Convex Optimization for Machine Learning and Computer Vision

Lecture: Dr. Virginia Estellers
Exercises: Emanuel Laude
Winter Semester 2017/18

Computer Vision Group
Institut für Informatik
Technische Universität München

Weekly Exercises 4

Room: 02.09.023
Friday, 24.11.2017, 09:15-11:00
Submission deadline: Monday, 20.11.2017, 10:15, Room 02.09.023

Theory: Proximal operators and Projections (8 Points)

Exercise 1 (4 Points)**.** Let $A \in \mathbb{R}^{n \times n}$ be orthonormal, meaning that $A^\top A = AA^\top = I$. Let the convex set C be given as

$$C := \{u \in \mathbb{R}^n : \|Au\|_\infty \leq 1\}.$$

Compute a formula for the projection onto C given as

$$\Pi_C(v) := \operatorname{argmin}_{u \in \mathbb{R}^n} \frac{1}{2}\|u - v\|_2^2, \quad \text{s.t. } u \in C.$$

Hint: Show that the ℓ_2-norm of a vector is invariant under a multiplication with an orthonormal matrix A, meaning that $\|u\|_2 = \|Au\|_2$.

Exercise 2 (4 Points)**.** Let $f \in \mathbb{R}^n$. Show that the ℓ_1-norm proximity operator of f defined as the solution u of the convex optimization problem

$$\arg\min_{u \in \mathbb{R}^n} \frac{1}{2\lambda}\|u - f\|^2 + \|u\|_1,$$

is given as

$$u \in \mathbb{R}^n, \quad u_i := \begin{cases} f_i + \lambda & \text{if } f_i < -\lambda \\ 0 & \text{if } f_i \in [-\lambda, \lambda] \\ f_i - \lambda & \text{if } f_i > \lambda. \end{cases}$$

Hint: Note that the above optimization problem is decoupled in the sense that one can look for the individual entries u_i of the optimal u separately.

Multinomial Logistic Regression (16 Points)

Exercise 3 (16 Points)**.** In this exercise you are asked to train a linear model for a multiclass classification task with Logistic regression. The idea is as follows: You

are given a set of training samples $\mathcal{I} = \{1, \ldots, N\}$ that are represented by their feature vectors $x_i \in \mathbb{R}^d$, for $i \in \mathcal{I}$. Each training sample i is associated with a class label $y_i \in \{1, \ldots, C\}$. The aim is to estimate a linear classifier parameterized by $W^* \in \mathbb{R}^{d \times C}, b^* \in \mathbb{R}^C$ so that $y_i = \text{argmax}_{1 \leq j \leq C} \, x_i^\top W_j^* + b_j^*$ for most training samples i. Once you have obtained this "optimal" classifier the hope is, that you are able to classify new unseen and unlabeled samples $x \in \mathbb{R}^d$. In machine learning this is called generalization. For this task you may query your trained model via the classifier rule

$$y = \text{argmax}_{1 \leq j \leq C} \, x^\top W_j^* + b_j^* \tag{1}$$

and y probably is the true class label of x if your model generalizes well.

In order to estimate the model we solve an optimization problem of the form

$$\min_{W \in \mathbb{R}^{d \times C}, b \in \mathbb{R}^C} \; \frac{1}{N} \sum_{i=1}^{N} \ell(W, b, x_i, y_i) + \frac{\lambda_1}{2} \|W\|_2^2 + \frac{\lambda_1}{2} \|b\|_2^2, \tag{2}$$

where

$$\ell(W, b, x_i, y_i) = -\log \left(\frac{\exp(\langle W_{y_i}, x_i \rangle + b_{y_i})}{\sum_{j=1}^{C} \exp(\langle W_j, x_i \rangle + b_j)} \right) \tag{3}$$

is called the softmax loss. Note that the above problem is smooth and strongly convex and can be solved with gradient descent. In practice however, it may happen, that some features (i.e. components of the vector x_i) do not contain any information about the true class labels, i.e. components that are just noise. In order to filter out the useless features we add the nonsmooth sparsity inducing ℓ_1-norm term on W. So overall we would like to optimize

$$\min_{W \in \mathbb{R}^{d \times C}, b \in \mathbb{R}^C} \; \frac{1}{N} \sum_{i=1}^{N} \ell(W, b, x_i, y_i) + \frac{\lambda_1}{2} \|W\|_2^2 + \frac{\lambda_1}{2} \|b\|_2^2 + \lambda_2 \|W\|_1. \tag{4}$$

You are asked to do the following:

- Download the toy data template from the homepage
- Implement a proximal gradient descent algorithm to optimize the above objective (Avoid for-loops)
- Make sure that your objective monotonically decreases. Plot the objective values. Stop your code if the difference of two successive iterates is less than 10^{-12}.
- In order to ensure that your derivative is computed correctly you may first optimize the fully differentiable model (2) with MATLABs *fminunc* with the options *'GradObj'*, *'On'* and *'DerivativeCheck'*, *'On'*.

- Iteratively compute the test error in percent, i.e. how many test samples are not classified correctly via the rule (1).
- Play around with different parameter settings for λ_1, λ_2. What do you observe? Can you identify the useless features? Explain why the model generalizes better to unseen test data if you add a sparsity inducing term.
- You may apply your code to the MNIST dataset `http://yann.lecun.com/exdb/mnist/` and see that your are now able to classify handwritten digits.