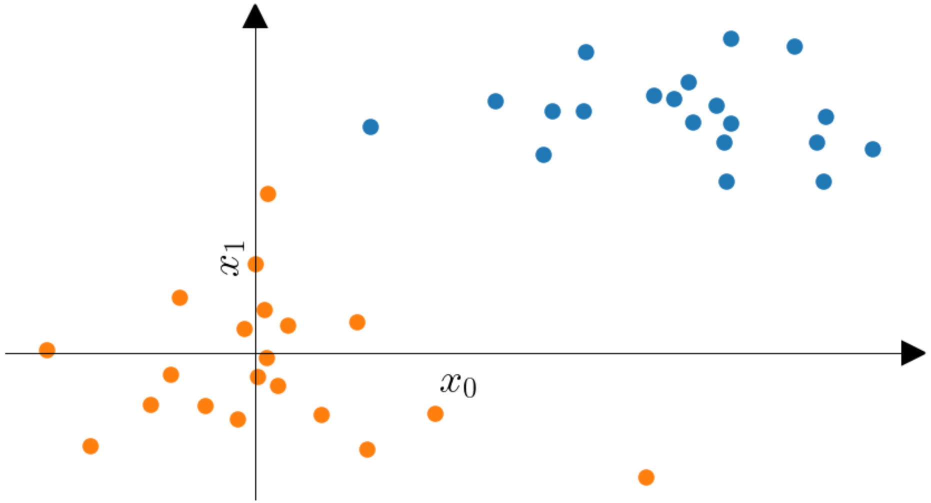
- Which of the two has the smaller error (does minimize our objective)?
- Split data set before fitting the model and test on unseen data. → Error shows when model overfits the training data.



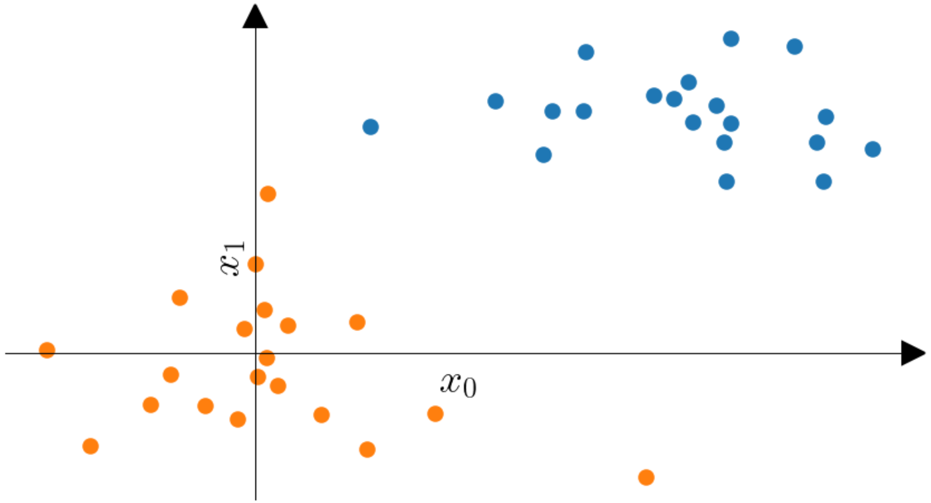
- A number of models with increasing complexity was fitted to some training data.
- What do you think what form the data generating distribution has?



- A number of models with increasing complexity was fitted to some training data.
- What do you think what form the data generating distribution has?

