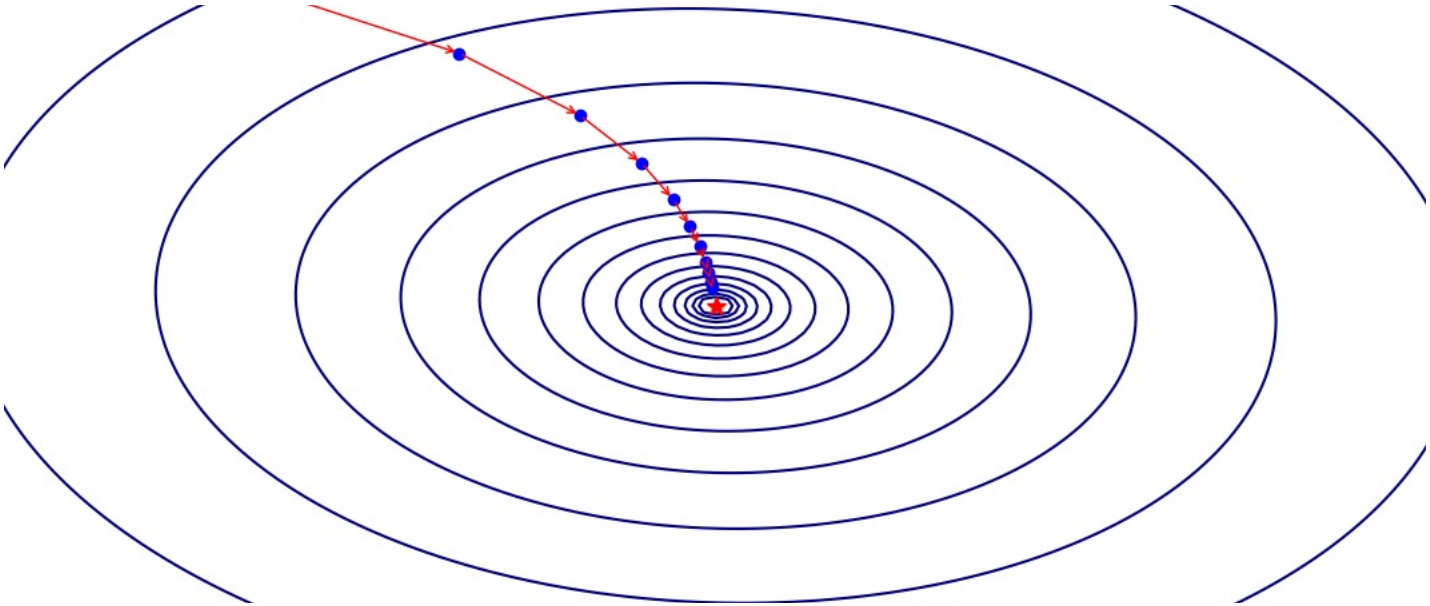


Machine Learning in Signal Processing

Winter Semester 2023/24

3. Linear Regression

30.10.2023

Prof. Dr. Vasileios Belagiannis

Chair of Multimedia Communications and Signal Processing

Friedrich-Alexander-Universität
Technische Fakultät

Course Topics

1. Introduction.
2. Basics and terminology.
3. **Linear regression.**
4. Linear classification.
5. Performance evaluation.
6. Neural networks.
7. Deep neural networks.
8. Decision trees.
9. Ensemble models.
10. Random forests.
11. Clustering / Unsupervised learning.
12. Dimensionality reduction.
13. Support vector machines.
14. Recap and Q&A.

- The exam will be written.
- We will have an exam preparation test before the end of the year.

Acknowledgements

Ideas and inspiration from:

- CSC311 Introduction to Machine Learning, University of Toronto.
- Introduction to Machine Learning: LMU Munich.
- Introduction to Machine Learning, CSAIL, MIT.
- CSE 574 Introduction to Machine Learning, University of Buffalo.
- Special thanks Arij Bouazizi, Julia Hornauer, Julian Wiederer, Adrian Holzbock and Youssef Dawoud for contributing to the lecture preparation.

Last Lecture Recap

- Machine learning definition.
- K-Nearest Neighbours algorithm.
- Learning types.
- Objective function.
- Continuous vs discrete value function minimisation.
- Gradient-based optimisation.

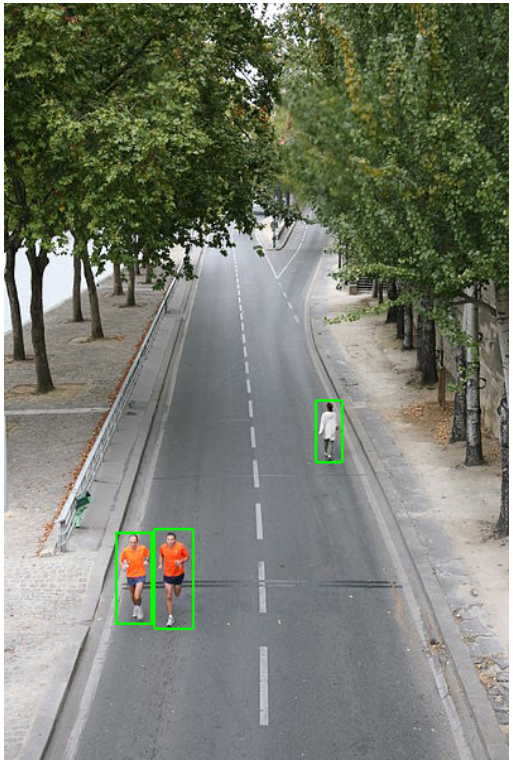
Today's Agenda and Objectives

- Linear regression
- Normal Equation Solution.
- Gradient-based Solution.
- Polynomial regression.
- Regularisation.
- Generalization and overfitting.
- Lasso regression.
- Ridge regression.

Special thanks to Mr Youssef Dawoud for contributing to the lecture preparation.

Continuous Values Prediction

- Regression in machine learning.
  - It is the task of predicting continuous scalar value(s) given a set of input value(s), which are discrete or continuous.
  - Model the relation between independent and dependent variables.
- We can still perform regression if the input or part of it is discrete. In some cases, it is convenient to work with discrete values, e.g. computer vision.
- Example I
  - Input → image and output → number of pedestrians.



