Machine Learning Primer 28.10.2024
WiSe 24/25 **C. Gros**, D. Nevermann

Exercise Sheet #3

Problem 1 (*Correlations*)

The correlation coefficient is defined by

$$\mathrm{Corr}(X,Y) = \frac{\mathrm{Cov(X,Y)}}{\sigma_X \sigma_Y} = \frac{\mathrm{E}[(X-\mathrm{E}(X))(Y-\mathrm{E}(Y))]}{\sigma_X \sigma_Y}$$

for random variables X, Y. Generate two NumPy arrays `x`, `y` of random numbers of length $N = 10\,000$, compute the correlation between them numerically, and create a scatter plot in Matplotlib for visualization. Generate arrays for four cases with

(a) a correlation $\mathrm{Corr}(\mathtt{x},\mathtt{y}) \approx 0$,

(b) a correlation $\mathrm{Corr}(\mathtt{x},\mathtt{y}) \approx 1$,

(c) a correlation $\mathrm{Corr}(\mathtt{x},\mathtt{y}) \approx -1$,

(d) a correlation $\mathrm{Corr}(\mathtt{x},\mathtt{y}) \approx 0.5$.

Problem 2 (*Bayesian Inference - Coin Flip*)

Consider the example from the lecture (see here) of flipping a biased coin defined by

$$p(\text{heads}) = \alpha, \qquad p(\text{tails}) = 1-\alpha,$$

where $\alpha \in [0,1]$. The code in the lecture uses Bayesian inference to estimate the bias α of a coin from observations of flipping the coin.
The idea behind Bayesian inference is to start with a hypothesis about some probability distribution (usually one starts with a uniform distribution) and use observed data to iteratively update the distribution via Bayes' theorem

$$P(\text{hypo} \mid \text{data}) = \frac{P(\text{data} \mid \text{hypo})}{P(\text{data})} \cdot P(\text{hypo}).$$

In the case of the coin flip, the hypothesis $P(\text{hypo})$ (called *prior*) is a probability distribution assigning a probability to a potential value of the bias α. Numerically, this distribution is discretized to `nX = 11` support points.

```
nX = 11                                # numerical discretization
xx = [i*1.0/(nX-1.0) for i in range(nX)]
pp = [1.0/nX  for _ in range(nX)] # starting prior
```

In this, `xx` resembles the discretized α values and `pp` the assigned probabilities.
In each call of the function `updatePrior`, the posterior $P(\text{hypo} \mid \text{data})$ is computed using Bayes' theorem, considering new data (a coin flip) generated by

```
20 evidence = 0.0 if (np.random.uniform()>trueBias) else 1.0
```

The posterior becomes the new prior. This procedure is repeated for 200 observations.

(a) Increase the number of support points to `nX = 101`.

(b) In lines 37 to 45 a table of the current prior distribution is generated. Replace that table with an animated bar plot visualizing for each new coin flip how the prior changes.
Hint: For animating plots you can use the `FuncAnimation` class from `matplotlib.animation`.

To smooth out the updates over time, include a memory into the update process. Define a weight parameter $\beta \in [0, 1]$ that controls the degree of belief in new data

$$\text{prior} \leftarrow (1 - \beta) \cdot \text{prior} + \beta \cdot \text{posterior},$$

where in this new information is weighted by β.

(c) Implement the memory, as defined above, in the program. How does this change things?

(d) Generate a plot of the variance of the posterior distribution after a sufficient number of updates as a function of β. How does the variance scale with β?