Linear Regression

Machine Learning – Foundations and Algorithms

Prof. Gerhard Neumann
Autonomous Learning Robots
KIT, Institut für Anthropomatik und Robotik

Announcements

- The Exam will take place on 23.02.2022 at 18:00. The place will be the Audimax, and potentially further rooms "next to it" if needed.
- According to the survey, we fixed the date for the exercise/recap sessions to Thursday, 17:00 virtually. - First session will be next week and it will be "Exercise 0: Organization and Intro to scientific python"
- We'll close the exercise group assignment next Tuesday evening and might re-assigin people who are in groups of < 3, to form groups of 3.

Learning Outcomes

- Get familiar with matrix computations and matrix calculus
- Understand the regression problem
- What do we mean with “linear representation”?
- Be able to derive the least squares solution
- What is the use of regularization in ridge regression?
- How to extend linear regression to non-linear function?

Today‘s Agenda!

Recap: Types of Machine Learning

Recap: Linear Algebra

- Vectors, Matrices and manipulation of those

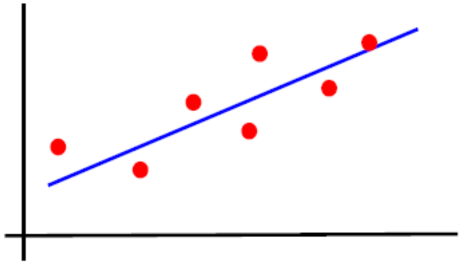
Linear Regression:

- Least-Squares Solution
- Generalized Linear Regression Models
- Ridge Regression

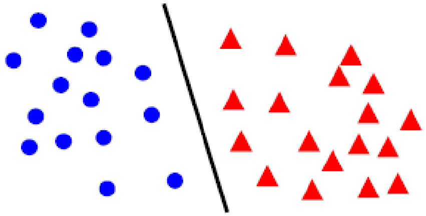
Supervised Learning

Training data includes targets

- **Regression:**
  - Learn continuous function
  - Example: line



- **Classification:**
  - Learn class labels
  - Example: Digit recognition

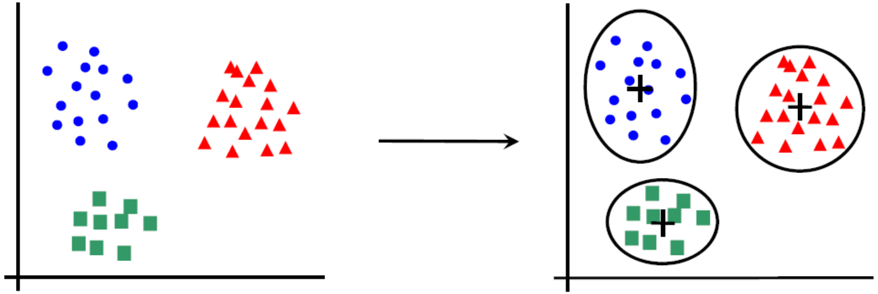


Unsupervised Learning

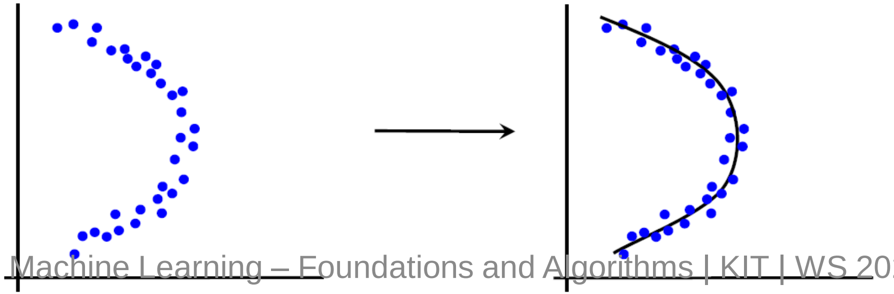
Trainings data does not include target values

- Model the data

- **Clustering:**



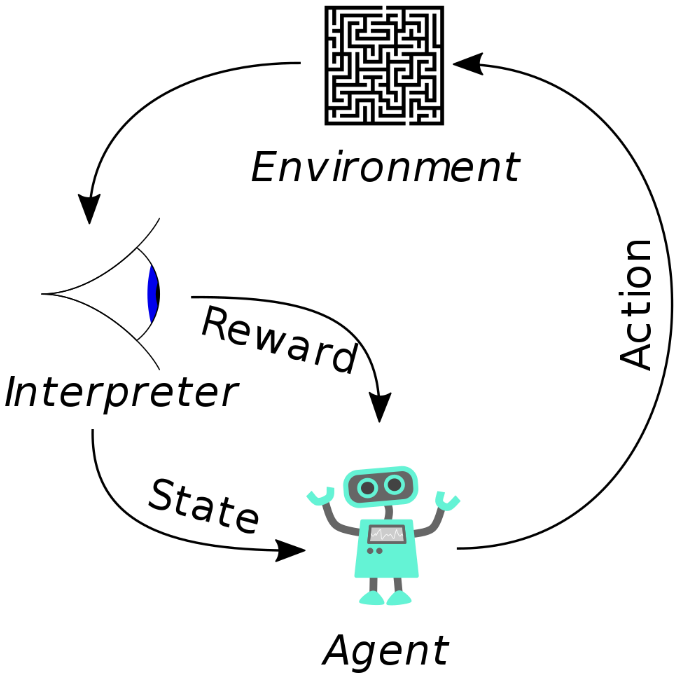
- **Dimensionality reduction:**



Reinforcement Learning

- No supervisor, but reward signal
- Selected actions also influence future states

Not part of this lecture!



Today‘s Agenda!