By Paris,_voie_Georges_Pompidou,_"_sans_voitures_".jpg: vincent desjardinsderivative work: Indif (talk) - Paris,_voie_Georges_Pompidou,_"_sans_voitures_".jpg, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=11911812

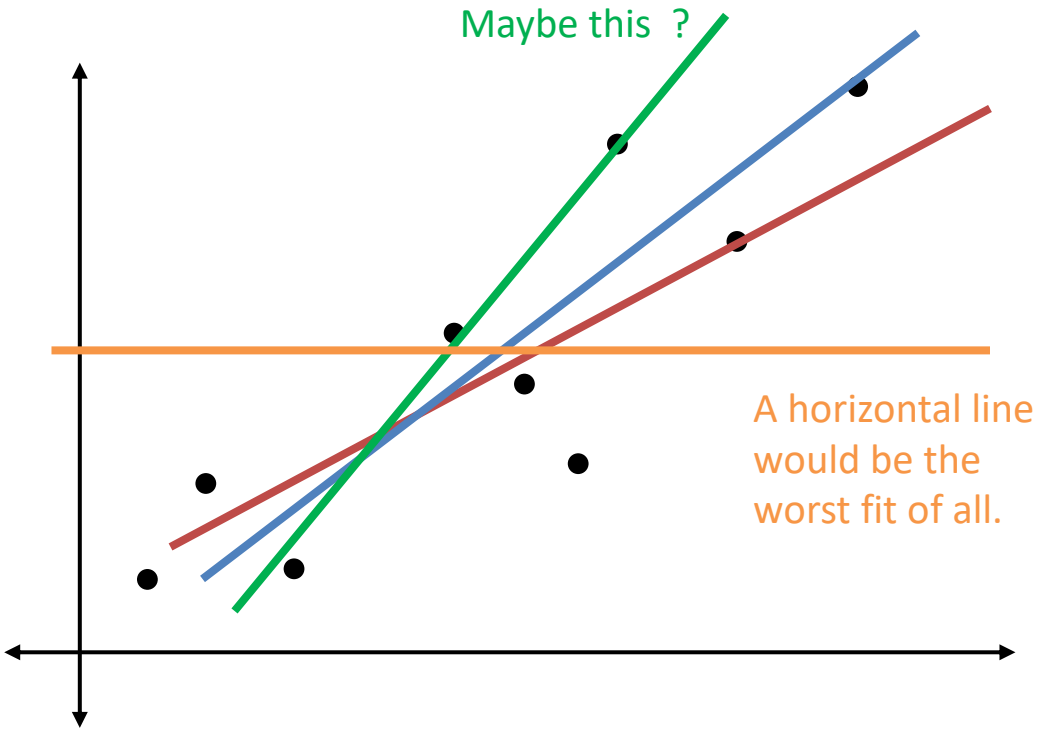
Continuous Values Prediction (Cont.)

- Example II
  - Object 2D detection from a single input RGB image.
- *Do we have a continuous or discrete output?*
- *How many output values?*
- More examples:
  - Forced exhalation volume (lung function) prediction from person's age.
  - Stock value prediction given the past values
  - Predict age of a person given a face picture.

More examples: https://online.stat.psu.edu/stat462/node/101/

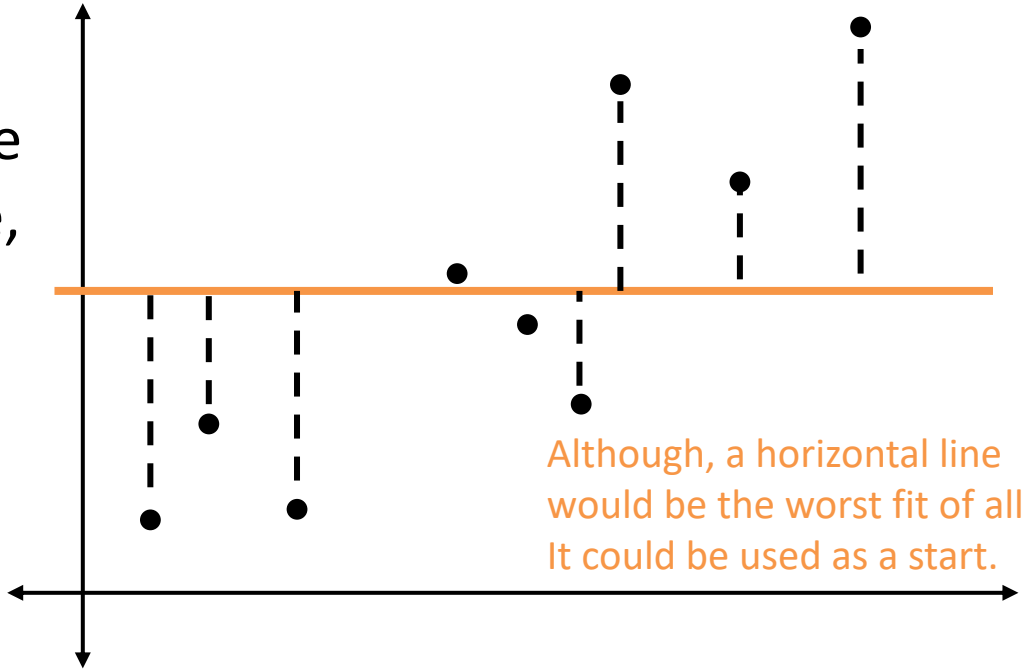
Line Fitting to the Data

- Suppose that we want to model the relationship between two variables.
- *Given a set of data, how can we describe their relation?*
- A linear function would fit the data.
- *What is the best line to fit to the data?*



Line Fitting to the Data (Cont.)

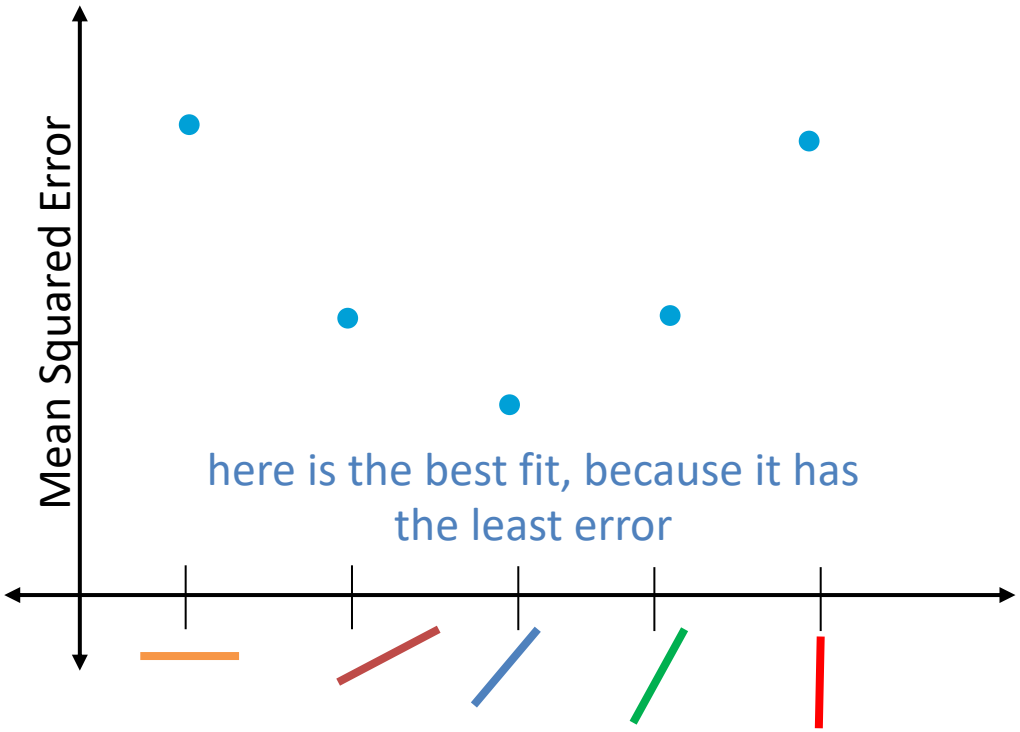
- To find the “best” (optimal) fitting line, we need a metric to tell us how good we fit the line to the data.
- *What would be a suitable metric in this case?*
- We can measure the distances (error) from the observed data to the line, sum and average over number of data points.
- We prefer squared distances to ensure the calculated distance are positive (mean squared error).



