Labs
Machine Learning Course
Fall 2019

EPFL
School of Computer and Communication Sciences
Martin Jaggi & Rüdiger Urbanke
www.epfl.ch/labs/mlo/machine-learning-cs-433

Problem Set 7, Oct 31, 2019 (SVM)

Goals. The goal of this exercise is to

- Implement and debug Support Vector Machine (SVM) using SGD and coordinate descent.
- Derive updates for the coordinate descent algorithm for the dual optimization problem for SVM.
- Implement and debug the coordinate descent algorithm.
- Compare it to the primal solution.

Setup, data and sample code. Obtain the folder `labs/ex07` of the course github repository

github.com/epfml/ML_course

We will finally depart from using the height-weight dataset and instead use the larger CERN dataset from Project 1 in this exercise. We have provided sample code templates that already contain useful snippets of code required for this exercise.

1 Support Vector Machines using SGD

Until now we have implemented linear and logistic regression to do classification. In this exercise we will use the Support Vector Machine (SVM) for classification. As we have seen in the lecture notes, 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, $1 \leq n \leq N$, 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.

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.

1. Fill in the notebook functions `calculate_accuracy(y, X, w)` which computes the accuracy on the training/test dataset for any $\boldsymbol{w}$ and `calculate_primal_objective(y, X, w, lambda_)` which computes the total primal objective (1).
2. Derive the SGD updates for the original SVM formulation and fill in the notebook function `calculate_stochastic_gradient()` which should return the stochastic gradient of the total cost function (loss plus regularizer) with respect to $\boldsymbol{w}$. Finally, use `sgd_for_svm_demo()` provided in the template for training.

2 Support Vector Machines using Coordinate Descent

As seen in class, another approach to train SVMs is by considering the dual optimization problem given by

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

where $\boldsymbol{Y} := \text{diag}(\boldsymbol{y})$, and $\boldsymbol{X} \in \mathbb{R}^{N\times D}$ again collects all N data examples as its rows, as usual. In this approach we optimize over the dual variables $\boldsymbol{\alpha}$ and map the solutions back to the primal vector $\boldsymbol{w}$.

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 fully 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 $\boldsymbol{w} = \boldsymbol{w}(\boldsymbol{\alpha}) := \frac{1}{\lambda}\boldsymbol{X}^\top \boldsymbol{Y}\boldsymbol{\alpha}$. 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. Fill in the notebook functions `calculate_coordinate_update()` which should compute the coordinate update for a single desired coordinate and `calculate_dual_objective()` which should return the objective (loss) for the dual problem (2) .
3. Finally train your model using coordinate descent (here ascent) using the given function `sgd_for_svm_demo()` in the template. 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).

Theory Excercises

Problem 3 (Kernels):

In class we have seen that many kernel functions $k(\boldsymbol{x}, \boldsymbol{x}')$ can be written as inner products $\phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}')$ for a suitably chosen feature map $\phi(\cdot)$. Let us say that such a kernel function is *valid*. We further discussed many operations on valid kernel functions that result again in valid kernel functions. Here are two more.

1. Let $k_1(\boldsymbol{x}, \boldsymbol{x}')$ be a valid kernel function. Let f be a polynomial with positive coefficients. Show that $k(\boldsymbol{x}', \boldsymbol{x}') = f(k_1(\boldsymbol{x}, \boldsymbol{x}'))$ is a valid kernel.
2. Show that $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(k_1(\boldsymbol{x}, \boldsymbol{x}')$ is a valid kernel assuming that $k_1(\boldsymbol{x}, \boldsymbol{x}')$ is a valid kernel. HINT: You are allowed to take limits.

$$\underset{\alpha}{\text{maximize}}\ f(\alpha) = \alpha^T 1 - \frac{1}{2}\alpha^T Q \alpha$$

subject to $\alpha \in [0,1]^N$

→ for one coordinate:

$$\underset{\gamma \in \mathbb{R}}{\text{maximize}}\ f(\alpha + \gamma e_n) \quad \text{subject to } \alpha_n + \gamma \leq 1$$

$e_n = [0, \ldots, \underset{\uparrow}{1}, \ldots, 0]^T$

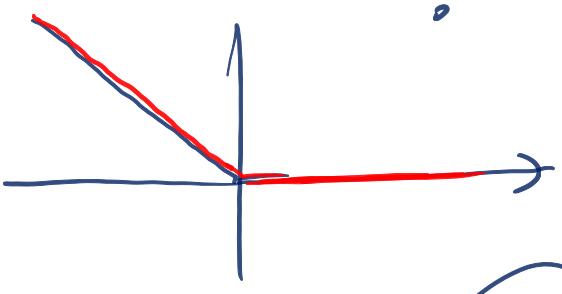
n-th position.

