Overview

This exercise session is about supervised learning algorithms.
In the first part, we will discuss how learning and optimization are related working on Stochastic Gradient Ascent and Gradient Boosting methods. In the second part, we will pass to the Jupyter notebook to see the effects of regularization in the housing prices prediction task. Finally, the third part will be focused on ensemble methods (gradient boosting tree in particular).
You can gain familiarity with training basic machine learning models in Python with `scikit-learn` by completing the preparatory exercises in the `bikes.ipynb` notebook on your own.
Have fun!

1 Learning as Optimization

1. Stochastic Gradient Ascent:

   You are using *stochastic gradient ascent* to approximate the optimal parameter values of a *logistic regression* model with L^2 *regularization* for a given set of training examples. Assume that $x_i = (x_{i,1}, x_{i,2}, x_{i,3})$, i.e. the number of features is 3.

   - Write down the the objective function being optimized.

$$
\begin{aligned}
L &= \sum_{i=1}^{n} (y_i \ln(P(y_i = 1|\mathrm{x}_i, w)) + (1 - y_i) \ln(1 - P(y_i = 1|\mathrm{x}_i, w))) - \lambda \sum_{j=1,2,3} w_j^2 \\
P(y_i = 1|\mathrm{x}_i, w) &= \frac{e^{w_0 + \sum_{j=1,2,3} w_j x_{i,j}}}{1 + e^{w_0 + \sum_{j=1,2,3} w_j x_{i,j}}}
\end{aligned}
\tag{1}
$$

   - Compute the gradient of the objective function and write down the equations for the parameter updates for a given example $x_i = (x_{i,1}, x_{i,2}, x_{i,3})$ with label y_i.

$$
\begin{aligned}
w_0^{t+1} &= w_0^t + \eta(y_i - P(y_i = 1|\mathrm{x}_i, w)) \\
w_j^{t+1} &= w_j^t + \eta(-\lambda w_j^t + x_{i,j}(y_i - P(y_i = 1|\mathrm{x}_i, w)) \text{ for } j \in \{1, 2, 3\}
\end{aligned}
\tag{2}
$$

   - The current estimate of the parameter vector is

$$w = (0.7, 0.3, 0.5, 0.6)$$

     Perform one step of the algorithm to update the weights using the following example:

x_1	x_2	x_3	y
0.8	0.5	0.5	1

     Let λ be equal to 0.1 and use a *learning rate* of 1.

$$
\begin{aligned}
P(y = 1|\mathrm{x}_i, w) &= \frac{e^{0.7 + 0.3 \cdot 0.8 + 0.5 \cdot 0.5 + 0.6 \cdot 0.5}}{1 + e^{0.7 + 0.3 \cdot 0.8 + 0.5 \cdot 0.5 + 0.6 \cdot 0.5}} = 0.8161 \\
w_0^{new} &= 0.7 + (1 - 0.8161) = 0.8839 \\
w_1^{new} &= 0.3 - 0.1 \cdot 0.3 + 0.8 \cdot (1 - 0.8161) = 0.4171 \\
w_2^{new} &= 0.5 - 0.1 \cdot 0.5 + 0.5 \cdot (1 - 0.8161) = 0.5420 \\
w_3^{new} &= 0.6 - 0.1 \cdot 0.6 + 0.5 \cdot (1 - 0.8161) = 0.6320
\end{aligned}
\tag{3}
$$

2. Gradient Boosting: You will now use the *gradient boosting* technique for a classification problem where the goal is to predict the probabilities of the outcome. In this case, you will use the *cross entropy* as loss function:

$$J(Y, F(x)) = -\left[Y \log\left(F(x)\right) + (1 - Y) \log\left(1 - F(x)\right)\right].$$

- Write the formula to derive $F_0(x)$.
  At the step 0, the method simply finds the best constant value:

$$\begin{aligned} F_0(x) &= \arg\min_{\gamma} \sum_{j=1}^{N} J(Y_j, \gamma) = \arg\min_{\gamma} \sum_{j=1}^{N} -\left[Y_j \log\left(\gamma\right) + (1 - Y_j) \log\left(1 - \gamma\right)\right] \\ &= \arg\min_{\gamma} [-P \log(\gamma) - R \log(1 - \gamma)], \end{aligned} \tag{4}$$

  where P is the number of times $Y_j = 1$, and R is the number of times $Y_j = 0$, for $j = 1, \ldots, N$.
- View $F(x_j)$s as parameters and take the derivative.

$$\frac{\partial J}{\partial F(x_j)} = \frac{\partial \sum_{i=1}^{N} -\left[Y_i \log\left(F(x_i)\right) + (1 - Y_i) \log\left(1 - F(x_i)\right)\right]}{\partial F(x_j)} = -\frac{Y_j - F(x_j)}{F(x_j)(1 - F(x_j))} \tag{5}$$

- Assuming that $F_m(x_j)$ at step m is given, derive $F_{m+1}(x_j)$, i.e. the update at step $m + 1$. For the sake of simplicity, you can assume that the step length is equal to 1, and that the update uses only one example, namely x_j.

$$\begin{aligned} h_m(x_j) &= -\frac{\partial J}{\partial F_m(x_j)} = \frac{Y_j - F_m(x_j)}{F_m(x_j)(1 - F_m(x_j))} \\ F_{m+1}(x_j) &= F_m(x_j) + h_m(x_j) = F_m(x_j) + \frac{Y_j - F_m(x_j)}{F_m(x_j)(1 - F_m(x_j))} \end{aligned} \tag{6}$$

- What is the residual at step m?
  The residual is not the negative gradient and can be derived from Eq. 5:

$$Y_j - F_m(x_j) = -\frac{\partial J}{\partial F(x_j)} \times F(x_j)(1 - F(x_j))$$

- Do you observe any potential issue due to the update? If yes, what could you apply to fix it? Advice: take into account the setting we described at the beginning of the question ... and check whether the method respects the restrictions. You can also try to substitute real values and comment on the result.
  Because we want to estimate the class probabilities, the residual should be in $[0, 1]$. However, it is likely that the updated values fall outside this interval, meaning that further work is required. One option could be to simply put the updated values into a logistic function

$$\frac{1}{1 + e^{-F_m}} \in [0, 1].$$

  You can check in python that the model XGBoost[1] requires the loss function being specified and transforms the outputs into probabilities.

[1] `https://xgboost.readthedocs.io/en/latest/python/python_intro.html`

2 Predicting Housing Prices

In this exercise, we will try to find out what influences the prices of residential homes rented in Ames, Iowa.
Download the data `houses.csv` and the notebook `houses.ipynb` from Toledo.

In the notebook, you will evaluate the performance of several linear regression methods. Specifically, the goal is to compare ordinary least squares linear regression with its regularized versions (Lasso and Ridge) in order to understand the differences among these approaches.

3 Ensemble Learning

In this exercise, you will have a chance to implement a gradient tree boosting algorithm for solving regression tasks.

Download the notebook called `gbr.ipynb` from Toledo.

In the notebook, you will find the skeleton code that you will need to complete to get a very naive implementation of gradient tree boosting.

You will then evaluate the performance of your learner, investigate the effects of various learning parameters and compare gradient boosting to other ensemble methods.