Supervised Learning

CS446-Machine Learning

Qi Long

Contents

0 Preliminaries

0.1 Matrix Derivative

0.1.1 First Order

$$\frac{\partial x^T a}{\partial x} = \frac{\partial a^T x}{\partial x} = a$$
$$\frac{\partial a^T X b}{\partial X} = ab^T$$

0.1.2 Second Order

$$\frac{\partial x^T B x}{\partial x} = (B + B^T)x$$
$$\frac{\partial b^T X^T X c}{\partial X} = X(bc^T + cb^T)$$

0.2 Dual Program

0.2.1 Primal Program

The goal is to Find the optimum of a function given a restriction of another function, for example,

$$\max_{x,y} f(x, y)$$

given restriction

$$g(x, y) = c$$

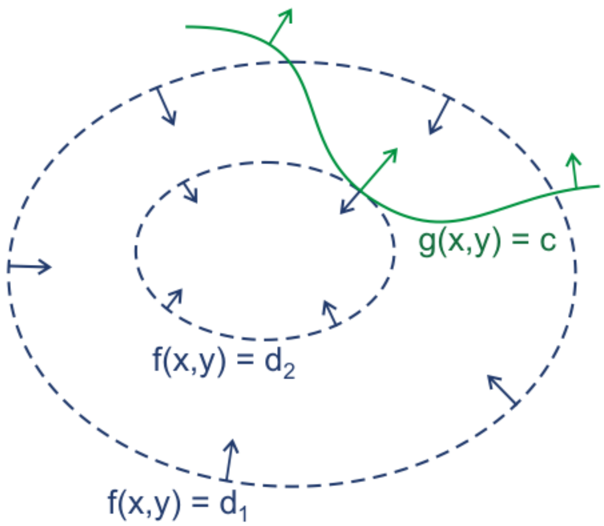
0.2.2 Lagrangian

$f(x, y) = d_n$ can be regarded as contours with **adjustable** values d_n.
$g(x, y) = c$ can be regarded as a fixed curve crossing contours.
Then the optimum is when $f(x, y) = d_n$ and $g(x, y) = c$ have same derivative.
Import λ to represent adjustable d_n,

$$\nabla \frac{1}{\lambda} f(x, y) = \nabla(g(x, y) - c)$$
$$\nabla[f(x, y) - \lambda(g(x, y) - c)] = 0 \tag{1}$$

The corresponding Lagrangian Program is

$$L(x, y, \lambda) = f(x, y) - \lambda(g(x, y) - c)$$



0.2.3 Dual Form

Solves equation (1) to represent primal parameters (x, y) using Lagrangian Multiplier λ, getting the dual form equation with only one parameter λ.
After solving λ, plug λ back to get optimal x^* and y^*.

0.3 Vector Norms

0.3.1 p-norm

$$||x||_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}, \quad p \geq 1$$

0.3.2 1-norm

Property of encouraging sparsity: even differences among all magnitudes.

$$||x||_1 = \sum_{i=1}^{n} |x_i|$$

0.3.3 2-norm

Analog to Euclidean distance: straight-line distance.

$$||x||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

0.3.4 max-norm

Largest one dimension.

$$||x||_\infty = \max_i |x_i|$$

1 Naive Bayes

1.1 Objective

Given evidence X, predict label Y.

$$\mathbf{Pr}(Y|X_1, \cdots, X_d) = \frac{\mathbf{Pr}(Y)\mathbf{Pr}(X_1, \cdots, X_d|Y)}{\mathbf{Pr}(X_1, \cdots, X_d)} = \frac{\mathbf{Pr}(Y)\mathbf{Pr}(X_1, \cdots, X_d|Y)}{\sum_{y \in [k]} \mathbf{Pr}(X_1, \cdots, X_d|Y = y)\mathbf{Pr}(Y = y)}$$

Since given evidence X, all labels Y has same denominator $\mathbf{Pr}(X_1, \cdots, X_d)$, the objective becomes

$$\mathbf{Pr}(Y|X_1, \cdots, X_d) \propto \mathbf{Pr}(Y)\mathbf{Pr}(X_1, \cdots, X_d|Y) \tag{2}$$

To predict based on (X, Y), $\mathbf{Pr}(Y)$ and $\mathbf{Pr}(X_1, \cdots, X_d|Y)$ are directly come from data.

