Lec 18: Support Vector Machine (SVM) II

Ailin Zhang

Agenda

- Dual form
- Digression: Convexity
- Kernel SVM
- Linear Inseparability
- Hinge Loss
- Connection to logistic regression

Support Vector Machine

Let u be an unit vector that has the same direction as β. $u = \dfrac{\beta}{|\beta|}$.

Suppose X_i is an example on the margin (i.e., support vector), the projection of X_i on u is

$$\langle X_i, u\rangle = \langle X_i, \frac{\beta}{|\beta|}\rangle = \frac{X_i^\top \beta}{|\beta|} = \frac{\pm 1}{|\beta|}.$$

So the margin is $1/|\beta|$. In order to maximize the margin, we should minimize $|\beta|$ or $|\beta|^2$. Hence, the SVM can be formulated as an optimization problem as follows:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}|\beta|^2, \\ \text{subject to} \quad & y_i X_i^\top \beta \geq 1, \forall i. \end{aligned}$$

Recall $X_i^\top \beta$ is the score, and $y_i X_i^\top \beta$ is the individual margin of observation i. This is the **primal form** of SVM.

Dual Form: Lagrange Multiplier

Let $\alpha = (\alpha_1, \alpha 2, \cdots, \alpha_n)$, where $\alpha_i \geq 0$

$L(\beta, \alpha) = \frac{1}{2}|\beta|^2 + \sum_{i=1}^{n} \alpha_i(1 - y_i X_i^\top \beta)$

$(\hat{\beta}, \hat{\alpha}) = \text{argmin}_\beta \text{argmax}_\alpha L(\beta, \alpha)$

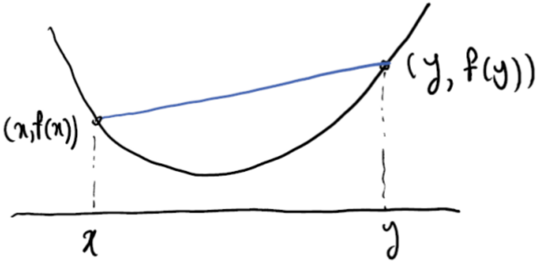
The idea is to solve an unconstrained problem because it is easier to solve.

Convexity

Definition

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if its domain is a convex set and for all x, y in its domain, and all $\lambda \in [0, 1]$, we have

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$$



Convex functions

- Strictly convex if $\forall x, y, x \neq y, \forall \lambda \in (0,1)$

$$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda)f(y)$$

- In words, this means that if we take any two points x, y, then f evaluated at any convex combination of these two points should be no larger than the same convex combination of $f(x)$ and $f(y)$.
- Geometrically, the line segment connecting $(x, f(x))$ to $(y, f(y))$ must sit above the graph of f.
- If f is continuous, then to ensure convexity it is enough to check the definition with $\lambda = \frac{1}{2}$ (or any other fixed $\lambda \in (0,1)$).
- We say that f is concave if $-f$ is convex.

Examples of univariate convex functions

- e^{ax}
- $-\log(x)$
- x^a (defined on $x > 0, a \geq 1$ or $a \leq 0$)
- $-x^a$ (defined on $x > 0, 0 \leq a \leq 1$)
- $|x|^a, a \geq 1$
- $x \log(x)$ (defined on $x > 0$)

Examples of multivariate convex functions

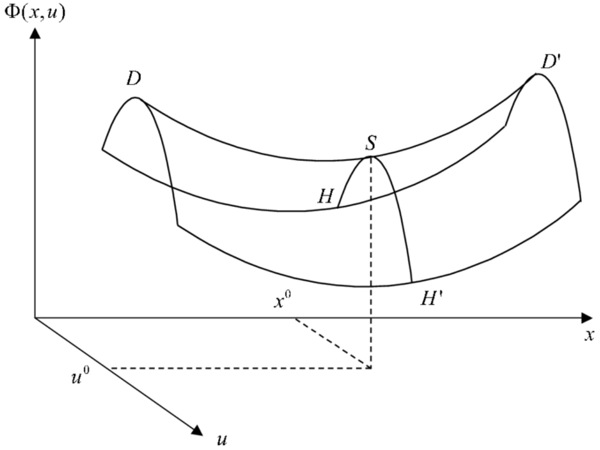
- Affine functions: $f(x) = a^T x + b$ (for any $a \in \mathbb{R}^n, b \in \mathbb{R}$). They are convex, but not strictly convex; they are also concave:

$$\begin{aligned}\forall \lambda \in [0,1], f(\lambda x + (1-\lambda)y) &= a^T(\lambda x + (1-\lambda)y) + b \\ &= \lambda a^T x + (1-\lambda) a^T y + \lambda b + (1-\lambda) b \\ &= \lambda f(x) + (1-\lambda) f(y).\end{aligned}$$

  In fact, affine functions are the only functions that are both convex and concave.

