The Chinese University of Hong Kong, Shenzhen
SDS · School of Data Science
Andre Milzarek · Fall Semester 2020/21

MDS 6106 – Introduction to Optimization

Final Project

Logistic Regression and Support Vector Machines

The goal of this project is to investigate different optimization models and to utilize minimization methodologies that were introduced in the lecture to solve logistic regression and support vector machine problems for large-scale classification tasks.

Project Description. Classification is the process of assigning elements to one of several classes. There are many real-world applications such as, e.g., deciding if an e-mail is spam or not. Other examples include applications from machine vision or bioinformatics. The goal is to automatically learn how to classify elements based on some given training data.

We focus on binary classification problems with two classes $\mathcal{C}_1$ and $\mathcal{C}_2$. We assume that we have some training data which consists of feature vectors $a_i \in \mathbb{R}^n$, $i \in \{1, 2, \ldots, m\}$ and corresponding class labels $z_i \in \{0, 1\}$ or $b_i = 2z_i - 1 \in \{-1, 1\}$. Here, each label b_i or z_i indicates whether the data point a_i belongs to the class $\mathcal{C}_1$ or $\mathcal{C}_2$.

The training data provides implicit information on how to distinguish the two classes and which elements to assign to which class. Based on this implicit information, we want to train a model that allows to assign new feature vectors correctly to one of the two classes. Therefore, our goal is to choose and learn a parametric model ℓ_θ which allows to separate the two classes. The model parameters θ in ℓ_θ have to be adapted to the training data such that as many feature vectors as possible are classified correctly by our trained model ℓ_θ. After the parameter estimation has been performed, the model can be used to predict the label and class of a new data point $a \in \mathbb{R}^n$ via evaluating the function $\ell_\theta(a)$.

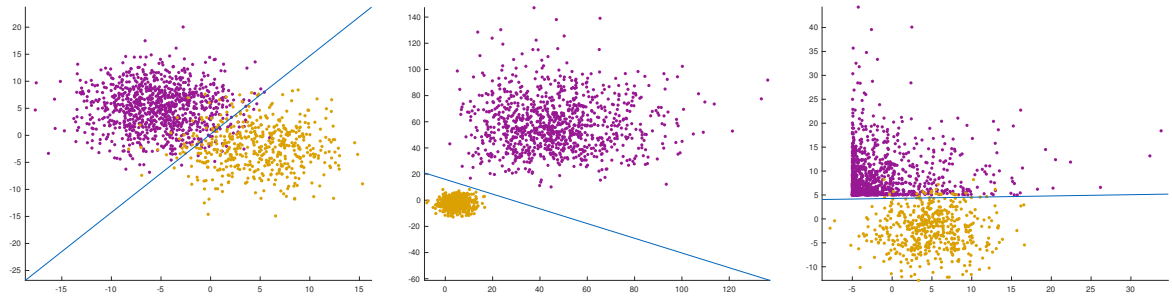

Figure 1: Illustration: Applying support vector machines to separate two-dimensional point clouds. The separating hyperplanes are shown in blue color.

There are different possibilities on how to build such a model ℓ_θ. Support vector machines try to separate the two data classes $\mathcal{C}_1$ and $\mathcal{C}_2$ via a hyperplane, i.e, points on the left side of the hyperplane are modeled to belong to class $\mathcal{C}_1$ while points on the right side of the hyperplane should belong to class $\mathcal{C}_2$ (or vice versa). Hence, $\ell_\theta = \ell_{(x,y)}$ is chosen as a linear function

$$\ell_{(x,y)} : \mathbb{R}^n \to \mathbb{R}, \quad \ell_{(x,y)}(a) := a^\top x + y$$

and our task is to choose the hyperplane parameters $\theta = (x, y) \in \mathbb{R}^n \times \mathbb{R}$ to separate the classes $\mathcal{C}_1$ and $\mathcal{C}_2$ and to represent the given data $(a_i, b_i) \in \mathbb{R}^n \times \{-1, 1\}$ optimally. A new data point a can then be classified via

$$\begin{cases} +1 & \text{if } \ell_{(x,y)}(a) > 0, \\ -1 & \text{if } \ell_{(x,y)}(a) \leq 0 \end{cases} \quad \text{or} \quad \begin{cases} \mathcal{C}_1 & \text{if } \ell_{(x,y)}(a) > 0, \\ \mathcal{C}_2 & \text{if } \ell_{(x,y)}(a) \leq 0 \end{cases}$$

