Support Vector Machine : Mathematical Development K-Fold Cross Validation

Sanchayan Bhunia (4849650)

University of Genova

4849650@studenti.unige.it

June 21, 2020

Instructors: Dr. Nicoletta Noceti and Prof. Lorenzo Rosasco

Università di Genova

Overview of the project

- Mathematical modelling of the problem.
  - The problem setup.
  - Choice of loss function.
  - Choice of Cost function.
  - Mathematical Modeling of SVM.
  - Algorithm design.
- Implementing the Model in python in Jupyter Notebook.
- K-Fold Cross Validation algorithm implementation for SVM and KNN in Jupyter Notebook.
- Comparison between KNN and SVM on the same data.

The Problem Set-up

Let say that we have an input space X which is a vector space could be a function, $\in \mathbb{R}^d$.
And an output space Y which is a structured object $y \in \{+1, -1\}$.

$$\exists f_* : X \to Y \text{ But that's unknown.}$$

Our job is to estimate f_* given (X, Y).

In order to do that we will choose a function f from our function space $\mathcal{F}$
The Loss Function :
Define a point-wise loss function ℓ which will calculate loss on every point for choosing f instead of f_*.

$$\ell(y, f(x)) : Y \times \mathbb{R} \to (0, \infty] \tag{1}$$

Input Output Probability Distribution

$$P(x, y) \approx P_X P(y|x) \tag{2}$$

Where, P_X is the Marginal Probability irrespective of X which is a way to express uncertainty in our way of sampling the data or the data-set is incomplete.
$P(y|x)$ is the conditional probability of the output given we choose x as input. Which essentially says that every input might not return the same output every single time, rather can output values from $P(y|x)$ distribution.
Classification:

$$P(y|x) = \{P(+1|x), P(-1|x)\} \tag{3}$$

Noise in the classification $\Delta\delta$ refers to the situations when we have *confusion in labeling the class*.

$$\Delta\delta = \{x \in X, \left| P(1|x) - \frac{1}{2} \right| \leq \delta\} \tag{4}$$

Cost Function

Empirical risk $L(f)$ is the *expectation* value of all the point-wise losses ℓ we made by choosing the *target function*, f instead of f_*.

$$L(f) = \mathbb{E}_{(x_i,y_i)\sim P}\ell(y_i, f(x_i)) \tag{5}$$

Our target function f is the function for which the **Cost** is minimum.

$$\begin{aligned} f_{P,\ell} &= \underset{f:X\to\mathbb{R}}{\operatorname{argmin}} \mathbb{E}_{(x,y)\sim P}\ell(y, f(x)) \\ &= \underset{f:X\to\mathbb{R}}{\operatorname{argmin}} \mathbb{E}_{X\sim P_X}\mathbb{E}_{X\sim P(y|x)}\ell(y, f(x)) \end{aligned} \tag{6}$$

- So, by choosing the loss function ℓ we are essentially constraining ourselves to look for a function inside our *hypothesis* space $\mathcal{H} \subset \mathcal{F} = \{f | f : X \to \mathbb{R}\}$
- But the probability distribution function $P(y|x)$ is **unknown** so, we can never know what the target function f is.

The Learning Algorithm

- The job of the Learning Algorithm is to estimate a function, $\hat{f}$ that approximate the target function f with the i.i.d. samples from $S_n = (x_i, y_i) \sim p^n$.
- The fact that we can never know the target function f does not matter a lot if the estimated function $\hat{f}$ can predict the y for the *future values* of x.
- We define $Excess\ Risk = L(\hat{f}) - \min\limits_{f \in \mathcal{H}} L(f)$
- As $\hat{f}$ is derived from the sample, for different samples *Excess Risk* is going to be a different random number and we want to see the distribution, $P(Excess\ Risk > \epsilon)$.
- For Infinite amount of data, $\lim_{n\to\infty} P(L(\hat{f}) - \min\limits_{f \in \mathcal{H}} L(f) > \epsilon) = 0$.
- For finite amount of data, $\lim_{n\to\infty} P(L(\hat{f}) - \min\limits_{f \in \mathcal{H}} L(f) > \epsilon) \leq \delta$

Hinge Loss and Target Function

- Our target function, $f_{P,\ell} = \underset{f:X\to\mathbb{R}}{\operatorname{argmin}} \mathbb{E}_{(x,y)\sim P}\ell(y, f(x)) = P(1|x) - P(-1|x)$ in classification.
- We choose Hinge Loss function $|1 - yf(x)|_+$
- The f_P we get is the Bays Rule (Lin02).