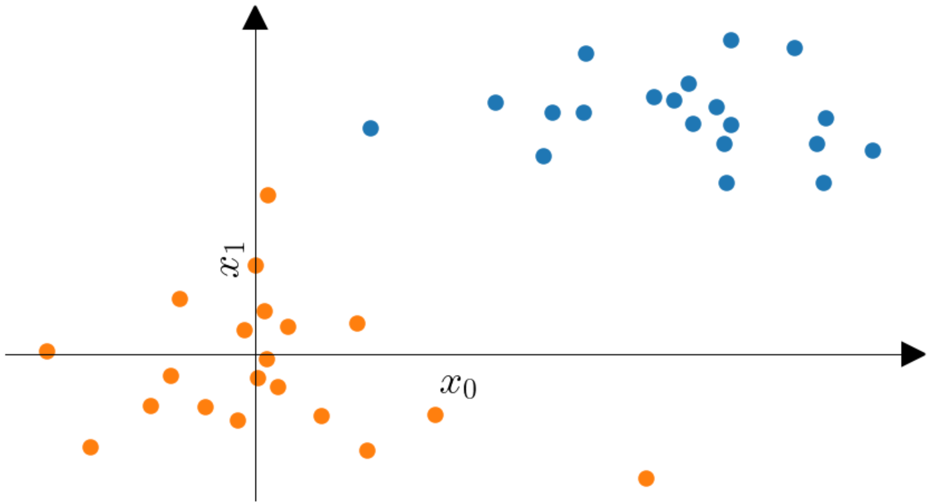


- So far we looked at data were the response variable y was quantitative.
  $\rightarrow$ This class of problems is referred to as regression problems.
- Now we want to look at problems were the response is qualitative or categorical.
- Examples: Categorization of facial expressions or objects in images.



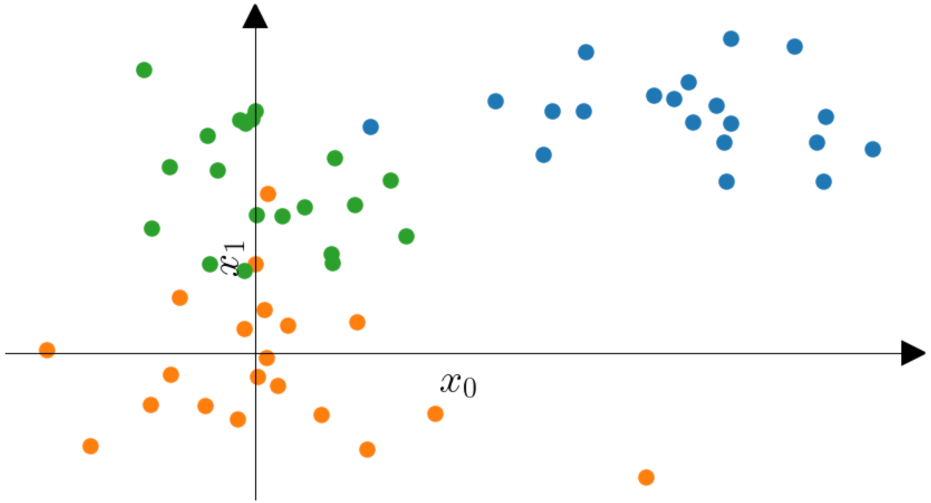
- To do this our goal is it to identify a boundary in between a set of training points that separates the two classes.
- Whether we call a problem a classification or a regression problem depends only on the response variable.
- As for regression, we look only at quantitative predictor variables here.
- When the predictor variable is categorical as e.g. in natural language processing they are usually embedded in a quantitative space.

Can we solve this with Linear Regression?



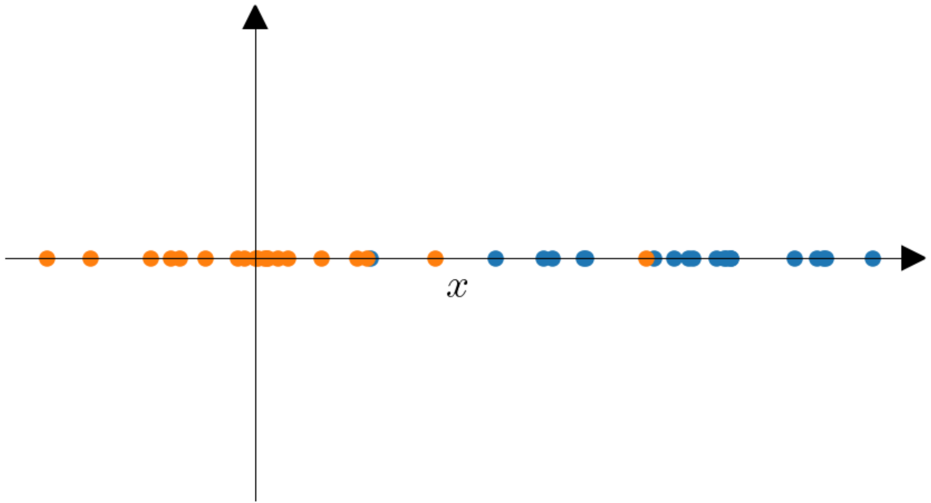
- We could define the blue class label as 0 and the orange class label as 1 and then apply linear regression.

