Catherine

Regression and Estimation

slides03 - regression
slides04 - MLE

Linear Models (denominator layout)

Input: vector $x \in \mathbb{R}^D$ ⟶ Output: $y \in \mathbb{R}$

Data: $\langle (x_i, y_i) \rangle_{i=1}^N$

Linear Model: $y = w_0 + w_1 x_1 + \cdots + w_D x_D + \varepsilon = w^T x + \varepsilon$

(w_0: bias / intercept; ε: noise / uncertainty)

$$\hat{y} \in \mathbb{R}^{N\times 1} = X \in \mathbb{R}^{N\times(D+1)} \quad w \in \mathbb{R}^{(D+1)\times 1}$$

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{bmatrix} = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix} \begin{bmatrix} w_0 \\ \vdots \\ w_D \end{bmatrix} = \begin{bmatrix} x_{10} & \cdots & x_{1D} \\ x_{20} & \cdots & x_{2D} \\ \vdots & \ddots & \vdots \\ x_{N0} & \cdots & x_{ND} \end{bmatrix} \begin{bmatrix} w_0 \\ \vdots \\ w_D \end{bmatrix}$$

training phrase: estimate w from data

testing phrase: predict $\hat{y}_{new} = \hat{w}^T x_{new}$

Method: minimize average squared error $\frac{1}{2N}\sum_{i=1}^N (y_i - \hat{y}_i)^2$

Loss function: $L(w) = \frac{1}{2N}\sum_{i=1}^N (x_i^T w - y_i)^2 = \frac{1}{2N}(Xw - y)^T(Xw - y)$

$$= \frac{1}{2N}\left[w^T(X^TX)w - w^TX^Ty - y^TXw + y^Ty\right]$$

$$= \frac{1}{2N}\left[w^T(X^TX)w - 2y^TXw + y^Ty\right]$$

$$\nabla_w L = \frac{1}{N}\left[(X^TX)w - X^Ty\right]$$

By setting $\nabla_w L = 0$, we get: $w = (X^TX)^{-1}X^Ty$ (rank $(X) = D+1$; complexity: $O(D^2N)$)

prediction: $\hat{y} = Xw = X(X^TX)^{-1}X^Ty$ (the "hat" matrix)

Assumptions for unbiasedness (OLS estimators)

1. Linear in parameters: linear in w
2. Random sampling: (x_i, y_i) iid, $E(y_i|x) = E(y_i|x_i)$, (not necessary)
3. No perfect collinearity: X^TX invertible, or rank $(X^TX) \xrightarrow{D=1} D+1$
4. Zero conditional mean: $E(\varepsilon_i|x_i) = 0$
   mean independence: $E(\varepsilon_i) = 0$, $E(\varepsilon_i x_j) = 0$, $E(\varepsilon_i|x_i) = 0$ (if independent)
   strictly exogenous: $E(\varepsilon_i|X)$, contemporaneously exogenous: $E(\varepsilon_i|x_i)$
5. (Gauss-Markov Assumptions): Conditional Homoskedasticity: $Var(\varepsilon_i|X)^2 = \sigma^2$
   $E(\varepsilon_i\varepsilon_j|X) = 0$ (conditional spatial / serial uncorrelations) $\Rightarrow Cov(\varepsilon_i, \varepsilon_j) = 0$

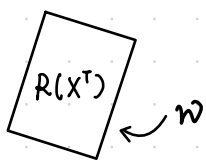
More on OLS

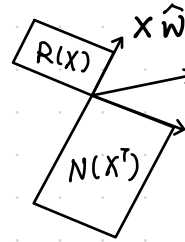
Linear Model: $y = w_0 + w_1 x_1 + \cdots + w_D x_D + \varepsilon = w^T x + \varepsilon$

$$\hat{y} \in \mathbb{R}^{N\times 1} = X \in \mathbb{R}^{N\times(D+1)} \quad w \in \mathbb{R}^{(D+1)\times 1}, \quad k = D+1$$

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{bmatrix} = \begin{bmatrix} X_1^T \\ X_2^T \\ \vdots \\ X_N^T \end{bmatrix} \begin{bmatrix} w_0 \\ \vdots \\ w_D \end{bmatrix} = \begin{bmatrix} x_{10} & \cdots & x_{1D} \\ x_{20} & \cdots & x_{2D} \\ \vdots & \ddots & \vdots \\ x_{N0} & \cdots & x_{ND} \end{bmatrix} \begin{bmatrix} w_0 \\ \vdots \\ w_D \end{bmatrix}$$

$\mathbb{R}^{D+1} \xrightarrow{N\left[\begin{smallmatrix} D+1 \\ X \end{smallmatrix}\right] \text{ sample matrix}} \mathbb{R}^N$





$X\hat{w} = X(X^TX)^{-1}X^TY$ projection matrix: $P^2=P$, $PX=X$

Y

related matrix

$\hat{\varepsilon} = (I-P)Y = MY$

$= M(Xw+\varepsilon) = M\varepsilon$

$= (I - X(X^TX)^{-1}X^T)\varepsilon$

symmetric and idempotent

$MX=0$, $PM=0$, $tr(M) = rank(M) = n-k$

$SSR = \hat{\varepsilon}'\hat{\varepsilon} = \varepsilon' M\varepsilon = Y'MY$

Estimation of $\sigma^2 (= E(\varepsilon_i^2))$:

1. Method of Moments (MOM): $\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^N \hat{\varepsilon}_i^2 = \frac{1}{N}\hat{\varepsilon}'\hat{\varepsilon}$
2. Unbiased Estimator: $s^2 = \frac{1}{n-k}\sum_{i=1}^N \hat{\varepsilon}_i^2 = \frac{SSR}{n-k}$

Variance Decomposition:

$$Y^TY = \hat{Y}^T\hat{Y} + \hat{\varepsilon}^T\hat{\varepsilon} \Rightarrow SST = SSE + SSR$$

$$R^2 = 1 - \frac{SSR}{SST} = 1 - \frac{\sum_{i=1}^N \hat{\varepsilon}_i^2}{\sum_{i=1}^N (Y_i - \bar{Y})^2}, \quad \bar{R}^2 = 1 - \frac{SSR/(n-k)}{SST/(n-1)} = 1 - \frac{n-1}{n-k}(1-R^2)$$

Proof of unbiasedness:

$$\hat{w} = (X^TX)^{-1}X^TY = (X^TX)^{-1}X^T(Xw+\varepsilon) = w + (X^TX)^{-1}X^T\varepsilon$$

$$E(\hat{w}|X) = w + E[(X^TX)^{-1}X^T\varepsilon|X] = w + (X^TX)^{-1}X^T E(\varepsilon|X) = w$$

$$E(s^2|X) = \frac{1}{n-k}E(\varepsilon^TM\varepsilon|X) = \frac{1}{n-k}E[tr(\varepsilon^TM\varepsilon)|X]$$

$$= \frac{1}{n-k}E(tr(M\varepsilon'\varepsilon)|X) = \frac{1}{n-k}tr[M\,E(\varepsilon'\varepsilon|X)] = \frac{\sigma^2}{n-k}tr(M) = \sigma^2$$

Variance of $\hat{w}$: $Var(\hat{w}|X) = Var((X^TX)^{-1}X^T\varepsilon|X) = (X^TX)^{-1}X^T\,Var(\varepsilon|X)[(X^TX)^{-1}X^T]^T$

$$= (X^TX)^{-1}X^T\sigma^2 I_N X(X^TX)^{-1} = \sigma^2(X^TX)^{-1}$$

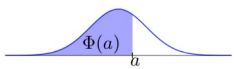
$$= \frac{\sigma^2}{SST_x}\ (D=1), \quad \frac{\sigma^2}{SST_j(1-R_j^2)}\ (D>1)$$

SLR: $\hat{w} = \frac{\sum_{i=1}^N (X_i-\bar{X})(Y_i-\bar{Y})}{\sum_{i=1}^N (X_i-\bar{X})^2}$ ($\frac{Cov(X,Y)}{Var(X)}$)