- Some quadratic functions: $f(x) = x^T Q x + c^T x + d$.
  - Convex if and only if $Q \succeq 0$.
  - Strictly convex if and only if $Q \succ 0$.
  - Concave if and only if $Q \preceq 0$; strictly concave if and only if $Q \prec 0$.

Dual Form: Lagrange Multiplier and saddle point



Dual Form

$$L(\beta, \alpha) = \frac{1}{2}|\beta|^2 + \sum_{i=1}^{n} \alpha_i (1 - y_i X_i^\top \beta)$$

1. $\frac{\partial L}{\partial \beta} = 0$

$$\hat{\beta} = \sum_{i=1}^{n} \alpha_i y_i X_i$$

2. Dual function:

$$Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}|\sum_{i=1}^{n} \alpha_i y_i X_i|^2$$

Dual Problem: $\max_{\alpha_i \geq 0} Q(\alpha)$

Dual Form

$$L(\beta, \alpha) = \frac{1}{2}|\beta|^2 + \sum_{i=1}^{n} \alpha_i (1 - y_i X_i^\top \beta)$$

1. $\dfrac{\partial L}{\partial \beta} = 0$

$$\hat{\beta} = \sum_{i=1}^{n} \alpha_i y_i X_i$$

2. Dual function:

$$Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}|\sum_{i=1}^{n} \alpha_i y_i X_i|^2$$

Dual Problem: $\max_{\alpha_i \geq 0} Q(\alpha)$ Solve this by coordinate descent

Coordinate Descent

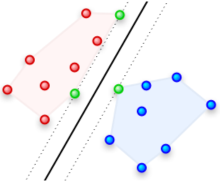
- Each iteration
  - For i in 1 to n:
    $\max_{\alpha_i} Q(\alpha)$ by fixing the rest $\alpha_j, j \neq i$
    Remark: All $\alpha_i \geq 0$

  Until Convergence

For prediction: $\hat{y} = \mathrm{sign}(< x, \hat{\beta} >)$

Dual Form

The primal form of SVM is max margin, and the dual form of SVM is min distance.



max margin = min distance

The margin between the two sets is defined by the minimum distance between two.

- Convex hull: the convex hull of a sample of points is the minimum convex set enclosing them all, yielding a polygon connecting the outermost points in the sample and all whose inner angles are less than 180 degrees

Dual Form - Convex Hull

Let $X_+ = \sum_{i\in +} c_i X_i$ and $X_- = \sum_{i\in -} c_i X_i$ ($c_i \geq 0, \sum_{i\in +} c_i = 1, \sum_{i\in -} c_i = 1$) be two points in the positive and negative convex hulls. The margin is $\min |X_+ - X_-|^2$.

$$
\begin{aligned}
|X_+ - X_-|^2 &= \left| \sum_{i\in +} c_i X_i - \sum_{i\in -} c_i X_i \right|^2 \\
&= \left| \sum_i y_i c_i X_i \right|^2 \\
&= \sum_{i,j} c_i c_j y_i y_j \langle X_i, X_j \rangle,
\end{aligned}
$$

$$
\text{subject to} \quad c_i \geq 0, \sum_{i\in +} c_i = 1, \sum_{i\in -} c_i = 1.
$$

Dual Form - Convex Hull

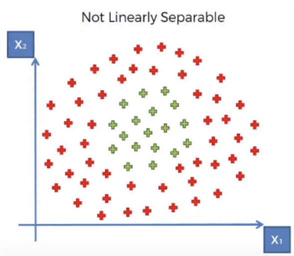
Let $X_+ = \sum_{i\in+} c_i X_i$ and $X_- = \sum_{i\in-} c_i X_i$ $(c_i \geq 0, \sum_{i\in+} c_i = 1, \sum_{i\in-} c_i = 1)$ be two points in the positive and negative convex hulls. The margin is $\min |X_+ - X_-|^2$.