Repeat the same process for other lines and select the line of best fit ≡ least squared distance.

Line Fitting to the Data (Cont.)

- The slope of the line affects how well it fits to the data.
- We could also shift the line to obtain a lower error.
- *How many parameter does our line-model have?*
- The problem of line fitting can be formulated as linear regression.



Linear Regression

- Consider an input feature vector $\mathbf{x} \in \mathbb{R}^D$, where D is the dimension of $\mathbf{x}$.
- An output scalar value (target) $\mathrm{y} \in \mathbb{R}$.
- The linear regression model to predict the the target is defined as $\hat{\mathrm{y}} = \mathrm{f}(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T\mathbf{x} + \mathrm{b}$.
- $\hat{\mathrm{y}}$ is the predicted value, $\mathbf{w} \in \mathbb{R}^n$ is the weights vector, and b is the bias (or intercept).
- This is a linear transformation, and our goal is to predict ($\hat{\mathrm{y}}$) the exact target value ($\hat{\mathrm{y}} = \mathrm{y}$) given the input feature vector $\mathbf{x}$.
- In the context of model fitting, we seek for the parameters $\mathbf{w}$ to best fit the model to our data.

Linear Regression: Learning

- The parameter values ($\mathbf{w}$, b) can be learned with the support of a training set.
- Consider the training dataset $\mathbf{X}^{train} \in \mathbb{R}^{m \times n}$, $\mathbf{y}^{train} \in \mathbb{R}^{m}$, where m denotes the number of samples.
- Similarly, a test dataset is required to measure performance of $\mathrm{f}(\mathbf{x}; \mathbf{w})$ after learning the parameters, $\mathbf{X}^{test} \in \mathbb{R}^{m \times n}$, $\mathbf{y}^{test} \in \mathbb{R}^{m}$.
- The test set can have different dimensions from the training set.
- **Important:** Our learning algorithm is allowed to experience only the training set to learn the parameters ($\mathbf{w}$, b).

Linear Regression: Loss Function

- Learning the parameter values from the training data set requires a loss function $\mathcal{L}(f\ (x; \boldsymbol{w}), y)$.
- The loss function $\mathcal{L}(f\ (x; \boldsymbol{w}), y)$ is the performance measure of how good is our prediction during learning the parameters on the training set. Also, the same measure is applied on the test set.
- For the task of linear regression, the performance measure, namely, mean squared error (MSE) is used on both training and test sets. Our goal is to minimize the MSE as loss function on the training set such that it results in low MSE on the test set.
  - $MSE = \frac{1}{m} \sum_{i=1}^{m} \left(\widehat{\mathrm{y}}_i^{train} - \mathrm{y}_i^{train}\right)^2 = \frac{1}{m} \sum_{i=1}^{m} \left(\mathbf{w}^T \mathbf{x}_i^{train} - \mathrm{y}_i^{train}\right)^2$.
  - For simplicity we assume that b is included in $\mathbf{w}$.

Linear Regression: Loss Function (Cont.)

- Minimizing the loss function is equivalent to finding the best $\mathbf{w}^*$ such that:
  - $\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} \left(\mathbf{w}^T \mathbf{x}_i^{train} - \mathrm{y}_i^{train}\right)^2$.
- *How do we obtain the optimal parameter values $\mathbf{w}^*$?*
- Solving for $\mathbf{w}^*$:
  - Normal equation: closed-form solution to find the parameter values. The gradient is set to zero and solve for $\mathbf{w}^*$. This is like an one-step algorithm (analytic approach).
  - Iterative: find iteratively the parameter values that minimize the MSE (loss function). Gradient-based optimization is a common iterative approach.
- *What are the advantages of each approach?*

Linear Regression: Normal Equation

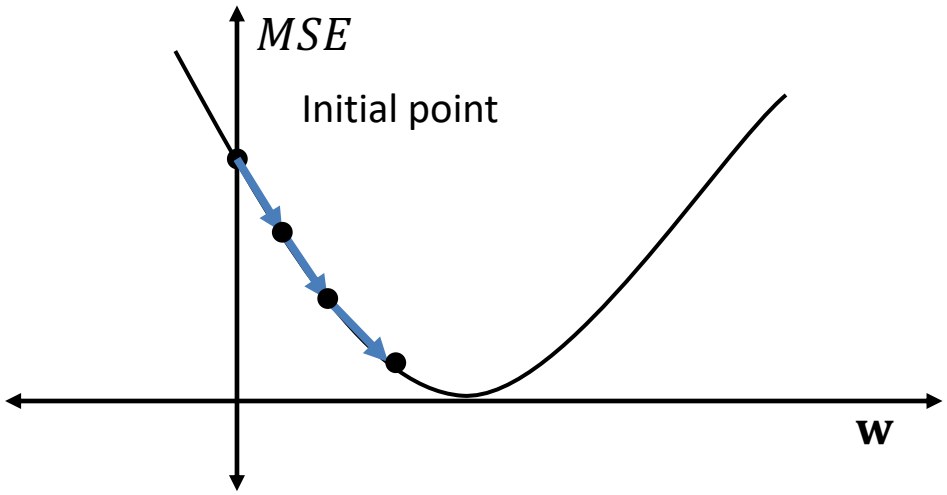
- Loss function:
  - $MSE = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}_i^{train} - y_i^{train}\right)^2 = \frac{1}{m}\sum_{i=1}^{m}\left(\mathbf{w}^T\mathbf{x}_i^{train} - y_i^{train}\right)^2$.
- We reformulate our loss function using matrix notation:
  - $MSE = \left(\mathbf{X}^{(train)}\mathbf{w} - \mathbf{y}^{train}\right)^T\left(\mathbf{X}^{(train)}\mathbf{w} - \mathbf{y}^{(train)}\right)$.
  - We skip the term $\frac{1}{m}$ because it has no impact to the optimal parameter values.
- Now we expand the above equation to obtain:
  - $MSE = (\mathbf{w}^T\mathbf{X}^{(train)T} - \mathbf{y}^{(train)T})\left(\mathbf{X}^{(train)}\mathbf{w} - \mathbf{y}^{(train)}\right) = \mathbf{w}^T\mathbf{X}^{(train)T}\mathbf{X}^{(train)}\mathbf{w} - \mathbf{w}^T\mathbf{X}^{(train)T}\mathbf{y}^{(train)} - \mathbf{y}^{(train)T}\mathbf{X}^{(train)}\mathbf{w} + \mathbf{y}^{(train)T}\mathbf{y}^{(train)} = \mathbf{w}^T\mathbf{X}^{(train)T}\mathbf{X}^{(train)}\mathbf{w} - 2\mathbf{w}^T\mathbf{X}^{(train)T}\mathbf{y}^{(train)} + \mathbf{y}^{(train)T}\mathbf{y}^{(train)}$.
  - Note that $\mathbf{y}^{(train)T}\mathbf{X}^{(train)}\mathbf{w} = \left(\mathbf{y}^{(train)T}\mathbf{X}^{(train)}\mathbf{w}\right)^T = \mathbf{w}^T\mathbf{X}^{(train)T}\mathbf{y}^{(train)}$.