Can we solve this with Linear Regression?

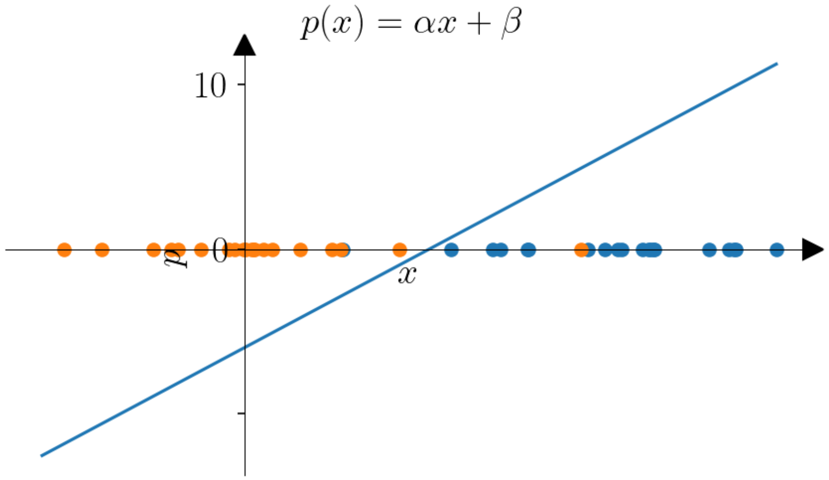


- However, this would not generalize to more than the binary case.
- For three classes we cannot define and order as e.g. orange > blue > green, which would be implied if we would assign numbers to our classes as before.

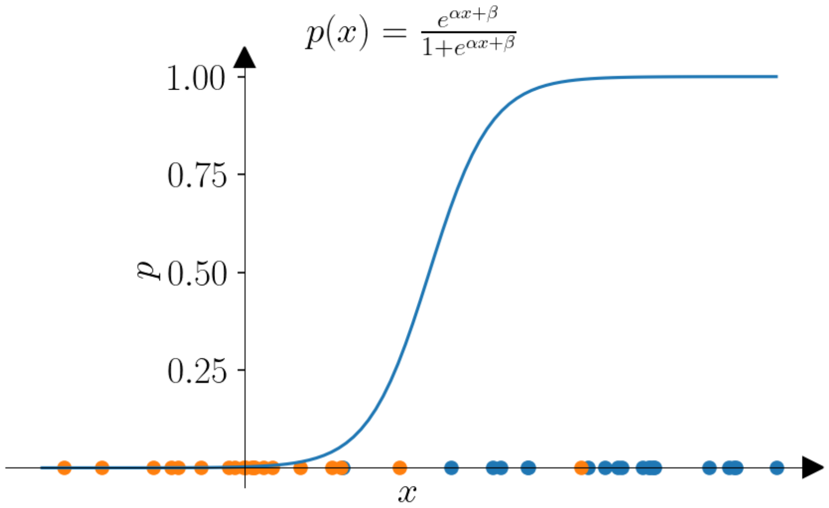
$$P(class = blue|x) \tag{3}$$



- There is a number of algorithms to approach this problem: LDA, SVM, Trees, Forests, K-nearest-neighbors, Boosting
- For this lecture however, we will first focus on Logistic Regression.
- The core idea is to formulate the problem as the regression of a probability function.
- This probability connects the predictor variables with the categorical response variable.
- For easier illustration, we switch to a two class problem with 1-dimensional input.