1.2 Naive Bayes Assumption

All feature attributes are statistically independent conditioned on label.

$$\mathbf{Pr}(X_1, \cdots, X_d|Y) = \prod_{i=1}^{d} \mathbf{Pr}(X_i = x_i|Y = y) \tag{3}$$

1.3 Algorithm

Combining (2), (3), objective becomes

$$\mathbf{Pr}(Y|X_1, \cdots, X_d) \propto \mathbf{Pr}(Y) \prod_{i=1}^{d} \mathbf{Pr}(X_i = x_i|Y = y) = p_y \prod_{i=1}^{d} p_{iy}$$

During training time, estimate probabilities by counting.

- c_y: number of samples with label y.
- c_{iy}: for each feature X_i, number of samples with label y and $X_i = 1$.
- $p_y = \frac{c_y}{n}$, $p_{iy} = \frac{c_{iy}}{c_y}$.

Only store p_y, p_{iy} for future inference.
During testing time, when $x_i \in \{0, 1\}$

$$\hat{y} = \mathbf{arg}\max_{y\in[k]} p_y \prod_{i=1}^{d} p_{iy}^{x_i}(1 - p_{iy})^{1-x_i} \tag{4}$$

To avoid computation reaching precision limit,

$$\hat{y} = \mathbf{arg}\max_{y\in[k]} \log p_y + \sum_{i=1}^{d} x_i \log p_{iy} + (1 - x_i)\log(1 - p_{iy})$$

1.4 Laplace Smoothing

Add pseudo-count $\lambda > 0$ for all out-of-domain (OOD) features x and labels y.
During training time

$$p_y = \frac{c_y + \lambda}{n + \lambda k}$$

$$p_{iy} = \frac{c_{iy} + \lambda}{c_y + \lambda|X_i|}$$

E.g., $X_1 = \{0, 1, \text{OOD}\}$, $Y = \{0, 1, \text{OOD}\}$, then

$$p_{y=0} = \frac{c_0 + \lambda}{c_0 + c_1 + 3\lambda} \qquad p_{y=\text{OOD}} = \frac{\lambda}{c_0 + c_1 + 3\lambda}$$

$$p_{xy=00} = \frac{c_{00} + \lambda}{c_0 + 3\lambda} \qquad p_{xy=OOD,0} = \frac{\lambda}{c_0 + 3\lambda}$$

where $c_0 = c_{00} + c_{01}$

1.5 Binary Classification

For $k = 2$, the decision rule is predicting $\hat{y} = 1$ when

$$\log p_1 + \sum_{i=1}^{d} x_i \log p_{i1} + (1 - x_i) \log(1 - p_{i1}) \geq \log p_0 + \sum_{i=1}^{d} x_i \log p_{i0} + (1 - x_i) \log(1 - p_{i0})$$

$$\log \frac{p_1}{p_0} + \sum_{i=1}^{d} \left(x_i \log \frac{p_{i1}}{p_{i0}} + (1 - x_i) \frac{1 - p_{i1}}{1 - p_{i0}} \right) \geq 0$$

which is a linear classifier w.r.t X.
(See Github:CS440/ECE448-MP2 for more details)

1.6 Gaussian Classification

Extend Binary Classification case to continuous type, assume Gaussian as underlying P(X|Y) distribution.

$$P(y = +1|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y = +1)p}{P(\boldsymbol{x})}$$

$$P(y = -1|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y = -1)(1 - p)}{P(\boldsymbol{x})}$$

$$\frac{P(y = -1|\boldsymbol{x})}{P(y = +1|\boldsymbol{x})} = \frac{P(\boldsymbol{x}|y = -1)(1 - p)}{P(\boldsymbol{x}|y = +1)p}$$

Express $P(y = +1|\boldsymbol{x})$ in terms of total probability,

$$P(y = +1|\boldsymbol{x}) = \frac{P(y = +1|\boldsymbol{x})}{P(y = +1|\boldsymbol{x}) + P(y = -1|\boldsymbol{x})} = \frac{1}{1 + \frac{P(y=-1|\boldsymbol{x})}{P(y=+1|\boldsymbol{x})}}$$

$$P(y = +1|\boldsymbol{x}) = \frac{1}{1 + \frac{P(\boldsymbol{x}|y=-1)(1-p)}{P(\boldsymbol{x}|y=+1)p}} = \frac{1}{1 + \exp(\log \frac{P(\boldsymbol{x}|y=-1)(1-p)}{P(\boldsymbol{x}|y=+1)p})} = \frac{1}{1 + \exp(\log \frac{A}{B})}$$

with $A = P(\boldsymbol{x}|y = -1)(1 - p)$, $B = P(\boldsymbol{x}|y = +1)p$.

