Hongyi Zhang

Lecture 1: Structural Risk Minimization (SRM)

Objective: minimizing a combination of loss function and model complexity penalty.

1.1 Machine Learning Problem:

dataset $D = \{(\vec{x}_i, y_i)\}_{i=1,\cdots,n}$

goal to learn $y = h(\vec{x})$ s.t. $\begin{cases} h(\vec{x}_i) = y_i, \forall i = 1,\cdots,n \\ h(\vec{x}^{test}) = y^{test}, \text{for unseen test data} \end{cases}$

$y = h(f(\vec{x}))$ with $f: \vec{x} \to R$ < classification $\{-1, 1\}$, $h(a) = \text{sign}(a)$; regression $h(a) = I(a)$

Empirical risk minimization (ERM) (minimize training error)

$$\min_h R_{emp}[h] = \min_h \frac{1}{n}\sum_{i=1}^{n} l(h(\vec{x}_i), y_i)$$ { tends to overfit training data; a simpler model is prefered (overfit↓)

1.2 Structural Risk Minimization (SRM)

balance fitting the training data against model complexity

$$\min_{\vec{w}} J(\vec{w}) = \min_{\vec{w}} \frac{1}{n}\sum_{i=1}^{n} l(h_{\vec{w}}(\vec{x}_i), y_i) + \lambda r(\vec{w})$$

$\vec{w}$: model parameter; l: training loss; r: regularizer; λ: hyperparameter (CV)

2. Loss Function

$l(f_{\vec{w}}(\vec{x}_i), y_i)$

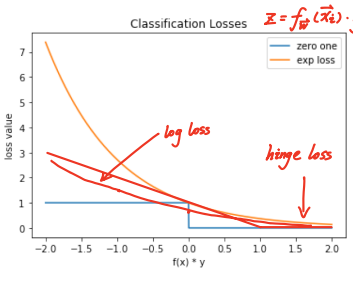
2.1 Binary Classification Loss Functions ($y \in \{-1, +1\}$)

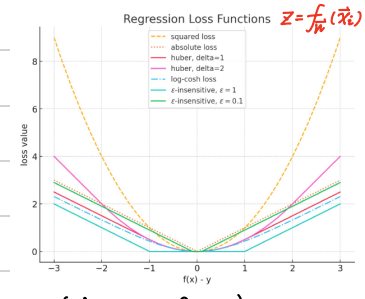
zero-one loss: $\delta(h_{\vec{w}}(\vec{x}_i) = y_i)$ non-continuous, impractical

hinge loss: $\max[1 - f_{\vec{w}}(\vec{x}_i)\cdot y_i, 0]^p$ standard SVM ($p=1$)

log loss: $\log(1 + e^{-f_{\vec{w}}(\vec{x}_i)\cdot y_i})$ logistic regression

exponential loss: $e^{-f_{\vec{w}}(\vec{x}_i)\cdot y_i}$ AdaBoost, aggressive, sensitive





2.2 Regression Loss Functions ($h_{\vec{w}}(\vec{x}) = f_{\vec{w}}(\vec{x})$)

Squared loss: $(f_{\vec{w}}(\vec{x}_i) - y_i)^2$ Ordinary Least Squares (OLS)

Absolute loss: $|f_{\vec{w}}(\vec{x}_i) - y_i|$ not differentiable at 0

(Smooth AL) Huber loss: $\begin{cases} \frac{1}{2}z_i^2, & \text{if } |z_i| < \delta \\ \delta(|z_i| - \frac{\delta}{2}), & \text{otherwise} \end{cases}$ where $z_i = f_{\vec{w}}(\vec{x}_i) - y_i$, δ - constant

Log-Cosh loss: $\log(\cosh(f_{\vec{w}}(\vec{x}_i) - y_i))$ where $\cosh = \frac{e^x + e^{-x}}{2}$

ε-Insensitive loss: $\begin{cases} 0, & \text{if } |z_i| < \varepsilon \\ |z_i| - \varepsilon, & \text{otherwise} \end{cases}$ where $z_i = f_{\vec{w}}(\vec{x}_i) - y_i$

3. Regularizer

$$\min_{\vec{w},b} \sum_{i=1}^{n} l(\vec{w}^T\vec{x}_i + b, y_i) + \lambda r(\vec{w}) \Leftrightarrow \min_{\vec{w},b} \sum_{i=1}^{n} l(\vec{w}^T\vec{x}_i + b, y_i) \text{ subject to } r(\vec{w}) \le B$$

l_2-regularizer: $r(\vec{w}) = \vec{w}^T\vec{w} = \|\vec{w}\|_2^2 = \sum_{i=1}^{n} w_i^2$

l_1-regularizer: $r(\vec{w}) = \|\vec{w}\|_1 = \sum_{i=1}^{n} |w_i|$ l_p-norm: $\|\vec{w}\|_p = \left(\sum_{i=1}^{n} |w_i|^p\right)^{\frac{1}{p}}$