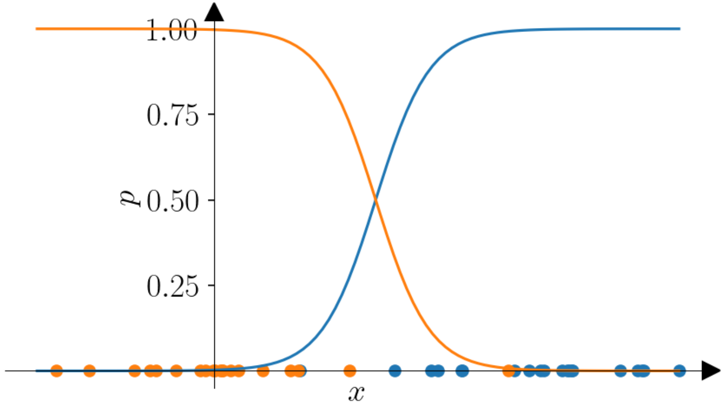


- How to model the probability mass function? As a linear mapping as for the regression?
- p gets arbitrarily big, > 1 and < 1



- How to model the probability mass function? The logistic function is one of many that makes the result look more like a probability.

$$p(blue|x) = \frac{e^{\alpha x+\beta}}{1+e^{\alpha x+\beta}}$$
$$p(orange|x) = 1 - p(blue|x)$$



- How to model the probability mass function? The logistic function is one of many that makes the result look more like a probability.

$$p(Y|X,\Theta) = \prod_{\forall i} p(y_i|x_i)$$

$$\log\ p(Y|X,\Theta) = \sum_{\forall i} \log\ p(y_i|x_i)$$

$$E(\Theta) = -\log\ p(Y|X,\Theta) = -\sum_{\forall i} \log\ p(y_i|x_i)$$

- If the samples in our data set are independent and identically distributed (iid assumption) we can write the probability of our dataset beeing generated by our model as a product of the probabilities of the samples.
- In our case $\Theta = (\alpha, \beta)$.
- If we fix the data and vary the parameters Θ, we call this the likelihood or log-likelihood respectively.
- We use the logarithm of the likelihood function for convenience.
- We define the error function as the negative log-likelihood and as for the linear regression we can use the derivatives of the error function to determine optimal estimates of α and β for the given dataset.

$$E(\Theta) = -\sum_{\forall i} q(x) \log \; p(y_i|x_i) = H(q, p)$$

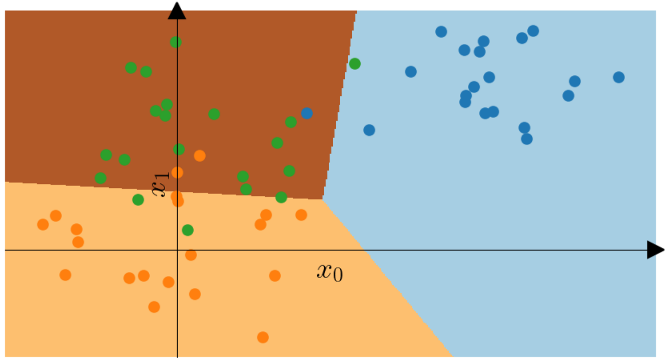
- The resulting error function describes the cross entropy between the modeled probability distribution and the distribution q which is 1 if the sample belongs to the respective class and 0 otherwise.
- For further reading we refer to Bishop p48ff.

$$p_i(x) = \sigma_i(x) = \frac{e^{z_i(x)}}{\sum_{\forall j} e^{z_j(x)}}$$

with

$$z_i(x) = \alpha_i x + \beta_i$$

- Using the *softmax* function which is a generalization of the logistic function, we can apply logistic regression to multi class problems.