Probability Review

univariate normal distribution: $f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, $X \sim N(\mu, \sigma^2)$

inverse transform sampling: $y = \Phi^{-1}(z; 0, 1)$, $Z \sim \text{Unif}([0,1])$



bivariate normal distribution: $X_1 \sim N(\mu_1, \sigma_1^2)$, $X_2 \sim N(\mu_2, \sigma_2^2)$

$$f(x_1, x_2) = \frac{1}{2\pi(\sigma_1^2\sigma_2^2)^{1/2}} \exp\left(-\left(\frac{(x_1-\mu_1)^2}{2\sigma_1^2} + \frac{(x_2-\mu_2)^2}{2\sigma_2^2}\right)\right)$$

$$= \frac{1}{2\pi|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

, where $\Sigma = \begin{bmatrix}\sigma_1^2 & 0 \\ 0 & \sigma_2^2\end{bmatrix}$, $\mu = \begin{bmatrix}\mu_1 \\ \mu_2\end{bmatrix}$, $x = \begin{bmatrix}x_1 \\ x_2\end{bmatrix}$

Covariance Matrix:

$$\text{Cov}(X) = E[(X-E(X))(X-E(X))^T] = \begin{bmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) & \cdots & \text{Cov}(X_1, X_D) \\ \text{Cov}(X_2, X_1) & \text{Var}(X_2) & \cdots & \text{Cov}(X_2, X_D) \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(X_D, X_1) & \text{Cov}(X_D, X_2) & \cdots & \text{Var}(X_D) \end{bmatrix}$$

$$N(\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right), \quad \Sigma = \text{Cov}(X)$$

Eigendecomposition: $\Sigma = Q \Lambda Q^{-1}$

(Q: eigenvectors, Λ: eigenvalues)

the eigenvector with the largest eigenvalue corresponds to the direction with the greatest variance and vice-versa

Maximum Likelihood

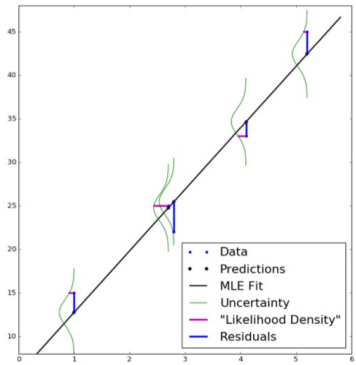
Linear Model: $y = w_0x_0 + w_1x_1 + \cdots w_Dx_D + \underbrace{\varepsilon}_{\text{aleatoric uncertainty}} = w \cdot x + \varepsilon$

prediction: $E[y|x, w] = f_w(x) = w^Tx$

$p(y|x, w) = N(y|w^Tx, \sigma^2)$, $y \sim w^Tx + N(0, \sigma^2)$

(alternatively, $\varepsilon \sim N(0, \sigma^2)$, Gaussian Noise)

Likelihood of Linear Regression (Gaussian Noise Model)



MLE Estimator

Find parameters which maximise the likelihood.

(product of "likelihood density" — segments)

Least Square Estimator

Find parameters which minimise the sum of squares of the residuals

(sum of squares of the | segments).

$p(y_1, y_2, \cdots y_N \mid X_1, X_2, \cdots X_N, w, \sigma) = \prod_{i=1}^{N} p(y_i \mid x_i, \underbrace{w, \sigma}_{\text{model parameters}})$

$= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(y_i - w^T x_i)^2}{2\sigma^2}\right)$

$= \left(\frac{1}{2\pi\sigma^2}\right)^{N/2} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - w^T x_i)^2\right)$

log-likelihood: $LL(y_1, y_2, \cdots y_N \mid X_1, X_2, \cdots X_N, w, \sigma)$

$= -\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - w^T x_i)^2$

$\Rightarrow LL(y \mid X, w, \sigma) = -\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(Xw - y)^T(Xw - y)$

$NLL(y \mid X, w, \sigma) = \frac{N}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}(Xw - y)^T(Xw - y)$

minimize $NLL \Rightarrow w, \sigma^2 \Leftrightarrow$ least square method

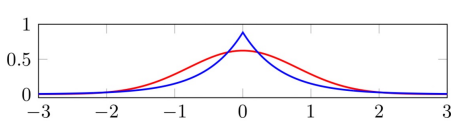
$\Rightarrow \begin{cases} w_{ML} = (X^T X)^{-1} X^T y \\ \sigma^2_{ML} = \frac{1}{N}(Xw_{ML} - y)^T(Xw_{ML} - y) \end{cases}$

Outliers and Laplace Distribution

Linear Model: $y = w_0 x_0 + w_1 x_1 + \cdots w_D x_D + \underbrace{\varepsilon}_{\varepsilon \sim Lap(0,b),\ pdf(x) = \frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right)} = w \cdot x + \varepsilon$

$p(y_1, \cdots y_N \mid X_1, \cdots X_N, w, b)$

$= \frac{1}{(2b)^N} \exp\left(-\frac{1}{b}\sum_{i=1}^{N}|y_i - w^T x_i|\right)$



$NLL(y \mid X, w, b) = \frac{1}{b}\sum_{i=1}^{N}|y_i - w^T x_i| + N\log(2b)$

Fitting, Regulization and Bayesian

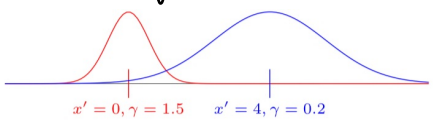
slides05 - fitting
slides06 - regulization
slides07 - Bayesian

Basis Function Expansion

polynomial basis expansion: $y = w \cdot \phi(x) + \varepsilon$

- linear model: $\phi(x) = [1, x_1, x_2]$
- quadratic model: $\phi(x) = [1, x_1, x_2, x_1^2, x_2^2, x_1 x_2]$

radial basis function (RBF) kernel: $\kappa(x', x) = \exp(-\gamma \|x - x'\|^2)$



$\kappa(x', x): \mathcal{X} \times \mathcal{X} \to \mathbb{R} = \phi(x') \cdot \phi(x)$

polynomial kernel: $\kappa(x', x) = (x^T \cdot x' + c)^d$

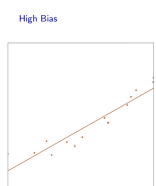
- underfitting: γ too small (large width)
- overfitting: γ too large (small width)

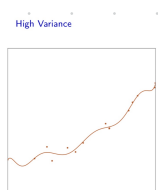
1. choose centres: $\mu_1, \mu_2, \cdots \mu_M$
2. feature map: $\phi(x) = [1, \kappa(\mu_1, x), \cdots \kappa(\mu_M, x)]$
3. model: $y = w_0 + w_1 \kappa(\mu_1, x) + \cdots w_M \kappa(\mu_M, x) + \varepsilon = w \cdot \phi(x) + \varepsilon$

curse of dimensionality: might need exponentially large (in the dimension) sample for using modest width kernels.

The Bias Variance Tradeoff

- underfitting: high bias (high similar errors)
- overfitting: high variance (training << test errors)





Learning Curves:

training set + validation + test

plot errors as a function of training data size

Regulization

Ridge Regression Objective $L_{ridge}(w) = (Xw - y)^T (Xw - y) + \lambda w^T w$

(*) before optimization, standardize all inputs (mean 0 and variance 1)

$$L_{ridge}(w) = (Xw - y)^T (Xw - y) + \lambda w^T w$$
$$= w^T (X^T X) w - 2 y^T X w + y^T y + \lambda w^T w$$

l_2 - regulization (weight - decay)

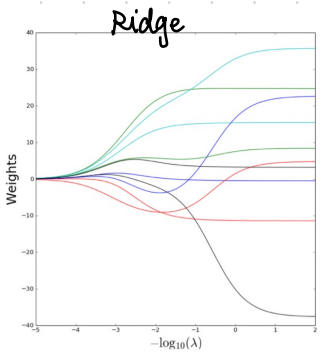
$$\nabla_w \mathcal{L}_{ridge} = 2(X^TX)w - 2X^Ty + 2\lambda w$$
$$= 2[(X^TX + \lambda I_D)w - X^Ty]$$
$$\Rightarrow w_{ridge} = (X^TX + \lambda I_D)^{-1} X^Ty$$

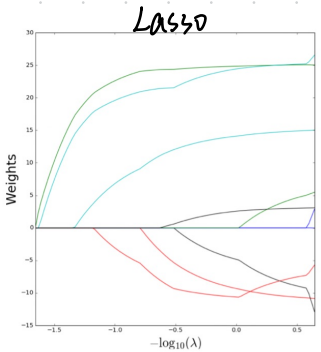
Lasso Regression Objective $\mathcal{L}_{ridge}(w) = (Xw-y)^T(Xw-y) + \lambda \sum_{i=1}^{D} |w_i|$

