Lec 17: Perceptron and Support Vector Machine (SVM)

Ailin Zhang

Recap: Bayesian Regression

- We can view a Rigde Regression as a MAP (Bayes Least Square) inference with a Gaussian prior.

  $\beta \sim \mathrm{N}(0, \tau^2 \mathbf{I}_p)$ be the prior distribution of β.

  $$\hat{\beta} = (\mathbf{X}^\top \mathbf{X}/\sigma^2 + \mathbf{I}_p/\tau^2)^{-1} \mathbf{X}^\top \mathbf{Y}/\sigma^2.$$

  which corresponds to the ridge regression with $\lambda = \sigma^2/\tau^2$.

- We can view a Lasso Regression as a MAP inference with a Laplace prior.

  $p(\beta) = (\frac{\gamma}{2})^p \exp(-\gamma||\beta||_1)$ be the prior distribution of β.

  corresponds to the lasso regression with $\lambda = \sigma^2\gamma$.

Bayesian Regression with multiparamters

$$p\left(Y \mid \beta, \sigma^2\right) = \left(2\pi\sigma^2\right)^{-n/2} \exp\left[-\frac{1}{2\sigma^2}||Y - X\beta||^2\right]$$

- looks normal as a function of β
- looks inverse gamma as a function of σ^2

Bayesian Regression Example (Multiparameter case)

Noninformative Prior: $[\beta, \sigma^2] \sim \frac{1}{\sigma^2}$

$$p\left(\beta, \sigma^2 \mid Y\right) \propto \sigma^{-n-2} \exp\left(-\frac{1}{2\sigma^2}||Y - X\beta||^2\right)$$

For Bayesian Inference, we would like to know:

- $$p(\sigma^2 \mid Y) = \int p(\beta, \sigma^2 \mid Y) d\beta$$
  - $\sigma^2 \mid y \sim \text{Inv}-\chi^2\left(n-p, s^2\right)$, where $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(y_i - \hat{y}\right)^2$
- $$p(\beta \mid Y) = \int p(\beta, \sigma^2 \mid Y) d\sigma^2$$
  - $\beta \mid Y \sim \mathrm{t}\left(n-p, \hat{\beta}_{LS}, s^2(X^TX)^{-1}\right)$

Agenda

- Classification, outcome, and logistic loss
- Perceptron model and margin
- Introduction to SVM

Warm up: Logsitic Regression

obs	$\mathbf{X}_{n\times p}$	$\mathbf{Y}_{n\times 1}$
1		
2		
...		
i	$X_i^\top$	y_i
...		
n		

Let $\eta_i = X_i^\top \beta$ be the score, then

$$p_i = \sigma(\eta_i) = \frac{1}{1+e^{-\eta_i}} = \frac{1}{1+e^{-X_i^\top \beta}} = \frac{e^{X_i^\top \beta}}{1+e^{X_i^\top \beta}},$$

$$1 - p_i = \frac{1}{1+e^{X_i^\top \beta}}$$

$\eta_i = \log(\frac{p_i}{1-p_i}) = logit(p_i) =$ log odds ratio

Warm up: Logsitic Regression

- Let the loss function $\mathrm{loss}(\beta)$ be the negative log-likelihood, then

$$\mathrm{loss}(\beta) = \sum_{i=1}^{n} \log\left[1 + \exp\left(-y_i X_i^\top \beta\right)\right].$$

- The gradient

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

- The gradient descent algorithm is

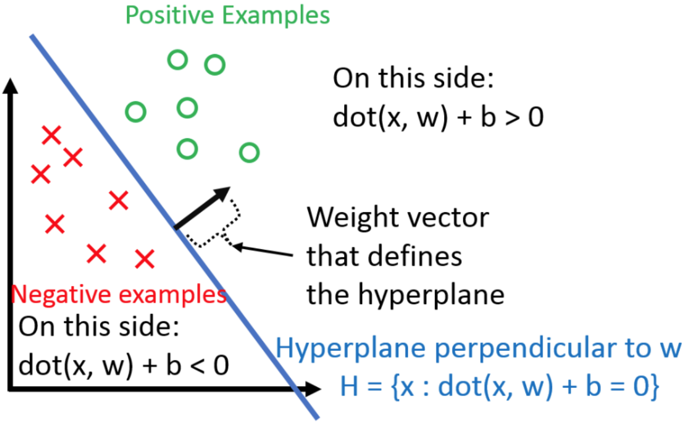
$$\beta^{(t+1)} = \beta^{(t)} + \gamma \sum_{i=1}^{n} \sigma\left(-y_i X_i^\top \beta\right) y_i X_i$$

Perceptron

- Logistic Regression is a soft version of perception
- Logistic Regression is also a generalized linear model (GLM)
- Perceptron is a binary classifier: $\hat{y}_i = sign(\mathbf{X}_i^\top \beta)$.

