Labs
Machine Learning Course
Fall 2025

Westlake University
Department of Artificial Intelligence, SOE
Tao Lin
https://github.com/LINs-lab/course_machine_learning

Problem Set 6, Nov 19, 2025 (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/lab06` of the course github repository

github.com/LINs-lab/cours_machine_learning

We will finally depart from using the height-weight dataset and instead use a toy dataset from scikit-learn. We have provided sample code templates that already contain useful snippets of code required for this exercise.

Additional theory questions. After the main SVM exercises, you will find some additional theory questions on convexity and interesting extensions of linear and logistic regression, which were the focus of previous labs.

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

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

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 $\mathbf{X}_n \in \mathbb{R}^D$ 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 $\mathbf{w}$ 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 $\mathbf{w}$ 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 $\mathbf{w}$. 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

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

where $\mathbf{1}$ is the vector of size N filled with ones, $\mathbf{Y} := \mathrm{diag}(\mathbf{y})$, and $\mathbf{X} \in \mathbb{R}^{N\times D}$ 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 $\mathbf{w}$ via the mapping we have seen in class: $\mathbf{w}(\boldsymbol{\alpha}) = \frac{1}{\lambda N}\mathbf{X}^T\mathbf{Y}\boldsymbol{\alpha}$.

Problem 2 (Coordinate Descent for SVM):

In this part we will derive the coordinate descent (or rather ascent in our case) algorithm seen in class to solve the dual (2) of the SVM formulation. That is, at every iteration, we pick a coordinate $n \in [N]$ uniformly at random, and fully optimize the objective (2) **with respect to that coordinate alone**. Hence one step of coordinate ascent corresponds to solving, for a given coordinate $n \in [N]$ and our current vector $\boldsymbol{\alpha} \in [0,1]^N$, the one dimensional problem:

$$\max_{\gamma\in\mathbb{R}} f(\boldsymbol{\alpha} + \gamma\mathbf{e}_n) \qquad \text{such that} \quad 0 \le \alpha_n + \gamma \le 1 \tag{3}$$

where $f(\boldsymbol{\alpha}) = \frac{1}{N}\boldsymbol{\alpha}^\top\mathbf{1} - \frac{1}{2\lambda N^2}\boldsymbol{\alpha}^\top\mathbf{Y}\mathbf{X}\mathbf{X}^\top\mathbf{Y}\boldsymbol{\alpha}$ and $\mathbf{e}_n = [0,\cdots,1,\cdots,0]^\top$ (all zero vector except at the n^{th} position). We denote by γ^* the value of γ which maximises problem 3. The coordinate update is then $\boldsymbol{\alpha}^{\text{new}} = \boldsymbol{\alpha} + \gamma^*\mathbf{e}_n$.

1. Solve problem 3 and give a closed form solution for the update $\boldsymbol{\alpha}^{\text{new}} = \boldsymbol{\alpha} + \gamma^*\mathbf{e}_n$, this update should only involve $\boldsymbol{\alpha}$, λ, N, $\mathbf{X}_n$, y_n and $\mathbf{w}(\alpha)$ (*Hint:* Notice that $\gamma \mapsto f(\alpha + \gamma\mathbf{e}_n)$ is polynomial and **don't forget the constraints !**). Convince yourself that this update can be computed in $O(D)$ time.
2. Find an efficient way of updating the corresponding $\mathbf{w}(\boldsymbol{\alpha}^{\text{new}})$. This should be computable in $O(D)$ time.
3. 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) .
4. 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 $\mathbf{w}$ iterates you obtain from each method). Is the gap going to 0?

Additional Theory Exercises

Convexity

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

$$f(\theta\mathbf{X} + (1-\theta)\mathbf{y}) \le \theta f(\mathbf{X}) + (1-\theta)f(\mathbf{y}), \text{ for all } \mathbf{X},\mathbf{y} \text{ in the domain of } f,\ 0 \le \theta \le 1.$$

And *strictly convex* if

$$f(\theta\mathbf{X} + (1-\theta)\mathbf{y}) < \theta f(\mathbf{X}) + (1-\theta)f(\mathbf{y}), \text{ for all } \mathbf{X} \ne \mathbf{y} \text{ in the domain of } f,\ 0 < \theta < 1.$$

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 $f_n(\mathbf{X})$ are convex over a fixed domain, then their sum $g(\mathbf{X}) = \sum_n f_n(\mathbf{X})$ 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 the function $g \circ f$ defined as $(g \circ f)(x) = g(f(x))$ 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 $g(\mathbf{X}) := f(\mathbf{w}^\top \mathbf{X} + b)$, is also convex. Here, $\mathbf{w}$ is a constant vector in $\mathbb{R}^D$, b is a constant in $\mathbb{R}$ and $\mathbf{X} \in \mathbb{R}^D$.