Recap: Types of Machine Learning

Recap: Linear Algebra

- Vectors, Matrices and manipulation of those

Linear Regression:

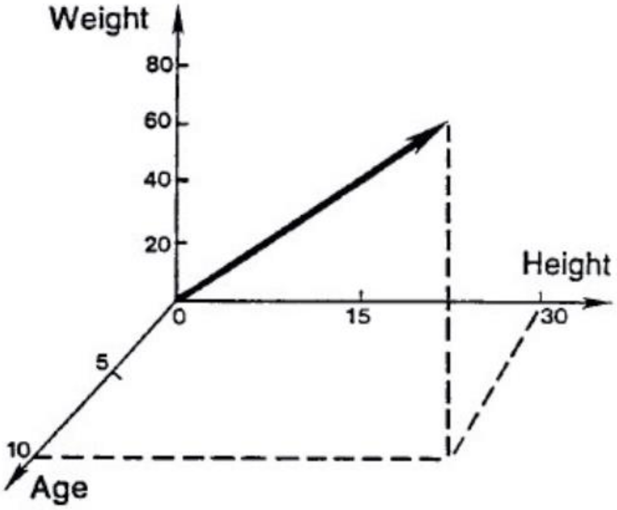
- Least-Squares Solution
- Generalized Linear Regression Models
- Ridge Regression

Vectors

- A vector is a multi-dimensional quantity

Joe $\begin{bmatrix} 37 \\ 72 \\ 175 \end{bmatrix}$ Mary $\begin{bmatrix} 10 \\ 30 \\ 61 \end{bmatrix}$

Carol $\begin{bmatrix} 25 \\ 65 \\ 121 \end{bmatrix}$ Brad $\begin{bmatrix} 66 \\ 67 \\ 155 \end{bmatrix}$



- Each dimension contains different information (Age, Height, Weight…)

Some notation

- Vectors will always be represented as bold symbols

$$\underbrace{x = 1}_{\text{scalar}}, \quad \underbrace{\boldsymbol{x} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}}_{\text{vector}}$$

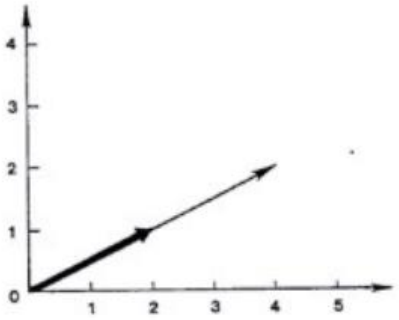
- A vector $\boldsymbol{x}$ is always a **column vector** $\boldsymbol{x} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$

- A transposed vector $\boldsymbol{x}^T$ is always a **row vector** $\boldsymbol{x}^T = \begin{bmatrix} 1 & 2 & 4 \end{bmatrix}$

What can we do with vectors?

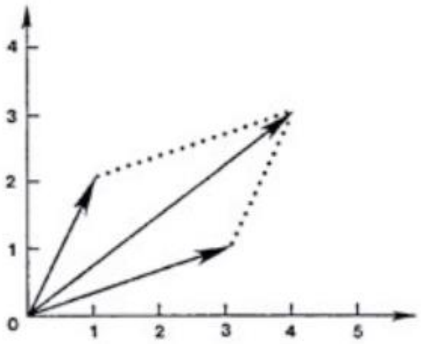
- Multiplication by scalars

$$2\begin{bmatrix}1\\2\\4\end{bmatrix}=\begin{bmatrix}2\\4\\8\end{bmatrix}$$



- Addition of vectors

$$\begin{bmatrix}1\\2\\4\end{bmatrix}+\begin{bmatrix}2\\1\\4\end{bmatrix}=\begin{bmatrix}3\\3\\8\end{bmatrix}$$



Scalar products and length of vectors

- Scalar (Inner) products:
  - Sum the element-wise products

$$\boldsymbol{v} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}, \quad \boldsymbol{w} = \begin{bmatrix} 2 \\ 4 \\ 8 \end{bmatrix}$$

$$\langle \boldsymbol{v}, \boldsymbol{w} \rangle = 1 \cdot 2 + 2 \cdot 4 + 4 \cdot 8 = 42$$

- Length of a vector
  - Square root of the inner product with itself

$$||\boldsymbol{v}|| = \langle \boldsymbol{v}, \boldsymbol{v} \rangle^{\frac{1}{2}} = \left(1^2 + 2^2 + 4^2\right)^{\frac{1}{2}} = \sqrt{21}$$

Matrices

- A matrix is a rectangular array of numbers arranged in rows and columns.

$$\boldsymbol{X} = \begin{bmatrix} 1 & 3 \\ 2 & 3 \\ 4 & 7 \end{bmatrix} \qquad \boldsymbol{A} = \begin{bmatrix} 1 & 3 & 5 & 4 \\ 2 & 3 & 7 & 2 \end{bmatrix}$$

  - $\boldsymbol{X}$ is a 3 x 2 matrix and $\boldsymbol{A}$ a 2 x 4 matrix
  - Dimension of a matrix is always *num rows times num columns*
  - Matrices will be denoted with bold upper-case letters (***A*****,*****B*****,*****W***)
  - Vectors are **special cases of matrices**

$$\underbrace{\boldsymbol{x} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}}_{3\times 1 \text{ matrix}} \qquad \underbrace{\boldsymbol{x}^T = \begin{bmatrix} 1 & 2 & 4 \end{bmatrix}}_{1\times 3 \text{ matrix}}$$

Matrices in Machine Learning

- In many cases, our data set can be represented as matrix, where single samples are vectors

$$\text{Joe: } \boldsymbol{x}_1 = \begin{bmatrix} 37 \\ 72 \\ 175 \end{bmatrix} \quad \text{Mary: } \boldsymbol{x}_2 = \begin{bmatrix} 10 \\ 30 \\ 61 \end{bmatrix} \quad \text{Carol: } \boldsymbol{x}_3 = \begin{bmatrix} 25 \\ 65 \\ 121 \end{bmatrix} \quad \text{Brad: } \boldsymbol{x}_4 = \begin{bmatrix} 66 \\ 67 \\ 175 \end{bmatrix}$$

- **Most typical representation:**
  - Each row represent a data sample (e.g. Joe)
  - Each column represents a data entry (e.g. age)

$\Rightarrow$ ***X*** is a *num samples x num entries* matrix

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \boldsymbol{x}_3^T \\ \boldsymbol{x}_4^T \end{bmatrix} = \begin{bmatrix} 37 & 72 & 175 \\ 10 & 30 & 61 \\ 25 & 65 & 121 \\ 66 & 67 & 175 \end{bmatrix}$$

What can you do with matrices?