g

$$p(c|x,\Theta) = \begin{pmatrix} P(blue|x) \\ P(orange|x) \\ P(green|x) \end{pmatrix} = \sigma\left(\begin{pmatrix} \alpha_{00} & \alpha_{01} \\ \alpha_{10} & \alpha_{11} \\ \alpha_{20} & \alpha_{21} \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} + \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix} \right)$$

- Using the *softmax* function which is a generalization of the logistic function, we can apply logistic regression to multi class problems.
- The matrix α_{ij} is often called the weight matrix W.
- The vector β_i is often called the bias b.
- Such a classifier can be and often is written as $y = \sigma(Wx)$ w.l.o.g. using an additional matrix column for the bias.

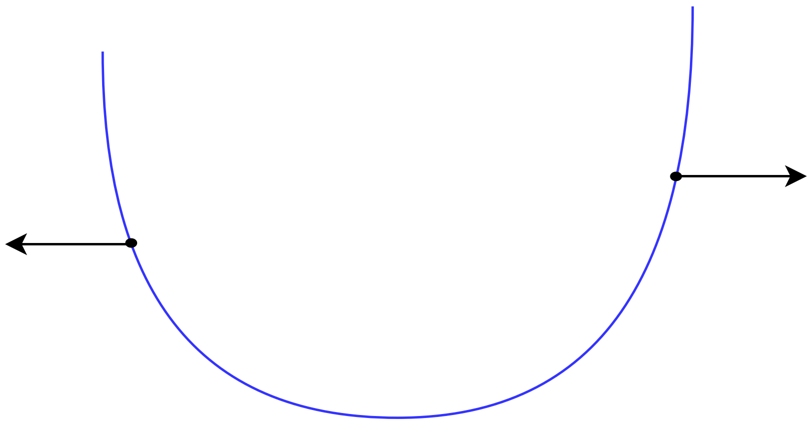
- The introduced formulation for Logistic Regression has no analytical solution.
- We can search for minima by walking on the error surface in the direction of steepest decent.

$$p(c|x) = y(x) = \sigma(Wx), \quad \sigma_i(x) = \frac{e^{x_i}}{\sum_{\forall j} e^{x_j}}$$

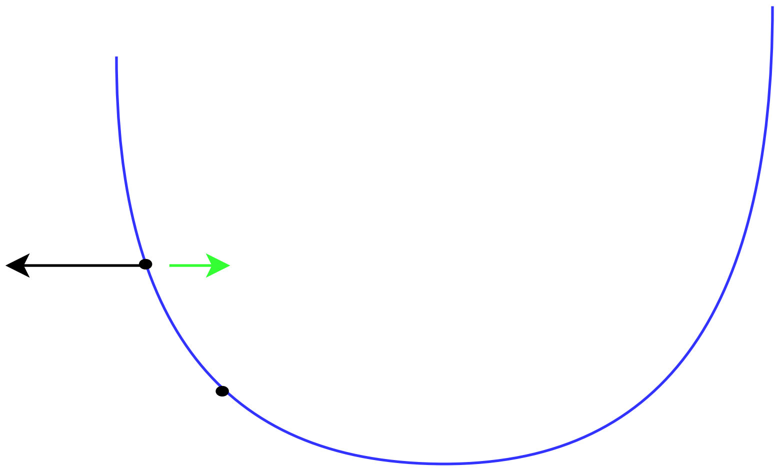
$$\nabla E(\theta) = \sum_i (y_i - t_i) x_i$$