Linear Regression: Normal Equation (Cont.)

- Next, we can obtain $\boldsymbol{w}^*$ by the setting the gradient of the MSE w.r.t $\mathbf{w}$ to zero and solve for it.
  - $\nabla_{\mathbf{w}} MSE = 0 \Rightarrow$
  - $\nabla_{\mathbf{w}} \left(\mathbf{w}^T \mathbf{X}^{(train)T} \mathbf{X}^{(train)} \mathbf{w} - 2\mathbf{w}^T \mathbf{X}^{(train)T} \mathbf{y}^{(train)} + \mathbf{y}^{(train)T} \mathbf{y}^{(train)} \right) = 0 \Rightarrow$
  - $2\mathbf{X}^{(train)T} \mathbf{X}^{(train)} \mathbf{w} - 2\mathbf{X}^{(train)T} \mathbf{y}^{(train)} = 0 \Rightarrow$
  - $\mathbf{X}^{(train)T} \mathbf{X}^{(train)} \mathbf{w} = \mathbf{X}^{(train)T} \mathbf{y}^{(train)} \Rightarrow$
  - $\mathbf{w} = \left(\mathbf{X}^{(train)T} \mathbf{X}^{(train)} \right)^{-1} \mathbf{X}^{(train)T} \mathbf{y}^{(train)}$.
- Our solution is $\mathbf{w} = \left(\mathbf{X}^{(train)T} \mathbf{X}^{(train)} \right)^{-1} \mathbf{X}^{(train)T} \mathbf{y}^{(train)}$.
- Using the above equation we can obtain the parameter value within <u>one step</u>.

Linear Regression: Gradient Descent

- We can solve for $\mathbf{w}^*$ iteratively using gradient descent.
- We need an initial value for the parameters and then iteratively update their values with a gradient descent variant.
- We consider the batch gradient for this example, but any other variant should work too.

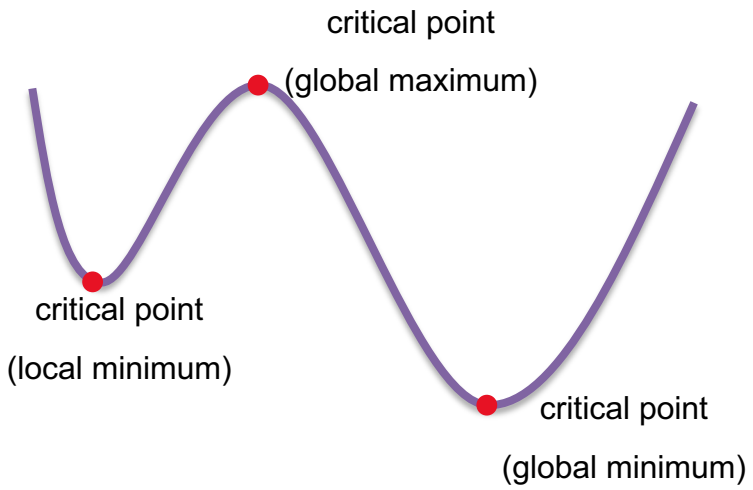


Linear Regression: Gradient Descent (Cont.)

- The gradient-descent update is given by:
  - $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} MSE = \mathbf{w} - \frac{\eta}{m} \sum_{i=1}^{m} \left(\hat{y}_i^{train} - y_i^{train} \right) \mathbf{x}_i^{train}$.
  - where $\eta > 0$ is the learning rate. The larger it is, the faster $\mathbf{w}$ value change.
- η is a hyper-parameter that we tune to reach a good performance. It usually takes a small value, e.g., 0.1 or 0.01.
- Ideally convergence would be reached when we reach a critical point $\nabla_{\mathbf{w}} MSE = 0$.
- Note that we will mostly work with problems where the reached minima will be only local because of the non-convexity of the problem.
- In machine learning, we rarely deal with convex problems with global minimum (or maximum).

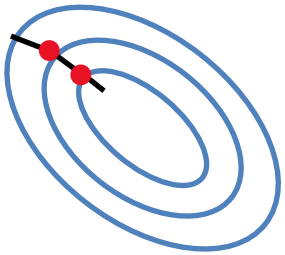
Linear Regression: Gradient Descent (Cont.)

- The final critical points, i.e., where the derivative is zero (or non-existent), will not necessarily provide the global maximum / minimum.
- For a convex function like the MSE, it is guaranteed to converge to a global optimum (recall its bell-shape on shows only one global minimum).



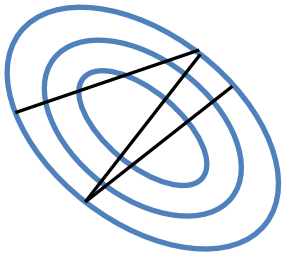
Linear Regression: Gradient Descent Hyper-parameters

- The speed of the local/global minimum is affected from the learning rate.



