Vision and Sports Summer School 2022
Best Practice in Machine Learning for Computer Vision – Exercises

In these exercises you will practice the techniques and principles introduced in the lecture. You won't need any machine learning packages or deep learning framework. Everything can be implemented in simple `numpy`.

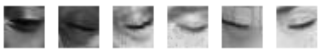
The example task we will solve is to classify images of eyes as open or closed.

We fix the following setting:

- **input**: x: images of eyes, grayscale, resolution 24×24 pixels
- **output**: $y = 1$: eye is open, $y = -1$: eye is closed.
- **model**: linear function, $f(x) = \langle w, x \rangle$ with model parameters $w \in \mathbb{R}^d$
- **quality measure**: $\ell(y, f(x)) = [\![\text{sign} f(x) \neq y]\!] = \begin{cases} 1 & \text{if sign} f(x) \neq y \\ 0 & \text{otherwise} \end{cases}$

examples:

open eyes 

closed eyes 

To learn the model parameters, we use the *Least-Squares Support Vector Machine (LS-SVM)* algorithm:

Algorithm 1 Least-Squares Support Vector Machine (LS-SVM)

input $X \in \mathbb{R}^{n \times d}$ data matrix for $x_1, \ldots, x_n$ with $x_i \in \mathbb{R}^d$
input $Y \in \mathbb{R}^n$ vector of output values $y_1, \ldots, y_n$ with $y_i \in \mathbb{R}$
input $\lambda \geq 0$ regularization hyperparameter

compute $A = X^\top X + \lambda \, \text{Id}_{d \times d}$ where $\text{Id}_{d \times d}$ is the $d \times d$ identity matrix
compute $b = X^\top Y$
compute w by solving $Aw = b$ (e.g. $w = A^{-1}b$ or $w =$ `linsolve`(A, b), etc.)

output w

Preparations

Download the file `http://cvml.ist.ac.at/VS3data.zip` (33MB) and unzip it. Note: if you're using *colab*, you can do this from inside a cell:

```
!wget http://cvml.ist.ac.at/VS3data.zip
!unzip VS3data.zip
```

Exercise 1 – Model Learning

1. Load the data files *"XtrainIMG.txt"* and *"Ytrain.txt"*. They contain the training images as row vectors of pixel intensities and the ground truth annotation, respectively.
2. Split the data randomly into two parts of (approximately) equal size: $Xtrn/Ytrn$, which we will use for learning the model, and $Xval/Yval$, which we will use as validation data.
   Hint: if the randomization makes you trouble, just split into first half/second half, or even/odd indices. This will also work.
3. implement a routine `train(X,Y,`λ`)` that
   - learns w from any given X, Y and λ, as specified in Algorithm 1
4. implement a routine `predict(w,x)` that
   - uses the model with parameter vector w to predict outputs for data x: $f(x) = \langle x, w \rangle$.
   - if x is a matrix instead of a vector, make one prediction for each row vector.
5. implement a routine `loss(y,pred)` that
   - computes the loss, i.e. outputs 1 if sign$(pred) \neq y$ and 0 otherwise (tip: *sign*$(pred) \neq y$ *is equivalent to* $pred \cdot y \leq 0$)
   - if y and *pred* are vectors, compute and output the losses for each component.
     Hint: don't use a loop, if you can avoid it! Ideally use only expressions that act componentswise on vectors.
6. use *Xtrn* and *Ytrn* to learn a model (a weight vector w) without regularization ($\lambda = 0$)
7. compute the training error:
   - use the trained model (i.e. `w`) to predict values on *Xtrn*
   - compute the average of losses between the predicted values and the ground truth *Ytrn*
8. compute the validation error:

- use the trained model (i.e. `w`) to predict values on *Xval*
- compute the average of losses between the predicted values and the ground truth *Yval*

9. Compare the training and the validation error. *What do you think you see? Overfitting, underfitting or neither?*

Exercise 2 – Model Selection: Regularization

1. repeat the above steps of training and evaluation for $\lambda \in A$ for $A = \{2^{-20}, 2^{-19}, \ldots, 2^{20}\}$
2. plot a graph with λ on the x-axis (in logarithmic units) and training and validation error on the y axis. *What do you observe?* (if plotting does not work, try to answer this question from the numeric values)
3. write a routine `modelselect(Xtrn,Ytrn,Xval,Yval,A)` that
   - automatizes the above steps and returns the value of λ with smallest validation error and the validation error itself. Print a warning if the largest or smallest value in A is selected.
4. run the `modelselect` routine on the training/validation data and confirm that the output coincides with the location of the minimum in the plot.
5. (optional) identify the 5 validation examples with highest prediction output and the 5 validation examples with lowest prediction output. Find and visualize the corresponding input images (you'll have to convert them from 576-dimensional vectors to 24×24 image for that). *What do you see?*

Exercise 3 – Distribution Mismatch (optional, skip if there isn't enough time)

1. load the data files *"XtrainIMG2.txt"* and *"Ytrain2.txt"*. This data is also images of eyes, but of *right eyes*, whereas for the previous exercises all images were *left eyes*.
2. as above, split the data into a training and a validation part.
3. use the `modelselect` routine to select a regularization constant and learn a model (your results should be similar to the ones in Exercise 2).
4. take the model learned on left eyes and evaluate how well it works on the validation set of right eyes. *What do you observe?*
5. *If you knew in advance that the training data are only left eyes but the test data are right eyes, could you think of a way to overcome the problem? What, if you didn't know what the images contained?*

Exercise 4 – Model Selection: Image Features (optional, skip if there isn't enough time)

1. load the data from files *"XtrainHOG.txt"*, *"XtrainDAISY.txt"* and *"XtrainCNN.txt"*.
   They contain the same training images as in Exercise 1 (so their labels are the same), but represented by *HoG* features, *Daisy* features and features extracted using a pre-trained *ConvNet*.
2. as above, split the data for each feature type into a training and a validation part.
3. runs the `modelselect` script for each of the three feature representations and determine the most promising feature type and regularization hyperparameter (if model selection for *Daisy* is too slow, use just $A = \{10^{-7}, 10^{-6}, \ldots, 10^{7}\}$)
4. with the obtained hyperparameters, retrain a model on all available data of the respective feature type

Exercise 5 – Model Evaluation

1. depending one which model you picked in the previous step, load the test data from the files *"XtestIMG.txt"*, *"XtestHOG.txt"*, *"XtestDAISY.txt"*, or *"XtestCNN.txt"* and the ground truth labels *"Ytest.txt"*.
2. compute the test error of the (one!) selected model
3. write your names, the hyperparameters of the (one!) selected model, and the test error on a piece of paper and give it to one of the organizers. You have now officially finished the exercise session. Enjoy your afternoon!

Exercise 6 – Challenge (optional, to be completed any time before the Thursday lectures)

1. do whatever you like to learn a model from the provided *training* data files (do not use the *test* data files!)
2. load the challenge data from the files *"XchallengeIMG.txt"*, *"XchallengeHOG.txt"*, *"XchallengeDAISY.txt"* and/or *"XchallengeCNN.txt"*.
3. predict outputs for the challenge data and write them to a text file, one value per line
4. send the file with predictions and a description of what you did to `chl@ist.ac.at` with subject line *"VS3 Challenge"*
5. everyone who submits a solution will get free drinks at the barbeque
6. the submission with lowest error will win the challenge!