SUPPORT VECTOR MACHINE with Hinge Loss

AMMI Project

GROUP 7

AFRICAN INSTITUTE FOR MATHEMATICAL SCIENCES

April 22, 2022

Group Members

1. ABOUBACRY MBARGANE
2. ALBERT AGISHA
3. KHADIJA IDDRISU
4. YASEEN ELTAHIR

Overview

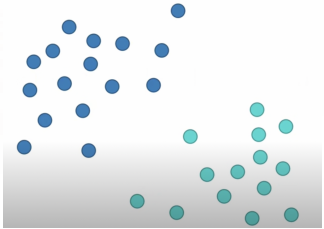
Overview

MOTIVATION AND CONCEPTS

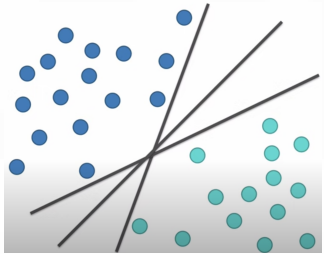
Support vector machine is a supervised learning system and used for classification and regression problems. Support vector machine is extremely favored by many as it produces notable correctness with less computation power. It is mostly used in classification problems.

Motivation

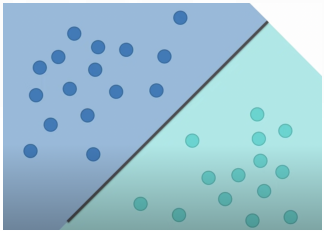
- Main Idea



(a) data points



(b) Looking for good split's line

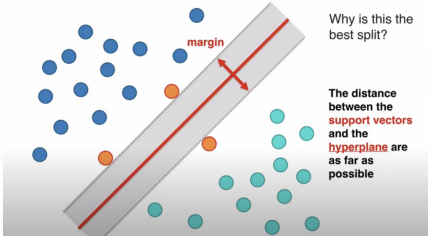


(c) Good Split line

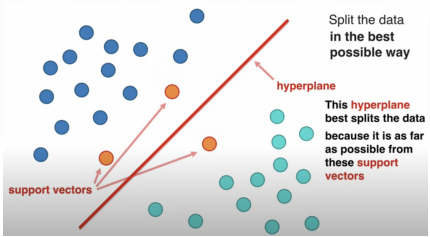
Figure 1: Three simple graphs

Concepts

- Margin, Support vectors, Hyperplane



(a) margin



(b) Looking for good split's line

Figure 2: Interesting concepts

Why SVM?

- Assumptions: our data-set is linearly separable from $+$'s and $-$'s, using logistic regression (or other), we may get fig.3

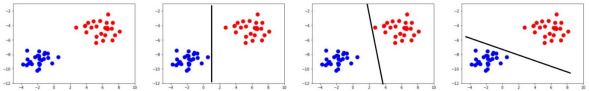


Figure 3: Other Classifiers

- Using SVM, with the same assumptions, we get

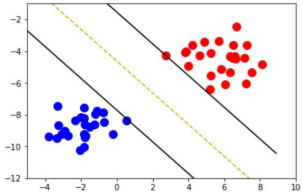


Figure 4: SVM

Concepts

- Margin, Support vectors, Hyperplane

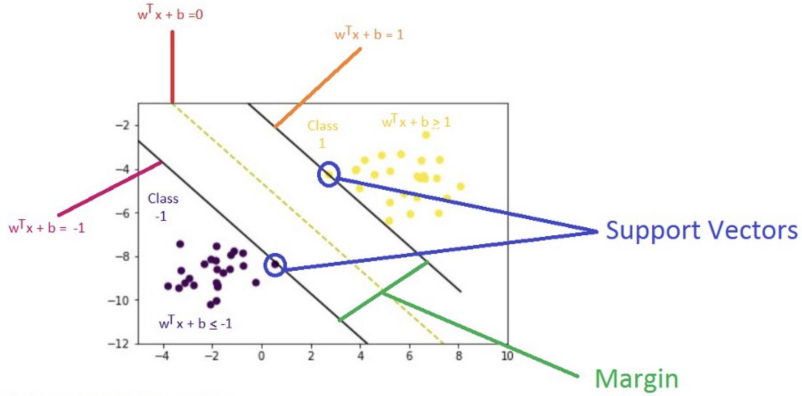


Figure 5: All concepts in one

Definitions

- The yellow line is called *Decision Boundary* (Hyperplane) in SVM.
- The other two lines are *Hyperplanes* that help us make the right decision boundary.
- *Hyperplane* is a "line in more than 3 dimensions" (in $1-D$ it's called a point, in $2-D$ it's called a line, in $3-D$ it's called a plane, more than $3D$ - Hyperplane).
- *Support vectors* are the closest points to the boundary.
- **GOAL:** Maximize margin, means to find the classifier whose decision boundary is furthest away from any data point.
- *Margin*: the distance between the left hyperplane and right hyperplane