$$\log(A) = \log(\prod_{i=1}^{d} P(x_i|y = -1)(1 - p)) = \sum_{i=1}^{d} \log(P(x_i|y = -1)) + \log(1 - p)$$

Fitting in the Gaussian distribution,

$$\log(A) = \sum_{i=1}^{d} \log(\frac{1}{\sqrt{2\pi}} \exp(\frac{-(x_i - \mu_{-,i})^2}{2})) + \log(1-p) = d \log(\frac{1}{\sqrt{2\pi}}) - \frac{1}{2} \sum_{i=1}^{d} (x_i - \mu_{-,i})^2 + \log(1-p)$$

Transform to matrix form,

$$\log(A) = d\log(\frac{1}{\sqrt{2\pi}}) - \frac{1}{2}(\boldsymbol{X} - \boldsymbol{\mu}_-)^T(\boldsymbol{X} - \boldsymbol{\mu}_-) + \log(1-p)$$

$$= d\log(\frac{1}{\sqrt{2\pi}}) + \log(1-p) - \frac{1}{2}(\boldsymbol{X^T X - 2\mu_- X + \mu_-^T \mu_-})$$

Similarly for $\log(B)$,

$$\log(B) = d\log(\frac{1}{\sqrt{2\pi}}) + \log(p) - \frac{1}{2}(\boldsymbol{X^T X - 2\mu_+ X + \mu_+^T \mu_+})$$

Combining these two with property of logarithm function,

$$\log(\frac{A}{B}) = \log(A) - \log(B) = \log(1-p) - \log(p) - \frac{1}{2}(\boldsymbol{2(\mu_+ - \mu_-)X + \mu_-^T\mu_- - \mu_+^T\mu_+})$$

$$= -\boldsymbol{(\mu_+ - \mu_-)X} + \log(1-p) - \log(p) - \frac{1}{2}(\boldsymbol{\mu_-^T\mu_- - \mu_+^T\mu_+}) = \boldsymbol{w^T x} + b$$

with $w^T = -\boldsymbol{(\mu_+ - \mu_-)}$, $b = \log(1-p) - \log(p) - \frac{1}{2}(\boldsymbol{\mu_-^T\mu_- - \mu_+^T\mu_+})$
Combining the two cases,

$$P(y|\boldsymbol{x}) = \frac{1}{1 + \exp(y(\boldsymbol{w^T x} + b))}$$

2 Linear Regression

2.1 Objective

Given evidence X, predict Y by linear function.

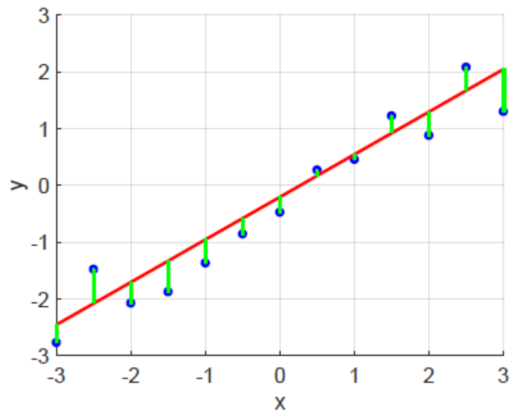
$$\hat{y} = w_1 \cdot x + w_2 \tag{5}$$

Then the objective is the difference between prediction and labels.

$$\arg\min_{w_1, w_2} \frac{1}{2}\sum_{i=1}^{N}(y - \hat{y})^2 = \frac{1}{2}\sum_{i=1}^{N}(y - w_1 \cdot x - w_2)^2$$

Loss function in matrix form (Quadratic Loss):

$$L = \frac{1}{2}||\boldsymbol{Y} - \boldsymbol{X}^T\boldsymbol{w}||_2^2$$



2.2 Closed Form Solution

Take derivative

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{X}\boldsymbol{X}^T\boldsymbol{w}^* - \boldsymbol{X}\boldsymbol{Y} = 0$$

$$\boldsymbol{w}^* = (\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}\boldsymbol{Y}$$