(*) before optimization, standardize all inputs (mean 0 and variance 1)

$$\mathcal{L}_{ridge}(w) = (Xw-y)^T(Xw-y) + \lambda w^Tw$$
$$= w^T(X^TX)w - 2y^TXw + y^Ty + \lambda w^Tw$$

l_1 - regulization
(Least Absolute Shrinkage and Selection Operator)





when using Lasso:
weights are often exactly 0
⇒ sparse models

Feature Selection

Forward Search: $O(n^2)$

1. Set set of selected features to $F := \phi$
2. Repeat the following until $F = \{1, 2, \cdots n\}$:
   - set $F_i := F \cup \{i\}$ for $i \in \{1, 2, \cdots n\} \setminus F$
   - evaluate generalization error when using only features from F_i
   - set new F to the best feature subset found
3. Return best overall feature subset found

Filter feature selection:

criterion (mutual information):

$$I(X, Y) = \sum_x \sum_y p(X=x, Y=y) \cdot \log \frac{p(X=x, Y=y)}{p(X=x) \cdot p(Y=y)}$$

compute mutual information for all features, retain top k features

Exercises

1. $$\mathop{\mathbb{E}}_{\mathcal{D}}\left[\|\hat{\mathbf{w}}(\mathcal{D}) - \mathbf{w}^*\|^2\right] = \|\mathop{\mathbb{E}}_{\mathcal{D}}\left[\hat{\mathbf{w}}(\mathcal{D})\right] - \mathbf{w}^*\|^2 + \mathop{\mathbb{E}}_{\mathcal{D}}\left[\|\hat{\mathbf{w}}(D) - \mathop{\mathbb{E}}_{\mathcal{D}}\left[\hat{\mathbf{w}}(\mathcal{D})\right]\|^2\right]$$

MSE — Bias² — Variance

$$\begin{aligned}
\mathop{\mathbb{E}}_{D}[\|\hat{w}(D) - w^*\|^2] &= \mathop{\mathbb{E}}_{D}[\|\hat{w}(D) - \mathop{\mathbb{E}}_{D}[\hat{w}(D)] + \mathop{\mathbb{E}}_{D}[\hat{w}(D)] - w^*\|^2] \\
&= \mathop{\mathbb{E}}_{D}[\|\hat{w}(D) - \mathop{\mathbb{E}}_{D}[\hat{w}(D)]\|^2] + \mathop{\mathbb{E}}_{D}[\|\mathop{\mathbb{E}}_{D}(\hat{w}(D)) - w^*\|^2] \\
&\quad + 2\mathop{\mathbb{E}}_{D}\left([\hat{w}(D) - \mathop{\mathbb{E}}_{D}[\hat{w}(D)] \cdot [\mathop{\mathbb{E}}_{D}[\hat{w}(D)] - w^*]\right) \\
&= \mathop{\mathbb{E}}_{D}[\|\hat{w}(D) - \mathop{\mathbb{E}}_{D}[\hat{w}(D)]\|^2] + \|\mathop{\mathbb{E}}_{D}[\hat{w}(D)] - w^*\|^2
\end{aligned}$$

$\mathop{\mathbb{E}}_{D}\left(\hat{w}(D) - \mathop{\mathbb{E}}_{D}[\hat{w}(D)]\right) = \mathop{\mathbb{E}}_{D}(\hat{w}(D)) - \mathop{\mathbb{E}}_{D}[\mathop{\mathbb{E}}_{D}(\hat{w}(D))] = 0$

unrelated to D, this constant wrt $\mathop{\mathbb{E}}_{D}(\cdot)$

2. Huber Loss

In this question, we will investigate the *Huber loss* in a linear regression setting. Given arbitrary but fixed parameters $\lambda, \mu \in \mathbb{R}$ such that $\lambda, \mu > 0$, the Huber loss is given by the function $h_{\lambda,\mu} : \mathbb{R} \to \mathbb{R}$ such that

$$h_{\lambda,\mu}(z) = \begin{cases} \lambda\left(|z| - \frac{\lambda}{4\mu}\right) & \text{if } |z| \geq \frac{\lambda}{2\mu}, \\ \mu z^2 & \text{otherwise.} \end{cases}$$ (regieme).

Given a vector $\mathbf{z} = (z_1, \ldots, z_D) \in \mathbb{R}^D$, we extend $h_{\lambda,\mu}$ such that $h_{\lambda,\mu}(\mathbf{z}) = \sum_{i=1}^{D} h_{\lambda,\mu}(z_i)$. Recall that when dealing with absolute values, the sign function defined as follows is often helpful:

$$\text{sign}(z) = \begin{cases} 1 & \text{if } z > 0, \\ -1 & \text{otherwise.} \end{cases}$$

$$\mathcal{H}(\mathbf{z}; \mathcal{D}) = h_{\lambda,\mu}(\mathbf{z}) + \frac{1}{N}\sum_{i=1}^{N} \ell(y_i - \mathbf{z}^\intercal \cdot \mathbf{x}_i),$$

$$\mathcal{S}(\mathbf{v}, \mathbf{w}; \mathcal{D}) = \lambda||\mathbf{v}||_1 + \mu||\mathbf{w}||_2^2 + \frac{1}{N}\sum_{i=1}^{N} \ell(y_i - (\mathbf{v} + \mathbf{w})^\intercal \cdot \mathbf{x}_i).$$

Sklearn Python Programming

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.linear_model import Ridge, Lasso
from sklearn.metrics import mean_squared_error

degree = 2
ridge_model = make_pipeline(PolynomialFeatures(degree),
                            StandardScaler(), Ridge(alpha=lamb),
                            verbose=True)
ridge_model.fit(X_train, y_train)
ridge_validation_pred = ridge_model.predict(X_validation)
ridge_mse = mean_squared_error(y_validation, ridge_validation_pred)
```

} First polynomialize Then standardize

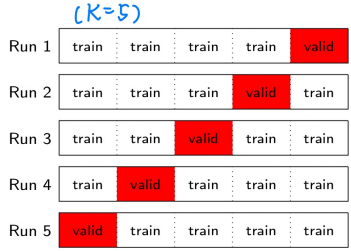
```
# model selection
from sklearn.moel_selection import GridSearchCV

param_grid_ridge = {
        'polynomialfeatures__degree': [2, 3, 4]
        'ridge__alpha': [0.01, 0.1, 1, 10, 100]
}
ridge_model = make_pipeline(PolynomialFeatures(),
                            StandardScaler(), Ridge())
ridge_cv = GridSearchCV(ridge_model, param_grid_ridge, cv=5,
                        scoring="neg_mean_squared_error")
ridge_model.fit(X_train, y_train)
best_ridge_model = ridge_cv.best_estimator_
ridge_test_pred = ridge_model.predict(X_test)
ridge_mse = mean_squared_error(y_test, ridge_test_pred)
```

dictionary key corresponds to preprocessing classes

K-fold Cross Validation

(K=5)



When K=N, LOOCV (Leave One Out Cross Validation)

Other Tips for programming

```
# bar chart
v = []
for i in range(3, 10):
    v.apppend(y_train[y_train == i].size()
plot.bar(range(3, 10), v)