$$\begin{aligned}
|X_+ - X_-|^2 &= \left|\sum_{i\in+} c_i X_i - \sum_{i\in-} c_i X_i\right|^2 \\
&= \left|\sum_{i} y_i c_i X_i\right|^2 \\
&= \sum_{i,j} c_i c_j y_i y_j \langle X_i, X_j\rangle, \\
\text{subject to} \quad & c_i \geq 0, \sum_{i\in+} c_i = 1, \sum_{i\in-} c_i = 1.
\end{aligned}$$

We can play the kernel trick to replace $\langle X_i, X_j\rangle$ by $K(X_i, X_j)$ Solvable with sequential minimal optimization

Kernel SVM

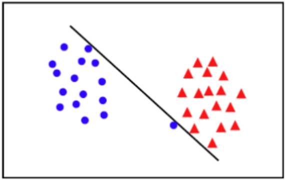
For a linearly non-separable dataset:



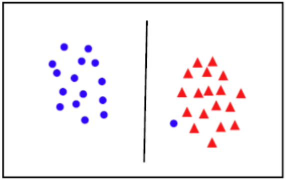
A popular kernel:

- Gaussian radial basis function $K(X, X') = \exp(-\gamma|X - X'|^2)$

Linear Inseperability

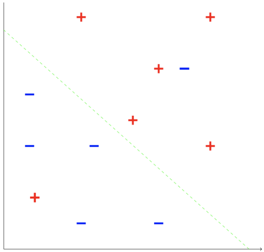


• the points can be linearly separated but there is a very narrow margin



• but possibly the large margin solution is better, even though one constraint is violated

Linear Inseparablity



we have a few examples that are incorrectly classified. We'd like to somehow move the bad examples to the other side of the hyperplane. But for this, we'd have to pay a price.

Linear Inseparability

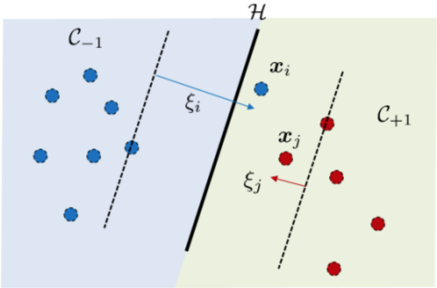
- Change

$$y_j \left(\boldsymbol{x}_j^T \beta \right) \geq 1$$

  to this one:

$$y_j \left(\boldsymbol{x}_j^T \beta \right) \geq 1 - \xi_j, \quad \text{and} \quad \xi_j \geq 0.$$

- If $\xi_j > 1$, then $\boldsymbol{x}_j$ will be misclassified.



Inseparability: Slack Variable

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}|\beta|^2 + C\sum_{i=1}^{n}\xi_i, \\ \text{subject to} \quad & y_i X_i^\top \beta \geq 1 - \xi_i, \forall i. \end{aligned}$$

Essentially, ξ_i is the amount which we move example i, and C is some positive constant.

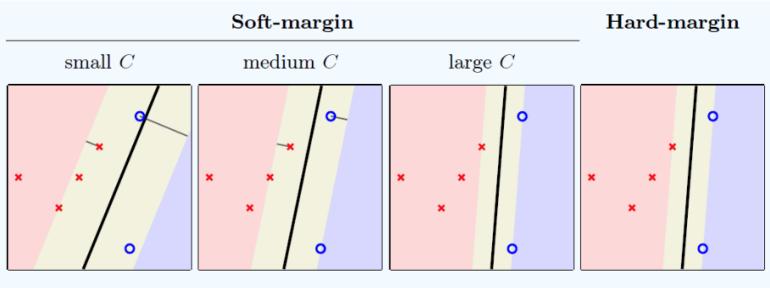
Dual form:

$$L(\beta, \xi, \alpha, \mu) = \frac{1}{2}|\beta|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i(1 - \xi_i - y_i X_i^\top \beta) + \sum_{i=1}^{n}\mu_i(-\xi_i)$$

$\max_{\alpha,\mu} \min_{\beta,\xi} L(\beta, \xi, \alpha, \mu)$

Role of C

- If C is big, then we enforce ξ to be small.
- If C is small, then ξ can be big.



