2. Regression
Motivation

- **Regression** aims at predicting a **numerical target feature** based on one or multiple other (numerical) features (we’ll stick to the case of **one other feature** for **today**).

- **Example**: Predict **fuel consumption** (in miles/gallon) of a car based on its **power** (in horsepowers).

- **Data**: **Auto MPG Dataset** from **UCI ML Repository**
  
  https://archive.ics.uci.edu/ml/datasets/Auto+MPG
  
  - **398 cars** (392 with all features)
  - **8 features** (mpg, cylinders, horsepower, weight, …)
Motivation
Agenda

- 2.1 Ordinary Least Squares
- 2.2 Gradient Descent
- 2.3 Multiple Linear Regression
- 2.4 Handling Non-Numerical Features
- 2.5 Polynomial Regression
- 2.6 Evaluation Fundamentals
- 2.7 Regularization
2.1 Ordinary Least Squares

- How can we predict the value of a **numerical feature** \( y \) based on the value of **another numerical feature** \( x \)?

- First, we need to make **some assumption** about their **relationship**, i.e., how \( x \) influences \( y \)

\[
\hat{y} = w_0 + w_1 x
\]

- Our **model** thus assumes that there is a **linear relationship** between the two features \( x \) and \( y \)

- How can we **determine the coefficients** \( w_0 \) and \( w_1 \)?
Linear Model

Different values of $w_0$ and $w_1$ correspond to different lines in our plot.

Which ones are best?
Data Points

- We determine the values of the coefficients $w_0$ and $w_1$ based on training data that is available to us.

- Our training data consists of $n$ data points $(x_i, y_i)$.

  In our example those are pairs of power (in hp) and fuel consumption (in mpg) of individual cars:

<table>
<thead>
<tr>
<th>hp</th>
<th>mpg</th>
</tr>
</thead>
<tbody>
<tr>
<td>130.0</td>
<td>18.0</td>
</tr>
<tr>
<td>165.0</td>
<td>15.0</td>
</tr>
<tr>
<td>150.0</td>
<td>18.0</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
Mean, Variance, and Standard Deviation

- We define the **mean** of feature $x$ and $y$ in our training data as

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \\
\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i
\]

- **Variance** of feature $x$ and $y$ in our training data is defined as

\[
\sigma_x^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \\
\sigma_y^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^2
\]

- The values $\sigma_x$ and $\sigma_y$ are referred to as the **standard deviation** of features $x$ and $y$
Covariance

- **Covariance** $\text{cov}_{x,y}$ measures the degree of joint variability between the two features $x$ and $y$

$$\text{cov}_{x,y} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

- Large covariance suggests that the two features vary jointly
  - a **positive value** indicates that they tend to deviate from their respective mean in the **same direction**
  - a **negative value** indicates that they tend to deviate from their respective mean in **opposite directions**
Pearson Correlation Coefficient

- **Pearson correlation coefficient** (also: Pearson’s r) is a **normalized measure of linear correlation** between the two features $x$ and $y$

$$\text{cor}_{x,y} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}} = \frac{\text{COV}_{x,y}}{\sigma_x \sigma_y}$$

- Pearson correlation coefficient takes values in [-1, +1]
  - a value of -1 indicates a **negative linear correlation**
  - a value of 0 indicates that there is **no linear correlation**
  - a value of +1 indicates a **positive linear correlation**
Pearson Correlation Coefficient

\[ \text{cor}_{hp, mpg} \approx -0.7784 \]
Anscombe’s Quartet

- All four datasets have the **same mean, variance, correlation coefficient, and optimal regression line**.

Source: https://en.wikipedia.org/wiki/Anscombe%27s_quartet
Loss Function

- **Loss function** measures how well our model, for a specific choice of coefficients $w_0$ and $w_1$, describes the training data (i.e., how much we lose by using the model).