- Multiplication with scalar

$$3\boldsymbol{M} = 3\begin{bmatrix} 3 & 4 & 5 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 9 & 12 & 15 \\ 3 & 0 & 3 \end{bmatrix}$$

- Addition of matrices

$$\boldsymbol{M} + \boldsymbol{N} = \begin{bmatrix} 3 & 4 & 5 \\ 1 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 2 & 1 \\ 3 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 6 & 6 \\ 4 & 1 & 2 \end{bmatrix}$$

- Matrices can also be transposed

$$\boldsymbol{M} = \begin{bmatrix} 3 & 4 & 5 \\ 1 & 0 & 1 \end{bmatrix}, \; \boldsymbol{M}^T = \begin{bmatrix} 3 & 1 \\ 4 & 0 \\ 5 & 1 \end{bmatrix}$$

Multiplication of a vector with a matrix

- Matrix-Vector Product: $\boldsymbol{u} = \boldsymbol{W}\boldsymbol{v} = \begin{bmatrix} 3 & 4 & 5 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \cdot 1 + 4 \cdot 0 + 5 \cdot 2 \\ 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 2 \end{bmatrix} = \begin{bmatrix} 13 \\ 3 \end{bmatrix}$
- Think of it as: $\underbrace{\begin{bmatrix} \boldsymbol{w}_1, \dots, \boldsymbol{w}_n \end{bmatrix}}_{\boldsymbol{W}} \underbrace{\begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}}_{\boldsymbol{v}} = \underbrace{\begin{bmatrix} v_1 \boldsymbol{w}_1 + \dots + v_n \boldsymbol{w}_n \end{bmatrix}}_{\boldsymbol{u}}$
  - Hence: $\boldsymbol{u} = v_1 \boldsymbol{w}_1 + \dots + v_n \boldsymbol{w}_n = 1 \begin{bmatrix} 3 \\ 1 \end{bmatrix} + 0 \begin{bmatrix} 4 \\ 0 \end{bmatrix} + 2 \begin{bmatrix} 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 13 \\ 3 \end{bmatrix}$
  - We sum over the columns $\boldsymbol{w}_i$ of ***W*** weighted by v_i
- Vector needs to have same **dimensionality as number of columns!**

Multiplication of a matrix with a matrix

- Matrix-Matrix Product:

$$\boldsymbol{U} = \boldsymbol{W}\boldsymbol{V} = \begin{bmatrix} 3 & 4 & 5 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \\ 2 & 4 \end{bmatrix} = \begin{bmatrix} 3\cdot 1 + 4\cdot 0 + 5\cdot 2 & 3\cdot 0 + 4\cdot 3 + 5\cdot 4 \\ 1\cdot 1 + 0\cdot 0 + 1\cdot 2 & 1\cdot 0 + 0\cdot 3 + 1\cdot 4 \end{bmatrix} = \begin{bmatrix} 13 & 32 \\ 3 & 4 \end{bmatrix}$$

- Think of it as: $\boldsymbol{W} \underbrace{\begin{bmatrix} \boldsymbol{v}_1, \dots, \boldsymbol{v}_n \end{bmatrix}}_{\boldsymbol{V}} = \begin{bmatrix} \underbrace{\boldsymbol{W}\boldsymbol{v}_1}_{\boldsymbol{u}_1}, \dots, \underbrace{\boldsymbol{W}\boldsymbol{v}_n}_{\boldsymbol{u}_n} \end{bmatrix} = \boldsymbol{U}$

  – Hence: Each column $\boldsymbol{u}_i = \boldsymbol{W}\boldsymbol{v}_i$ in ***U*** can be computed by a matrix-vector product

Multiplication of a matrix with a matrix

- **Dimensions:** $$\underbrace{m \times n}_{\boldsymbol{W}} \cdot \underbrace{n \times j}_{\boldsymbol{V}} = \underbrace{m \times j}_{\boldsymbol{U}}$$
  - Number of columns of left matrix must match number of rows of right matri
- **Non-commutative (in general):** $\boldsymbol{V}\boldsymbol{W} \neq \boldsymbol{W}\boldsymbol{V}$
- **Associative:** $\boldsymbol{V}(\boldsymbol{W}\boldsymbol{X}) = (\boldsymbol{V}\boldsymbol{W})\boldsymbol{X}$
- **Transpose Product:** $(\boldsymbol{V}\boldsymbol{W})^T = \boldsymbol{W}^T\boldsymbol{V}^T$

Important special cases

- **Scalar (Inner) product:**

$$\boldsymbol{w}^T\boldsymbol{v} = [w_1, \dots, w_n]\begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = w_1v_1 + \dots + w_nv_n = \langle \boldsymbol{w}, \boldsymbol{v} \rangle$$

  - The scalar product can be written as vector-vector product

Important special cases

- **Compute row/column averages of matrix** $\boldsymbol{X} = \underbrace{\begin{bmatrix} X_{1,1} & \dots & X_{1,m} \\ \vdots & & \vdots \\ X_{n,1} & \dots & X_{n,m} \end{bmatrix}}_{n \text{ (samples) } \times m \text{ (entries)}}$

  - Vector of row averages (average over all entries per sample)

$$\begin{bmatrix} \frac{1}{m}\sum_{i=1}^{m} X_{1,i} \\ \vdots \\ \frac{1}{m}\sum_{i=1}^{m} X_{n,i} \end{bmatrix} = \boldsymbol{X} \begin{bmatrix} \frac{1}{m} \\ \vdots \\ \frac{1}{m} \end{bmatrix} = \boldsymbol{X}\boldsymbol{a}, \quad \text{with } \boldsymbol{a} = \begin{bmatrix} \frac{1}{m} \\ \vdots \\ \frac{1}{m} \end{bmatrix}$$

  - Vector of column averages (average over all samles per entry)

$$\left[\frac{1}{n}\sum_{i=1}^{n} X_{i,1}, \dots, \frac{1}{n}\sum_{i=1}^{n} X_{i,m}\right] = \left[\frac{1}{n}, \dots, \frac{1}{n}\right] \boldsymbol{X} = \boldsymbol{b}^T \boldsymbol{X}, \text{ with } \boldsymbol{b} = \begin{bmatrix} \frac{1}{n} \\ \vdots \\ \frac{1}{n} \end{bmatrix}$$