Elastic Net: $\alpha\|\vec{w}\|_1 + (1-\alpha)\|\vec{w}\|_2^2$, $\alpha \in [0,1)$

4 Famous SRM model (Notation: $X \in R^{d \times n} = [\vec{x}_1, \cdots, \vec{x}_n]$, $\vec{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$)

Ordinary (OLS) Least Square: $\min_{\vec{w}} \frac{1}{n}\sum_{i=1}^{n} (\vec{w}^T\vec{x}_i - y_i)^2 \xrightarrow{\frac{\partial L}{\partial \vec{w}} = 0} \vec{w} = (XX^T)^{-1}X\vec{y}$

$(X^T\vec{w} - \vec{y})^T(X^T\vec{w} - \vec{y})$

Ridge Regression: $\min_{\vec{w}} \frac{1}{n}\sum_{i=1}^{n} (\vec{w}^T\vec{x}_i - y_i)^2 + \lambda\|\vec{w}\|_2^2 \to \vec{w} = (XX^T + n\lambda I)^{-1}X\vec{y}$

Lasso: $\min_{\vec{w}} \frac{1}{n}\sum_{i=1}^{n} (\vec{w}^T\vec{x}_i - y_i)^2 + \lambda\|\vec{w}\|_1$

Logistic Regression: $\min_{\vec{w}} \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i(\vec{w}^T\vec{x}_i + b)})$ GD or Newton

Lecture 2: Optimization for Machine Learning

2. Gradient Descent GD Algorithm:

Initialize $\vec{w}_0$
repeat
  $\vec{w}_{t+1} \leftarrow \vec{w}_t + s$
until $\|\vec{w}_{t+1} - \vec{w}_t\| \le \varepsilon$

learning objective $l(\vec{w})$, gradient step s

$l(\vec{w}_{t+1}) \le l(\vec{w}_t) \leftrightarrow l(\vec{w}_t + s) \le l(\vec{w}_t)$

If $\|s\|$ is small, Taylor expansion: $l(\vec{w}+s) = l(\vec{w}) + \nabla l(\vec{w})^T s + \frac{1}{2}s^T\nabla^2 l(\vec{w})s + \cdots$

Gradient descent use $\nabla l(\vec{w})^T$ $l(\vec{w}_t + s) \approx l(\vec{w}_t) + \nabla l(\vec{w})^T s \le l(\vec{w}_t)$

$\Rightarrow \nabla l(\vec{w})^T s \le 0$ $(g(\vec{w}) = \nabla l(\vec{w})^T) \Rightarrow s = -\alpha g(\vec{w}_t)$ learning rate or stepsize

sufficient small - gd converge; too large - gd diverge

3. Newton's Method

$l(\vec{w}+s) = l(\vec{w}) + \nabla l(\vec{w})^T s + \frac{1}{2}s^T\nabla^2 l(\vec{w})s$ $l(\vec{w})$ is twice differentiable

$\nabla^2 l(\vec{w}) = H(\vec{w})$ $H_{ij} = \frac{\partial^2 l}{\partial \vec{w}_i \partial \vec{w}_j}$ symmetric, positive semi-definite

choose s, we want $l(w+s)$ to be small (*)

$s = \arg\min_s l(\vec{w}) + \nabla l(\vec{w})^T s + \frac{1}{2}s^T\nabla^2 l(\vec{w})s = \arg\min_s l(\vec{w}) + g(\vec{w})^T s + \frac{1}{2}s^T H(\vec{w})s$

$\frac{\partial(*)}{\partial s} = 0 \Rightarrow g(\vec{w}) + H(\vec{w})s = 0 \Rightarrow s = -H(\vec{w})^{-1}g(\vec{w})$

Lecture 3: Estimating Probabilities from Data

Discriminative learning, estimate $p(y|\vec{x})$ directly

Generative learning, estimate $p(\vec{x}, y) = p(\vec{x}|y)p(y)$

可用 Bayes rule 计算分类 $p(y|\vec{x}) = \frac{p(\vec{x}|y)p(y)}{p(\vec{x})}$

2. Maximum Likelihood Estimation (MLE)

(the goal is to choose θ s.t. the observed data D is most likely)

MLE principle: Find θ max likelihood $p(D|\theta)$: $\hat{\theta}_{MLE} = \arg\max_\theta p(D|\theta)$