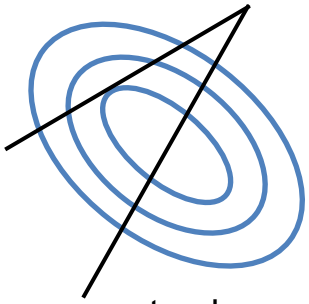
η too small

slow learning progress



η large

oscillations



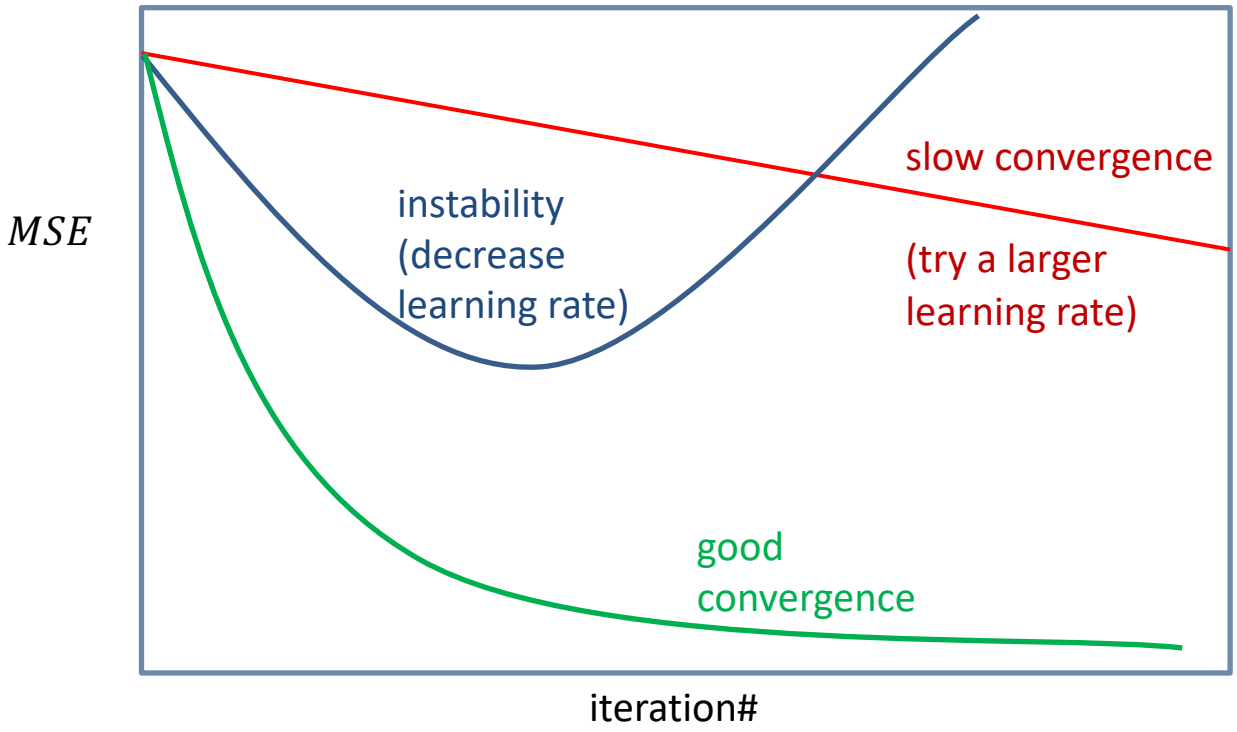
η too large

instability

- The above circles depict the landscape of the loss function from a top view.
- Ideally, we would like to have low oscillations with progressive convergence.
- We can use grid search to choose a good learning rate i.e. try a set of values e.g. {0.1, 0.01, 0.001}

Linear Regression: Gradient Descent Congergence

- The training curve could be used to monitor the effect of the learning rate value and observe the convergence. We plot the training loss against the number of training iterations.

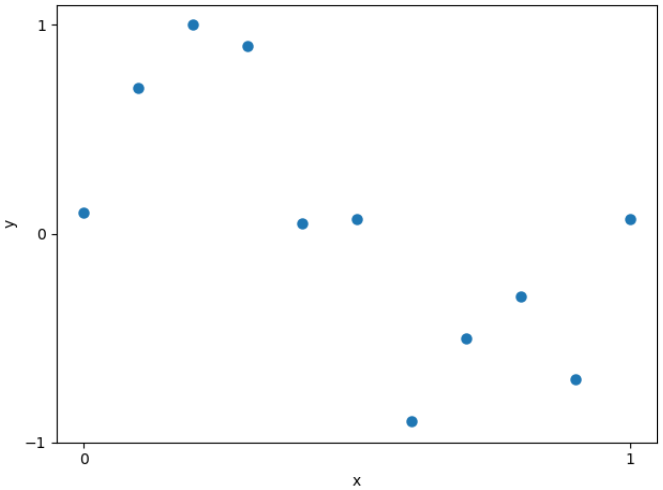


Linear Regression: Gradient Descent Observations

- This approach is meaningful only when there is not an analytical solution, or the analytical solution is too expensive for our computational budget.
- In a higher dimensional space:
  - A direct solution may not be feasible because $\mathbf{X}^{(train)}$ maybe non-invertible (singular) matrix.
  - Gradient decent is more efficient sometimes as finding the inverse matrix maybe computationally expensive $O(D^3)$ while vanilla gradient descent costs $O(mD)$.
- Termination criteria:
  - Reaching maximum number of update iterations.
  - Achieving a certain loss e.g., $MSE < 10^{-3}$.
- Gradient descent applies with the same ideas on any problem given a differentiable loss function. The loss function and hyper-parameters mainly differ between the problems.

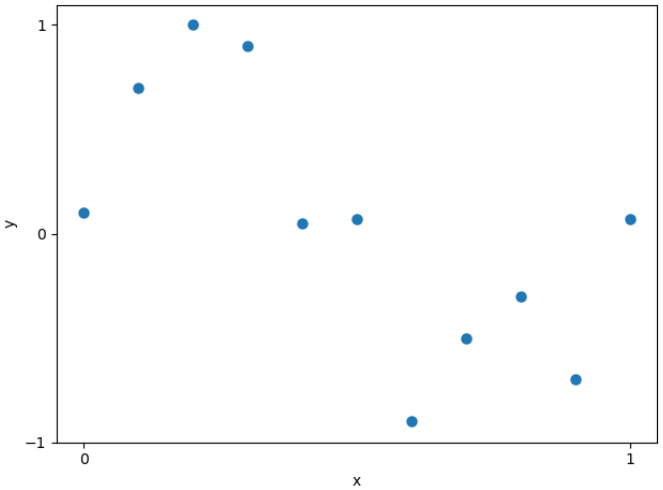
Polynomial Regression

- The relationship between $\mathbf{x}$ and $\mathbf{y}$ may be non-linear too.
- A polynomial function $\mathrm{f}(.)$ of degree M can be more suitable for the data.
- The input features are mapped to a polynomial space $\psi(\mathbf{x})$: $\mathbb{R}^D \rightarrow \mathbb{R}^d$.
- The mapped features are then treated as input to linear regression.
- The degree M is a hyper-parameter.
  - A higher M has the capacity to fit higher non-linear relations.
- Linear regression is still applicable since y is linear w.r.t. $\mathbf{w}$.
  - Our unknown is $\mathbf{w}$ while $\mathbf{x}$, which is the input is known (it is the observed features).