Proof of $\boldsymbol{X}\boldsymbol{X}^T$ invertible:
According to the SVD of $\boldsymbol{X}$,

$$\boldsymbol{X} = \boldsymbol{U\Sigma V^T}$$

$$\boldsymbol{XX^T} = \boldsymbol{(U\Sigma V^T)(U\Sigma V^T)^T} = \boldsymbol{U\Sigma V^T V\Sigma^T U^T}$$

Since $\boldsymbol{V}$ is unitary, Σ is diagonal matrix, $\boldsymbol{V^TV} = \boldsymbol{I}$.

$$\boldsymbol{XX^T} = \boldsymbol{U\Sigma\Sigma^T U^T} = \boldsymbol{U\Sigma^2 U^T}$$

Since $\boldsymbol{X}$ has size $n \times d$ and rank n, $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ should have size $n \times n$, and

$$rank(\boldsymbol{U}) = rank(\boldsymbol{U^T}) = rank(\boldsymbol{\Sigma}) = n$$

Therefore, $\boldsymbol{XX^T}$ has size $n \times n$ and

$$rank(\boldsymbol{XX^T}) = n$$

and so $\boldsymbol{XX^T}$ is invertible.
During Training time, $\boldsymbol{w}$ is calculated from X, Y and stored.
During Inference time, apply (5).

(See Github:CS446/ECE449-MP1 for more details)

2.3 Kernel Method

Enlarge feature space X can help better fitting when underfit. E.g.,

$$y = w_2 x^2 + w_1 x + w_0$$

$$\mathbf{\Phi} = \begin{bmatrix} x_1^2 & x_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n^2 & x_n & 1 \end{bmatrix}$$

$$\boldsymbol{w}^* = (\mathbf{\Phi}\mathbf{\Phi}^T)^{-1}\mathbf{\Phi}\boldsymbol{Y}$$

2.4 Probabilistic Derivation

Given evidence X, predict Y by conditioned probability based on Gaussian distribution.

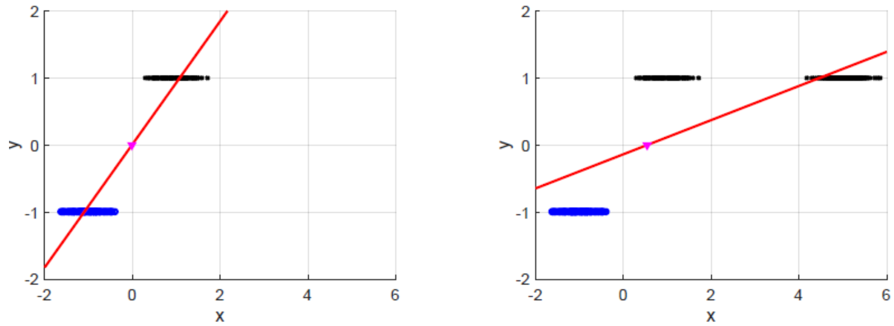
$$p(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left(y - w^T\phi(x)\right)^2\right)$$

then objective becomes MLE on i.i.d. samples.

$$\begin{aligned} \arg\max_w p(D) = \arg\max_w \prod_i^N p(y_i|x_i) = \arg\max_w \sum_i^N \log p(y_i|x_i) \\ = \arg\max_w \sum_i^N \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left(y_i - w^T\phi(x_i)\right)^2\right) \\ = \arg\max_w \sum_i^N \left(-\frac{1}{2\sigma^2}\left(y_i - w^T\phi(x_i)\right)^2\right) + C \\ = \arg\min_w \sum_i^N \left(y_i - w^T\phi(x_i)\right)^2 \end{aligned}$$

2.5 Limitations

Bad performance on Classification Problems: boundary shifting.