$\min_{\beta,\xi} L(\beta, \xi, \alpha, \mu)$

$$\frac{\partial L}{\partial \beta} = 0 \rightarrow \hat{\beta} = \textstyle\sum_{i=1}^{n} \alpha_i y_i X_i$$

$$\frac{\partial L}{\partial \xi_i} = 0 \rightarrow \alpha_i = C - \mu_i \leq C$$

$$Q(\alpha) = \textstyle\sum_{i=1}^{n} \alpha_i - \frac{1}{2} |\sum_{i=1}^{n} \alpha_i y_i X_i|^2$$

Hinge Loss

Another way to interpret ξ_i

- $y_i \beta X_i \geq 1 \rightarrow \xi_i = 0$
- $y_i \beta X_i < 1 \rightarrow \xi_i = 1 - y_i \beta X_i$

$\hat{\xi}_i = \max(0, 1 - y_i X_i^T \beta)$, this is usually called hinge loss

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}|\beta|^2 + C \sum_{i=1}^{n} \xi_i, \\ & \rightarrow \frac{1}{2}|\beta|^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i \beta X_i) \end{aligned}$$

Recall the loss for perceptron is $\max(0, -y_i X_i^\top \beta)$, which penalizes mistakes or negative margins $y_i X_i^\top \beta$. In comparison, the hinge loss does not only penalize the negative margins $y_i X_i^\top \beta$, it also penalizes margins less than 1.

SVM and ridge logistic regression

Rewrite the

$$loss(\beta) = \sum_{i=1}^{n} \max(0, 1 - y_i X_i^\top \beta) + \frac{\lambda}{2}|\beta|^2,$$

we can solve β by gradient descent. The gradient is

$$loss'(\beta) = -\sum_{i=1}^{n} 1(y_i X_i^\top \beta < 1) y_i X_i + \lambda\beta,$$

where $1(\cdot)$ is the indicator function.

SVM and ridge logistic regression

Rewrite the

$$loss(\beta) = \sum_{i=1}^{n} \max(0, 1 - y_i X_i^\top \beta) + \frac{\lambda}{2}|\beta|^2,$$

we can solve β by gradient descent. The gradient is

$$loss'(\beta) = -\sum_{i=1}^{n} 1(y_i X_i^\top \beta < 1) y_i X_i + \lambda\beta,$$

where $1(\cdot)$ is the indicator function.

This is similar to the ridge logistic regression

$$loss(\beta) = \sum_{i=1}^{n} \log[1 + \exp(-y_i X_i^\top \beta)] + \frac{\lambda}{2}|\beta|^2,$$

$$loss'(\beta) = -\sum_{i=1}^{n} \sigma(-y_i X_i^\top \beta) y_i X_i + \lambda\beta.$$

R code for SVM

```
my_SVM <- function(X_train, Y_train, X_test, Y_test, lambda = 0.01,
                   num_iterations = 1000, learning_rate = 0.1)
{
  n         <- dim(X_train)[1]
  p         <- dim(X_train)[2] + 1
  X_train1  <- cbind(rep(1, n), X_train)
  Y_train   <- 2 * Y_train - 1
  beta      <- matrix(rep(0, p), nrow = p)

  ntest     <- nrow(X_test)
  X_test1   <- cbind(rep(1, ntest), X_test)
  Y_test    <- 2 * Y_test - 1

  acc_train <- rep(0, num_iterations)
  acc_test  <- rep(0, num_iterations)

  for(it in 1:num_iterations)
  {
    s         <- X_train1 %*% beta
    db        <- s * Y_train < 1
    dbeta     <- matrix(rep(1, n), nrow = 1) %*%((matrix(db*Y, n, p)*X1))/n;
    beta      <- beta + learning_rate * t(dbeta)
    beta[2:p] <- beta[2:p] - lambda * beta[2:p]

    acc_train[it] <- mean(sign(s * Y_train))
    acc_test[it]  <- mean(sign(X_test1 %*% beta * Y_test))
    }
  model <- list(beta = beta, acc_train = acc_train, acc_test = acc_test)
  model
}
```