Reminders to finish last exercises and to our way of working: -1, 0
Prepare before class such that you can present your result: 1, 2, 4, 5
Worked on in class, but you can prepare them at home: 3, 6

Carefully read the instructions below!

(-1) Finalize last exercise sheet

Finalize all exercises of the last exercise sheet and upload the results to your defined GitHub repository.

(0) Setup a git project on GitHub

In order to get a bit of `git` training done we work for all the exercises on GitHub.

(a) Use the project created for the previous exercises on GitHub

(b) Give the instructor (kandolfp) access to the project (see `GitHubdocumentation` for help)

(c) Create a `pdm` project in your repository and commit the necessary files.

(d) Create an appropriate structure in your repository for the rest of the exercise sheet (maybe have a look at the exercises to have a better idea first), not everything should be in the main folder.

(e) Try to structure your work on the exercises with git, i.e.

- Don't commit things that do not belong together in one single commit. Each exercise can be considered as a separate thing. Subparts of an exercise might be independent as well.
- Use meaningful commit messages `https://www.conventionalcommits.org/en/v1.0.0/`
- Make sure that you do not commit something that does not work - produces an error. If you have difficulties with an exercise you can also commit your best effort in this case.

(f) Add a README.md that explains what you are doing, how to run the exercises and anything else that is necessary (quick guide to `pdm`), maybe note your name somewhere.

(g) Optional

- Work with issues, you can reference the issue in the commit message, GitHub documentation

(1) Dataset investigation

Investigate the palmerpenguins dataset so that you are able to work on it in the successive exercises. Follow these steps:

(a) Find the correct package for Python to install and load the data.

(b) Check if you have incomplete entries and clean up the data

(c) What are the features available for each penguin?

(d) Create a series of plots where you show the different feature combinations and color the individuals per species of penguins.

(e) Which two features are a good choice to continue with a classification?

(2) Support Vector Machines - Linear classification

We continue working on the palmerpenguins dataset and you selected two features in the above exercise. For the SVM use the `sklearn` package.

(a) Let us reduce to two species, depending of what you choose above, drop one of the `species` such that only two groups are present.

(b) Can you use a hard margin classification in this case?

- If not, can you remove outliers from the classes and allow for it to converge?
- Visualize the separation, including the *street*.

(c) Train a `LinearSVC` for a soft margin classification for the entire dataset (including potential outliers that where deleted in the previous step) and test it with and without `StandardScaler`

- Include the *street* that separates the two sets in a plot.

(3) Doing the math?

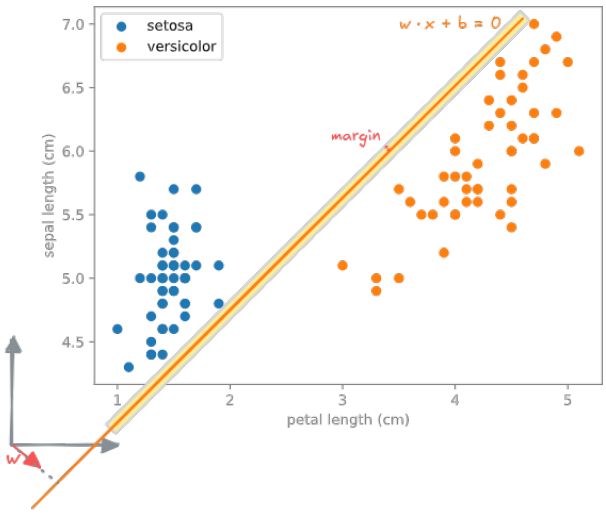
We start of with the linear SVM where we construct a hyperplane

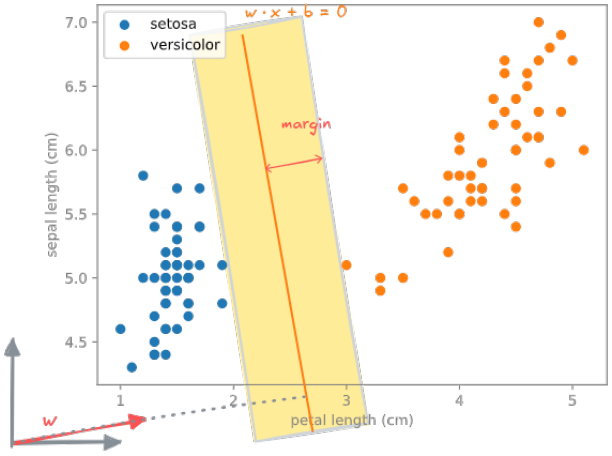
$$\langle w, x\rangle + b = 0$$

with a vector w and a constant b. The classification is than performed via

$$\operatorname{sign}(\langle w, x_j\rangle + b) = \begin{cases} +1 \\ -1 \end{cases}$$

There is a natural degree of freedom in this selection of the hyperplane, see next pictures.