Matrix Inverse

	scalar	matrices
Definition:	$w \cdot w^{-1} = 1$	$\boldsymbol{W}\boldsymbol{W}^{-1} = \boldsymbol{I}, \quad \boldsymbol{W}^{-1}\boldsymbol{W} = \boldsymbol{I}$

- **Unit Element:** Identity matrix, e.g., 3 x 3: $\boldsymbol{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- **Verify it!**

$$\boldsymbol{W} = \begin{bmatrix} 1 & \frac{1}{2} \\ -1 & 1 \end{bmatrix} \qquad \boldsymbol{W}^{-1} = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} \\ \frac{2}{3} & \frac{2}{3} \end{bmatrix}$$

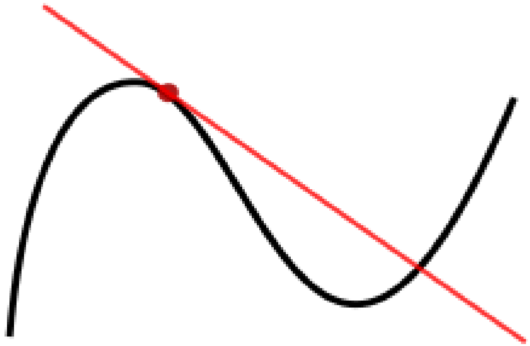
$$\boldsymbol{W}\boldsymbol{W}^{-1} = \begin{bmatrix} 1 & \frac{1}{2} \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} \\ \frac{2}{3} & \frac{2}{3} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- **Note:** We can only invert quadratic matrices (num rows = num cols)

Calculus

We also need to talk about derivatives…

“The derivative of a function of a real variable measures **the sensitivity to change of a quantity** (a function value or dependent variable) which is determined by another quantity (the independent variable)” (Wikipedia)



Function: $f(x)$

Derivative: $\frac{\partial f(x)}{\partial x}$

Minimum/Maximum: $\frac{\partial f(x)}{\partial x} = 0$

Matrix Calculus

Derivatives of a scalar function w.r.t a vector…

- Yields the gradient vector: $$\nabla_{\boldsymbol{x}} f = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial x_d} \end{bmatrix}$$
- Example: Quadratic form $\nabla_{\boldsymbol{x}} \boldsymbol{x}^T \boldsymbol{x} = 2\boldsymbol{x}$ $\nabla_{\boldsymbol{x}} \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} = 2\boldsymbol{A}\boldsymbol{x}$

Derivatives of a vector-valued function w.r.t a vector…

- Yields a matrix (the Jacobian) $$\nabla_{\boldsymbol{x}} \boldsymbol{f} = \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_k(\boldsymbol{x})}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_1(\boldsymbol{x})}{\partial x_d} & \cdots & \frac{\partial f_k(\boldsymbol{x})}{\partial x_d} \end{bmatrix}$$
- Example: Linear form $\nabla_{\boldsymbol{x}} \boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T$

Matrix Calculus

Derivatives of a scalar function w.r.t. a matrix…

- … is again a matrix $$\nabla_{\boldsymbol{W}} f = \frac{\partial f(\boldsymbol{W})}{\partial \boldsymbol{W}} = \begin{bmatrix} \frac{\partial f(\boldsymbol{W})}{\partial W_{11}} & \cdots & \frac{\partial f(\boldsymbol{W})}{\partial W_{1d}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(\boldsymbol{W})}{\partial W_{k1}} & \cdots & \frac{\partial f(\boldsymbol{W})}{\partial W_{kd}} \end{bmatrix}$$

Derivatives of a vector-valued function w.r.t. a matrix…

- … is a 3D tensor!
- So that gets a bit tricky… luckily we (almost) do not need that

Matrix Calculus

We need to know some rules from **Matrix Calculus** (see wikipedia)

	scalar	vector	matrix
• **Linear:**	$\frac{\partial ax}{\partial x} = a$	$\nabla_{\boldsymbol{x}} \boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T$	$\nabla_{\boldsymbol{X}}\ \boldsymbol{a}^T\boldsymbol{X}\boldsymbol{b} = \boldsymbol{a}\boldsymbol{b}^T$ $\nabla_{\boldsymbol{X}}\ \mathrm{tr}(\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}) = \boldsymbol{A}^T\boldsymbol{B}^T$
• **Quadratic:**	$\frac{\partial x^2}{\partial x} = 2x$	$\nabla_{\boldsymbol{x}} \boldsymbol{x}^T\boldsymbol{x} = 2\boldsymbol{x}$ $\nabla_{\boldsymbol{x}} \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} = (\boldsymbol{A}^T + \boldsymbol{A})\boldsymbol{x}$ $= 2\boldsymbol{A}\boldsymbol{x}$ if $\boldsymbol{A}$ is symetric	

Today‘s Agenda!

Recap: Types of Machine Learning

Recap: Linear Algebra

- Vectors, Matrices and manipulation of those

Linear Regression:

- Least-Squares Solution
- Generalized Linear Regression Models
- Ridge Regression

Regression

Regression:

– Learn continuous function

$$y = f(x) + \epsilon$$

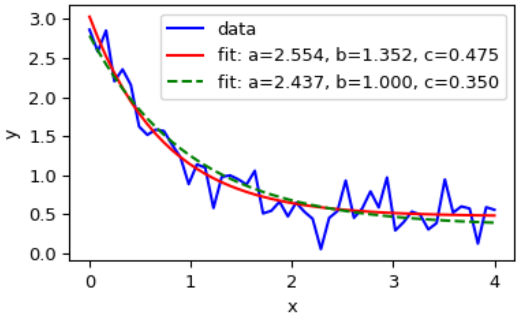
Linear Regression:

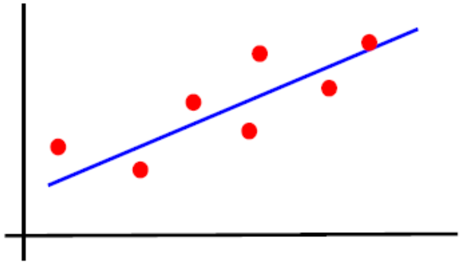
– We “just” fit a line

$$y = f(x) + \epsilon = w_0 + w_1 x + \epsilon$$

We assume that the outputs are affected by (typically) Gaussian noise:

$$\epsilon \sim \mathcal{N}(0, 1)$$





Objective of Regression

We want to minimize the summed (or mean) squared error

$$\mathrm{SSE} = \sum_{i=1}^{N} \left(y_i - f(\boldsymbol{x}_i)\right)^2$$

- … where the input ***x*** is a d-dimensional vector

Why do we use the squared error?