$$\nabla E(\theta) = \sum_i (y_i - t_i) x_i$$

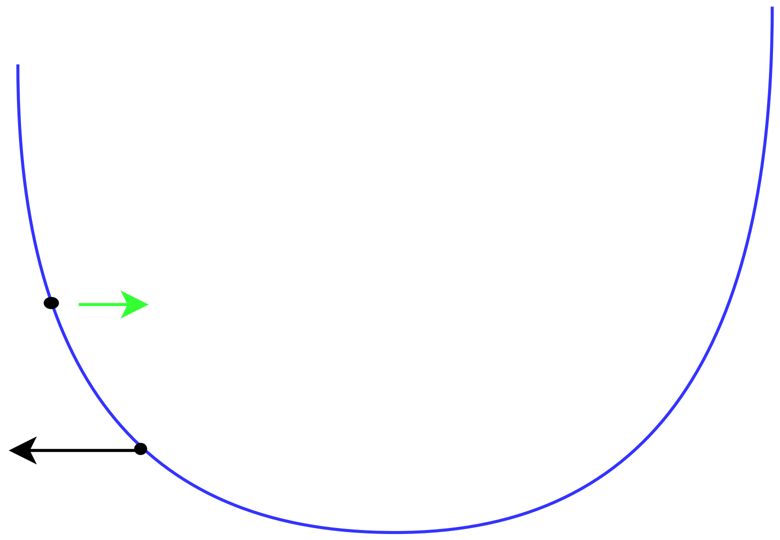
- We start at a random point and search for a minimum by walking on the error surface in the direction of steepest decent.



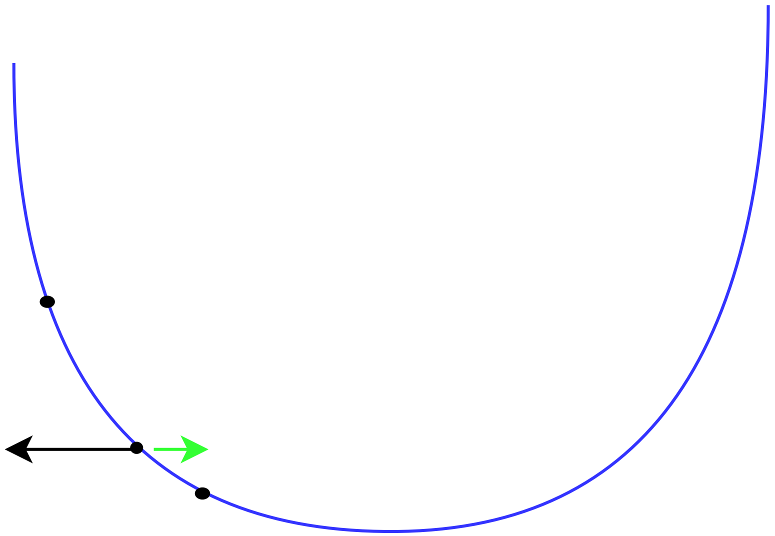
- For the error $E(\Theta)$ blue, the gradient points into the direction of steepest ascent.



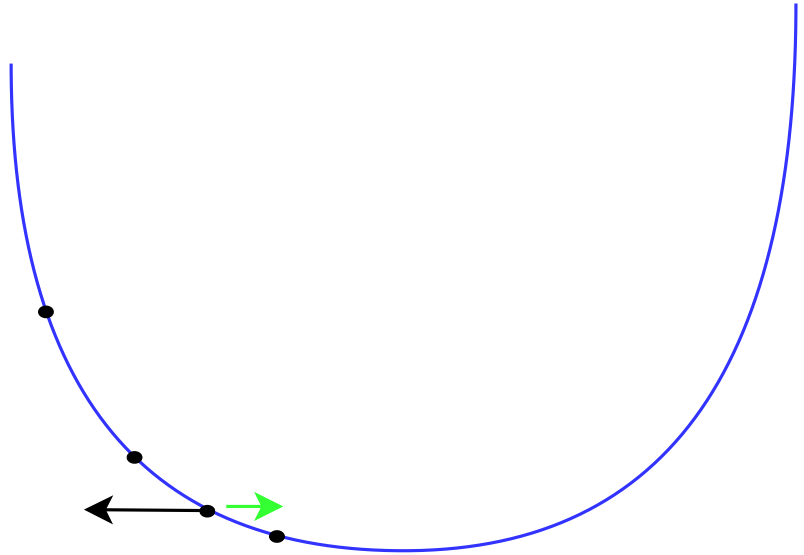
- Given a random initialization for θ we can evaluate the derivative and move into opposite direction.