# numpy
means = np.mean(X_train, axis=0)
stds = np.std(X_train, axis=0)
X_train_n = (X_train - means) / stds
X_train_n = np.hstack((X_train_n, np.full((N_train, 1), 1)))
W = np.linalg.inv(X_train_n.transpose().dot(X_train_n))
      .dot(X_train_n.transpose()).dot(y_train)
y_hat_train = X_train_n.dot(W)
```

along 'columns' (vertically)

(horizontal)

shape ; fill value

tuple

X_train = [[; [; [ ] = [ ] NxD

Bayesian Approach

- Frequentists: fixed parameters, $P(D|w)$
- Bayesians: fixed data, $p(w|D)$

- Aleatoric uncertainty: stochastic nature of variables (more data useless)
- Epistemic uncertainty: about choice of model (reducible with more data)

Bayes' Theorem: $\underbrace{p(w|D)}_{\text{posterior}} = \frac{p(D|w)\cdot p(w)}{p(D)}$ ($p(w)$: prior)

$$\Rightarrow p(w|D) = \frac{p(y|w,X)\cdot p(w)}{p(D)}$$

prior: $p(\mu_s) = \mathcal{N}(\mu_{0,s}, \sigma^2_{0,s})$

posterior: $p(\mu_s|D) = \mathcal{N}(\mu_{N,s}, \sigma^2_{N,s})$ from samples $t_i \sim N(\mu, \sigma^2)$

, where: $\mu_{N,s} = \frac{1}{\frac{\sigma^2}{\sigma^2_{0,s}} + N}\left[\frac{\sigma^2}{\sigma^2_{0,s}}\mu_{0,s} + \sum_{i=1}^{N} t_i\right]$, $\sigma^2_{N,s} = \left(\frac{1}{\sigma^2_{0,s}} + \frac{N}{\sigma^2}\right)^{-1}$

Bernstein-von Mises Theorem:

under certain regularity assumptions, as $N \to \infty$, posterior converges to a normal distribution $\mathcal{N}(w_{mle}, N^{-1} I_F(w_{mle})^{-1})$ centred on maximum likelihood estimate (w_{mle}) where $I_F(w_{mle})$ is the Fisher information matrix at w_{mle}

$$\Rightarrow p(t|s,D) = \int_w \underbrace{p(t|s,w)}_{\text{model}} \cdot \underbrace{p(w|D)}_{\text{posterior}}\, dw = \mathcal{N}(\mu_{N,s}, \underbrace{\sigma^2}_{\text{aleatoric}} + \underbrace{\sigma^2_{N,s}}_{\text{epistemic}})$$

$$E[t|s,D] = \int t\cdot p(t|s,D)\,dt = \mu_{N,s}$$

$$pdf(y|X,w) = \frac{1}{(2\pi\sigma^2)^{N/2}} \prod_{i=1}^{N} \exp\left(-\frac{(y_i - w^T x_i)^2}{2\sigma^2}\right) \Rightarrow \exp\left(-\frac{1}{2\sigma^2}(y-Xw)^T(y-Xw)\right)$$

Gaussian prior: $pdf(w) = \mathcal{N}(w|0,\Lambda) \propto \exp\left(-\frac{1}{2} w^T \Lambda^{-1} w\right)$

posterior: $pdf(w|D) \propto \underbrace{\mathcal{N}(y|Xw,\Sigma)}_{\text{likelihood}} \cdot \underbrace{\mathcal{N}(w|0,\Lambda)}_{\text{prior}} = \mathcal{N}(w|w_N, \Sigma_N)$

, where $\Sigma_N = (\Lambda^{-1} + \frac{1}{\sigma^2} X^T X)^{-1}$, $w_N = \frac{1}{\sigma^2}\Sigma_N X^T y$ } complexity!

$$pdf(y|D, X_{new}, \sigma^2) = \int_w pdf(y|X_{new}, w)\, pdf(w|D)\, dw$$

$$= \mathcal{N}(y|w_N^T X_{new}, \underbrace{\sigma^2}_{\text{aleatoric}} + \underbrace{X_{new}^T \Sigma_N X_{new}}_{\text{epistemic}})$$

Maximum a Posteriori (MAP)

prior over w takes the more general form: $pdf(w) \propto \exp(-R(w))$, $R: \mathbb{R}^D \to \mathbb{R}$

$w_{MAP} \in \arg\max_w pdf(w|D) = \arg\max_w \log pdf(w|D)$

$\Rightarrow pdf(w|D) \propto \exp\left(-\frac{1}{2\sigma^2}(y-Xw)^T(y-Xw)\right) \cdot \exp(-R(w))$

$= \exp\left(-\frac{1}{2\sigma^2}(y-Xw)^T(y-Xw) - R(w)\right)$

the MAP loss function:

$$L_{MAP}(w) \propto -\log(pdf(w|D)) = \frac{1}{2\sigma^2}(y-Xw)^T(y-Xw) - R(w)$$

(least squares; regularization)

$$L_{ridge}(w) \propto \frac{1}{2\sigma^2}(y-Xw)^T(y-Xw) + \frac{\lambda}{2\sigma^2}w^Tw = L_{MAP}(w)$$

the prior corresponds to $\exp\left(-\frac{\lambda}{2\sigma^2}w^Tw\right) \Rightarrow \mathcal{N}\left(0, diag\left(\frac{\sigma^2}{\lambda}\right)\right)$

$$L_{lasso}(w) \propto \frac{1}{2\sigma^2}(y-Xw)^T(y-Xw) + \frac{\lambda}{2\sigma^2}\sum_{i=1}^D |w_i| = L_{MAP}(w)$$

the prior corresponds to $\exp\left(-\frac{\lambda}{2\sigma^2}\sum_{i=1}^D |w_i|\right) \Rightarrow pdf(w) = \prod_{i=1}^D Lap\left(0, \frac{2\sigma^2}{\lambda}\right)$

$$L_{mle}(w) \propto \frac{1}{2\sigma^2}(y-Xw)^T(y-Xw) + \lim_{\lambda \to 0}\frac{1}{2\sigma^2}w^Tw = L_{MAP}(w)$$

the prior corresponds to $\exp\left(-\lim_{\lambda\to 0}\frac{1}{2\sigma^2}w^Tw\right) \Rightarrow \mathcal{N}\left(0, \lim_{\lambda\to 0} diag\left(\frac{\sigma^2}{\lambda}\right)\right)$ improper prior $\Rightarrow$ entire $\mathbb{R}^D$

Optimization

slides08-optimization
slides09-optimization2

Convex Optimization

Definition: the entire set $\mathbb{R}^D$: $\lambda x + (1-\lambda) y \in \mathbb{R}^D$, $\forall x, y \in \mathbb{R}^D$
function $f: \mathbb{R}^D \to \mathbb{R}$: $f(\lambda x + (1-\lambda) y) \leq \lambda f(x) + (1-\lambda) f(y)$

Theorem: the set of positive semi-definite matrices is convex
positive semi-definite. $A = A^T$, $x^T A x \geq 0$ for $\forall x \in \mathbb{R}^D$
set S_+^D = positive semidefinite cone
$\forall A, B \in S_+^D$, $x^T(\lambda A + (1-\lambda) B) \cdot x = \lambda \cdot x^T A x + (1-\lambda) x^T B x \geq 0$

Examples for concave functions:

1. Affine functions: $f(x) = b^T x + c$
2. Quadratic functions: $f(x) = \frac{1}{2} x^T A x + b^T x + c$,
   A is symmetric positive semidefinite
3. Norms, in particular L^p-norms
4. Nonnegative weighted sums of convex functions

Theorem: For a convex optimization problem, all locally optimal points are globally optimal

Convex Optimization

Given convex functions $f, g_1, \cdots g_m$ and affine functions $h_1, \cdots h_n$:

minimize $f(x)$
subject to $g_i(x) \leq 0$, $i \in \{1, 2, \cdots m\}$
$h_j(x) = 0$, $j \in \{1, 2, \cdots n\}$
$\Rightarrow$ optimal value, optimal point

1. Linear Programming
   minimize $c^T x + d$ s.t. $Ax \leq e$, $Bx = f$
2. Quadratically Constrained Quadratic Programming
   minimize $\frac{1}{2} x^T B x + c^T x + d$ s.t. $\frac{1}{2} x^T Q_i x + r_i^T x + s_i \leq 0$, $Ax = b$