- It is fully differentiable
- Easy to optimize
- It also makes sense as:

$$f^*(\boldsymbol{x}) = \mathrm{argmin}_{f(\boldsymbol{x})} \mathrm{SSE} \Rightarrow f^*(\boldsymbol{x}) = \mathbb{E}[y|\boldsymbol{x}]$$

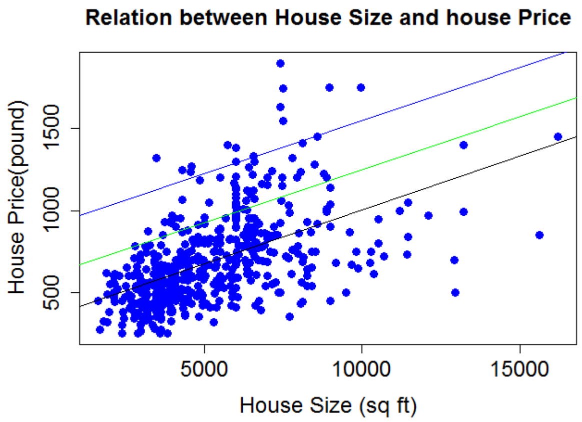
  – Hence, we always estimate the mean of the target function!

Linear regression models

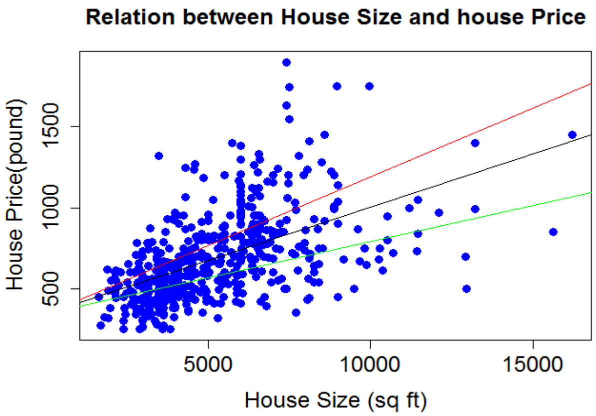
- In linear regression, the output *y* is modelled as linear function of the input $\boldsymbol{x}_i$

$$y = f(x) + \epsilon = w_0 + w_1 x + \epsilon$$

Effect of w_0



Effect of w_1



Objective for Linear Regression

We want to consider linear functions with multiple inputs

$$f(\boldsymbol{x}_i) = w_0 + \sum_j w_j x_{i,j}$$

Our SSE objective now looks the following

$$\mathrm{SSE} = \sum_{i=1}^{N} \left(y_i - \left(w_0 + \sum_j w_j x_{i,j} \right) \right)^2$$

Can we simplify it using matrices??

Linear regression models in matrix form

- Equation for the i-th sample

$$\hat{y}_i = w_0 + \sum_{j=1}^{D} w_j x_{i,j} = \tilde{\boldsymbol{x}}_i^T \boldsymbol{w}, \quad \text{with } \tilde{\boldsymbol{x}}_i = \begin{bmatrix} 1 \\ \boldsymbol{x}_i \end{bmatrix} \text{ and } \boldsymbol{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_D \end{bmatrix}$$

- Equation for full data set

$$\hat{\boldsymbol{y}} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{x}}_1^T \boldsymbol{w} \\ \vdots \\ \tilde{\boldsymbol{x}}_n^T \boldsymbol{w} \end{bmatrix} = \boldsymbol{X}\boldsymbol{w}$$

  - $\hat{\boldsymbol{y}}$ is a vector containing the output for each sample
  - $\boldsymbol{X} = \begin{bmatrix} \tilde{\boldsymbol{x}}_1^T \\ \vdots \\ \tilde{\boldsymbol{x}}_n^T \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{x}_1^T \\ \vdots & \vdots \\ 1 & \boldsymbol{x}_n^T \end{bmatrix}$ is the data-matrix containing a vector of ones as the first column as bias

Linear regression models in matrix form

- Error vector: $$\boldsymbol{e} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \boldsymbol{y} - \hat{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}$$
- Sum of squared errors (SSE)

$$\text{SSE} = \sum_i (y_i - \hat{y}_i)^2 = \sum_i e_i^2 = \boldsymbol{e}^T\boldsymbol{e} = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

- We have now written the **SSE completely in matrix form**!

Deriving Linear Regression

- How do we obtain the optimal $\boldsymbol{w}$? (which minimizes the SSE)

$$\boldsymbol{w}^* = \operatorname{argmin}_{\boldsymbol{w}} \mathrm{SSE} = \operatorname{argmin}_{\boldsymbol{w}} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

At a minimal value of a function, its derivative is zero

I.e., find a $\boldsymbol{w}$ where $\dfrac{\partial \mathrm{SSE}}{\partial \boldsymbol{w}} = \mathbf{0}^T$

Estimation of w

$$
\begin{aligned}
\mathrm{SSE}(\boldsymbol{w}) &= (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) \\
&= \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{y} \\
&= \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{y}^T\boldsymbol{X}\boldsymbol{w} + \boldsymbol{y}^T\boldsymbol{y}
\end{aligned}
$$

Take the derivative w.r.t $\boldsymbol{w}$:

$$
\begin{aligned}
\nabla_{\boldsymbol{w}}\mathrm{SSE}(\boldsymbol{w}) &= \frac{\partial}{\partial \boldsymbol{w}}\left\{\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{y}^T\boldsymbol{X}\boldsymbol{w} + \boldsymbol{y}^T\boldsymbol{y}\right\} \\
&=
\end{aligned}
$$

Setting the gradient to $\mathbf{0}$ yields

$$
\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}
$$

Discussion

We have now derived our first ML algorithm: **Linear Regression!**

- The solution is called Least Squares solution
- One of the rare cases where we can obtain a closed form solution

This was only possible because:

- The cost-function (SSE) is convex for linear f(x)
  - There is only one minimum
- The cost function is quadratic in w
  - The minimum is easy to obtain

Ask questions!!!