- **Residual** for data point $(x_i, y_i)$ measures how much the observed value $y_i$ differs from the prediction of our model:

  $$(y_i - \hat{y}_i) = (y_i - (w_0 + w_1 x_i)) = (y_i - w_0 - w_1 x_i)$$
Loss Function

- Ordinary least squares (OLS) uses the **sum of squared residuals** (also: sum of squared errors) as a loss function

\[
L(w_0, w_1) = \sum_{i=1}^{n} (y_i - w_0 - w_1 x_i)^2
\]

- Since we’re interested in finding the coefficients \( w_0 \) and \( w_1 \) that minimize the loss, we obtain the **optimization problem**

\[
\arg \min_{w_0, w_1} \sum_{i=1}^{n} (y_i - w_0 - w_1 x_i)^2
\]
Let’s **plot the loss function** for our example dataset.
Determining Optimal Coefficients Analytically

- **Optimal values** for the coefficients \( w_0 \) and \( w_1 \) can be determined *analytically* in the case of OLS

1) compute **partial derivatives** of loss function w.r.t. \( w_0 \) and \( w_1 \)

\[
\frac{\partial L}{\partial w_0} = -2 \sum_{i=1}^{n} (y_i - w_0 - w_1 x_i)
\]

\[
\frac{\partial L}{\partial w_1} = -2 \sum_{i=1}^{n} (y_i - w_0 - w_1 x_i) x_i
\]

2) identify **common zero** by solving system of equations

\[
\frac{\partial L}{\partial w_0} = 0 \quad \frac{\partial L}{\partial w_1} = 0
\]
Determining Optimal Coefficients Analytically

- We obtain the following **closed-form solutions** to compute optimal values of the coefficients based on our data.

\[
\begin{align*}
    w_0^* &= \frac{1}{n} \sum_{i=1}^{n} y_i - w_1^* \frac{1}{n} \sum_{i=1}^{n} x_i \\
    w_1^* &= \frac{n \sum_{i=1}^{n} x_i y_i - \left( \sum_{i=1}^{n} x_i \right) \left( \sum_{i=1}^{n} y_i \right)}{n \sum_{i=1}^{n} x_i^2 - \left( \sum_{i=1}^{n} x_i \right)^2}
\end{align*}
\]
Determining Optimal Coefficients Analytically

- For our example dataset, we obtain the optimal coefficients

\[ w_0^* = 39.9359 \quad \quad \quad w_1^* = -0.1578 \]
**R² Coefficient of Determination**

- **R² coefficient of determination** (short: ”R squared”) measures **how well** the determined **regression line** approximates the **data**, or put differently, **how much of the variation observed** in the data is explained by it.

\[
R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}
\]
2.2 Gradient Descent

- Unfortunately, it is not always possible to determine optimal coefficients for a loss function analytically.

- Gradient descent is an optimization algorithm that we can use to determine the minimum of a loss function.

- Idea:
  - start with a random choice of the coefficients (here: $w_0$ and $w_1$)
  - repeat for a specified number of rounds or until convergence
    - compute the gradient for this choice of coefficients
    - update coefficients based on the gradient
Gradient Descent

- **Intuition**: Think of the loss function as a **surface** on which you want to find the **lowest point**
  - start your journey at a **random position**
  - repeat the following
    - identify **direction** with **steepest descent**
    - **walk a few steps** in the identified direction
Gradient Descent

- **Gradient** of a multivariable function is defined as the vector of its *partial derivatives*; when evaluated at a *specific point*, it indicates the *direction* of steepest ascent.

- For our loss function, the gradient is thus defined as

\[
\nabla L(w_0, w_1) = \begin{bmatrix}
\frac{\partial L}{\partial w_0} \\
\frac{\partial L}{\partial w_1}
\end{bmatrix}
\]
Gradient Descent

- We think of our current choice of coefficients as a vector

\[
\begin{bmatrix}
    w_0 \\
    w_1
\end{bmatrix}
\]

- Coefficients are updated in each round as

\[
\begin{bmatrix}
    w_0 \\
    w_1
\end{bmatrix}' = \begin{bmatrix}
    w_0 \\
    w_1
\end{bmatrix} - \alpha \nabla L(w_0, w_1)
\]

with \(0 < \alpha \leq 1\) as the learning rate (also: step size)
Extensions of Gradient Descent

- **Gradient ascent** as a counterpart to maximize functions

- **Stochastic gradient descent** (SGD) does not compute the true gradient, but **approximates the gradient** based on a single or few **randomly chosen data points** in each round

- **Gradient** can be approximated when the function at hand is **non-differentiable** or when partial derivatives are expensive to compute
Summary

- **Regression** aims at predicting a **numerical target feature** based on one multiple other (numerical) features.

- **Loss function** measures how well our model, for a specific choice of coefficients, describes the training data.

- **Ordinary least squares (OLS)** uses sum of squared errors as a loss function; closed-form optimal coefficients.

- **Gradient descent** as an optimization algorithm to find the minimum of a multivariable function.
References