Note: $\gamma = 0 \iff \nabla_n^P f(\alpha) = 0$ where ∇_n denotes the n-th component of the gradient.

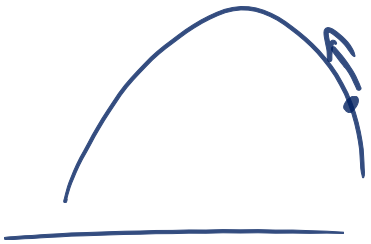
P stands for projected.

$$\nabla_n^P f(\alpha) = \begin{cases} \nabla_n f(\alpha) & \text{if } 0 < \alpha_n < 1 \\ \min\{0, \nabla_n f(\alpha)\} & \text{if } \alpha_n = 0 \\ \max\{0, \nabla_n f(\alpha)\} & \text{if } \alpha_n = 1 \end{cases}$$

$\alpha_n = 0$? ?



$\alpha_n = 0$ ✓



$$\nabla f(\alpha) = 1 - \frac{1}{2\lambda}(Q + Q^T)\alpha$$

$$= 1 - \frac{1}{\lambda} Q\alpha.$$

$$\nabla_n f(\alpha) = 1 - \frac{1}{\lambda} e_n^T Q\alpha.$$

$$f(\alpha + \gamma e_n) = (\alpha + \gamma e_n)^T 1 - \frac{1}{2\lambda}(\alpha + \gamma e_n)^T Q(\alpha + \gamma e_n)$$

$$= \alpha^T + \gamma e_n^T - \frac{1}{2\lambda}(\alpha^T + \gamma e_n^T)(Q\alpha + Q\gamma e_n)$$

$$= \alpha^T + \gamma e_n^T - \frac{1}{2\lambda}(\alpha^T Q\alpha + \alpha^T Q\gamma e_n + \gamma e_n^T Q\alpha + \gamma^2 e_n^T Q e_n)$$

$$= \alpha^T - \frac{1}{2\lambda}\alpha^T Q\alpha + \gamma e_n^T - \frac{1}{2\lambda}(\alpha^T Q\gamma e_n + \gamma e_n^T Q\alpha + \gamma^2 e_n^T Q e_n)$$

$$= f(\alpha) + \gamma e_n^T - \frac{1}{2\lambda}\gamma^2 Q_{nn} - \frac{1}{2\lambda}\alpha^T Q\gamma e_n - \frac{1}{2\lambda}\gamma e_n^T Q\alpha$$

$$= f(\alpha) - \frac{1}{2\lambda}\gamma^2 Q_{nn} + \gamma\left(1 - \frac{1}{2\lambda}\alpha^T Q e_n - \frac{1}{2\lambda} e_n^T Q\alpha\right)$$

1/ $k(x,x')$ kernel ?

is $k(x,x') = f(k_1(x,x'))$ a valid kernel ?

$$f(k_1(x,x')) = f(\phi^T(x)\,\phi(x'))$$

$$= \sum_{i=0}^{n} \alpha_i (\phi^T(x)\,\phi(x'))^i$$

$\phi^T(x)\,\phi(x')$ is a valid kernel (assumption)

$(\phi^T(x)\,\phi(x'))^i$ is a valid kernel. product of valid kernel is a valid kernel.

sum of valid kernels is a valid kernel.

$k(x,x')$ is a valid kernel. ✓

2/ $\exp(k_1(x,x'))$ is a valid kernel ?

$$\exp(x) = \lim_{n\to+\infty} \sum_{i=0}^{n} \frac{x^i}{i!}$$ of positive coefficients

from 1 every polynomial of a valid kernel is a valid kernel $\Rightarrow$ $\sum_{i=0}^{n} \frac{x^i}{i!}$ is valid.

the limit of a valid kernel is a valid kernel. ✓