In order to improve the separation of the classes $\mathcal{C}_1$ and $\mathcal{C}_2$ and to reduce potential misclassifications, one typically considers the following hinge-loss formulation

$$\min_{x,y} \ \frac{\lambda}{2}\|x\|^2 + \sum_{i=1}^{m} \max\{0, 1 - b_i(a_i^\top x + y)\} \tag{1}$$

to determine the parameters x and y. Here, $\lambda > 0$ is a regularization parameter that balances the margin and the misclassification error. The images in Figure 1 show several typical classification results using support vector machines. Here, the data consists of two point clouds in $\mathbb{R}^2$ that we want to separate via appropriate hyperplanes.

In this project, we want to study different optimization approaches for the classification model (1) and for an alternative logistic regression model and compare their performance on several real-word data sets and tasks.

Project Tasks.

1. *Data Preparation.* This first part of the project concerns data generation and data preparation for our optimization problems. We want to discuss two categories of data: self-generated data point clouds in the two-dimensional plane and datasets coming from real-world large-scale applications.

   – Given $m_1, m_2 \in \mathbb{N}$, generate the data points $a_1, ..., a_m \in \mathbb{R}^2$, $m = m_1 + m_2$, such that

   $$a_1, \ldots, a_{m_1} \text{ belong to class } \mathcal{C}_1 \quad \text{and} \quad a_{m_1+1}, \ldots, a_{m_1+m_2} \text{ belong to class } \mathcal{C}_2.$$

   In this case, the corresponding data labels satisfy $b_i = +1$ for all $i = 1, ..., m_1$ and $b_i = -1$ for $i = m_1 + 1, ..., m_1 + m_2$. The data points a_i can be generated in many different ways. For instance, let $c_1, c_2 \in \mathbb{R}^2$ be two reference points. Then, we can set

   $$a_i = c_1 + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \end{pmatrix}, \quad \varepsilon_1, \varepsilon_2 \sim N(0, \sigma_1^2), \quad a_j = c_2 + \begin{pmatrix} \delta_1 \\ \delta_2 \end{pmatrix}, \quad \delta_1, \delta_2 \sim N(0, \sigma_2^2)$$

   for all $i = 1, ..., m_1$ and $j = m_1 + 1, ..., m$. Here, ε_i and δ_j are error terms that follow a Normal distribution with zero mean and variance σ_1^2 and σ_2^2, respectively. Data clouds generated by this exemplary model will then cluster around the two reference points c_1 and c_2 and the spread of the data points is controlled by the selected variance.

   Generate a variety of data test sets (at least three–four) for the optimization problems following this outlined strategy. Vary the size of the datasets and their shape (see also Figure 1 for an illustration) to cover interesting situations.

   – On Blackboard, we have provided ten different real-world datasets for binary classification. The datasets are taken from the `LIBSVM` library `https://www.csie.ntu.edu.tw/~cjlin/libsvm/` and are stored in `MATLAB`'s `mat`-format. You can access the data points and labels via

   `load('dataset_train.mat','A'), load('dataset_train_label.mat'),'b').`

data set	m	n	data set	m	n
`a9a`	32561	123	`mushrooms`	8124	112
`breast-cancer`	638	10	`news20`	19996	1355191
`covtype`	581012	54	`phishing`	11055	68
`gisette`	6000	5000	`rcv1`	20242	47236
`ijcnn1`	49990	22	`sido0`	12678	4932

Table 1: A description of the datasets used in the numerical comparison.

An overview of the prepared datasets is given in Table 1. Notice that the ith-row of A here corresponds to $a_i^\top$. Some of the datasets also contain additional testing data for validation of the results. Download the datasets (from BB or the mentioned website) and check if you can access and store the labels and data points in your workspace. Additional tools and different datasets can also be found on the `LIBSVM` website.