5. Let $f : \mathbb{R}^D \to \mathbb{R}$ be strictly convex. Let $\mathbf{X}^\star \in \mathbb{R}^D$ be a global minimizer of f. Show that this global minimizer is unique. Hint: Do a proof by contradiction.

Extension of Logistic Regression to Multi-Class Classification

Suppose we have a classification dataset with N data example pairs $\{\mathbf{X}_n, y_n\}$, $n \in [1, N]$, 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 $\mathbf{w}_k$, and define $\eta_{nk} = \mathbf{w}_k^\top \mathbf{X}_n$ and compute the probability distribution of the output as follows,

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

Note that we indeed have a probability distribution, as $\sum_{k=1}^{K} \mathbb{P}[y_n = k \,|\, \mathbf{X}_n, \mathbf{w}_1, ..., \mathbf{w}_K] = 1$. To make the model *identifiable*, we will fix $\mathbf{w}_K$ 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.,

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

1. Derive the log-likelihood for this model.
   Hint: It might be helpful to use the indicator function $1_{y_n=k}$, that is equal to one if $y_n = k$ and 0 otherwise

2. Derive the gradient with respect to each $\mathbf{w}_k$.

3. Show that the negative of the log-likelihood is jointly convex in $\mathbf{w}_1, \ldots, \mathbf{w}_K$.
   Hint: you can use Hölder's inequality:

$$\sum_k |x_k y_k| \leq \left(\sum_k |x_k|^p\right)^{\frac{1}{p}} \left(\sum_k |y_k|^q\right)^{\frac{1}{q}},$$

   where $\frac{1}{p} + \frac{1}{q} = 1$.

Mixture of Linear Regression

If you have a regression dataset with two or more distinct clusters, a mixture of linear regression models is preferred over just one linear regression model.

Consider a regression dataset with N pairs $\{y_n, \mathbf{X}_n\}$. Similar to Gaussian mixture model (GMM), let $r_n \in \{1, 2, \ldots, K\}$ index the mixture component. The distribution of the output y_n under the kth linear model is defined as follows:

$$p(y_n | \mathbf{X}_n, r_n = k, \mathbf{w}) := \mathcal{N}(y_n | \mathbf{w}_k^\top \tilde{\mathbf{X}}_n, \sigma^2)$$

Here, $\mathbf{w}_k$ is the regression parameter vector for the kth model with $\mathbf{w}$ being a vector containing all $\mathbf{w}_k$. Also, denote $\tilde{\mathbf{X}}_n := [1, \mathbf{X}_n^\top]^\top$.

1. Define $\mathbf{r}_n$ to be a binary vector of length K such that all the entries are 0 except the kth entry, i.e., $r_{nk} = 1$, implying that $\mathbf{X}_n$ is assigned to the kth mixture. Rewrite the likelihood $p(y_n | \mathbf{X}_n, \mathbf{w}, \mathbf{r}_n)$ in terms of r_{nk}.

2. Write the expression for the joint distribution $p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \mathbf{r})$ where $\mathbf{r}$ is the set of all $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N$.

3. Assume that r_n follows a multinomial distribution $p(r_n = k|\boldsymbol{\pi}) = \pi_k$, with $\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_K]$. Derive the marginal distribution $p(y_n|\mathbf{X}_n, \mathbf{w}, \boldsymbol{\pi})$ obtained after marginalizing r_n out.

4. Write the expression for the maximum likelihood estimator $\mathcal{L}(\mathbf{w}, \boldsymbol{\pi}) := -\log p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \boldsymbol{\pi})$ in terms of data $\mathbf{y}$ and $\mathbf{X}$, and parameters $\mathbf{w}$ and $\boldsymbol{\pi}$.

5. (a) Is $\mathcal{L}$ jointly convex with respect to $\mathbf{w}$ and $\boldsymbol{\pi}$?
(b) Is the model identifiable? That is, can the MLE estimator return the true parameters $\mathbf{w}$ and $\boldsymbol{\pi}$, assuming we have infinitely many samples.
Prove your answers.