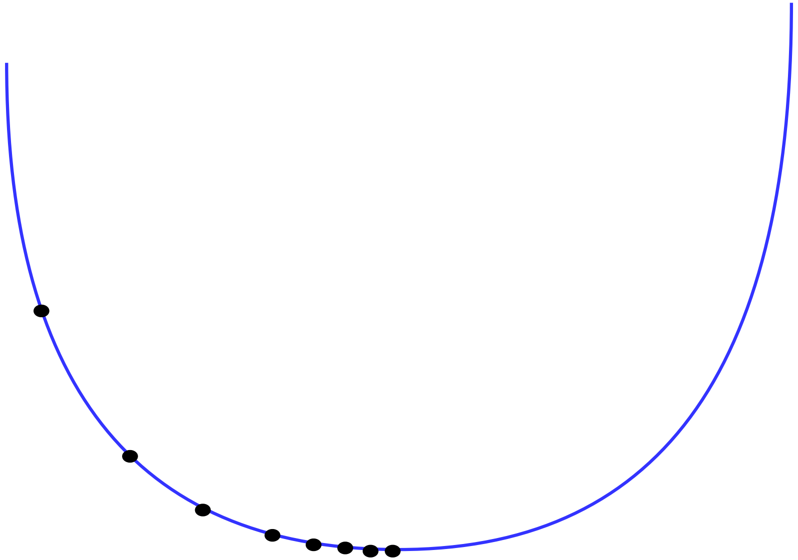
- We repeat the procedure at the new θ.



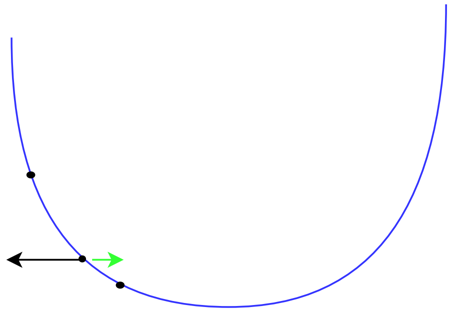
- We repeat the procedure at the new θ.



- We repeat the procedure at the new θ.



- And end up at a local minimum.



$$\theta_{i+1} = \theta_i - \eta \nabla E(\theta)$$

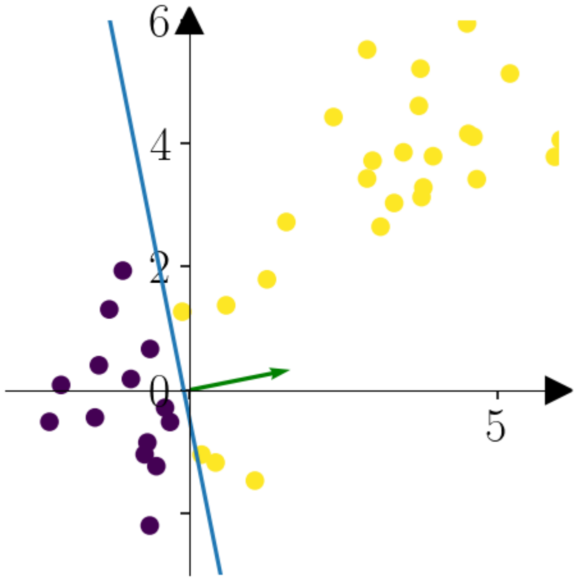
- We can write the update step formally including the learning rate (step size) η.
- Whereas ∇ is the gradient operator.