2. *Support Vector Machines.* We first consider a smooth variant of the support vector machine problem (1) that was already introduced in the lectures:

$$\min_{x,y} \; f_{\texttt{svm}}(x,y) := \frac{\lambda}{2}\|x\|^2 + \sum_{i=1}^{m} \varphi_+(1 - b_i(a_i^\top x + y)). \tag{2}$$

Here, $\varphi_+(t)$ denotes a Huber-type version of the max-function $\max\{0,t\}$:

$$\varphi_+(t) = \begin{cases} \frac{1}{2\delta}(\max\{0,t\})^2 & \text{if } t \le \delta, \\ t - \frac{\delta}{2} & \text{if } t > \delta. \end{cases}$$

– Implement the basic gradient method with backtracking ($\gamma = 0.1$, $\sigma = 0.5$, and $s = 1$) and the accelerated gradient method (AGM) for the smoothed support vector machine problem (2).

You can either implement the variant of AGM with fixed step size and basic extrapolation strategy

$$\alpha_k = \frac{1}{L}, \quad \beta_k = \frac{t_{k-1} - 1}{t_k}, \quad t_k = \frac{1}{2}(1 + \sqrt{1 + 4t_{k-1}^2}), \quad t_{-1} = t_0 = 1,$$

(calculate and estimate the true Lipschitz constant of $\nabla f_{\texttt{svm}}$ in this case) or you can use the adaptive version that estimates the Lipschitz constant shown in Algorithm 1.

– Implement the globalized BFGS method (Lecture L-11, slide 11) for problem (2). You can use backtracking to perform the line search and you can choose $H_0 = I$ or $H_0 = \rho I$ (for a suitable $\rho > 0$) as initial matrix for the BFGS-updates. In order to guarantee positive definiteness of the BFGS updates, the pair $\{s^k, y^k\}$ should only be added to the current curvature pairs if the condition $(s^k)^\top y^k > 10^{-14}$ is satisfied.

(Notice that here $y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$ denotes the vector within the BFGS-framework and not the bias term in the model $\ell_{(x^k,y^k)}$)).

– Run the three different methods first on several of the synthetic datasets and compare their performance with respect to the number of iterations and the required cpu-time. Plot and compare the convergence w.r.t. the norm of the gradient $\nabla f_{\texttt{svm}}(x^k, y^k)$ or w.r.t. the relative error $|f_{\texttt{svm}}(x^k, y^k) - f^*|/\max\{1, |f^*|\}$ where f^* is the best obtained function value. You can choose $\delta \in [10^{-4}, 10^{-1}]$ and $\lambda = \frac{1}{m}$ or $\lambda = 0.1$ (other suitable choices are also possible).

Algorithm 1: The Accelerated Gradient Method for Unknown Lipschitz Constants

Initialization: Choose $x^0 \in \mathbb{R}^n$, set $x^{-1} = x^0$ and $t_{-1} = t_0 = 1$. Choose $\alpha_{-1} > 0$ and $\eta \in (0,1)$

for $k = 0, 1, 2, ...$ **do**

  Compute the extrapolation parameter $\beta_k = t_k^{-1}(t_{k-1} - 1)$ and set $y^{k+1} = x^k + \beta_k(x^k - x^{k-1})$.

  Set $\alpha_k = \alpha_{k-1}$ and $\bar{x}^{k+1} = y^{k+1} - \alpha_k \nabla f(y^{k+1})$

  while $f(\bar{x}^{k+1}) - f(y^{k+1}) > -\frac{\alpha_k}{2}\|\nabla f(y^{k+1})\|^2$ **do**

    Set $\alpha_k = \eta\alpha_k$ and recompute $\bar{x}^{k+1} = y^{k+1} - \alpha_k \nabla f(y^{k+1})$.

  Set $t_{k+1} = \frac{1}{2} \cdot (1 + \sqrt{1 + 4t_k^2})$ and $x^{k+1} = \bar{x}^{k+1}$.

