Labs
Machine Learning Course
Fall 2016

EPFL
School of Computer and Communication Sciences
Martin Jaggi & Rüdiger Urbanke
mlo.epfl.ch/page-136795.html
epfmlcourse@gmail.com

Problem Set 7, Nov 3, 2016 (Theory Questions, SVM)

1 Convexity

Recall that we say that a function f is *convex* if the domain of f is a convex set and

$$f(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leq \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y}), \text{ for all } \boldsymbol{x}, \boldsymbol{y} \text{ in the domain of } f,\ 0 \leq \theta \leq 1.$$

And *strictly convex* if

$$f(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) < \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y}), \text{ for all } \boldsymbol{x} \neq \boldsymbol{y} \text{ in the domain of } f,\ 0 < \theta < 1.$$

Prove the following assertions.

1. The affine function $f(x) = ax + b$ is convex, where a, b and x are scalars.
2. If multiple functions $f_n(\boldsymbol{x})$ are convex over a fixed domain, then their sum $g(\boldsymbol{x}) = \sum_n f_n(\boldsymbol{x})$ is convex over the same domain.
3. Take $f, g : \mathbb{R} \to \mathbb{R}$ to be convex functions and g to be increasing. Then $g \circ f$ is also convex.
   Note: A function g is increasing if $a \geq b \Leftrightarrow g(a) \geq g(b)$. An example of a convex and increasing function is $\exp(x), x \in \mathbb{R}$.
4. If $f : \mathbb{R} \to \mathbb{R}$ is convex, then $g : \mathbb{R}^D \to \mathbb{R}$, where $g(\boldsymbol{x}) := f(\boldsymbol{w}^\top \boldsymbol{x} + b)$, is also convex. Here, $\boldsymbol{w}$ is a constant vector in $\mathbb{R}^D$, b is a constant in $\mathbb{R}$ and $\boldsymbol{x} \in \mathbb{R}^D$.
5. Let $f : \mathbb{R}^D \to \mathbb{R}$ be strictly convex. Let $\boldsymbol{x}^\star \in \mathbb{R}^D$ be a global minimizer of f. Show that this global minimizer is unique. Hint: Do a proof by contradiction.

2 Extension of Logistic Regression to Multi-Class Classification

Suppose we have a classification dataset with N pairs $\{y_n, \boldsymbol{x}_n\}$, $n \in [1, N]$, and y_n is a categorical variable over K categories, $y_n \in \{1, 2, ..., K\}$. We wish to fit a linear model in a similar spirit to logistic regression, and we will use the softmax function to link the linear inputs to the categorical output, instead of the logistic function.

We will have K sets of parameters $\boldsymbol{w}_k$, and define $\eta_{nk} = \boldsymbol{w}_k^\top \boldsymbol{x}_n$ and compute the probability distribution of the output as follows,

$$\mathbb{P}[y_n = k \,|\, \boldsymbol{x}_n, \boldsymbol{w}_1, ..., \boldsymbol{w}_K] = \frac{\exp(\eta_{nk})}{\sum_{j=1}^K \exp(\eta_{nj})}.$$

Note that we indeed have a probability distribution, as $\sum_{k=1}^K \mathbb{P}[y_n = k \,|\, \boldsymbol{x}_n, \boldsymbol{w}_1, ..., \boldsymbol{w}_K] = 1$. To make the model *identifiable*, we will fix $\boldsymbol{w}_K$ to $\mathbf{0}$, which means we have $K-1$ sets of parameters to learn. As in logistic regression, we will assume that each y_n is i.i.d., i.e.,

$$\mathbb{P}[\boldsymbol{y} \,|\, \mathbf{X}, \boldsymbol{w}_1, ..., \boldsymbol{w}_K] = \prod_{n=1}^N \mathbb{P}[y_n \,|\, \boldsymbol{x}_n, \boldsymbol{w}_1, ..., \boldsymbol{w}_K].$$

1. Derive the log-likelihood for this model.
2. Derive the gradient with respect to each $\boldsymbol{w}_k$.
3. Show that the negative of the log-likelihood is convex with respect to $\boldsymbol{w}_k$.

3 Support Vector Machines and Coordinate Descent

The original optimization problem for the Support Vector Machine (SVM) is given by

$$\min_{\boldsymbol{w}\in\mathbb{R}^D} \sum_{n=1}^{N} \ell(y_n \boldsymbol{x}_n^\top \boldsymbol{w}) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \tag{1}$$

where $\ell : \mathbb{R} \to \mathbb{R}$, $\ell(z) := \max\{0, 1-z\}$ is the *hinge loss* function. Here for any n, $1 \le n \le N$, the vector $\boldsymbol{x}_n \in \mathbb{R}^D$ is the n^{th} data example, and $y_n \in \{\pm 1\}$ is the corresponding label.

The dual optimization problem for the SVM is given by

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^N} \boldsymbol{\alpha}^\top \mathbf{1} - \tfrac{1}{2\lambda}\boldsymbol{\alpha}^\top \boldsymbol{Y}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{Y}\boldsymbol{\alpha} \quad \text{such that} \quad 0 \le \alpha_n \le 1 \ \forall n \tag{2}$$

where $\boldsymbol{Y} := \text{diag}(\boldsymbol{y})$, and $\boldsymbol{X} \in \mathbb{R}^{D\times N}$ again collects all N data examples as its columns.

Problem 1 (SGD for SVM):

Implement stochastic gradient descent (SGD) for the original SVM formulation (1). That is in every iteration, pick one data example $n \in [N]$ uniformly at random, and perform an update on $\boldsymbol{w}$ based on the (sub)gradient of the n^{th} summand of the objective (1). Then iterate by picking the next n.

Problem 2 (Coordinate Descent for SVM):

Derive the coordinate descent algorithm updates for the dual (2) of the SVM formulation. That is in every iteration, pick a coordinate $n \in [N]$ uniformly at random, and optimize the objective (2) with respect to that coordinate alone. After updating that coordinate α_n, update the corresponding primal vector $\boldsymbol{w}$ such that the first-order correspondence is maintained, that is that always $\boldsymbol{w} = \boldsymbol{w}(\boldsymbol{\alpha}) := \frac{1}{\lambda}\boldsymbol{X}\boldsymbol{Y}\boldsymbol{\alpha}$. Then iterate by picking the next coordinate n.

1. Mathematically derive the coordinate update for one coordinate n (finding the closed-form solution to maximization over just that coordinate), when given $\boldsymbol{\alpha}$ and corresponding $\boldsymbol{w}$.

2. Implement the coordinate descent (here ascent) algorithm in Python, and compare to your SGD implementation. Which one is faster? (Compare the training objective values (1) for the $\boldsymbol{w}$ iterates you obtain from each method).