Evaluating linear regression models

How can we estimate the quality of the model?

- The SSE can take arbitrary values depending on the range of the output
- Make the evaluation invariant to the variance of y

R-Square (or R^2) determines how much of the total variation in y is explained by the variation in x. Mathematically, it can be written as

$$R^2 = 1 - \frac{\text{Regression sum of squares}}{\text{Total sum of squares}} = 1 - \frac{\sum_{n=1}^{N}(\hat{y}_n - y_n)^2}{\sum_{n=1}^{N}(y_n - \bar{y})^2}$$

where $\bar{y}$ is the mean of the outputs. R^2 tells how well the regression line approximates the real data points. An R^2 of 1 indicates that the regression line perfectly fits the data.

Today‘s Agenda!

Recap: Types of Machine Learning

Recap: Linear Algebra

- Vectors, Matrices and manipulation of those

Linear Regression:

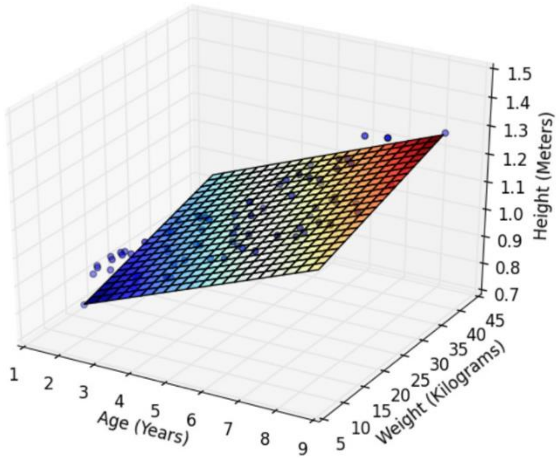
- Least-Squares Solution
- **Generalized Linear Regression Models**
- Ridge Regression

Linear Functions

So far, we modelled our function f as linear in **x** and **w**

$$f(\boldsymbol{x}) = \tilde{\boldsymbol{x}}^T \boldsymbol{w}$$

However, this equation can only represent hyper-planes in the D-dimensional input space



General Form

In a more general writing, we could rewrite it as

$$f(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{w}$$

Where $\phi(\boldsymbol{x})$ is a vector valued function of the input vector $\boldsymbol{x}$. This is also called **linear basis function models**, and $\phi_i(\boldsymbol{x})$ are known as **basis functions.**

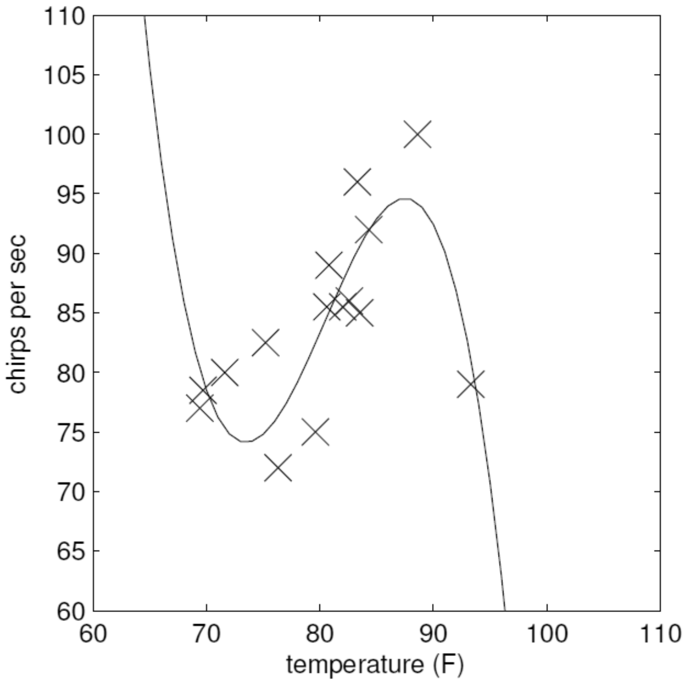
The model is linear in the parameter $\boldsymbol{w}$, not necessarily linear in $\boldsymbol{x}$.

Example of Polynomial Curve Fitting

$$f(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{w}$$

where

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}, \qquad \boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \end{bmatrix}$$

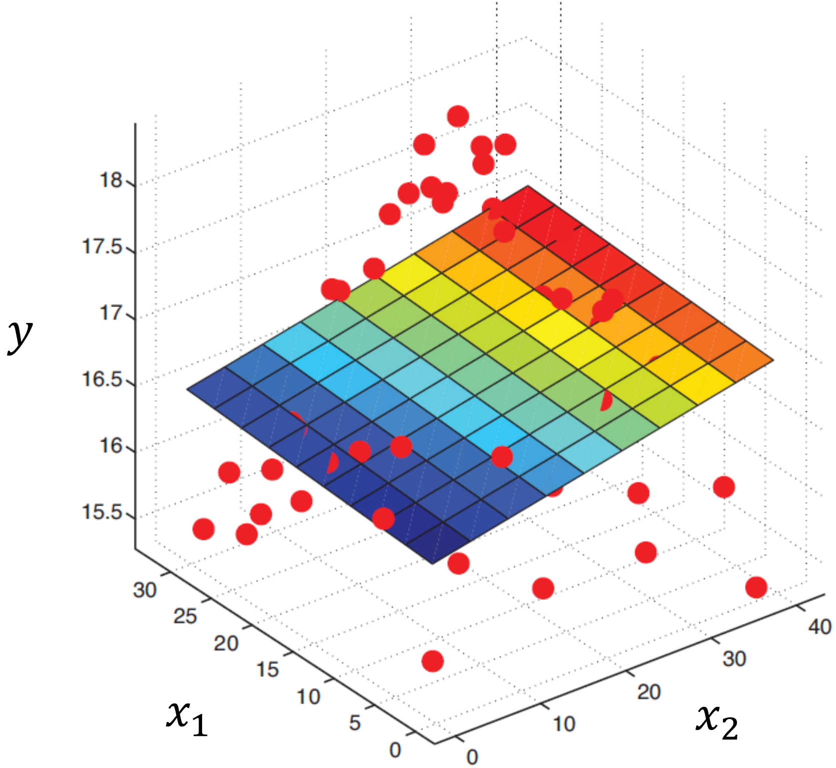


Example of Multiple Linear Regression

$$f(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{w}$$

where

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}, \qquad \boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