3. Semidefinite Programming
   minimize $\mathrm{tr}(CX)$ s.t. $\mathrm{tr}(A_i X) = b_i$, X positive semidefinite

Exercises

1. Optimization Method for ℓ_1-regularization

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{N} |\mathbf{w}^\top \mathbf{x}_i - y_i| + \lambda \sum_{i=1}^{D} |w_i|$$

Consider a linear program with $2D+N$ variables:

$w_1, \cdots w_D, k_1, \cdots k_D, \xi_1, \cdots \xi_N$

The objective function can be expressed in:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{i=1}^{N} \xi_i + \lambda \sum_{i=1}^{D} k_i \\ \text{subject to} \quad & w^\top x_i - y_i \le \xi_i, \; i = 1, 2, \cdots N \\ & y_i - w^\top x_i \le \xi_i, \; i = 1, 2, \cdots N \\ & w_i \le k_i, \; i = 1, 2, \cdots D \\ & -w_i \le k_i, \; i = 1, 2, \cdots D \end{aligned}$$

$$\mathcal{L}_{\text{lasso}}(\mathbf{w}) = \sum_{i=1}^{N} (\mathbf{w}^\top \mathbf{x}_i - y_i)^2 + \lambda \sum_{i=1}^{D} |w_i|$$

Set $f(w) = \lambda \sum_{i=1}^{D} |w_i|$

$$\nabla_w f(w) = \lambda \begin{bmatrix} \text{sign}(w_1) \cdot \mathbb{I}(w_1 \neq 0) \\ \vdots \\ \text{sign}(w_D) \cdot \mathbb{I}(w_D \neq 0) \end{bmatrix}, \text{ where } \mathbb{I}(w_i \neq 0) = \begin{cases} 1, & w_i \neq 0 \\ 0, & w_i = 0 \end{cases}$$

Thus $\nabla_w \mathcal{L}_{\text{lasso}}(w) = 2(X^T X w - X^T y) + \lambda \begin{bmatrix} \text{sign}(w_1) \cdot \mathbb{I}(w_1 \neq 0) \\ \vdots \\ \text{sign}(w_D) \cdot \mathbb{I}(w_D \neq 0) \end{bmatrix}$

sub-gradient descent update rule:

$$w_{t+1} = w_t - \eta \left(2(X^T X w - X^T y) + \lambda \begin{bmatrix} \text{sign}(w_1) \cdot \mathbb{I}(w_1 \neq 0) \\ \vdots \\ \text{sign}(w_D) \cdot \mathbb{I}(w_D \neq 0) \end{bmatrix} \right)$$

Gradient Descent

Gradients: $\nabla_w f(w) = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \vdots \\ \frac{\partial f}{\partial w_D} \end{bmatrix}$

Hessians: $H = \begin{bmatrix} \frac{\partial^2 f}{\partial w_1^2} & \frac{\partial^2 f}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_1 \partial w_D} \\ \frac{\partial^2 f}{\partial w_2 \partial w_1} & \frac{\partial^2 f}{\partial w_2^2} & \cdots & \frac{\partial^2 f}{\partial w_2 \partial w_D} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial w_D \partial w_1} & \frac{\partial^2 f}{\partial w_D \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_D^2} \end{bmatrix}$

Symmetric, capture the curvature of the surface

gradient descent: $w_{t+1} = w_t - \eta_t \nabla f(w_t)$, $\eta_t > 0$: learning rate / step size

gradient descent for least squares regression:

$$L(w) = (Xw - y)^T (Xw - y)$$
$$\nabla_w L = 2(X^T X w - X^T y)$$

complexity: $O(ND)$

($X^T X$ only calculate once)

Newton's Method

$$w_{t+1} = w_t - H_t^{-1} g_t$$

$$f_{quad}(w) = f(w_t) + g_t^T (w - w_t) + \frac{1}{2}(w - w_t)^T H_t (w - w_t)$$
$$\nabla_w f_{quad} = g_t + H_t(w - w_t) := 0$$

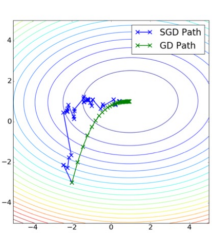
Sub-gradient Descent: range between the left and right derivatives

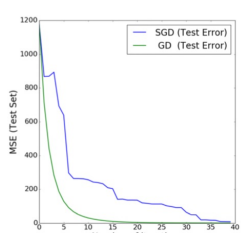
Stochastic Gradient Descent (SGD)

objective function: $L(w; D) = \frac{1}{N} \sum_{i=1}^{N} l(w; x_i, y_i) + \underline{\lambda R(w)}$ regularization term

pick a random datapoint (x_i, y_i) and evaluate $g_i = \nabla_w l(w; x_i, y_i)$

$$E(g_i) = \frac{1}{N} \sum_{i=1}^{N} \nabla_w l(w; x_i, y_i)$$





Batch / offline Learning:

$$w_{t+1} = w_t - \frac{\eta}{N} \sum_{i=1}^{N} \nabla_w l(w; x_i, y_i) - \lambda \nabla_w R(w)$$

Online Learning:

$$w_{t+1} = w_t - \eta \nabla_w l(w; x_i, y_i) - \lambda \nabla_w R(w)$$

Minibatch Online Learning

$$w_{t+1} = w_t - \frac{\eta}{b} \sum_{i=1}^{b} \nabla_w l(w; x_i, y_i) - \lambda \nabla_w R(w)$$

Other Optimization Techniques

First Order Methods / (Sub) Gradient Methods:

1. Nesterov's Accelerated Gradient
2. Line-Search to Find Step-Size
3. Momentum-based Methods
4. AdaGrad, AdaDelta, Adam, RMSProp

$$w_{t+1,i} \leftarrow w_{t,i} - \frac{\eta}{\sqrt{\sum_{s=1}^{t} g_{s,i}^2}} g_{t,i}$$

rare features can be most predictive

suitable in NLP

Second Order / Newton / Quasinewton Methods:

1. Conjugate Gradient Method
2. BGFS and L-BGFS

Classification

slides 10 - classification (NBC)
slides 11 - logistic regression
slides 12 - SVM
slides 13 - SVM (2)

Discriminative Model (判别性模型): distinguish decision boundaries through observed data
model the conditional distribution $p(y|x,\theta)$

Generative Model (生成式模型): include the distribution of data itself
model the full joint distribution $p(x,y|\theta)$

Classification Background

target / output: $y \in \{1,2,\cdots C\}$

input: $X = (X_1, X_2, \cdots X_D)$, where $x_i \in \{1,2,\cdots K\} \in \mathbb{R}$

$$p(y=c|X_{new},\theta) = \frac{p(y=c|\theta)\cdot p(X_{new}|y=c,\theta)}{\sum_{c'=1}^{C} p(y=c'|\theta)p(X_{new}|y=c',\theta)}$$

class-conditional $\Rightarrow$ joint distribution $p(X_{new}, y|\theta)$

$\Rightarrow$ marginal distribution $p(X_{new}|\theta)$

$$\Rightarrow \hat{y} = \text{argmax}_c\, p(y=c|X_{new},\theta)$$

joint distribution: $p(X,y|\theta,\pi) = p(y|\pi)\cdot p(X|y,\theta)$

$p(y=c|\pi) = \pi_c$, $\sum \pi_c = 1$ class-conditional densities: $p(x|y=c,\theta_c)$ for class $c=1,2,\cdots C$

Naïve Bayes Classifier (NBC)

To estimate the parameters π_c, θ_c for $c=1,2,\cdots C$.

Assumption: the features are conditionally independent given the class label

$$p(X|y=c,\theta_c) = \prod_{j=1}^{D} p(x_j|y=c,\theta_{jc})$$

if not independent, $O(C\cdot 2^D)$ parameters

Example: use a Gaussian Model, so $\theta_{jc} = (\mu_{jc}, \vec{\nabla_{jc}})$