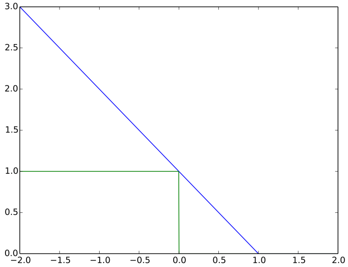


Figure: $\ell = max(0, 1 - yf(x))$

Linear Model with Hinge Loss and ERM

- We constrain our hypothesis space to be *linear*.

$$\text{i.e. } \mathcal{H} \cong \mathbb{R}^d \text{ and } f(x) = \mathbf{w}^\intercal \mathbf{x} \text{ and } X = \mathbb{R}^d$$
$$f \iff w \text{ and } \langle f, \overline{f} \rangle = \langle w^\intercal, \overline{w} \rangle$$

- We choose our loss function as *Hinge Loss* $\ell = max(0, 1 - yw^\intercal x)$
- If we choose $L(f)$ to be the Empirical Risk with square norm penalty or Ridge Penalty, $\lambda \|w\|^2$,

$$L^\lambda(w) = \min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{n=1}^{\infty} |1 - yw^\intercal x|_+ + \lambda \|w\|^2$$

Empirical Risk Minimizer

- In order to get the target function we have to minimize the Empirical Risk Function $L^\lambda(w)$.
- Existence of a minimum:
  - $\frac{1}{n}\sum_{n=1}^{\infty} |1 - yw^\intercal x|_+$ is Convex in w and $\lambda\|w\|^2$ is a parabola.
  - $L^\lambda(w)$ is Coercive in w for sure if $\lambda \geq 0$. i.e. $\lim_{W\to\infty} L^\lambda(w) \to \infty$
  - Minimum **exist**.
  - The uniqueness requires $L^\lambda(w)$ to be *strongly* convex.
  - If $\lambda > 0$ then $\exists\ \hat{w}^\lambda$ which minimizes $L^\lambda(w)$ and **unique**.
- In order to compute the minimum we have to take gradient of $L^\lambda(w)$ and set it to 0.

Gradient Decent and SVM

To minimize the Cost function $L^{\lambda}(w)$ we use iterative update.

$$w_{t+1} \leftarrow w_t - \gamma \Delta_w L^{\lambda}(w)$$

where γ is the learning rate and

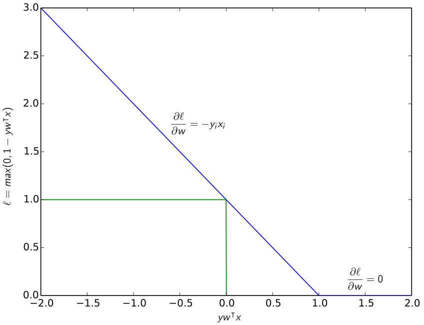
$$L^{\lambda}(w) = \min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{n=1}^{\infty} |1 - yw^{\intercal}x|_{+} + \lambda \|w\|^2$$

But the Hinge Loss Function is not differenciable as at $yw^{\intercal}x = 1$, The left-hand derivative $\neq$ right-hand derivative.
Other than that point the function is continuous and differenciable so, we can compute the set subdifferencials at $yw^{\intercal}x = 1$ and check if 0 is included in the set.

Sub-gradient for hinge loss

We will take Left Derivative on $\ell = max(0, 1 - yf(x))$

$$\begin{aligned} \frac{\partial \ell}{\partial w} &= -y_i x_i \quad \textit{When } yw^\intercal x \leq 1 \\ &= \quad 0 \qquad \textit{Otherwise} \end{aligned} \tag{7}$$



Sub-gradient descent algorithm for SVM

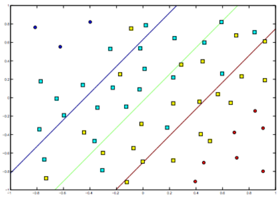
$$\frac{\partial L^{\lambda}(w)}{\partial w} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\partial \ell}{\partial w}\right)_{left} + 2\lambda w \tag{8}$$