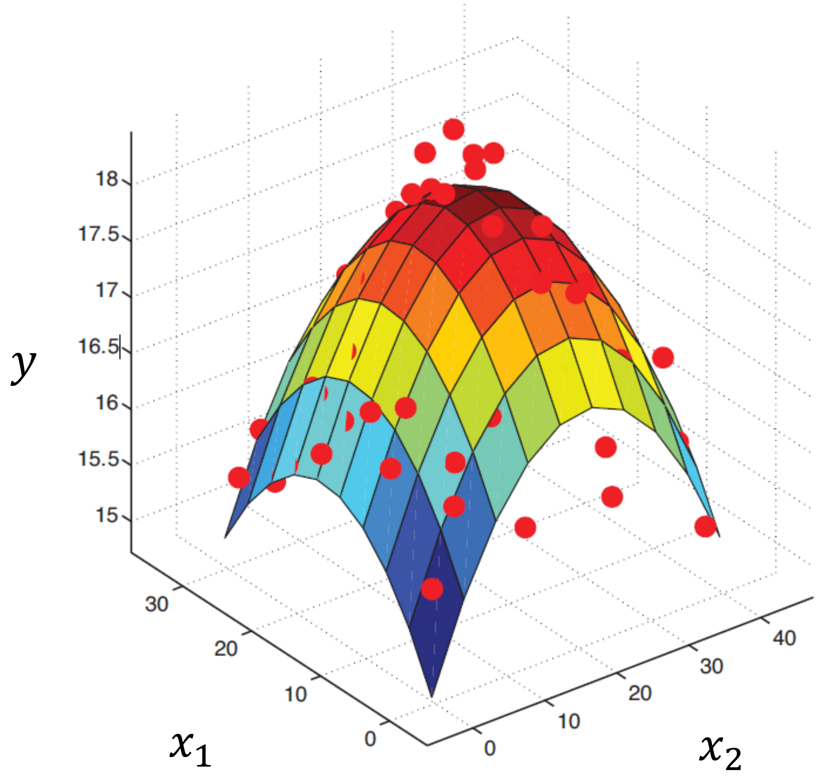


Example of Fitting Quadratic Form

$$f(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{w}$$

where

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}, \qquad \boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$



Generalized Linear Regression

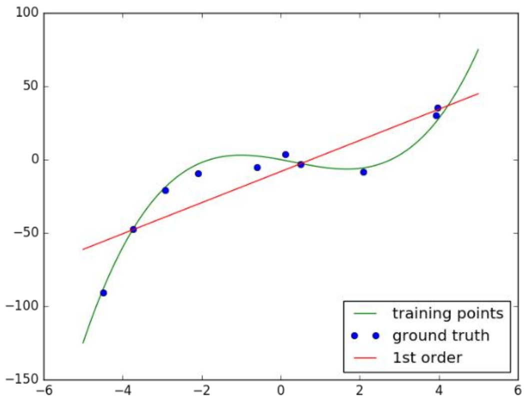
The derivations stay exactly the same, just the data matrix is now replaced by the basis function matrix, i.e.:

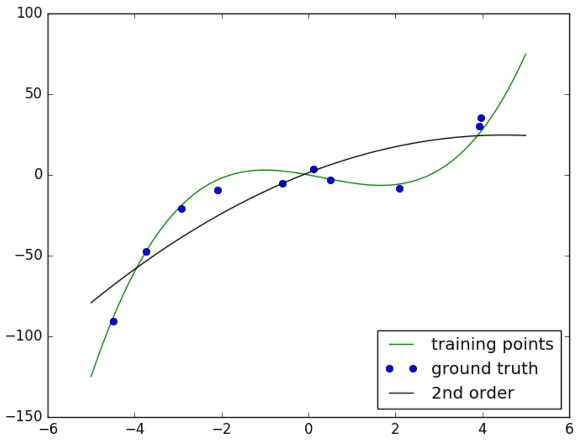
$$\boldsymbol{w}^* = (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{y},$$

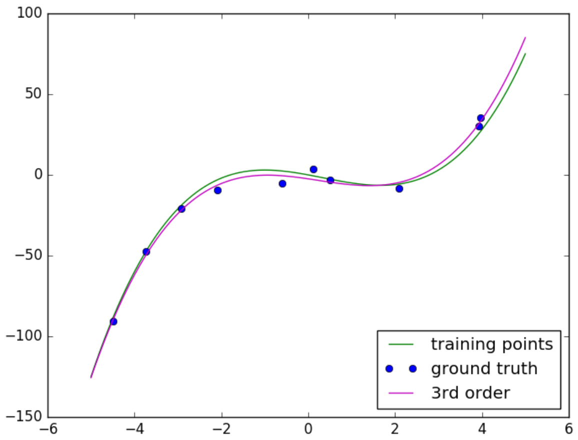
with $\boldsymbol{\Phi} = \begin{bmatrix} \phi_1^T \\ \vdots \\ \phi_n^T \end{bmatrix}$

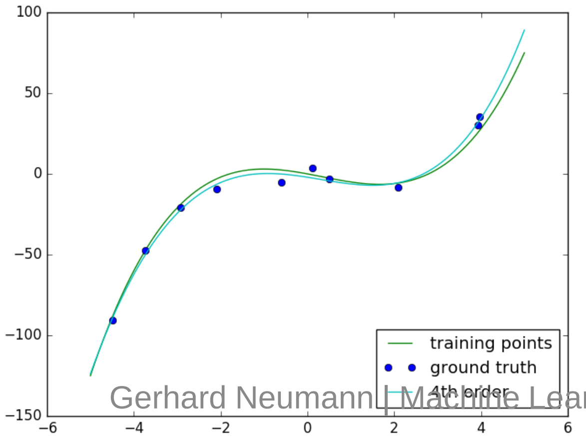
- In principle, this allows us to **learn any non-linear function**, if we know suitable basis functions (which is typically not the case).