$\sum_{i=1}^{K} \mu_{jc,i} = 1$ (j: factor $\le D$, c: choice $\le C$, i: factor range $\le K$)

assume all features binary, $K=1$. $x_j \in \{0,1\}$. $O(CD)$ params

Maximum Likelihood for NBC

Data $\langle (x_i, y_i)\rangle_{i=1}^{N}$ i.i.d. from some joint distribution $p(X,y)$

probability for a single datapoint:

$$p(x_i,y_i|\theta,\pi) = p(y_i|\pi)\cdot p(x_i|\theta,y_i) = \prod_{c=1}^{C} \pi_c^{\mathbb{I}(y_i=c)} \prod_{c=1}^{C}\prod_{j=1}^{D} p(x_{ij}|\theta_{jc})^{\mathbb{I}(y_i=c)}$$

Denote N_c = number of datapoints with $y_i = c$, s.t. $\sum_{c=1}^{C} N_c = N$

$\Rightarrow \log p(\mathcal{D} \mid \theta, \pi) = \sum_{c=1}^{C} N_c \log \pi_c + \sum_{c=1}^{C} \sum_{j=1}^{D} \sum_{i: y_i = c} \log p(x_{ij} \mid \theta_{jc})$

1. $\max \sum_{c=1}^{C} N_c \log \pi_c$ s.t. $\sum_{c=1}^{C} \pi_c = 1 \Rightarrow \pi_c = \frac{N_c}{N}$
2. $\max \sum_{c=1}^{C} \sum_{j=1}^{D} \sum_{i: y_i = c} \log p(x_{ij} \mid \theta_{jc}) \Rightarrow \hat{\theta}_{jc}$ only depends on j^{th} feature with $y_i = c$

Pro: handle missing data at test time, just marginalize it out:

$$p(y = c \mid x_{new}, \theta) = \frac{\pi_c \cdot \prod_{j=2}^{D} p(x_j \mid y = c, \theta_{jc})}{\sum_{c'=1}^{C} p(y = c' \mid \theta) \prod_{j=2}^{D} p(x_j \mid y = c', \theta_{jc})}$$

Loosen the independence assumption:

$p(x \mid y = c, \theta_c) = \mathcal{N}(x \mid \mu_c, \Sigma_c)$

($x \in \mathbb{R}^D$, model class-conditional density as a multivariate normal distribution with Σ_c).

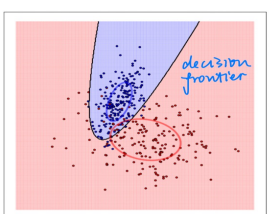
$$p(y = c \mid x_{new}, \theta) = \frac{p(y = c \mid \theta) \cdot p(x_{new} \mid y = c, \theta)}{\sum_{c'=1}^{C} p(y = c' \mid \theta) p(x_{new} \mid y = c', \theta)}$$

$$= \frac{\pi_c |2\pi\Sigma_c|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x - \mu_c)^T \Sigma_c^{-1} (x - \mu_c)\right)}{\sum_{c'=1}^{C} \pi_{c'} |2\pi\Sigma_{c'}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x - \mu_{c'})^T \Sigma_{c'}^{-1} (x - \mu_{c'})\right)}$$

the boundary between classes:

$$\frac{\pi_c |2\pi\Sigma_c|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x - \mu_c)^T \Sigma_c^{-1} (x - \mu_c)\right)}{\pi_{c'} |2\pi\Sigma_{c'}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x - \mu_{c'})^T \Sigma_{c'}^{-1} (x - \mu_{c'})\right)} = 1$$

— quadratic discriminant analysis (QDA)



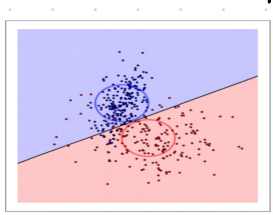
(*) Special case: shared/tied covariance matrices across different classes

$$p(y = c \mid x, \theta) \propto \pi_c \exp\left(-\frac{1}{2}(x - \mu_c)^T \Sigma^{-1} (x - \mu_c)\right)$$

$$= \exp\left(\underbrace{\mu_c^T \Sigma^{-1}}_{\beta_c = \Sigma^{-1}\mu_c} x - \underbrace{\frac{1}{2}\mu_c^T \Sigma^{-1} \mu_c + \log \pi_c}_{\text{denote as } \gamma_c}\right) \cdot \exp\left(-\frac{1}{2} x^T \Sigma^{-1} x\right)$$

$$= \exp(\beta_c^T x + \gamma_c) \Rightarrow \frac{\exp(\beta_c^T x + \gamma_c)}{\sum_{c'} \exp(\beta_{c'}^T x + \gamma_{c'})} = \text{softmax}(\eta)_c$$

$$, \eta = [\beta_1^T x + \gamma_1, \cdots \beta_C^T x + \gamma_C]$$



— linear discriminant analysis (LDA)

*softmax maps a set of numbers to a probability distribution with mode at the maximum, $\text{softmax}([a, b, c]) = ([p_1, p_2, p_3])$

(*) **Two class LDA**

when we have only 2 classes, say 0 and 1.

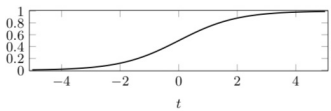
$$p(y=1|x,\theta) = \frac{\exp(\beta_1^T x + \gamma_1)}{\exp(\beta_1^T x + \gamma_1) + \exp(\beta_0^T x + \gamma_0)}$$

$$= \frac{1}{1 + \exp[-((\beta_1 - \beta_0)^T x + (\gamma_1 - \gamma_0))]}$$

$$= \text{sigmoid}((\beta_1 - \beta_0)^T x + (\gamma_1 - \gamma_0))$$

sigmoid function:

$$\text{sigmoid}(t) = \frac{1}{1+e^{-t}}$$



MLE for QDA and LDA:

$$\begin{cases} \pi_c = \frac{N_c}{N} \\ \hat{\mu}_c = \frac{1}{N_c} \sum_{i:y_i=c} x_i \\ \hat{\Sigma}_c = \frac{1}{N_c} \sum_{i:y_i=c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)^T \end{cases}$$

(number of parameters: $O(CD^2)$).

Logistic Regression

Binary classification: 0 and 1
This is a discriminative method: $p(y|w,x)$

$Z \sim \text{Bernoulli}(\theta)$, $\theta \in [0,1]$: $\begin{cases} =1 & \text{with prob} = \theta \\ =0 & \text{with prob} = 1-\theta \end{cases}$, $p(x|\theta) = \theta^x (1-\theta)^{1-x}$

model: $y \sim \text{Bernoulli}(f(x,w))$, where $f \in [0,1]$, x is input, w is param
logistic regression builds up on a linear model, composed with sigmoid:

$$p(y|w,x) = \text{Bernoulli}(y|\text{sigmoid}(wx))$$

$$\begin{cases} p(y_{new}=1|x_{new},w) = \text{sigmoid}(w \cdot x_{new}) \\ \hat{y}_{new} = \mathbb{I}(\text{sigmoid}(w \cdot x_{new}) \geq \frac{1}{2}) = \mathbb{I}(w \cdot x_{new} \geq 0) \end{cases}$$

$$p(y|X,w) = \prod_{i=1}^{N} \nabla(w^T x_i)^{y_i} \cdot (1-\nabla(w^T x_i))^{1-y_i}, \quad y_i \in \{0,1\}$$

$$\Rightarrow NLL(y|X,w) = -\sum_{i=1}^{N} (y_i \log \mu_i + (1-y_i)\log(1-\mu_i)), \quad \mu_i = \nabla(w^T x_i)$$

the cross-entropy between y_i and μ_i

Entropy 熵: $H(X) = -\sum_x p(x) \log p(x)$, Bernoulli has $H(X) = -\theta \log\theta - (1-\theta)\log(1-\theta)$

$$\nabla_w NLL(y|X,w) = \sum_{i=1}^{N} x_i(\mu_i - y_i) = X^T(\mu - y)$$

$$\begin{cases} \text{Hessian: } H = X^T S X, \ S \text{ is a diagonal matrix where } S_{ii} = \mu_i(1-\mu_i) \\ \text{Gradient: } g = -X^T(\mu - y) \end{cases}$$

The Newton Update Rule:

$$\begin{aligned} w_{t+1} &= w_t - H_t^{-1} g_t \\ &= w_t + (X^T S_t X)^{-1} X^T (y - \mu) \\ &= (X^T S_t X)^{-1} X^T S_t (X w_t + S_t^{-1}(y - \mu)) \\ &= (X^T S_t X)^{-1} X^T S_t z_t, \text{ where } z_t = X w_t + S_t^{-1}(y - \mu) \end{aligned}$$