– Let (x^*, y^*) be a solution returned by one of the algorithms. In order to validate the quality of the found hyperplane $\ell_{(x^*,y^*)}(a) = a^\top x^* + y^*$, a part of the data $a_1, ..., a_m$ is typically reserved for testing. I.e., we split the dataset into two disjoint (and randomly selected) groups $A_{\texttt{train}} = (a_i)_{i \in \mathcal{T}}$ and $A_{\texttt{test}} = (a_i)_{i \in \mathcal{T}^\complement}$ where $\mathcal{T}^\complement = \{1, ..., m\} \backslash \mathcal{T}$ denotes the complement of the index set $\mathcal{T} \subset \{1, ..., m\}$. In the optimization problem (2), we then only access data from the set $A_{\texttt{train}}$. We then validate the obtained result on the test data and define the so-called *testing accuracy* as

$$\tau(x^*, y^*) := 100\% \cdot \frac{\sum_{i \in \mathcal{T}^\complement} |p_i + b_i|}{2|\mathcal{T}^\complement|} \quad \text{where} \quad p_i = \begin{cases} +1 & \text{if } \ell_{(x^*,y^*)}(a_i) > 0 \\ -1 & \text{if } \ell_{(x^*,y^*)}(a_i) \leq 0 \end{cases} \quad \forall\, i \in \mathcal{T}^\complement.$$

The higher the testing accuracy the more data points have been classified correctly by our model $\ell_{(x^*,y^*)}$. Use the testing accuracy to calibrate your model. In particular, save and compute the testing accuracy $\{\tau(x^k, y^k)\}_k$ within the used algorithm and plot $\{\tau(x^k, y^k)\}_k$ for different choices of the model parameters λ and δ. Discuss your results and observations.

Use different datasets for this calibration test. Some of the real-world datasets already provide additional test data which can be used for testing. Otherwise one typically sets $|\mathcal{T}| = 30\% \cdot m$, $|\mathcal{T}| = 50\% \cdot m$, or $|\mathcal{T}| = 60\% \cdot m$ – depending on the size of the dataset.

Hints and Further Guidelines:

– Notice that the data matrix A for the examples given in Table 1 is stored in a `sparse` format (we only save nonzero entries). Make sure that your code does not contain any operations that converts this matrix into a `full` matrix (especially when working with some of the large-scale sets).

– Notice that the efficiency of the BFGS-method will depend on the dimension n since we need to store a full $n \times n$ matrix for the BFGS-updates.

3. *Logistic Regression and L-BFGS.* We now consider a second model for binary classification – the so-called *logistic regression model*. Here, the corresponding optimization problem is given by:

$$\min_{x,y} \; f_{\texttt{log}}(x, y) = \frac{1}{m} \sum_{i=1}^{m} \log(1 + \exp(-b_i \cdot (a_i^\top x + y))) + \frac{\lambda}{2}\|x\|^2, \tag{3}$$

As before the vectors $a_i \in \mathbb{R}^n$, $i = 1, ..., m$ denote the given data points with binary labels $b_i \in \{+1, -1\}$ and $\lambda > 0$ is a model parameter. This methodology is based on a probabilistic idea. Specifically, we try to find a good parametric model of the probability that a given feature vector belongs to the first class $\mathcal{C}_1$. The parameters x, y of the model function then have to be chosen such that the predicted probability is close to 1 for every feature vector

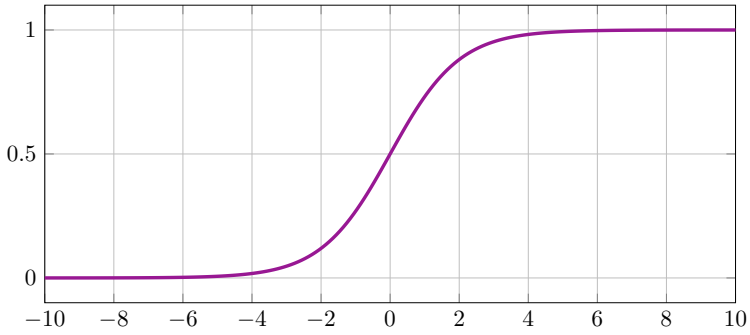

Figure 2: Plot of the sigmoid function σ.

which was assigned to the class $\mathcal{C}_1$. After the training has been finished, the model can be used to classify any feature vector according to the probabilistic estimate.