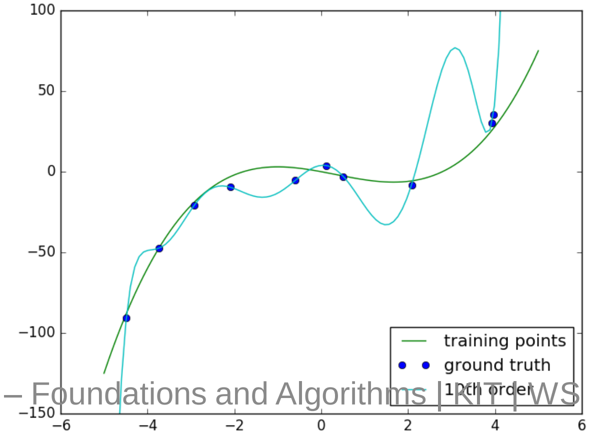
Example: Selecting the order of the polynom











Overfitting for polynomial regression

The error on the training set **is not an indication for a good fit**!!

- We always need an **independent test-set!**

Overfitting:

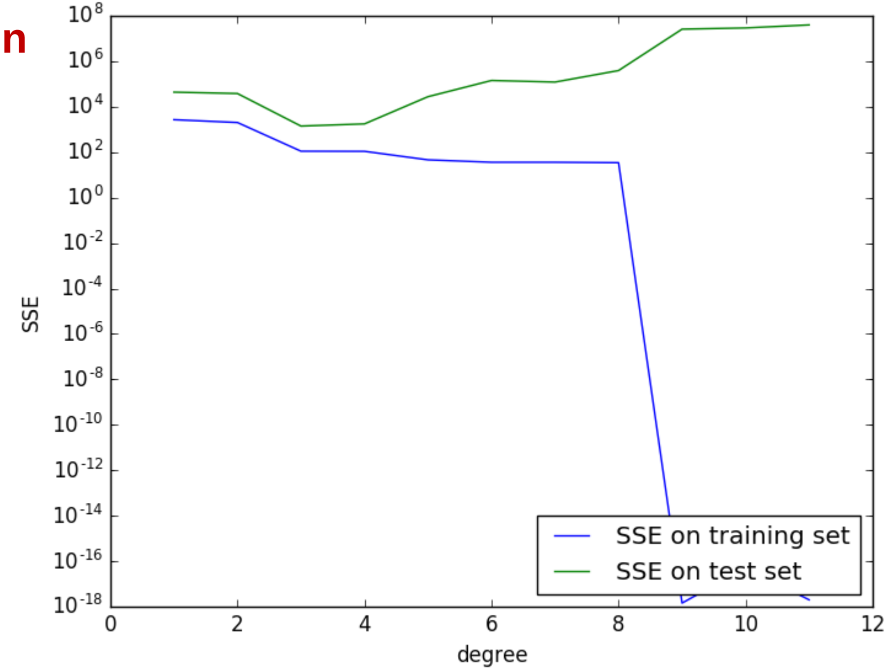
- Training error goes down
- Test error goes up

The model is too complex. It fits the noise and has unspecified behavior between the training points.

Underfitting:

- Training + Test error are high

The model is too simple to fit the data



Regularization

Regularization:

- Limit the model such that it can not fit the training data perfectly any more

Simple form of regularization: forcing the weights **w** to be small

- Small weights will lead to a smoother function
- Introduce "regularization term" in cost-function

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

Data term + Regularization term

- where λ is the regularization factor
- Needs to be tuned manually in most cases

Regularized Least Squares

With the sum-of-squares error function and a quadratic regularizer, we get

$$L_{\text{ridge}} = (\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}) + \lambda \boldsymbol{w}^T\boldsymbol{w}$$

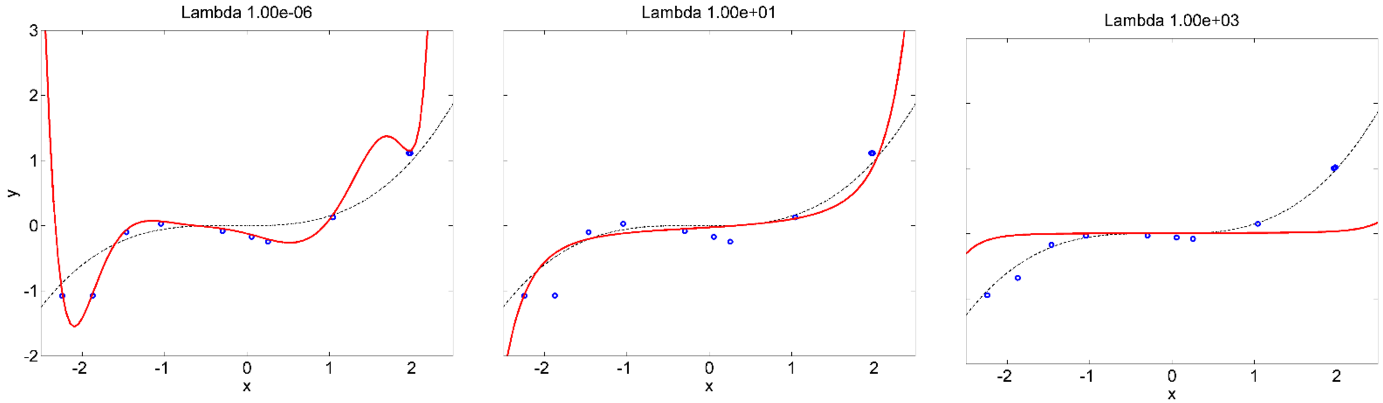
- This particular choice of regularizer is known as **weight decay** because in sequential learning algorithms, it encourages weight values to decay towards zero, unless supported by the data.
- In statistics, it is called **ridge regression**.

Derivations can be done similarly as before. The solution is given by

$$\boldsymbol{w}^*_{\text{ridge}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}$$

- ***I*** is the Identity matrix
- The matrix $\left(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda \boldsymbol{I}\right)$ is now full rank and can be more easily inverted

Ridge regression: Degree n=15



Influence of the regularization constant

Takeaway messages

What have we learned today?

- Familiarized with matrix manipulations and matrix calculus
- What a regression problem is
- How to obtain the Least-Squares solution in closed form
  - Only possible as the cost function is quadratic in the weights
- Generalized Linear Regression
  - Non-linear functions in x are fine as long as linear in w
- Avoid overfitting by keeping the weights small