Then w_{t+1} is a solution of the following:
$\min \sum_{i=1}^{N} S_{t,ii}(z_{t,i} - w^T x_i)^2$, weighted least squares

Multiclass Logistic Regression

(also a discriminative classifier)

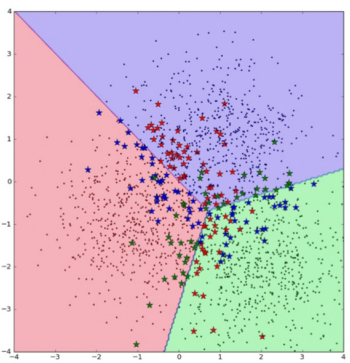
input $x \in \mathbb{R}^D$, output $y \in \{1, 2, \cdots C\}$

parameters $w_c \in \mathbb{R}^D$ for every class $c = 1, \cdots C$, denoted as $W_{D \times C}$

$$p(y=c \mid x, W) = \frac{\exp(w_c^T x)}{\sum_{c'=1}^{C} \exp(w_{c'}^T x)} \qquad W = \begin{bmatrix} | & | & & | \\ w_1 & w_2 & \cdots & w_c \\ | & | & & | \end{bmatrix}_{D \times C}$$

$$\Rightarrow p(y=c \mid x, W) = \text{softmax}([w_1^T x, \ldots, w_c^T x])$$

$$\text{softmax}([a_1, \cdots a_c]^T) = \left[\frac{e^{a_1}}{z}, \cdots \frac{e^{a_c}}{z}\right]^T, \quad z = \sum_{c=1}^{C} e^{a_c}$$



Extension to multiclass:

sigmoid → softmax

Exercises

1. Maximum Likelihood for Logistic Regression

Consider the sigmoid function, defined as $\sigma(z) = \frac{1}{1+e^{-z}}$. Note that $\lim_{z\to-\infty}\sigma(z) = 0$ and $\lim_{z\to\infty}\sigma(z) = 1$. Thus, for binary classification problems, we can compose a linear function with the sigmoid function to model the probability that a given input $\mathbf{x}$ belongs to one of the two classes $\{0, 1\}$. More precisely, for parameter vector $\mathbf{w} \in \mathbb{R}^D$, and input vector $\mathbf{x} \in \mathbb{R}^D$,[1] the label $y \in \{0, 1\}$ is given by the following model:

$$\Pr(y = 1 \mid \mathbf{w}, \mathbf{x}) = \sigma(\mathbf{x}^\mathsf{T}\mathbf{w}) \tag{2.1}$$

$$\Pr(y = 0 \mid \mathbf{w}, \mathbf{x}) = 1 - \sigma(\mathbf{x}^\mathsf{T}\mathbf{w}) \tag{2.2}$$

$\sigma'(z) = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right) = \sigma(z)(1-\sigma(z))$

$P(y_1, y_2, \cdots y_N \mid w, X) = \prod_{i:y_i=1}\sigma(x_i^T w)\prod_{i:y_i=0}(1-\sigma(x_i^T w))$

$NLL(y \mid w, x) = -\log P(y \mid w, x) = -\sum_{i:y_i=1}\log(\sigma(x_i^T w)) - \sum_{i:y_i=0}\log(1-\sigma(x_i^T w))$

$\nabla_w NLL(y \mid w, x) = -\sum_{i:y_i=1}\frac{\sigma(x_i^T w)(1-\sigma(x_i^T w))}{\sigma(x_i^T w)}x_i + \sum_{i:y_i=0}\frac{\sigma(x_i^T w)(1-\sigma(x_i^T w))}{1-\sigma(x_i^T w)}x_i$

$= -\sum_{i:y_i=1}(1-\sigma(x_i^T w))x_i + \sum_{i:y_i=0}\sigma(x_i^T w)x_i$

$= -\sum_{i:y_i=1}x_i + \sum_{i=1}^N \sigma(x_i^T w)x_i$

$H = \nabla_w^2 NLL(y \mid w, x) = \left(\sum_{k=1}^N \sigma(x_k^T w)(1-\sigma(x_k^T w))x_{ki}x_{kj}\right)_{i,j}$

$= \begin{bmatrix} x_{11} & \cdots & x_{N1} \\ \vdots & \ddots & \vdots \\ x_{1D} & \cdots & x_{ND} \end{bmatrix} \begin{bmatrix} \sigma(x_1^T w)(1-\sigma(x_1^T w)) & & \\ & \ddots & \\ & & \sigma(x_N^T w)(1-\sigma(x_N^T w)) \end{bmatrix} \begin{bmatrix} x_{11} & \cdots & x_{1D} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{ND} \end{bmatrix}$

$= X^T S X$

, where S is a diagnol matrix s.t. $S_{ii} = \sigma(x_i^T w)(1-\sigma(x_i^T w))$

S is positive definite because

$\forall y \in \mathbb{R}^{N\times 1}$, $y^T S y = \sum_{i=1}^N y_i^2 S_{ii} \geq 0$, and $y^T S y = 0$ iff. $y = 0$

Thus, H is positive semi-definite because

$\forall y \in \mathbb{R}^{D\times 1}$, $y^T(X^T S X)y = (Xy)^T S (Xy) \geq 0$,

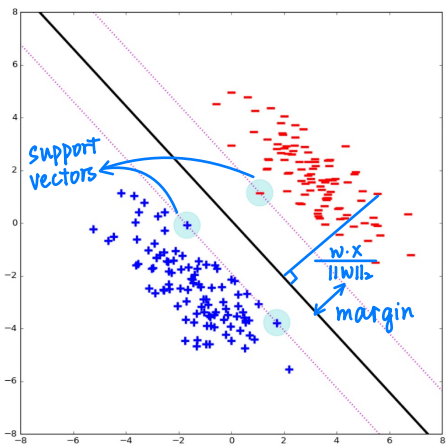
and $(Xy)^T S (Xy) = 0$ iff. $Xy = 0$, which doesn't necessarily mean $y = 0$

Since the Hessian is positive semi-definite, NLL is a convex function of w, and the obtained solution is the global minimum.

SVM (Support Vector Machines)

Binary Classification

1. Goal: find a linear separator
   data is linearly separable iff. all points are classified correctly
2. Maximum Margin Principle: max the distance of the closest point from the decision boundary (support vectors)



Hyperplane: $H \equiv w \cdot x + w_0 = 0$

Distance between $x \in \mathbb{R}^D$ and H: $\frac{w \cdot x + w_0}{\|w\|_2}$

Separable Case:

$\min \frac{1}{2}\|w\|_2^2$ s.t. $y_i(w \cdot x_i + w_0) \geq 1$

(where $y_i \in \{-1, 1\}$) — quadratic program

Non-separable Case:

$\min \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{N} \zeta_i$ ← penalty for slack terms

(regularizer) (loss function)

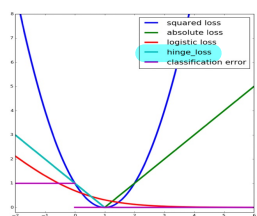
$\zeta_i = \max\{0, 1 - y_i(w \cdot x_i + w_0)\}$ ← hinge loss function

s.t. $y_i(w \cdot x_i + w_0) \geq 1 - \zeta_i$ ← slack to violate constraints

Logistic Regression: Loss Function

$$NLL(y_i; w, x_i) = -\left(y_i \log\left(\frac{1}{1+e^{-w \cdot x}}\right) + (1-y_i)\log\left(\frac{1}{1+e^{-w \cdot x}}\right)\right)$$

$$= \log(1 + e^{-z_i(w \cdot x_i)}) = \log(1 + e^{-(2y_i - 1)(w \cdot x_i)})$$



hinge loss: $y(w \cdot x + w_0)$

logistic loss: $(2y-1)(w \cdot x + w_0)$

Multiclass Classification

- one-vs-one: train $\binom{K}{2}$ classifiers, choose the most commonly occurring y
- one-vs-rest: train K classifiers (one class vs the rest)

One-vs-One	One-vs-Rest
▶ Training roughly $K^2/2$ classifiers	▶ Training only K classifiers
▶ Each training procedure only uses on average $2/K$ of the training data	▶ Each training procedure uses all the training data
▶ Resulting learning problems are more likely to be "natural"	▶ Resulting learning problems are less likely to be "natural"

Exercises

1. SVM problems:

(a) minimize $\frac{1}{2}\|w\|_2^2$
subject to: $y_i(wx_i + w_0) \geq 1$

(b) maximize α
subject to: $y_i(wx_i + w_0) \geq \alpha$, $\|w\|_2 = 1$

(a) $\Leftrightarrow$ (b)

Suppose you use the primal SVM formulation for the non-separable case, *i.e.*, with slack variables ζ_i, but your data is actually linearly separable. Do you always recover the "true" max-margin separating hyperplane?