In the case of logistic regression, the *sigmoid function* $\sigma : \mathbb{R} \to \mathbb{R}$, $\sigma(a) = \frac{1}{1+\exp(-a)}$ is chosen as underlying probability model. A plot of σ is shown in Figure 2. In logistic regression, we want to train the linear model $\ell_{(x,y)}(a) = a^\top x + y$ such that

$$\sigma(\ell_{(x,y)}(a_i)) = \sigma(a_i^\top x + y) \approx \begin{cases} 1 & \text{if } a_i \text{ belongs to class } \mathcal{C}_1, \text{ i.e., } b_i = +1, \\ 0 & \text{if } a_i \text{ belongs to class } \mathcal{C}_2, \text{ i.e., } b_i = -1. \end{cases}$$

A new data point $a \in \mathbb{R}^n$ can then be classified via

$$\begin{cases} +1 & \text{if } \sigma(\ell_{(x,y)}(a)) > \frac{1}{2}, \\ -1 & \text{if } \sigma(\ell_{(x,y)}(a)) \leq \frac{1}{2} \end{cases} \quad \text{or} \quad \begin{cases} \mathcal{C}_1 & \text{if } \sigma(\ell_{(x,y)}(a)) > \frac{1}{2}, \\ \mathcal{C}_2 & \text{if } \sigma(\ell_{(x,y)}(a)) \leq \frac{1}{2}. \end{cases}$$

The optimization problem (3) is based on a corresponding maximum likelihood approach to estimate the optimal model parameters x and y for this probabilistic strategy.

- Solve the optimization problem (3) using either the standard gradient method, the accelerated gradient method, or Newton's method (the globalized variant). The Lipschitz constant of $\nabla f_{\texttt{log}}$ can be calculated explicitly and is given by $L = \frac{1}{4m} \sum_{i=1}^m \|a_i\|^2$.
- Implement the globalized L-BFGS method (Lecture L-11, slide 11 – now with L-BFGS updates) for the smooth logistic loss problem (3). You can use backtracking to perform the line search and the two-loop recursion to calculate the L-BFGS update. You can choose

  $$H_k^0 = \frac{(s^{k-1})^\top y^{k-1}}{\|y^{k-1}\|^2} \cdot I, \quad s^{k-1} = x^k - x^{k-1}, \quad y^{k-1} = \nabla f(x^k) - \nabla f(x^{k-1})$$

  as initial matrix for the update in iteration k. In order to ensure positive definiteness of the L-BFGS update, the pair $\{s^k, y^k\}$ should again only be added to the current curvature pairs if the condition $(s^k)^\top y^k > 10^{-14}$ is satisfied. Suitable choices for the memory parameter are $m_{\texttt{L-BFGS}} \in \{5, 7, 10, 12\}$.
- Run your code for different data sets and report your results. You can use $\lambda = \frac{1}{m}$ for your first experiments.
- Repeat the calibration test mentioned in the second part of the project for the logistic regression and analyze the performance of the logistic loss model. Here, we can use a similar validation test:

  $$\rho(x^*, y^*) := 100\% \cdot \frac{\sum_{i \in \mathcal{T}^\mathsf{C}} |q_i + b_i|}{2|\mathcal{T}^\mathsf{C}|} \quad \text{where} \quad q_i = \begin{cases} +1 & \text{if } \sigma(\ell_{(x^*,y^*)}(a_i)) > \frac{1}{2}, \\ -1 & \text{if } \sigma(\ell_{(x^*,y^*)}(a_i)) \leq \frac{1}{2}, \end{cases}$$

for all $i \in \mathcal{T}^{\mathsf{C}}$ (where $\mathcal{T}^{\mathsf{C}}$ again corresponds to a chosen testing data set $A_{\texttt{test}}$). Plot $\{\rho(x^k, y^k)\}_k$ for different choices of λ. Compare the your results with the ones obtained in part 1.) – does the logistic regression model achieve better results?

Hints and Guidelines:

- For debugging, you can test your implementation of the L-BFGS strategy first on simpler and low-dimensional examples.

4. *Performance, Extensions, and Stochastic Optimization.* In this final part of the project, we try to investigate additional improvements and variants of the algorithms discussed in the second and third part.