Overview

Hyperplane and Margin

- $$Hyperplane = \{x \in \mathbb{R} \mid a^T x = b\}$$
- Let consider two parallel hyperplanes $h_1 \equiv a^T x = b_1$ and $h_2 \equiv a^T x = b_2$. Let consider two points two point x_1 on h_1 and x_2 on h_2, intersection of the normal respectively with h_1 and h_2. Where $x_1, x_2, a \in \mathbb{R}^n$ and $b1, b2 \in \mathbb{R}$

$$\begin{cases} x_1 \text{is on } h_1 \text{ it satisfies } a^T x_1 = b_1 \\ x_1 = c_1 a \end{cases}$$

$$\implies c_1 = \frac{b_1}{a^T a} = \frac{b_1}{||a||^2}$$

$$\implies x_1 = \frac{b_1}{||a||^2} a. \text{ Similarly, } x_2 = \frac{b_2}{||a||^2} a$$

Margin

The distance: $||x_1 - x_2|| = ||\frac{b_1}{||a||^2}a - \frac{b_2}{||a||^2}a|| = \frac{|b_1-b_2|}{||a||^2}||a|| = \frac{|b_1-b_2|}{||a||}$.
In our case we have $h_1 \equiv w^T x = 1 + b$ and $h_2 \equiv w^T x = -1 + b$. Thus we have

$$Margin = \frac{|1 + b - (-1 + b)|}{||w||} = \boxed{\frac{2}{||w||}}$$

Rules

-

how to choose the maximum margin?
pick one which has minimum Magnitude of w

Linear Model and Hinge Loss

We use a linear model to get a linear classifier

1. **Linear Model**
   - $$w^T x - b = 0$$
     with constraints
     $$\begin{cases} w^T x_i - b \geq 1 & \text{if } y_i = 1 \\ w^T x_i - b \leq -1 & \text{if } y_i = -1 \end{cases}$$
     $$\implies y_i(w^T x_i - b) \geq 1 \text{ constraints in one equation .}$$
2. **Cost function: Hinge Loss**
   $$l = \max(0, 1 - y_i(w^T x_i - b))$$

Linear Model and Hinge Loss

We use a linear model to get a linear classifier

1. **Linear Model**
2. **Cost function: Hinge Loss**

$$l = \max(0, 1 - y_i(w^T x_i - b))$$

$$l = \begin{cases} 0 & \text{if } y \cdot f(x) \geq 1 \\ 1 - y \cdot f(x) & \text{otherwise} \end{cases} \qquad \text{with } f(x) = (w^T x - b)$$

We want to maximize the margin $\frac{2}{||w||}$, therefore we need to minimize the Magnitude of our weights($||w||$). Thus, we have to add it to our cost function .

Linear Model and Hinge Loss

We use a linear model to get a linear classifier

1. **Linear Model**
2. **Cost function: Hinge Loss**
3. **Add Regularisation**: We want to maximize the margin $\frac{2}{||w||}$, therefore we need to minimize the Magnitude of our weights($||w||$). Thus, we have to add it to our cost function .

$$L = \lambda||w||^2 + \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i(w^T x_i - b))$$

The parameter tries to find the trade-off between the two terms.

- If $\boxed{y \cdot f(x) \geq 1}$: $L_i = \lambda||w||^2$
- Else: $L_i = \lambda||w||^2 + (1 - y_i(w^T x_i - b))$

Gradient

- If $y \cdot f(x) \geq 1$:

$$\begin{cases} \frac{\partial L_i}{\partial w_k} = 2\lambda w_k \\ \frac{\partial L_i}{\partial b} = 0 \end{cases} \tag{1}$$

- Else: $L_i = \lambda ||w||^2 + (1 - y_i(w^T x_i - b))$

$$\begin{cases} \frac{\partial L_i}{\partial w_k} = 2\lambda w_k - y_i x_i \\ \frac{\partial L_i}{\partial b} = y_i \end{cases} \tag{2}$$

Stochastic gradient descent for SVM

Given a training set $S = \{(x_i, y_i)\}$ $X \in \Re^d, y \in \{-1, 1\}$

1. Initialize $w_0 = 0_{\Re^d}$
2. For epoch $= 1$ ... T:

   Pick a random example (x_i, y_i) from the training set S.
   Treat (x_i, y_i) as full dataset and take the derivative of the SVM.
   objective $\hat{L}$ at the current w_{t-1}. Call it $\nabla\ \hat{L}(w_{t-1})$
   Update : $w_t \leftarrow w_{t-1} - \gamma_t \nabla L(w_{t-1})$
3. return Final w

This algorithme is guaranteed to converge to the minimum of L if γ_t is small enough. Because the objective $L(w)$ is convex.

sub-gradients of the hinge loss

Definition

Formally, a vector g is subgradient to f at point x if $f(y) \geq f(x) + g^T(y - x)$ for all y.