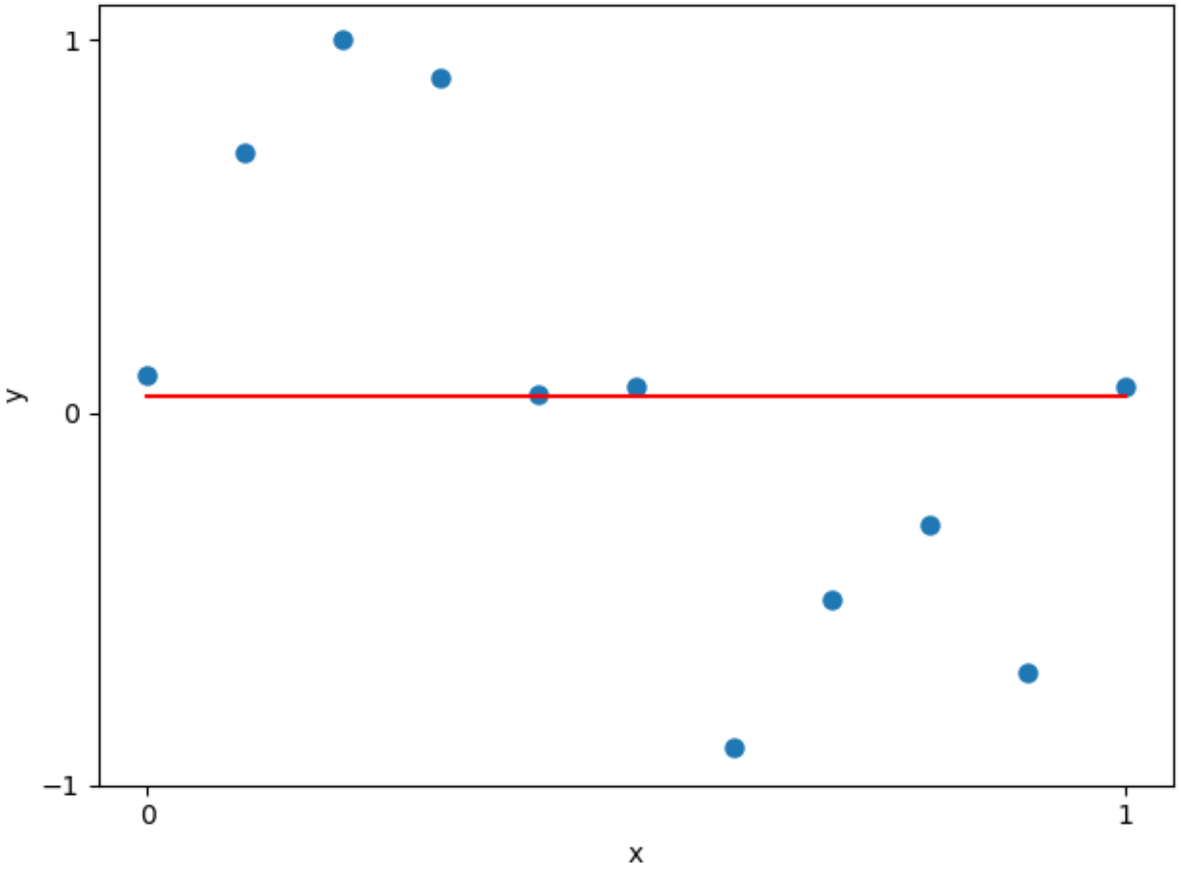
Polynomial Regression (Cont.)

- The regression model is defined as:
  - $\hat{y} = f(\mathbf{x}; \mathbf{w}) = b + w_1 x + w_2 x^2 + \cdots + w_M x^M$.
- The problem is still linear since we are interested in optimizing the parameters of the model.
- We can re-write the model in a vectorized form as with the linear regression example.
- The loss function is the same:
  - $\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i^{train})^2$.



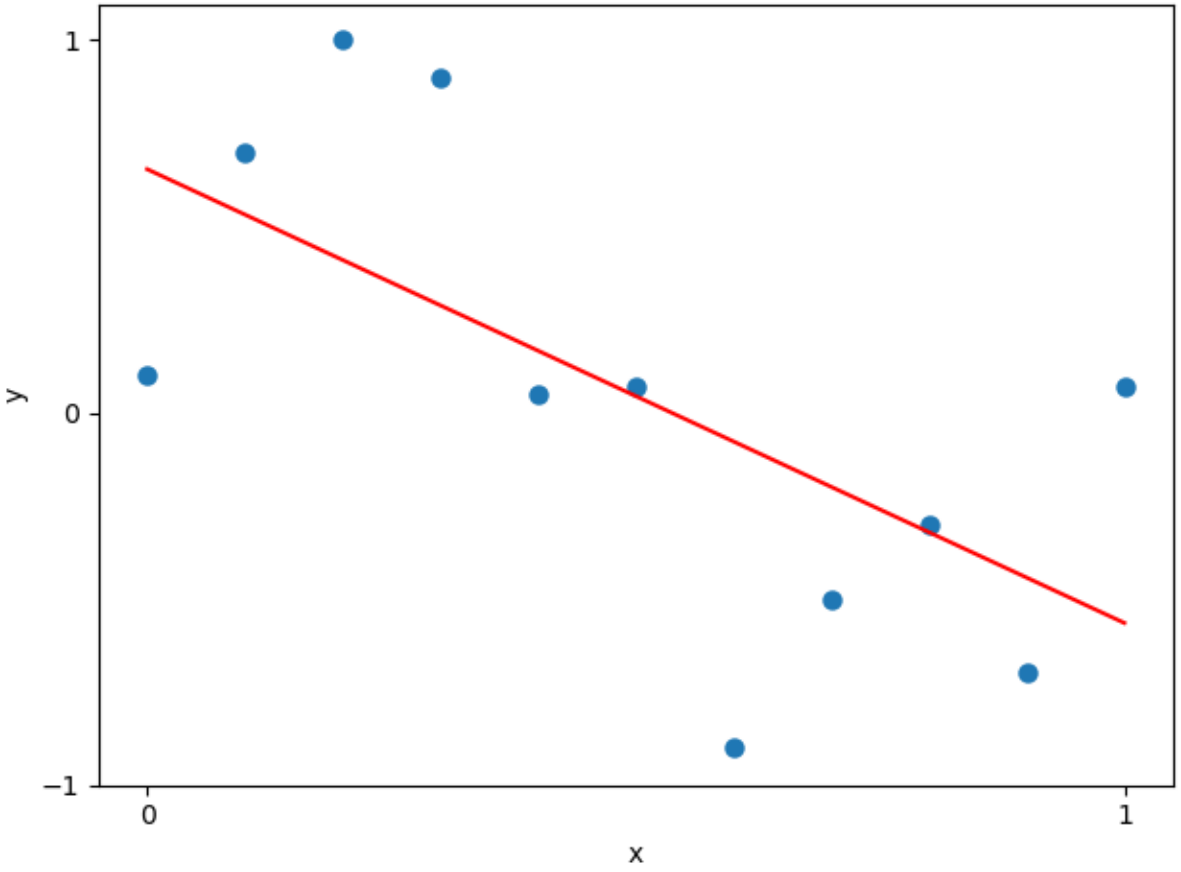
Polynomial Feature Mapping with $M = 0$

- Linear relationship.