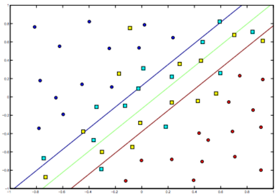
So the iterative update becomes: for t = 0, ...,T initialize w_0

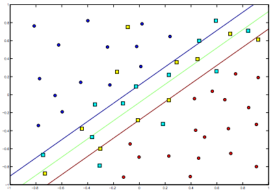
$$\begin{aligned} W_{t+1} &\leftarrow w_t - \gamma_t\left(2\lambda w - \tfrac{1}{n}\textstyle\sum_{i=1}^{n} y_i x_i\right) && \text{if } yw^{\intercal}x < 1 \\ &\leftarrow w_t - \gamma_t \lambda w_t && \text{Otherwise} \end{aligned}$$

Unlike the Gradient descent, a descent is not immediately guaranteed on every iteration that is why a variable step size γ_t is required.

Decision Boundary and Support Vectors

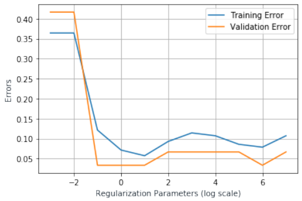

$(\lambda = 0.1)$


$(\lambda = 1)$


$(\lambda = 1000)$

- For a centered data $\hat{w}^\intercal x$ is the decision boundary.
- λ is the regularization parameter which has been chosen by performing cross validation on the data.
- λ serves as a degree of importance that is given to miss-classifications. As λ grows larger, less wrongly classified examples are allowed thus narrower margin between support vectors.
- $\lambda \to \infty$, allow no miss-classification(hard-margin).
- $\lambda \to 0$, more mis-classifications are allowed (soft-margin).

Training and Validation Error in SVM



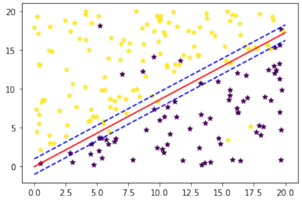
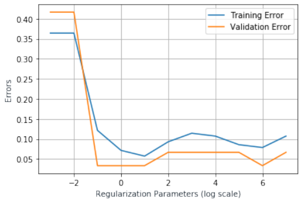


Figure: 3-fold Cross Validation and SVM for Best λ

- We generate synthetic data set of 2 dimensions and size 200 for training and validation of SVM mode.
- The first graph shows the training and testing error for increasing order of magnitude of λ starting from 10^{-3} till 10^{7} using svmTrainTestAnalysis().

Cross Validation and best λ



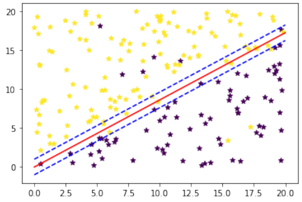
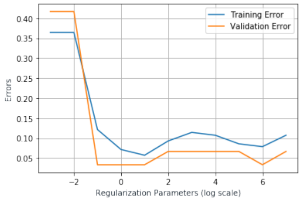


Figure: 3-fold Cross Validation and SVM for Best λ

- Then we will run 3-Fold Cross validation on more specific range $[1, 100]$ of λ to get best value of λ. Which takes around 972 sec.
- The Best value of the regularization parameter, $\lambda = 40$ for which the minimum validation error is 0.03333.
- The second graph shows the decision boundary and support vectors for best λ.
- Some points on the decision boundary explains consistent error even with very large values of λ.

Comparison Between KNN and SVM



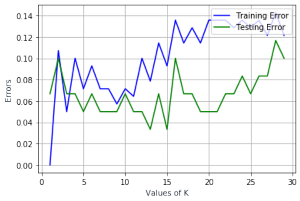


Figure: Error Comparison KNN vs SVM

- We perform KNN on the same data set with 10-fold cross validation and found the best value of $k = 15$.
- For best value of k the error on the validation set is 0.03333, same as SVM but it took a fraction of the time taken by SVM.
- From this comparison we can conclude that for low dimensional data of this type, **KNN out-performs SVM**.

References I

[Lin02] Yi Lin, *Support Vector Machines and the Bayes Rule in Classification*, Data Mining and Knowledge Discovery **6** (2002), no. 3, 259–275 (en).