(disad: MLE overfit data if n small; x correct model, MLE terrible way)

3. The Bayesian Way and Maximum-a-posterior Estimation (MAP)

Bayes rules: $p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$

MAP principle: Find θ max $p(\theta|D)$

$\hat{\theta}_{MAP} = \arg\max_\theta p(\theta|D) = \arg\max_\theta \frac{p(D|\theta)p(\theta)}{p(D)} = \arg\max_\theta \log p(D|\theta) + \log p(\theta)$

4. True Bayesian Approach

Posterior Mean: $\hat{\theta}_{mean} = E(\theta|D) = \int_\theta \theta\, p(\theta|D)\, d\theta$

Lecture 4: MLE and MAP for Discriminative Supervised Learning

(Discriminative learning, estimate $p(y|\vec{x})$ directly)

Assumption for Discriminative Supervised Learning:

(1) $\vec{x}_i$ are known → $\vec{x}_i$ independent of the model parameters $\vec{w}$

→ $p(X|\vec{w}) = p(X)$, $p(\vec{w}|X) = p(\vec{w})$

(2) y_i are independent given the input feature $\vec{x}_i$ and $\vec{w}$

Goal: estimate $\vec{w}$ from $D = \{(\vec{x}_i, y_i)\}$ using joint conditional likelihood $p(\vec{y}|X, \vec{w})$

Lemma: Max likelihood $p(D|\vec{w}) = p(\vec{y}, X|\vec{w})$ is equivalent to max the joint conditional likelihood $p(\vec{y}|X,\vec{w})$.

Since $p(\vec{y}, X|\vec{w}) = p(\vec{y}|X,\vec{w}) \cdot p(X|\vec{w}) = p(\vec{y}|X,\vec{w}) \cdot p(X)$ ← not depends on $\vec{w}$

✓ MLE: choose $\vec{w}$ to max $p(\vec{y}|X,\vec{w})$ — log-likelihood

$\hat{\vec{w}}_{MLE} = \arg\max_{\vec{w}} p(\vec{y}|X,\vec{w}) \overset{(1)}{=} \arg\max_{\vec{w}} \prod_{i=1}^{n} p(y_i|\vec{x}_i,\vec{w}) = \arg\max_{\vec{w}} \sum_{i=1}^{n} \log p(y_i|\vec{x}_i,\vec{w})$

✓ MAP: choose $\vec{w}$ to max $p(\vec{w}|X,\vec{y})$

$\hat{\vec{w}}_{MAP} = \arg\max_{\vec{w}} p(\vec{w}|X,\vec{y}) = \arg\max_{\vec{w}} p(X,\vec{y}|\vec{w}) p(\vec{w})$

$= \arg\max_{\vec{w}} \underbrace{p(\vec{y}|X,\vec{w})}_{\text{likelihood}} \underbrace{p(\vec{w})}_{\text{prior}} = \arg\max_{\vec{w}} \sum_{i=1}^{n} \log p(y_i|\vec{x}_i,\vec{w}) + \log p(\vec{w})$

Lecture 5: Naive Bayes Classifier

$p(\vec{x}|y,D)$

Bayes rule $p(y|\vec{x}) = \dfrac{p(\vec{x}|y)p(y)}{p(\vec{x})}$ — $p(y|D)$ — $p(\vec{x}|D)$

2. Naive Bayes

2.1 Estimate $p(y)$ counting

binary classification $\hat{\pi}_{y=+1} = p(y=+1)$, $\hat{\pi}_{y=-1} = p(y=-1)$

multi-class classification $\hat{\pi}_c = p(y=c) = \frac{1}{n}\sum_{i=1}^{n} I(y_i=c)$

2.2 Estimate $p(\vec{x}|y)$

✓ Naive Bayes Assumption: features are independent given the label

$p(\vec{x}|y) = \prod_{\alpha=1}^{d} p([\vec{x}]_\alpha | y)$

$y = h(\vec{x}) = \arg\max_y p(y|\vec{x}) = \arg\max_y \dfrac{p(\vec{x}|y)p(y)}{p(\vec{x})}$

$= \arg\max_y p(\vec{x}|y)p(y) = \arg\max_y \prod_{\alpha=1}^{d} p([\vec{x}]_\alpha|y)p(y)$

Case 1: Categorical Features; Case 2: Multinomial Features

Case 3: Continuous Features (Gaussian Naive Bayes)

$p([\vec{x}]_\alpha|y=c) = N(\mu_{\alpha c}, \sigma^2_{\alpha c}) = \dfrac{1}{\sqrt{2\pi\sigma^2_{\alpha c}}} \exp\left(-\dfrac{(x-\mu_c)^2}{2\sigma^2_{\alpha c}}\right)$