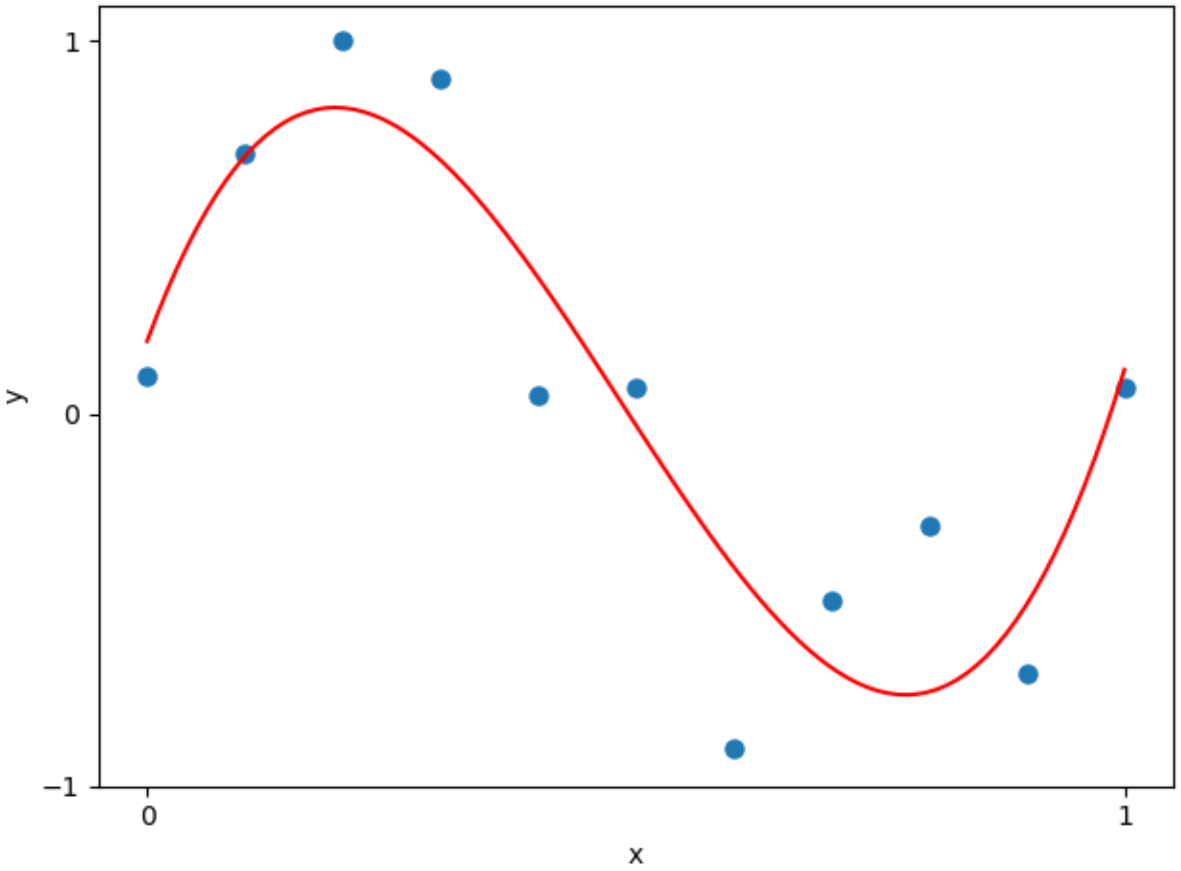
Polynomial Feature Mapping with $M = 1$

- Linear relationship.



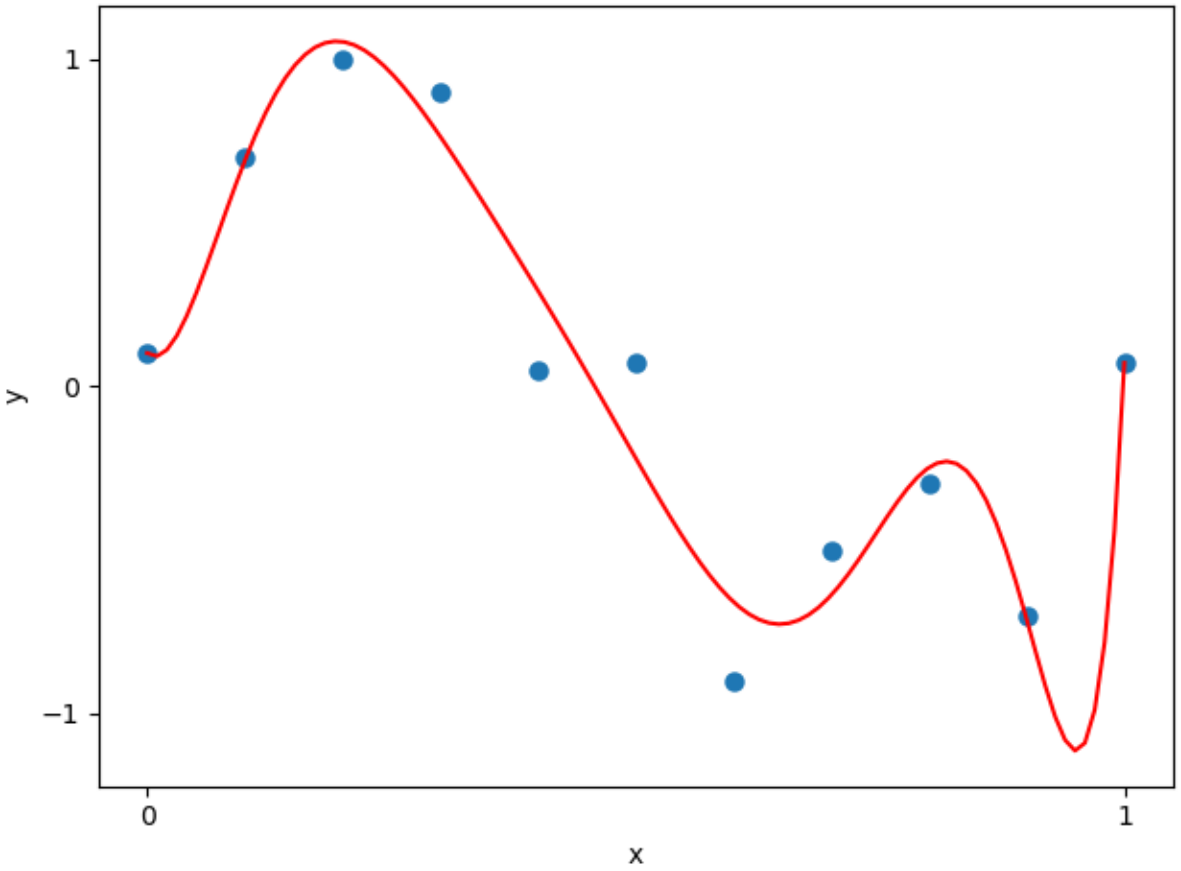
Polynomial Feature Mapping with $M = 3$

- Non-linear modelling of the features.



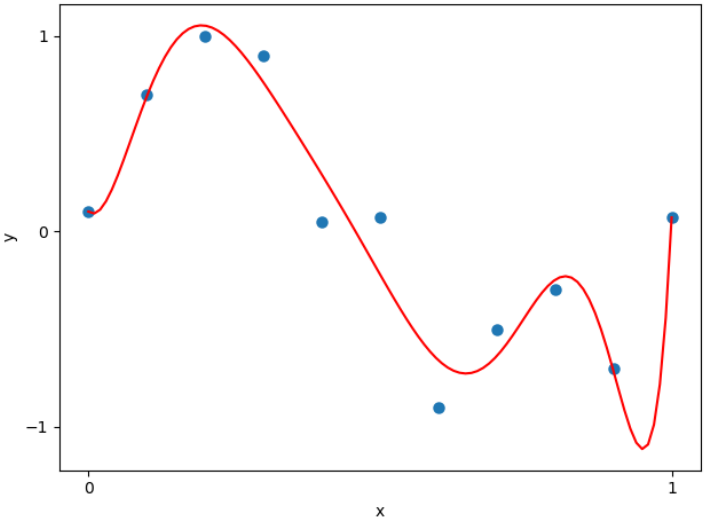
Polynomial Feature Mapping with $M = 9$

- Which M leads to the performing model?