3 Logistic Regression

3.1 Objective

Given evidence X, predict class label Y with sigmoid function. For binary classification,

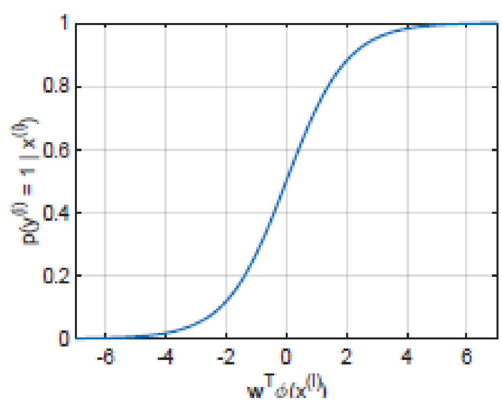
$$p(y=1|x) = \frac{1}{1+\exp(-\boldsymbol{w}^T\phi(x))}$$

$$p(y=-1|x) = 1 - p(y=1|x) = \frac{1}{1+\exp(\boldsymbol{w}^T\phi(x))}$$

Putting together,

$$p(y|x) = \frac{1}{1+\exp(-y\boldsymbol{w}^T\phi(x))} \tag{6}$$

Then objective is maximizing the probability of predicting true label given evidence.



$$\arg\max_w \prod_{(x,y)\in D} p(y|x) = \arg\min_w \sum_{(x,y)\in D} -\log p(y|x)$$

$$= \arg\min_{w} \sum_{(x,y)\in D} \log(1+\exp(-y\boldsymbol{w}^T\phi(x)))$$

Loss function:

$$L = \sum_{(x,y)\in D} \log(1+\exp(-y\boldsymbol{w}^T\phi(x)))$$

3.2 Solution

Take gradient,

$$\nabla_w L = \sum_{(x,y)\in D} \frac{-y\exp(-y\boldsymbol{w}^T\phi(x))\phi(x)}{1+\exp(-y\boldsymbol{w}^T\phi(x))}$$

During Training time, by gradient descent with stepsize α,

$$w_{t+1} \leftarrow w_t - \alpha\nabla_w L$$

During Inference time, apply (6).
(See Github:CS444-MP1 for more details)

4 SVM

4.1 Linear Separable

4.1.1 Objective

Given evidence X, predict decision boundary to classify to class Y.
Motivation: find the boundary that maximize the distance to nearest data points from two classes.

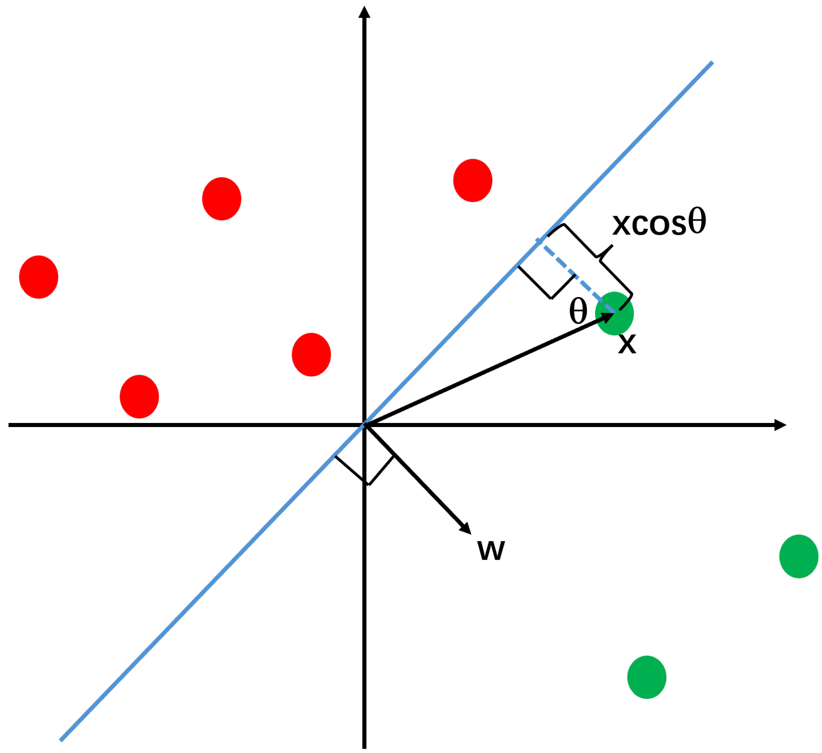
$$\max_{w\in\mathbb{R}^d} \min_{i\in[n]} \frac{y_i w^T x_i}{||w||_2} \tag{7}$$

Derivation: Set blue line as boundary, w is vertical to boundary pointing to one class, the distance from nearest point to boundary is

$$\textbf{Dist} = \mathbf{x}\cos\theta = \frac{||w||_2||x||_2\cos\theta}{||w||_2} = \frac{w^T x}{||w||_2}$$