$\hat{\mu}_{\alpha c} \leftarrow \frac{1}{n_c}\sum_{i=1}^{n} I(y_i=c)\vec{x}_i$; $\sigma^2_{\alpha c} \leftarrow \frac{1}{n_c}\sum_{i=1}^{n} I(y_i=c)(\vec{x}_i - \hat{\vec{\mu}}_c)(\vec{x}_i - \hat{\vec{\mu}}_c)$

where $n_c = \sum_{i=1}^{n} I(y_i=c)$

without naive assumption: $\hat{\vec{\mu}}_c \leftarrow \begin{pmatrix}\hat{\mu}_{1c}\\ \hat{\mu}_{2c}\end{pmatrix}$, $\hat{\Sigma}_c \leftarrow \begin{pmatrix}\hat{\sigma}^2_{1c} & \hat{\sigma}_{12c}\\ \hat{\sigma}_{21c} & \hat{\sigma}^2_{2c}\end{pmatrix}$

with naive assumption: $\hat{\vec{\mu}}_c \leftarrow \begin{pmatrix}\hat{\mu}_{1c}\\ \hat{\mu}_{2c}\end{pmatrix}$, $\hat{\Sigma}_c = \begin{pmatrix}\sigma^2_{1c} & 0\\ 0 & \sigma^2_{2c}\end{pmatrix}$

✓ 2.3 NB is a linear classifier → $h(\vec{x}) = +1 \iff \vec{w}^T\vec{x} + b > 0$

Suppose $y_i \in \{-1,+1\}$:

features-binary $h(\vec{x}) = \arg\max_c p(y=c)\prod_{\alpha=1}^{d} p([\vec{x}]_\alpha|y=c) = \text{sign}(\vec{w}_{BER}^T\vec{x} + b_{BER})$

features-multinomial $h(\vec{x}) = \arg\max_c p(y=c)p(\vec{x}|m,y=c) = \text{sign}(\vec{w}_{multT}^T\vec{x} + b_{multT})$

feature-continuous $p(y|\vec{x}) = \dfrac{1}{1+\exp(-y(\vec{w}_{GNB}^T\vec{x} + b_{GNB}))}$

log odds: $\log\dfrac{p(y=1|\vec{x})}{p(y=-1|\vec{x})} = \vec{w}^T\vec{x}$ (logistic regression)

Multinomial: $P(X_1=x_1,\cdots,X_k=x_k) = \dfrac{n!}{x_1!\cdots x_k!}p_1^{x_1}\cdots p_k^{x_k}$

Gaussian: $f(x) = \dfrac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\dfrac{(x-\mu)^2}{2\sigma^2}\right)$

Bernoulli: $P(X=x|p) = \begin{cases}p, & x=1\\ 1-p, & x=0\end{cases}$

$P(X=x) = p^x(1-p)^{1-x}$, $x \in \{0,1\}$

Binomial Distribution: $P(X=k) = \binom{n}{k}p^k(1-p)^{n-k}$, $k = 0,1,\cdots,n$ ($\frac{n!}{k!(n-k)!}$ → 成功次数)

($p(w|D) \propto p(D|w)p(w)$, Bayes rule)

Lecture A1: Unsupervised Learning - Clustering

1.2 Clustering Given: $D = \{\vec{x}_1,\cdots,\vec{x}_n\} \subset R^d$

2. K-means clustering

Assumption: 1) number of clustering K is known; 2) each input can only belong to a single cluster

Notation: 1) $\vec{z}_i = \begin{pmatrix}0\\ \vdots\\ 1\\ \vdots\\ 0\end{pmatrix} \in R^K$ one-hot coding, 每个$\vec{x}_i$对应一个$\vec{z}_i$

如果数据点$\vec{x}_i$归属于簇α, 则 $[\vec{z}_i]_\alpha = 1$,

2) cluster center (also cluster mean)

$\vec{\mu}_\alpha = \dfrac{\sum_{i=1}^{n}[\vec{z}_i]_\alpha \vec{x}_i}{\sum_{i=1}^{n}[\vec{z}_i]_\alpha}$ ← 属于α簇的所有点的坐标和 / ← 属于α簇的所有点的个数

distance $d_{i\alpha} = \|\vec{x}_i - \vec{\mu}_\alpha\|^2$

2.1 Objective and Algorithm

objective function: $D(\vec{\mu}_1,\cdots,\vec{\mu}_K, \vec{z}_1,\cdots,\vec{z}_n) = \sum_{i=1}^{n}\sum_{\alpha=1}^{K}[\vec{z}_i]_\alpha d_{i\alpha}$

