## Gradient Descent

Consider the function

$$f(w,b) = -\log\sigma(w+b) - \log\sigma(1.5w+b) - \log\sigma(-2w-b)$$

where $w, b \in \mathbb{R}$ and

$$\sigma(x) = \frac{1}{1+\exp(-x)} = \frac{\exp(x)}{\exp(x)+1}$$

is the sigmoid function. We will encounter objective functions as this one later in a more complex way when we discuss neural networks. The objective function here is actually the one of logistic regression for three data points. You might have learned about logistic regression in another course such as data analytics for engineers.

We try to find the minimum of $f$ with the gradient descent algorithm. In particular, we evaluate various strategies to define a step-size policy. Therefore:

- Implement a function that takes a point $(w,b)$ and returns the the gradient of $f$ at this point.
- Implement a function
  `gradient_descent(f, grad_f, eta, (w_0,b_0), max_iter=100)`
  that performs `max_iter` gradient descent steps

  $$x_{t+1} \leftarrow x_t - \eta(t)\nabla f(x_t)$$

  where $f$ is the function to be minimized, $\nabla f$ returns the gradient (implemented by `grad_f`), $\eta(t)$ returns the step-size at iteration $t$ (implemented by `eta`) and $(w_0, b_0)$ is the starting point (initialization).

Perform 100 iterations, starting at $(w_0, b_0) = (1, 1)$ and return the function value of $f(w_{100}, b_{100})$ and the lowest (best) function value achieved throughout the 100 steps for the step size policies below.

8.0p **1a** Use a constant stepsize strategy: implement a function `eta_const(t,c=0.2)` that returns for each iteration `t` the constant `c` as stepsize. Using this step-size policy, what are the results?

What is the final function value after 100 iterations?

$f(w_{100}, b_{100}) =$ [ ] .

What is the best function value obtained throughout the training process?

$\min_{1\le t\le 100} f(w_t, b_t) =$ [ ]

6.0p **1b** Use a continuously decreasing step-size strategy: implement a function `eta_sqrt(t,c=0.2)` that returns for iteration $t$ the step size $c/\sqrt{t+1}$. Using this step-size policy, what are the results?

What is the final function value after 100 iterations?

$f(w_{100}, b_{100}) =$ [ ]

What is the best function value obtained throughout the training process?

$\min_{1 \le t \le 100} f(w_t, b_t) =$

6.0p **1c** Use a multi-step step-size strategy: implement a function `eta_multistep(t, milestones=[20,50,80], c=0.2, eta_init=0.2)` that returns a step size that is initially set to `eta_init`, but is decayed at each milestone by multiplying it with factor `c`. For example:

$$\text{eta\_multistep}(\text{t}, [20, 50], \text{c} = 0.1, \text{eta\_init} = 1) = \begin{cases} 1, & t < 20 \\ 0.1 & 20 \le t < 50 \\ 0.01 & 50 \le t \end{cases}$$

What is the final function value after 100 iterations?

$f(w_{100}, b_{100}) =$

What is the best function value obtained throughout the training process?

$\min_{1 \le t \le 100} f(w_t, b_t) =$

## Coordinate Descent

Consider the function

$$f(\mathbf{x}) = \frac{1}{2}x_1^4 - x_1 x_2 + x_2^2 + x_2 x_3 + x_3^2$$

We try to find the minimum of $f$ with coordinate descent.

3.0p **2a** Implement for each coordinate $x_i$ ($i \in \{1, 2, 3\}$) a function `argmin_xi(x)` that returns $\arg\min_{x_i} f(x)$. Compute the $\arg\min$ for each coordinate on $\mathbf{x}_0 = (2, 3, 4)$.

$\arg\min_{x_1} f(\mathbf{x}_0) =$

$\arg\min_{x_2} f(\mathbf{x}_0) =$

$\arg\min_{x_3} f(\mathbf{x}_0) =$

9.0p **2b** Implement a function `coordinate_descent(f, argmin, x_0, max_iter=100)` that performs `max_iter` coordinate descent steps, where

- `f` is the function to be minimized (check the function values at each iteration),
- `argmin` is an array of the `argmin_xi` functions for each coordinate, and
- `x_0` is the starting point (initialization).