Taken class label into account, positive distance for one class and negative for the other, with $y = \{-1, +1\}$,

$$\textbf{Dist} = \frac{y w^T x}{||w||_2}$$



Simplify (7),

$$\max_{w\in\mathbb{R}^d}\min_{i\in[n]}\frac{y_iw^Tx_i}{||w||_2}=\max_{w\in\mathbb{R}^d}\frac{\min_{i\in[n]}y_iw^Tx_i}{||w||_2}=\max_{w\in\mathbb{R}^d}\frac{C\min_{i\in[n]}y_iw^Tx_i}{C||w||_2}$$

One can always find C to multiply into w that makes $\min_{i\in[n]}y_iw^Tx_i=1$, so translate to optimization problem

$$\max_{w\in\mathbb{R}^d}\frac{1}{||w||_2}=\min_{w\in\mathbb{R}^d}\frac{1}{2}||w||_2,\quad s.t.\quad y_iw^Tx_i\geq 1,\forall i\in[n] \tag{8}$$

(See Github:CS444-MP1 for more details)

4.1.2 Dual Form

By applying Lagrangian multiplier to (8),

$$\min_{w\in\mathbb{R}^d}\max_{\alpha\in\mathbb{R}^n_+}L(w,\alpha)=\frac{1}{2}||w||_2+\sum_{i\in[n]}\alpha_i(1-y_iw^Tx_i)$$

By weak duality property, for any function $f(x, y)$,

$$\min_x \max_y f(x, y) \geq \max_y \min_x f(x, y)$$

By strong duality property, for convex function $f(x, y)$,

$$\min_x \max_y f(x, y) = \max_y \min_x f(x, y)$$

Therefore, Lagrangian becomes Primal problem:

$$P(w) = \max_{\alpha \in \mathbb{R}_+^n} L(w, \alpha)$$

and Dual problem:

$$D(\alpha) = \min_{w \in \mathbb{R}^d} L(w, \alpha)$$

By strong duality,

$$\min_{w \in \mathbb{R}^d} P(w) = \max_{\alpha \in \mathbb{R}_+^n} D(\alpha)$$

For Dual problem,

$$\nabla_w D(\alpha) = \nabla_w \left(\frac{1}{2} ||w||_2 + \sum_{i \in [n]} \alpha_i (1 - y_i w^T x_i) \right) = 0$$

$$w = \sum_{i \in [n]} \alpha_i y_i x_i$$

Plugging back get

$$D(\alpha) = \sum_{i \in [n]} \alpha_i - \frac{1}{2} \sum_{i,j \in [n]} \alpha_i \alpha_j y_i y_j x_i^T x_j = \mathbf{1}_n^T \alpha - \frac{1}{2} \alpha^T K \alpha$$

where $K_{ij} = (y_i x_i)^T (y_j x_j)$.
By solving $\max_{\alpha^* \in \mathbb{R}_+^n} D(\alpha^*)$ as a quadratic program,

$$w^* = \sum_{i \in [n]} \alpha_i^* y_i x_i$$

4.1.3 Kernel Method

Replacing evidence x in (8) with kernel features $\phi(x)$,

$$\min_{w\in\mathbb{R}^d} \frac{1}{2}||w||_2, \quad s.t. \quad y_i w^T \phi(x_i) \geq 1, \forall i \in [n]$$

Dual form becomes

$$D(\alpha) = \sum_{i\in[n]} \alpha_i - \frac{1}{2} \sum_{i,j\in[n]} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) = \mathbf{1}_n^T \alpha - \frac{1}{2}\alpha^T K \alpha$$

where $K_{ij} = (y_i \phi(x_i))^T (y_j \phi(x_j))$.
E.g., Gaussian Kernel:

$$k(x, x^{'}) = \phi(x)^T \phi(x^{'}) = \exp\left(-\frac{||x - x^{'}||_2^2}{2\sigma^2}\right)$$

(See Github:CS446/ECE449-MP2 for more details)

4.2 Not Linear Separable

4.2.1 Objective

Introduce slack variables $\xi_i \geq 0$ as smallest pseudo movement for each data points (x_i, y_i) to make the dataset linear separable.