The perceptron model

$$h(x_i) = \text{sign}(\mathbf{w}^\top \mathbf{x}_i + b)$$



The Perceptron Model

The perceptron model $y_i = \mathrm{sign}(X_i^\top \beta)$, where $y_i \in \{+1, -1\}$

- Define the loss function:

$$loss(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} \mathrm{Loss}\left(y_i, f_{\boldsymbol{\beta}}\left(\mathbf{x}_i\right)\right)$$

  where

$$\mathrm{Loss}(y, \hat{y}) = \mathbf{1}\{\hat{y} \neq y\} = \begin{cases} 1 & \hat{y} \neq y \\ 0 & \text{otherwise} \end{cases}$$

- Solve β by perceptron update

The Perceptron Model: perceptron update

Starting from $\beta_0 = 0$,

$$\beta^{(t+1)} = \beta^{(t)} + \sum_{i=1}^{n} \delta_i y_i X_i,$$

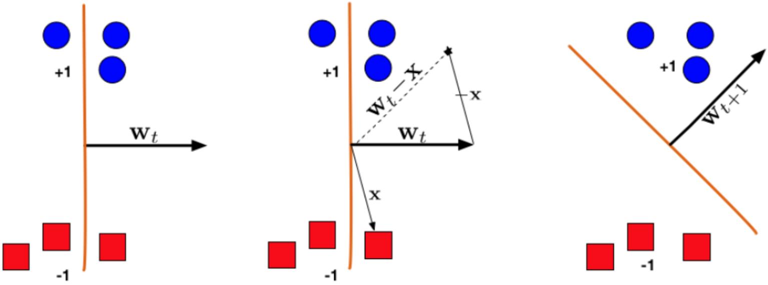
where $\delta_i = 1(y_i \neq \mathrm{sign}(X_i^\top \beta^{(t)}))$ to determine whether $\beta^{(t)}$ makes a mistake in classifying y_i.

- Reasoning: (only consider one data point (X_t, y_t), where $y_t \neq X_i^\top \beta^{(t)}$)
  $\beta_{(t+1)} = \beta_{(t)} + y_t X_t$

  $$\begin{aligned} y_t \beta_{(t+1)}^T X_t &= y_t \left(\beta_{(t)} + y_t X_t\right)^T X_t \\ &= y_t \beta_{(t)}^T X_t + y_t^2 X_t^T X_t \\ &= y_t \beta_{(t)}^T X_t + \|X_t\|^2 \end{aligned}$$

  so this quantity either becomes "less negative", or even better, shifts to being positive.

Perceptron Update: Geometric Intuition



Perceptron Update and Margin

- The algorithm can also be considered as the gradient descent algorithm for the loss function

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

- The term $y_i X_i^\top \beta$ can be defined as the margin for this observation.
  - If the margin is large, it means the classification is confident.
  - If the margin is small, it means the classification is uncertain.
  - If the margin is negative, it means β makes a mistake. The more negative it is, the bigger the mistake is.

Perceptron and Margin

- If a data set is **linearly separable**, the perceptron model will find a separating hyperplane in a finite number of updates.
- Margin: the distance between the hyperplane and the observations closest to the hyperplane

$$loss(\beta) = \sum_{i=1}^{n} \max(0, -y_i X_i^\top \beta) = \sum_{i=1}^{n} \max(0, -Margin_i),$$

- $\max(0, -Margin_i) = 0$ if $Margin_i \geq 0$, i.e., no mistake is made
- $\max(0, -Margin_i) = -Margin_i$ if $Margin_i < 0$.
- the algorithm learns from the mistakes.

Perceptron Convergence

Theorem

For simplicity, we consider the case where the linear separator must pass through the origin (intercept = 0). If the following conditions hold:

1. there exists β^* such that $Y_i \frac{X_i^\top \beta^*}{\|\beta^*\|} \geqslant \gamma$ for all $i = 1, \ldots, n$ and for some $\gamma > 0$ and

2. all the examples have bounded magnitude: $\|X_i\| \leqslant R$ for all $i = 1, \ldots n$,

then the perceptron algorithm will make at most $\left(\frac{\mathrm{R}}{\gamma}\right)^2$ mistakes. At this point, its hypothesis will be a linear separator of the data.