So, at iteration `t` we have to go through all the coordinates (indexed by `i`, going from the first to the last coordinate index) and update each coordinate with the update rule

`x_t[i] = argmin[i](x_t)`

Starting at `x_0=(1,20,5)`, run your coordinate descent implementation and answer the following questions.

What are the first three coordinate update results (for the first iteration)?

`x_t[0] = argmin[0](x_t)}=` ☐

`x_t[1] = argmin[1](x_t)}=` ☐

`x_t[2] = argmin[2](x_t)}=` ☐

What is the minimizer coordinate descent converges to?

$x^* =$( ☐ , ☐ , ☐ )

## Regression - polynomial features

In the accompanying notebook, the California housing dataset is loaded. This is a regression dataset which poses the task to predict house prices.
Compute the design matrix for this dataset when using a polynomial with degree 2. You can use the function `PolynomialFeatures` to do so (there's a code snippet in the accompanying notebook to help with this). Unfortunately, there is a problem when using polynomial features: the quadratic features explode when the original feature values are in a bigger range. This results in a warning about an ill-conditioned matrix when we try to solve for the regression parameters directly. For this reason, apply first the `StandardScaler` from `sklearn.preprocessing` to the data matrix before computing the design matrix.

6.0p **3a** Compute the regression model minimizing the RSS for the polynomial design matrix. Denote the regression parameters for the following features.

$\beta_{\text{MedInc}} =$ ☐

$\beta_{\text{MedIncAveBedrms}} =$ ☐

$\beta_{\text{HouseAgeAveBedrms}} =$ ☐

6.0p **3b** Compute the regression parameter vector $\beta$ of the ridge regression with the following objective:

$$\min_{\beta} \frac{1}{n}\|y - X\beta\|^2 + \lambda\|\beta\|^2$$

Note that this objective is a bit different from the one presented in the lecture and correspondingly, the solution for $\beta$ looks a bit different than presented in the lecture. However, it's not difficult to

derive the solution for this objective in the same way as it has been done for the objective from the lecture. This objective is more frequently used for penalized regression, since it's more easy to compare the effect of $\lambda$ in comparison to the average fit to the target vector.
Denote the obtained ridge regression parameters when using $\lambda = 0.1$ for the following features.

$\beta_{\text{MedInc}} =$

$\beta_{\text{MedIncAveBedrms}} =$

$\beta_{\text{HouseAgeAveBedrms}} =$

## Bias-var trade off

Consider the following true regression function:

$$f^*(x) = \tan(\pi x)$$

Imagine you fit three regression models on the i.i.d. data samples $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$ and obtain the following models:

$$f_{\mathcal{D}_1}(x) = x + 0.2$$
$$f_{\mathcal{D}_2}(x) = 3x + 0.3$$
$$f_{\mathcal{D}_3}(x) = 5x + 0.1$$

6.0p **4** Compute for $x_0 = 0$ the sample

$bias^2$=

$variance$=

Hint: you may use the corresponding formulas given in the lecture and interpret $\mathbb{E}_{\mathcal{D}}[f_{\mathcal{D}}(x_0)]$ as the mean over the samples listed above at $x_0$.

## Naive Bayes - 20news

In this exercise we use the naive Bayes method for text classification. In the accompanying notebook, the 20Newgroups dataset is loaded for four classes of newsgroups: alt.atheism, comp.graphics, sci.space and talk.politics.guns. The text documents are transformed to a bag of words representation, given by a data matrix $D \in \{0, 1\}^{n \times d}$ where each row represents a document and every column a word. $D$ is an indicator matrix of the words that occur in each document.

3.0p **5a** Compute the class prior probabilities $p(y)$.

$p(y = 0) =$

$p(y = 1) =$

$p(y = 2) =$

4.0p **5b** What are the log-probabilities of the word 'naive' given each class? Use Laplace smoothing with $\alpha = 1e - 5$. Note that the log is in ML as a default the natural logarithm to the base of $e$.