$$\min_{w\in\mathbb{R}^d, \xi\in\mathbb{R}^n\geq 0} \frac{1}{2}||w||_2 + C\sum_{i\in[n]} \xi_i = \min_{w\in\mathbb{R}^d, \xi\in\mathbb{R}^n\geq 0} \sum_{i\in[n]} \xi_i + \frac{\lambda}{2}||w||_2, \quad s.t. \quad y_i w^T x_i \geq 1 - \xi_i, \forall i \in [n] \tag{9}$$

Hinge Loss:

$$L_{\text{hinge}} = \max\{0, 1 - t\}$$

Support Vector Machine (SVM):

$$\min_{w\in\mathbb{R}^d} \sum_{i\in[n]} L_{\text{hinge}}(y_i w^T x_i) + \frac{\lambda}{2}||w||_2$$

4.2.2 Dual Form

Apply Lagrangian to (9).

$$\arg \underset{\boldsymbol{\alpha \geq 0, \beta \geq 0}}{max} \arg \underset{\boldsymbol{w,\xi}}{min}[\frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\alpha_i(1-\xi_i - y_i\boldsymbol{w^T x_i}) + \sum_{i=1}^{N}\beta_i(-\xi_i)]$$

Set derivative of $\boldsymbol{w^T} = 0$,

$$\nabla_w[\frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\alpha_i(1-\xi_i - y_i\boldsymbol{w^T x_i}) + \sum_{i=1}^{N}\beta_i(-\xi_i)] = \boldsymbol{w} - \Sigma_{i=1}^{N}\alpha_i y_i \boldsymbol{x_i} = 0$$

$$\boldsymbol{w}^* = \Sigma_{i=1}^{N}\alpha_i y_i \boldsymbol{x_i}$$

Set derivative of $\boldsymbol{\xi} = 0$,

$$\nabla_\xi[\frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\alpha_i(1-\xi_i - y_i\boldsymbol{w^T x_i}) + \sum_{i=1}^{N}\beta_i(-\xi_i)] = CN - \Sigma_{i=1}^{N}\alpha_i - \Sigma_{i=1}^{N}\beta_i = 0$$

$$\beta_i = C - \alpha_i$$

Plug $\boldsymbol{w^*}$ and $\boldsymbol{\beta}$ back, get dual form

$$\arg \underset{\boldsymbol{0 \leq \alpha \leq C}}{max} \frac{1}{2}||\Sigma_{i=1}^{N}\alpha_i y_i \boldsymbol{x_i}||^2 + C\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(\Sigma_{i=1}^{N}\alpha_i y_i \boldsymbol{x_i})\boldsymbol{x_i}) + \sum_{i=1}^{N}(\alpha_i - C)\xi_i$$

$$\arg \underset{\boldsymbol{0 \leq \alpha \leq C}}{max} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(\boldsymbol{x_i^T x_j}) + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(\boldsymbol{x_i^T x_j})$$

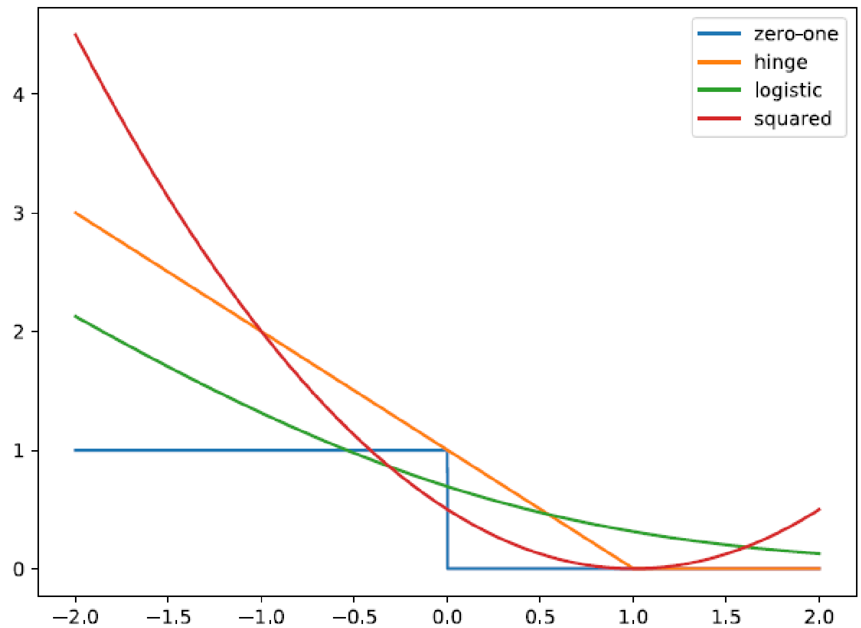
$$\arg \underset{\boldsymbol{0 \leq \alpha \leq C}}{max} \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(\boldsymbol{x_i^T x_j})$$

5 Regularization

5.1 Losses

5.2 Ridge Regularization

To avoid overfitting to outliers, regularization term $\frac{\lambda}{2}||w||_2$ is added as penalty on loss functions.