Polynomial Regression: Generalisation

- The model with $M = 9$ approximates the training data samples very well. So good that it actually can fail to generalize to other samples (test set).
- Generalization is the ability to perforr well on unobserved samples (more in following lecture).
- The problem of fitting too well the training data and failing to generalise on unobserved samples is called over-fitting.
- Regularisation helps to avoid over-fitting.



Regularization

- Regularization imposes a preference to the solution.
- The regularization is an additional minimization term next to the loss function, as:
  - $\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\mathrm{f}\left(\mathrm{x}^{\mathrm{i}}; \mathbf{w}\right), \mathrm{y}^{\mathrm{i}}\right) + \lambda \mathrm{R}(\mathbf{w}).$
- $\mathrm{R}(\mathbf{w})$ is the regularization term and λ is a hyper-parameter for tuning the contribution of the regularizer.
- There are two main types of regularization:
  - L1-regularisation.
  - L2-regularization.
- Some know but less frequency approaches:
  - Early stopping.

Regularization: L1

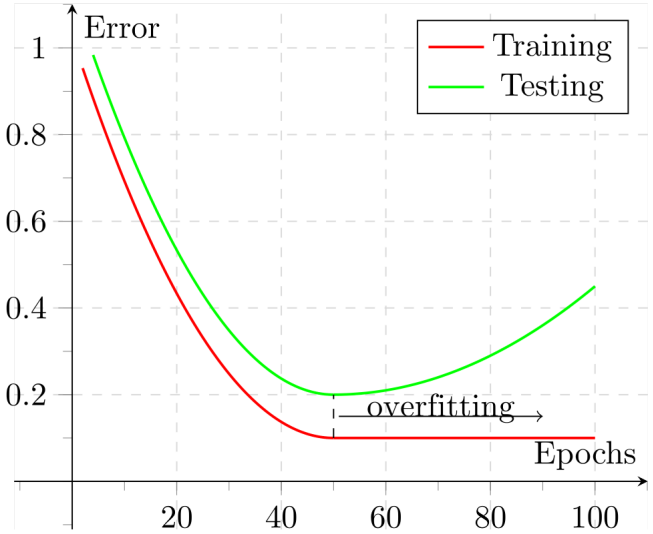
- We write our objective as:
  - $\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\mathrm{f}\left(\mathrm{x}^{\mathrm{i}}; \mathbf{w}\right), \mathrm{y}^{\mathrm{i}}\right) + \lambda \|\mathbf{w}\|.$
- During the gradient update in gradient descent, we have:
  - $\mathbf{w} = \mathbf{w} - \eta \nabla_{\mathrm{w}} \left(\frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\mathrm{f}\left(\mathrm{x}^{\mathrm{i}}; \mathbf{w}\right), \mathrm{y}^{\mathrm{i}}\right) + \lambda \|\mathbf{w}\| \right) \Rightarrow$
  - $\mathbf{w} = \mathbf{w} - \eta \nabla_{\mathrm{w}} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\mathrm{f}\left(\mathrm{x}^{\mathrm{i}}; \mathbf{w}\right), \mathrm{y}^{\mathrm{i}}\right) + \eta\lambda \frac{\mathbf{w}}{|\mathbf{w}|} \Rightarrow$
  - $\mathbf{w} = \left(1 - \frac{\eta\lambda}{|\mathbf{w}|}\right) \mathbf{w} - \eta \nabla_{\mathrm{w}} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\mathrm{f}\left(\mathrm{x}^{\mathrm{i}}; \mathbf{w}\right), \mathrm{y}^{\mathrm{i}}\right).$
- The regularization forces the weights to become 0 i.e sparse weights.

Regularization: L2

- We write our objective as:
  - $\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\mathrm{f}\left(\mathrm{x}^{\mathrm{i}}; \mathbf{w}\right), \mathrm{y}^{\mathrm{i}}\right) + \frac{\lambda}{2} \|\mathbf{w}\|^2.$
- During the gradient update in gradient descent, we have:
  - $\mathbf{w} = \mathbf{w} - \eta \nabla_{\mathrm{w}} \left(\frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\mathrm{f}\left(\mathrm{x}^{\mathrm{i}}; \mathbf{w}\right), \mathrm{y}^{\mathrm{i}}\right) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \right) \Rightarrow$
  - $\mathbf{w} = \mathbf{w} - \eta \nabla_{\mathrm{w}} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\mathrm{f}\left(\mathrm{x}^{\mathrm{i}}; \mathbf{w}\right), \mathrm{y}^{\mathrm{i}}\right) - \eta\lambda\mathbf{w} \Rightarrow$
  - $\mathbf{w} = (1 - \eta\lambda)\mathbf{w} - \eta \nabla_{\mathrm{w}} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\mathrm{f}\left(\mathrm{x}^{\mathrm{i}}; \mathbf{w}\right), \mathrm{y}^{\mathrm{i}}\right)$
  - where $(1 - \eta\lambda)$ is the weight decay. λ is usually 0.1 or 0.01.
- Both L1 and L2 are common regularizations for different machine learning algorithms.

Early stopping Regularization

- Early Stopping is another type of regularization which does not have an explicit form.
- It refers to terminating the training before over-fitting starts.
- For that purpose, we can extract a subset of the training set and use it to observe the training performance, this set is referred to as validation set.
- We stop when the validation error starts increasing.



Lasso Regression

- It is a variation of linear regression with L1-regularization.
- Loss function:
  - $\frac{1}{m}\sum_{i=1}^{m}\left(\mathbf{w}^T\mathbf{x}_i^{train} - y_i^{train}\right)^2 + \lambda\sum_{j=1}^{p}|w_j|$.
  - $\frac{1}{m}$ can be skipped.
- The tuning parameter λ controls the influence of the regularizer.
- The L1-reugalarisation is good at brining some of the input features to 0 (feature elimination).
- Thus, the model can have sparse parameters.

Ridge Regression

- It is a variation of linear regression with L2-regularization.
- Loss function:
  - $\frac{1}{m}\sum_{i=1}^{m}\left(\mathbf{w}^T\mathbf{x}_i^{train} - y_i^{train}\right)^2 + \lambda\sum_{j=1}^{p}|w_j|^2$.
  - $\frac{1}{m}$ can be skipped.
- The tuning parameter λ controls the influence of the regularizer.
- It is used when there is data collinearity (occurrence of intercorrelation among two or more input features).
- It leads to faster convergence compared to Lasso regression.

Study Material

Not for sharing (LMS, Friedrich-Alexander-Universität Erlangen-Nürnberg)

- *The Elements of Statistical Learning, Trevor Hastie et. al, Chapter 3.*
- *Understanding Machine Learning, Shai Shalev-Schwarz, Chapter 9, Section 9.2.*
- *Pattern Recognition and Machine Learning, Christopher Bishop, Chapter 3.*

Next Lecture

Linear Classification