Assuming that $x_{naive}$ denotes the random variable for the feature-word 'naive', compute the following probabilities:

$\log p(x_{naive} = 1 \mid y = 0) =$

$\log p(x_{naive} = 1 \mid y = 1) =$

$\log p(x_{naive} = 1 \mid y = 2) =$

6.0p **5c** Compute the class-conditioned log-probabilities $\log p(x_k, y)$ for each word and class combination. Apply the naive Bayes algorithm to compute the class-conditioned log probabilites for the first document $\mathbf{x}_0$ in the training dataset. Use again the Laplace-smoothing with $\alpha = 1e - 5$.

$\log p(\mathbf{x} = \mathbf{x}_0, y = 0) =$

$\log p(\mathbf{x} = \mathbf{x}_0, y = 1) =$

$\log p(\mathbf{x} = \mathbf{x}_0, y = 2) =$

## Decision Tree - Iris

In the accompanying notebook, the Iris dataset is loaded. We use this classification dataset to get our hands on decision tree computations.

3.0p **6a** What is the Gini impurity of the root node, containing all data points?

$G(\{(\mathbf{x}_i, y_i) \mid 1 \leq i \leq n\}) =$

4.0p **6b** Compute the cost for making the first split at the mean value of the first feature 'sepal length'. That is, the leaves would be $L_0 = \{(\mathbf{x}_i, y_i) \mid x_{i1} \leq 5.84, 1 \leq i \leq n\}$ and $L_1 = \{(\mathbf{x}_i, y_i) \mid x_{i1} > 5.84, 1 \leq i \leq n\}$, where $x_{i1}$ denotes the value of the first feature for datapoint $i$. Use the Gini impurity as the impurity measure.

$cost(L_0, L_1) =$

## Kernel SVM - Digits (+open question)

In the accompanying notebook the digits dataset is loaded. This dataset contains 8x8 pixel images of digits from 0-9. We train a kernel SVM in this exercise to predict the digits.

3.0p **7a** Train an RBF kernel SVM with parameters `gamma=0.0005, C=0.9`. Use the SVC SVM model from sklearn to do so. Train the model on the `D_train` dataset (70-30 split) and test the model on the `D_test` dataset. What is the accuracy of the model on the test data?

ACC =

5.0p **7b** The multiclass SVM from sklearn uses a one-vs-one scheme: one SVM is learned for each combination of two classes. Correspondingly, we can interpret the support vectors based on our knowledge on what happens "under the hood". To that end, explain first how the prediction of a class $y \in \{-1, 1\}$ is determined by the support vectors of that SVM. State the prediction formula for SVMs and explain where we find the support vectors in the formula and how the prediction works depending on the kernel, the support vectors and the learned parameters.

4.0p **7c** Sklearn has a peculiar way to denote the learned support vectors. To understand how this works read section 1.4.1.1 on this website, including the multi-class strategies and answer the following questions for the model obtained in question part a.

How many support vectors are there to distinguish between classes 0 and 1?

12.0p **7d**

1. Explain how you extract the support vectors for the SVM classifying between 0 and 1 from the sklearn model. Include screenshots of your code to make clear how you arrive at your result for Question c.
2. Use the plotting function from the notebook to plot four of the support vectors for each class (four support vectors for class 0 and four for class 1) that are most influential for the SVM discriminating between class 0 and 1. Explain how and why you chose the plotted support vectors.

3. Based on the role that the support vectors have in the prediction, what would you expect what the plotted support vectors look like, or what characteristic they would have? Do you see these characteristics in the plotted support vectors or are you surprised by the result?

6.0p **7e** Use the sklearn function `GridSearchCV` to determine the best combination for the parameters `gamma` and `C` according to a 5-fold cross validation of the SVC SVM with RBG kernel. Train the model on the whole dataset `D`, not just `D_train`. Use as the scoring method the accuracy and set as the candidate parameters `gamma` $\in \{0.0001, 0.0005, 0.001, 0.005\}$and `C` $\in \{0.6, 0.8, 1, 2, 4\}$.

What are the parameters resulting in the highest cross-validated scores?

`gamma` =

`C` =

What is the mean cross-validates accuracy of these parameters?

ACC=

## File Upload

0.0p **8** Upload here the well readable material that you generated to derive the assignment solutions (notebooks, scans of computations, etc.). Please make sure that its easy to read and well structured, such that we can find the corresponding code to each exercise quickly.

Upload

Upload a file with a maximum of 25 MB