No, not necessarily.
In non-separable case:
minimum $\frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{N}\zeta_i$
subject to $y_i(w \cdot x_i + w_0) \geq 1 - \zeta_i$

Correctly classified examples: $0 \leq \zeta_i \leq 1$
Misclassified examples: $\zeta_i > 1$

If we set the penalty C to a very small number, the motivation to minimize $\|w\|_2$ weighs more than that to separate the data correctly, which may not be the "true" hyperplane.

2. Kernel methods

Kernel methods are widely applicable in machine learning and not just restricted to support vector machines. For instance, linear regression using kernel basis expansion, is nothing but a *kernelised* form of linear regression. Let us briefly revisit the Ridge regression objective:

$$\mathcal{L}_{\text{ridge}}(\mathbf{w}) = \sum_{i=1}^{N}(\mathbf{w} \cdot \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|_2^2$$

We observe that the optimal solution $\mathbf{w}$ can always be written as $\mathbf{w} = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i$. Why? Suppose not. Let us write $\mathbf{w} = \mathbf{w}_s + \mathbf{w}_\perp$, where $\mathbf{w}_s$ is the component of $\mathbf{w}$ that lies in the span$\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and $\mathbf{w}_\perp$ is orthogonal to this linear subspace spanned by the inputs. Then, since $\mathbf{w}_\perp \cdot \mathbf{x}_i = 0$ for all $i = 1, \ldots, N$, $\mathbf{w}_\perp$ cannot affect $(\mathbf{w} \cdot \mathbf{x}_i - y_i)^2$ at all. However, $\|\mathbf{w}\|_2^2 = \|\mathbf{w}_s\|_2^2 + \|\mathbf{w}_\perp\|_2^2$. Thus, setting $\mathbf{w}_\perp = \mathbf{0}$ can only decrease $\mathcal{L}_{\text{ridge}}$. Thus, equivalently we can try to solve for α_i rather than $\mathbf{w}$, and we get a *kernelised* form of linear regression. You will work out the details on Problem Sheet 3.

Dual Formulation of SVM

primal form: minimize $F(z) = \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{N}\zeta_i$ here (concave, N+D+1 variables)

subject to $g_i(z) \geq 0$ $(i=1,2,\cdots m)$ (convex)

$h_j(z) = 0$ $(j=1,2,\cdots \ell)$ (affine $(w_0 + w_1 z_1 + \cdots w_D z_D)$)

Lagrange Function: $\Lambda(z;\alpha,\mu) = F(z) - \sum_{i=1}^{m}\alpha_i g_i(z) - \sum_{j=1}^{\ell}\mu_j h_j(z)$

For convex problems, Karush-Kuhn-Tucker (KKT) conditions provide necessary and sufficient conditions for a critical point of Λ to be the minimum of the original constrained optimization problem (z^*, α^*, μ^*)

1. Primal Feasibility: $g_i(z^*) \geq 0$, $h_j(z^*) = 0$, $i=1,2,\cdots m$; $j=1,2,\cdots \ell$ * $\nabla_{w,b}\Lambda(w,b,\alpha) = 0$
2. Dual Feasibility: $\alpha_i^* \geq 0$, $i=1,2,\cdots m$
3. Complementary Slackness: $\alpha_i^* g_i(z^*) = 0$, $i=1,2,\cdots m$

$$\Rightarrow \Lambda(w, w_0, \zeta; \alpha, \mu) = \frac{1}{2}w^T w + C\sum_{i=1}^{N}\zeta_i - \sum_{i=1}^{N}\alpha_i(y_i(w\cdot x_i + w_0) - 1 + \zeta_i) - \sum_{i=1}^{N}\underbrace{\mu_i\zeta_i}_{\zeta_i \geq 0}$$

$$\begin{cases} \frac{\partial \Lambda}{\partial w_0} = -\sum_{i=1}^{N}\alpha_i y_i \\ \frac{\partial \Lambda}{\partial \zeta_i} = C - \alpha_i - \mu_i \\ \nabla_w \Lambda = w - \sum_{i=1}^{N}\alpha_i y_i x_i \end{cases} \Rightarrow \quad FOC = 0, \quad 0 \leq \alpha_i \leq C$$

Thus $\Rightarrow g(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i y_i x_i \cdot (\sum_{j=1}^{N}\alpha_j y_j x_j)$

dual form: maximize $\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j)$ (concave, N variables (N^2 terms to express))

subject to $\sum_{i=1}^{N}\alpha_i y_i = 0$, $0 \leq \alpha_i \leq C$

Solution: $w = \sum_{i=1}^{N}\alpha_i y_i x_i$ $\alpha_i = 0$ or $y_i(w\cdot x_i + w_0) = 1 - \zeta_i$

$w_0 = y_j - \sum_{i=1}^{N}\alpha_i y_i x_i^T x_j$ (choose one $\alpha_j \neq 0$)

prediction: $w \cdot x_{new} = \sum_{i=1}^{N}\alpha_i y_i (x_i \cdot x_{new})$ $\Rightarrow$ thus Kernel$(\cdot,\cdot)$ is sufficient

1. Polynomial Basis Expansion: $K_d(x,x') = \phi_d(x)\cdot\phi_d(x') = (1 + x\cdot x')^d$ (time complexity $= O(D + \log d) = O(D\log d)$)

for $d=2$, $\phi_2([x_1,x_2]^T) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2]^T$ $O(D^d)$ variables (D: input features, d: power)

2. Mercer Kernels ($\mathcal{K}: \mathcal{X}\times\mathcal{X} \to \mathbb{R}$, for inputs $\langle x_i \rangle_{i=1}^N$, each $x_i \in \mathcal{X}$)

Gram matrix $K = \begin{bmatrix} K(X_1,X_1) & K(X_1,X_2) & \cdots & K(X_1,X_N) \\ K(X_2,X_1) & K(X_2,X_2) & \cdots & K(X_2,X_N) \\ \vdots & \vdots & \ddots & \vdots \\ K(X_N,X_1) & K(X_N,X_2) & \cdots & K(X_N,X_N) \end{bmatrix}$

$g(\alpha) = 1^T\alpha - \frac{1}{2}(\alpha\odot y)^T K(\alpha\odot y)$ $\Rightarrow$ concave if K is positive definite

is always positive semi-definite

(Here, $K(x,x')$ is some measure of similarities between x and x')

3. Gaussian / RBF Kernels: $K(x,x') = \exp\left(-\frac{\|x-x'\|_2^2}{2\sigma^2}\right)$ bandwidth

Measuring Performance

prediction	(Actual Labels) Y	N
yes	true positive	false positive
no	false negative	true negative

Type I errors ↑ ⇒ Precision: $P = \frac{TP}{TP+FP}$

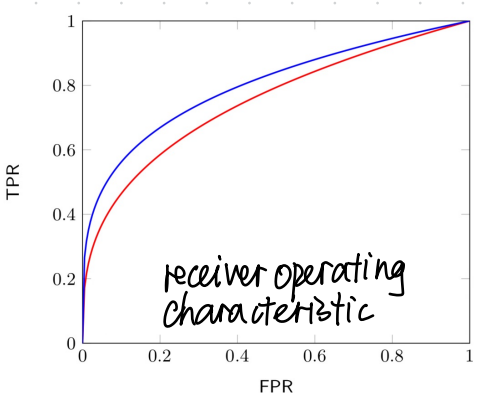
Type II errors

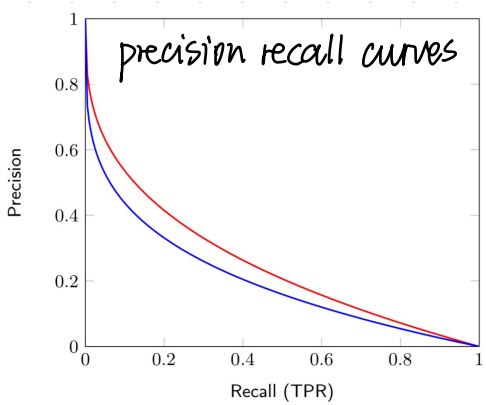
⇓ True Positive Rate $TPR = \frac{TP}{TP+FN}$

⇓ False positive Rate $FPR = \frac{FP}{FP+TN}$

For multi-class classification, it's common to write confusion matrix

prediction	Actual Labels 1	2	...	K
1	N_{11}	N_{12}	...	N_{1K}
2	N_{21}	N_{22}	...	N_{2K}
⋮	⋮	⋮	⋱	⋮
K	N_{K1}	N_{K2}	...	N_{KK}





Neural Network

slides 14~16 - Neural Networks