   - Based on your numerical experience, is it possible to further improve the performance of your implementation – i.e., by choosing different linesearch strategies, parameters (within the algorithm), or update rules? Try to revise your code and implement one (some) of the algorithms as efficient as possible. Report your changes and adjustments.

     Potential Key-Words and Ideas: Barzilai-Borwein step sizes; compact representation of the L-BFGS update (see, e.g., chapter 7 in Nocedal & Wright: "*Numerical Optimization*"); adjust line-search parameters; minimize the number of data-calls and computations involving the data matrix A; ...

   - Both the support vector machine and the logistic regression problem can be expressed as a so-called empirical risk minimization problem of the form

     $$\min_{x,y} \; f(x,y) = \frac{1}{m} \sum_{i=1}^{m} f_i(x,y),$$

     where each $f_i : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ corresponds to our loss model being evaluated at a certain data point $(a_i, b_i) \in \mathbb{R}^n \times \{-1, +1\}$. If the number of data points m is large or if the dimension $n+1$ of the problem is large, the evaluation of the full gradient $\nabla f(x,y)$ can be very time-consuming and – in some cases – is not even possible. Stochastic gradient methods are based on the idea of using simpler stochastic approximations of the gradient in each iteration. In particular, in each iteration we sample one index i_k from $\{1, ..., m\}$ uniformly at random and perform the update

     $$\begin{pmatrix} x^{k+1} \\ y^{k+1} \end{pmatrix} = \begin{pmatrix} x^k \\ y^k \end{pmatrix} - \alpha_k \nabla f_{i_k}(x^k, y^k).$$

     Convergence of this simple approach can be guaranteed if the step size $\alpha_k = \eta \beta_k$ are diminishing and satisfy $\sum_{k=0}^{\infty} \beta_k = \infty$, $\sum_{k=0}^{\infty} \beta_k^2$, and $\eta > 0$. There are also mini-batch version, where we select a (larger) subset $\mathcal{S}_k \subset \{1, ..., m\}$ at random and set

     $$\begin{pmatrix} x^{k+1} \\ y^{k+1} \end{pmatrix} = \begin{pmatrix} x^k \\ y^k \end{pmatrix} - \frac{\alpha_k}{|\mathcal{S}_k|} \sum_{i \in \mathcal{S}_k} \nabla f_i(x^k, y^k).$$

     Get familiar with such type of stochastic approach and implement the outlined scheme for one of the models. Investigate the behavior of the stochastic gradient method on large-scale data sets and test different choices of β_k. Compare the performance with other deterministic approaches.

This part of the project is more open and optional and not all of the mentioned points need to be addressed. In particular, other extensions and discussions are possible. Add comments if you have already improved your code and implementations while working on part 1 or 2.

Project Report and Presentation. This project is designed for groups of four (or five) students. Please send the following information to `217012017@link.cuhk.edu.cn` until **December, 23th, 11:00 pm**:

- Name and student ID number of the participating students in your group, group name.

Please contact the instructor in case your group is smaller to allow adjustments of the project outline and requirements.

A report should be written to summarize the project and to collect and present your different results. The report itself should take no more than 15–20 typed pages plus a possible additional appendix. It should cover the following topics and items:

- What is the project about?
- What have you done in the project? Which algorithmic components have you chosen to implement? What are your main contributions?
- Summarize your main results and observations.
- Describe your conclusions about the different problems and methods that you have studied.

You can organize your report following the outlined structure in this project description. As the different parts in this project only depend very loosely on each other, you can choose to distribute the different tasks and parts among the group members. Please clearly indicate the responsibilities and contributions of each student and mention if you have received help from other groups, the teaching assistant, or the instructor.

Try to be brief, precise and fact-based. Main results should be presented by highly condensed and organized data and not just piles of raw data. To support your summary and conclusions, you can put more selected and organized data into an appendix which is not counted in the page limit. Please use a cover page that shows the names and student ID numbers of your group members.

The deadline for the report submission is **December, **th, **:00 pm**. Please send your report and supporting materials to `217012017@link.cuhk.edu.cn`.

The individual presentations of the project are scheduled for **December, **th**. More information will follow here soon.