(cluster center, cluster assignment)

K-means clustering Algorithm

Initialize $\vec{\mu}_1,\cdots\vec{\mu}_K$ to distinct data points in D

repeat

for all $i=1,\cdots,n$ do $[\vec{z}_i]_\alpha = \begin{cases}1, & \text{if } \alpha = \arg\min_\alpha \|\vec{x}_i - \vec{\mu}_\alpha\|^2\\ 0, & \text{otherwise}\end{cases}$ (所有数据点; 计算距离 判断属于哪一簇; $\|\vec{x}_i - \vec{\mu}_\alpha\|^2 = d_{i\alpha}$)

end for

for all $\alpha=1,\cdots,K$ do $\vec{\mu}_\alpha = \dfrac{\sum_{i=1}^{n}[\vec{z}_i]_\alpha \vec{x}_i}{\sum_{i=1}^{n}[\vec{z}_i]_\alpha}$

end for

until converge $(\vec{z}_i^{(t)} = \vec{z}_i^{(t+1)}$ or $\vec{\mu}^{(t)} = \vec{\mu}_\alpha^{(t+1)})$

2.3 Choosing the number of Clusters K — Kink

3. Gaussian Mixture Model Clustering

$\sum_{\alpha=1}^{k}[\vec{z}_i]_\alpha = 1$

Define $[\vec{z}_i]_\alpha \in [0,1]$ as the probability that $\vec{x}_i$ belongs to cluster α.

Assumption: data set D comes from a mixture of Gaussian

$\Longrightarrow$ $\vec{x}_i \sim N(\vec{\mu}_\alpha, \Sigma_\alpha) = p(\vec{x}_i|C_\alpha)$ ← each cluster C_α

3.2 Generative Model and GMM Algorithm

Initialize $\vec{\mu}_1,\cdots,\vec{\mu}_K$ to be distinct random data points in D

Initialize $\Sigma_1,\cdots,\Sigma_K$ to $\sigma^2 I$ for any $\sigma > 0$

Initialize $\pi_1,\cdots,\pi_K$ to $\frac{1}{K}$ (uniform prior)

repeat

for all i and α, do

E-step: $[\vec{z}_i]_\alpha = \dfrac{p(\vec{x}_i|\vec{\mu}_\alpha,\Sigma_\alpha)\pi_\alpha}{\sum_{i=1}^{k}p(\vec{x}_i|\vec{\mu}_i,\Sigma_i)\pi_i}$

end for

for $\alpha=1,\cdots,K$, do

M-step: $\vec{\mu}_\alpha = \dfrac{\sum_{i=1}^{n}[\vec{z}_i]_\alpha\vec{x}_i}{\sum_{i=1}^{n}[\vec{z}_i]_\alpha}$ $\Sigma_\alpha = \dfrac{\sum_{i=1}^{n}[\vec{z}_i]_\alpha(\vec{x}_i-\vec{\mu}_\alpha)(\vec{x}_i-\vec{\mu}_\alpha)^T}{\sum_{i=1}^{n}[\vec{z}_i]_\alpha}$

$\pi_\alpha = \frac{1}{n}\sum_{i=1}^{n}[\vec{z}_i]_\alpha$

end for

until converge

missing features $y = \arg\max_c p(y=c|\vec{x}) = \arg\max_c \hat{\pi}_c \prod_{\alpha\in OBS} p([\vec{x}]_\alpha|y=c)$

$\dfrac{\partial(\vec{w}^TA\vec{w})}{\partial\vec{w}} \overset{(A=A^T)}{=} 2A\vec{w}$ $\overset{D}{=}$ $(A+A^T)\vec{w}$ $\quad \dfrac{\partial(\vec{w}^TB)}{\partial\vec{w}} = B^T$ $\quad \text{sign}(x) = \begin{cases}+1, & x>0\\ 0, & x=0\\ -1, & x<0\end{cases}$

$p(\vec{w}|X,\vec{y})$ posterior distribution (over parameters)

$p(y|x,w)$ model $p(y^*|x^*,w)$ (posterior) predictive distribution

$p(\vec{y}|X,\vec{w})$ conditional likelihood (i.i.d. $= \prod_{i=1}^{N} p(y_i|\vec{x}_i,\vec{w})$)

$p(D|\vec{w})$ likelihood, MLE $\hat{\vec{w}} = \arg\max_{\vec{w}} p(D|\vec{w})$ ← $X,\vec{y}$

$p(w)$ prior distribution (over parameters w)