Compute the vector w in the two cases of the above images. The vector w is normal to the line. For the left image, two points on the line are $v_1 = [1.25, 4.1]^{\mathrm{T}}$, $v_2 = [5, 7.4]^{\mathrm{T}}$. For right image, two points on the line are $z_1 = [2.6, 4.25]^{\mathrm{T}}$, $z_2 = [2, 7]^{\mathrm{T}}$.

Classify the two points

$$a = [1.4, 5.1]^{\mathrm{T}},$$
$$b = [4.7, 7.0]^{\mathrm{T}}.$$

(4) Support Vector Machines - Nonlinear classification

We continue working on the palmerpenguins dataset. From the first exercise, select two features (for two species) that you see fit for a polynomial separation.

For the SVM use the `sklearn` package.

(a) Use a `sklearn.pipeline` to train a polynomial SVC.

(b) Visualize your results for different degrees of the polynomial (start a line).

(c) Use a `kernel` to produce similar results with the *kernel trick*.

(d) Also try with a Gaussian RBF kernel and find parameters that work well.

(5) Linear Regression

The fit of a linear regression model depends on the selected norm for the optimization. Let us take a look at the 1-norm, the 2-norm, and the ∞-norm, i.e.

$$E_1(h) = \frac{1}{m}\sum_{k=1}^{m} |h(X^{(k)}, \Theta) - y^{(k)}|,$$

for the 1-norm or mean absolute error,

$$E_2(h) = \sqrt{\frac{1}{m}\sum_{k=1}^{m} |h(X^{(k)}, \Theta) - y^{(k)}|^2},$$

for the 2-norm or least-square error,

$$E_\infty(h) = \max_k |h(X^{(k)}, \Theta) - y^{(k)}|$$

for the ∞-norm or maximum norm error.

For a set of data points with x and y coordinates as stated below find optimal solutions according to the different norms with the help of `scipy.optimize.fmin`. Compare the results if you switch from y to z.

Our linear regression has the form

$$h(x_i) = \Theta_1 + \Theta_0 x_i = y_i$$

and as initial guess for Θ^0 we can use $[\Theta_0^0 = 1, \Theta_1^0 = 1]$.

```
x = np.arange(1, 11).reshape(-1, 1)
y = np.array([0.2, 0.5, 0.3, 0.5, 1.0, 1.5, 1.8, 2.0, 2.3, 2.2]).reshape(-1, 1)
z = np.array([0.2, 0.5, 0.3, 3.5, 1.0, 1.5, 1.8, 2.0, 2.3, 2.2]).reshape(-1, 1)
```

To do this follow the instructions:

(a) Create three functions corresponding to the norms for the minimization process use as function signature $h(\Theta, x, y)$.

(b) Create your initial guess Θ^0.

(c) Call the optimization function form `scipy` with something along

`p1 = scipy.optimize.fmin(fit1, theta0, args=(x, y))`.

(d) You can use the function `np.polyval` (check the order or Θ) for the evaluation at `np.arange(0, 11, 0.1)`.

(e) Visualize the findings in a single plot with the data points you are interpolating.

(f) Redo with `args=(x, z)`.

(6) Polynomial Regression

In order to do some polynomial fitting we use `sklearn.preprocessing.PolynomialFeatures`.

```
np.random.seed(42)
m = 100
X = 6 * np.random.rand(m, 1) - 3
y = 0.5 * X ** 2 + X + 2 + np.random.randn(m, 1)
```

For the above dataset fit a degree 1, 2, 5, 100 polynomial and visualize and explain the results.

Try to solve these problems on your own. If you get stuck, maybe some of your fellow students can help you. Why not write down your question in the course forum in SAKAI. Please try not to write back entire solutions or exchange solutions in the forum, this is not the idea and will backfire in the long run (also we might decide to delete such posts).