- γ is also defined as the margin for the hyperplane β^*

Convergence Proof (1)

- We initialize $\beta^{(0)} = 0$, and let $\beta^{(k)}$ define our hyperplane after the perceptron algorithm has made k mistakes. Assume that the k^{th} mistake occurs on the i^{th} example
- So, let's think about the cosine of the angle between $\beta^{(k)}$ and β^*,

$$\cos\left(\beta^{(k)}, \beta^*\right) = \frac{\beta^{(k)\top}\beta^*}{\|\beta^*\|\,\|\beta^{(k)}\|}$$

- Prove by induction

$$\frac{\beta^{(k)\top}\beta^*}{\|\beta^*\|} = \frac{\left(\beta^{(k-1)} + Y_i X_i\right)^\top \beta^*}{\|\beta^*\|} = \frac{\beta^{(k-1)\top}\beta^*}{\|\beta^*\|} + \frac{Y_i X_i^\top \beta^*}{\|\beta^*\|}$$
$$\geqslant \frac{\beta^{(k-1)\top}\beta^*}{\|\beta^*\|} + \gamma \geqslant k\gamma$$

Convergence Proof (2)

- Prove by Induction

$$
\begin{aligned}
\left\|\beta^{(k)}\right\|^2 &= \left\|\beta^{(k-1)} + Y_i X_i\right\|^2 \\
&= \left\|\beta^{(k-1)}\right\|^2 + 2 Y_i X_i^\top \beta^{(k-1)} + \|X - i\|^2 \\
&\leqslant \left\|\beta^{(k-1)}\right\|^2 + R^2 \\
&\leqslant kR^2
\end{aligned}
$$

- Returning to the definition of the dot product, we have

$$
\cos\left(\beta^{(k)}, \beta^*\right) = \left(\frac{\beta^{(k)\top}\beta^*}{\|\beta^*\|}\right)\frac{1}{\left\|\beta^{(k)}\right\|} \geqslant (k\gamma)\cdot\frac{1}{\sqrt{k}R} = \sqrt{k}\cdot\frac{\gamma}{R}
$$

Convergence Proof (3)

- Since the value of the cosine is at most 1 , we have

$$1 \geqslant \sqrt{k} \cdot \frac{\gamma}{R}$$
$$k \leqslant \left(\frac{R}{\gamma}\right)^2.$$

This result endows the margin γ of $\mathcal{D}_n$ with an operational meaning: when using the Perceptron algorithm for classification, at most $(R/\gamma)^2$ classification errors will be made, where R is an upper bound on the magnitude of the training vectors.

Support Vector Machine - Motivation

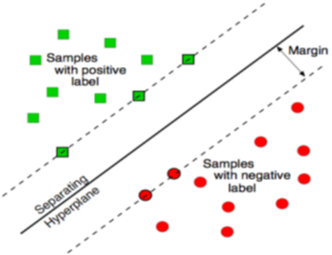
Consider the perceptron $y_i = \mathrm{sign}(X_i^\top \beta)$:

1. It separates the positive examples and negative examples by projecting the data on vector β,
2. It separates the examples by a hyperplane that is perpendicular to β.

If the positive examples and negative examples are separable, there are be many separating hyperplanes.

We want to choose the one with the **maximum margin** in order to guard against the random fluctuations in the unseen testing examples.

Support Vector Machine Geometry



The idea of support vector machine (SVM) is to find the β, so that

1. for positive examples $y_i = +$, $X_i^\top \beta \geq 1$,
2. for negative examples $y_i = -$, $X_i^\top \beta \leq -1$.

Here we use $+1$ and -1, because we can always scale β.

The decision boundary is decided by the training examples that lies on the margin. Those are the support vectors.

Support Vector Machine

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

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.

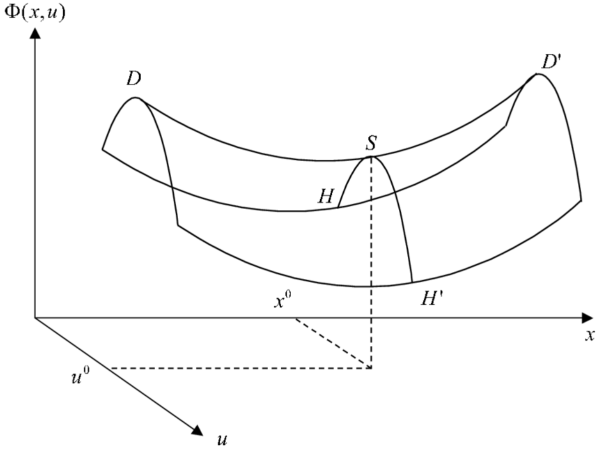
Dual Form: Lagrange Multiplier

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

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

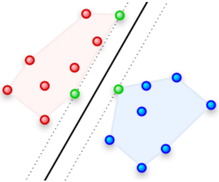
The idea is to solve an unconstrained problem because it is easier to solve.

Dual Form: Lagrange Multiplier and saddle point



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.

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 $\min \left|X_+ - X_-\right|^2$.

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

Solvable with sequential minimal optimization