Linear Regression:

$$L = \sum_{i \in [n]} (y_i - w^T x_i)^2 + \frac{\lambda}{2} ||w||_2$$

Logistic Regression:

$$L = \sum_{i \in [n]} L_{log}(y_i w^T x_i) + \frac{\lambda}{2} ||w||_2$$

SVM:

$$L = \sum_{i \in [n]} L_{\text{hinge}}(y_i w^T x_i) + \frac{\lambda}{2} ||w||_2$$

This regularization term can avoid large magnitude of all features of evidence x, so that if some outliers have several strong features while other common data points do not have, it will be less likely overfitting.

5.3 Lasso Regularization

Add regularization term $\lambda \sum_{i=1}^{d} |w_i|$. Instead of Ridge Regularization's sensitive to any huge w_i, it allows huge w_i and balanced out by some zero w_i, so it has benefits like sparsity, reducing model complexity to store.

6 Decision Tree

6.1 Objective

Given evidence and labels (X, Y), build a binary tree that each tree node is a splitting rule on one feature X_i, each leaf node corresponds to a label y.

6.2 Information Gain

Metric for finding best split on tree node.

$$IG(D, f) = I(D) - I(D|f) = I(D) - \sum_{j=1}^{N} \frac{|D_j|}{|D|} I(Dj)$$

larger IG means larger relative group purity, which is better.
$I(D)$ is the measurement of uncertainty within dataset D. Larger $I(D)$ means more uncertainty.

- Entropy
$$I(D) = -\sum_{c=1}^{C} p(c|D) \log p(c|D)$$
- Gini Impurity
$$I(D) = 1 - \sum_{c=1}^{C} p(c|D)^2$$
- Classification Error
$$I(D) = 1 - \max_{c \in \{1, \cdots, C\}} p(c|D)$$

6.3 Bagging & Random Forest

Assumption: independent errors when combining trees.
Bagging: running T times for each node to learn T classifier. For each time, randomly pick n examples with replacement from training data to determine split rule. Finally infer by a majority vote over all T classifiers.
Random Forest: running T times for each node to learn T classifier. For each time, randomly pick n examples with replacement from training data to determine split rule. Besides, randomly pick a subset of features to choose split rule from. Finally infer by a majority vote over all T classifiers.

6.4 Adaptive Boosting

input Training data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$ from $\mathcal{X} \times \{-1, +1\}$.

1: **initialize** $\gamma_1^{(i)} := 1/n$ for each $i = 1, 2, \dots, n$ (a probability distribution).

2: **for** $t = 1, 2, \dots, T$ **do**

3: Get weak classifier f_t from γ_t-weighted samples.

4: Update weights:

$$z_t := \sum_{i=1}^{n} \gamma_t^{(i)} \cdot y^{(i)} f_t(\mathbf{x}^{(i)}) \in [-1, +1] \quad \text{(weighted error rate)}$$

$$\alpha_t := \frac{1}{2} \ln \frac{1 + z_t}{1 - z_t} \in \mathbb{R} \quad \text{(weight of } f_t\text{)}$$

$$\gamma_{t+1}^{(i)} := \gamma_t^{(i)} \exp\left(-\alpha_t \cdot y^{(i)} f_t(\mathbf{x}^{(i)})\right) / Z_t \quad \text{for each } i \quad \text{(sample weight)},$$

where $Z_t > 0$ is normalizer that makes D_{t+1} a probability distribution.

5: **end for**

6: **return** Final classifier $\operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t \cdot f_t(x)\right)$.

Interpretation:

- Initialize with same weights on all samples.
- Find first split rule.
- Adjust weights z of samples based on accuracy. (larger weights on misclassified samples)
- Compute classifier weight α based on accuracy.
- Repeat for next split until error rate within range.
- Final classifier is linear combination of all splits with weights α

7 Deep Learning

See another file.