Hinge Loss is not ddifferentiable!

$L(w) = min\frac{1}{2}W^T W + C * max(0, 1 - y_i W^T x_i)$, C is the regularization hyper-parameter.

- Big C, the loss is more important better recognition rate but smaller margin (worse generalization).
- Small, the generalization is more important, larger margin (more robust classifier) but worse recognition rate.

We get the derivative of the hinge loss with respect to W by using :

a general strategy, it first solve the max and compute the gradient for each case.

sub-derivatives of the hinge loss

Base on the max, we can rewrite :

- $L(w) = min\frac{1}{2}W^T W$ if $max(0, 1 - y_i W^T x_i) = 0$
- $L(w) = min\frac{1}{2}W^T W + C * (1 - y_i W^T x_i)$ *otherwise*

And that gives :

$$\nabla L = \begin{cases} W & if \quad max(0, 1 - y_i W^T x_i) = 0 \\ W - Cy_i x_i & otherwise \end{cases}$$

Stochastic sub-gradient descent for SVM

Given a training set $S = \{(x_i, y_i)\}, X \in \Re^d \; y \in \{-1, 1\}$.

1. Initialize $W = 0 \in \Re^d$
2. For epoch $= 1 \ldots$ T:
   For each training example $(x_i, y_i) \in S$:
   if $y_i W^T x_i \leq 1$:
   $W \leftarrow (1 - \gamma_t) W + \gamma_t C y_i x_i$
   else:
   $W \leftarrow (1 - \gamma_t) W$
3. return W

Overview

Preprocessing

- Preprocessing and TFIDF output

	review	sentiment
0	one review mention watch 1 oz episod youll hoo...	positive
1	wonder littl product br br film techniqu unass...	positive
2	thought wonder way spend time hot summer weeke...	positive
3	basic there famili littl boy jake think there ...	negative
4	petter mattei love time money visual stun film...	positive

Figure 6: processed data

```
[('class', 0.023), ('agenda', 0.034), ('episod', 0.053), ('dodgi', 0.036), ('gangsta', 0.03
9), ('side', 0.024), ('saw', 0.022), ('focus', 0.03), ('injustic', 0.039), ('GO', 0.037), ('g
lass', 0.031), ('maximum', 0.034), ('scuffl', 0.036), ('classic', 0.022), ('darker', 0.033),
('inward', 0.039), ('privaci', 0.039), ('charm', 0.027), ('nickel', 0.039), ('hook', 0.03)]
```

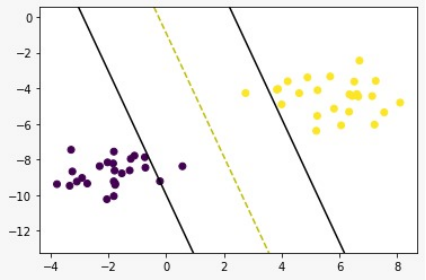
Figure 7: TFIDF some word in a doc

SVM-training

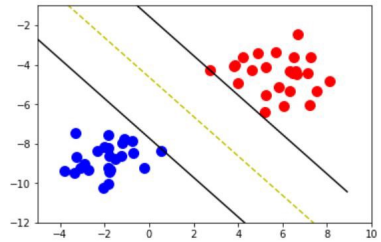
```
def fit(self, X, y):
    ##
        #fuction for training
    ##
    n_samples, n_features = X.shape
    #insure data in correct form
    y_ = np.where(y <= 0, -1, 1)
    #how to initialize the wights (width=n_features as the input has):
    self.w = np.zeros(n_features)
    #initialize bias
    self.b = 0

    #training loop:
    for iteration in range(self.n_iters):
        #Loop through X (input) i.e. SGD
        for idx, x_i in enumerate(X):
            #update rule:
            condition = y_[idx] * (np.dot(x_i, self.w) - self.b) >= 1
            if condition:
                self.w -= self.lr * (2 * self.lambda_param * self.w)
            else:
                self.w -= self.lr * (
                    2 * self.lambda_param * self.w - np.dot(x_i, y_[idx])
                )
                self.b -= self.lr * y_[idx]

        #training accuarcy:
        if(iteration %100==0):
          print(metrics.accuracy_score(y, self.predict(X)))
```

Figure 8: Fit Method

SVM



(a) SVM results from scratch



(b) SVM results Built-in

Figure 9: SVM Compared Results

References I

1. Shai Shalev-Shwartz and Shai Ben-David,Understanding Machine Learning: From Theory to Algorithms,Cambridge University Press
2. 7.3.4. Loss Function for Support Vector Machine Classifier - Hinge Loss: https://www.youtube.com/watch?v=CL2pUVLB7eI
3. art 24-SVM Classification (hard margin and soft margin) : https://www.youtube.com/watch?v=0OLR3If-qS0