$$E(\theta) = \sum_i E_i(\theta)$$

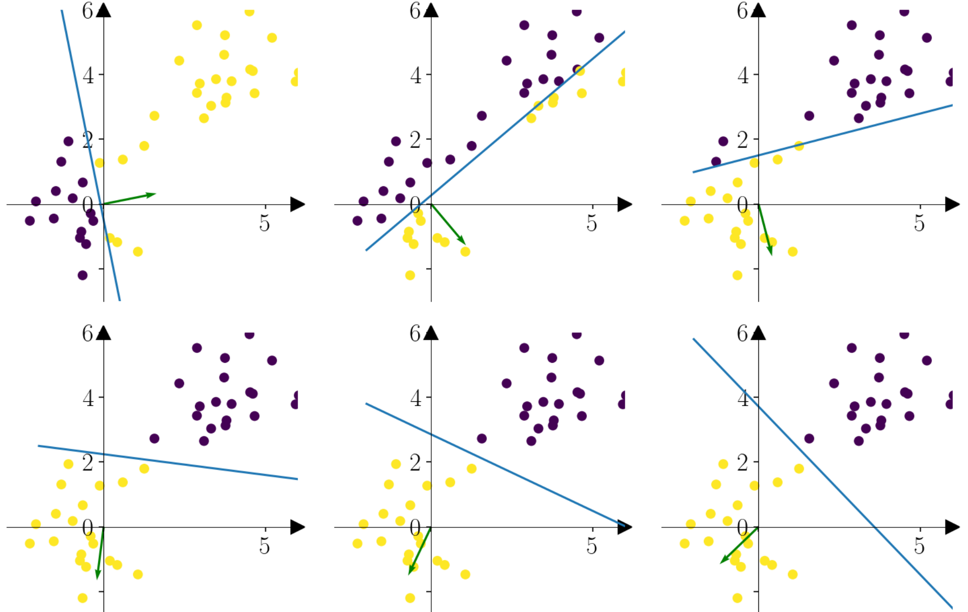
$$\theta_{i+1} = \theta_i - \eta \nabla \sum_j E_j(\theta)$$

- The error function includes a sum over all data points.
- If we use all data points for the computation of the gradient (batch methods) there would be better ways of doing that than gradient descent.
- Furthermore, the size of the data set often would make it very expensive to use all data points.
- However what we usually do when training neural networks is online learning.
- This means we use only one sample or a subset of samples j (mini-batch) at a time.

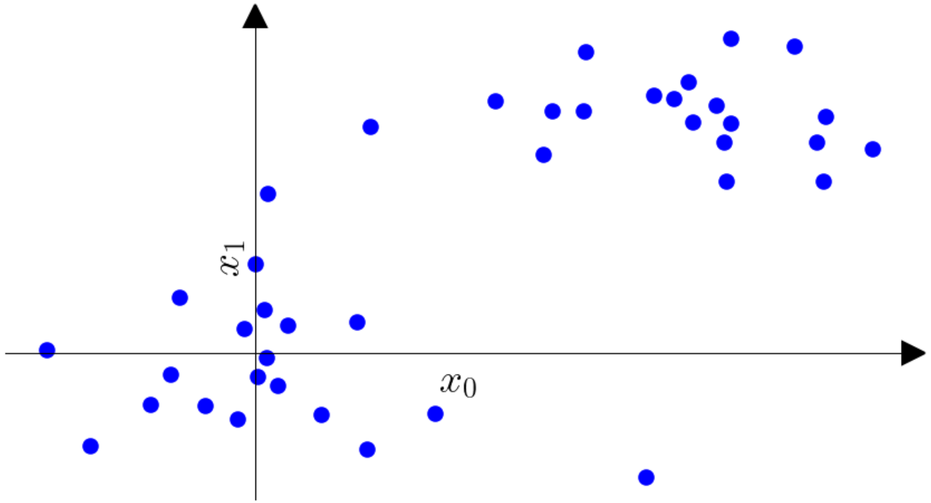
- How to choose samples?
  → Draw randomly without replacement.
- How many samples?
  → In CV often as many as possible (VRAM limiting factor)
  → Higher batch size → less gradient noise → higher learning rate η
- However, gradient noise allows to escape local optima!
  → Too big batch sizes possible.



- Random initialization for classification problem with logistic regression and gradient descent.
- Green arrow is the normalized weight vector.
- Blue line indicates decision boundary including the bias.



- The six graphs show step 0, 20, 60, 80, 100, 180 of the gradient descent.



- How to find structure in data if we don't have any labels?