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) $\Rightarrow Cov(\varepsilon_i, \varepsilon_j) = 0$

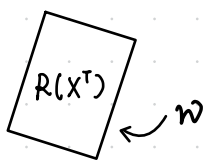
More on OLS

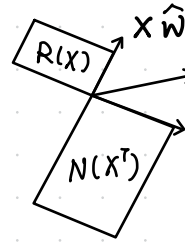
Linear Model: $y = w_0 + w_1 x_1 + \cdots + w_D x_D + \varepsilon = w^T x + \varepsilon$

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

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

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





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

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

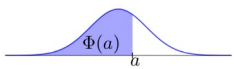
$$= \frac{\sigma^2}{SST_x}\ (D=1), \quad \frac{\sigma^2}{SST_j(1-R_j^2)}\ (D>1)$$

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

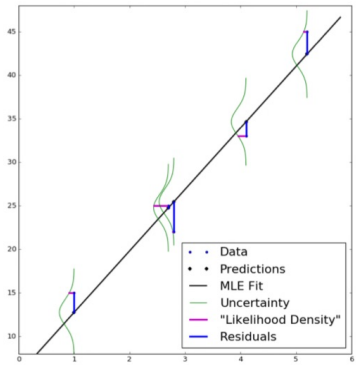
Linear Model: $y = w_0x_0 + w_1x_1 + \cdots w_Dx_D + \underbrace{\varepsilon}_{\text{aleatoric uncertainty}} = w \cdot x + \varepsilon$

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

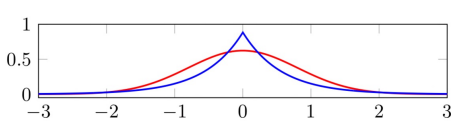
$\Rightarrow \begin{cases} w_{ML} = (X^T X)^{-1} X^T y \\ \sigma^2_{ML} = \frac{1}{N}(Xw_{ML} - y)^T(Xw_{ML} - y) \end{cases}$

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

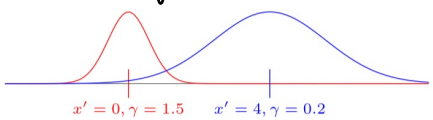
slides05 - fitting
slides06 - regulization
slides07 - Bayesian

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

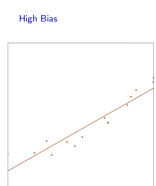
- underfitting: γ too small (large width)
- overfitting: γ too large (small width)

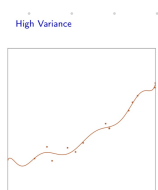
1. choose centres: $\mu_1, \mu_2, \cdots \mu_M$
2. feature map: $\phi(x) = [1, \kappa(\mu_1, x), \cdots \kappa(\mu_M, x)]$
3. model: $y = w_0 + w_1 \kappa(\mu_1, x) + \cdots w_M \kappa(\mu_M, x) + \varepsilon = w \cdot \phi(x) + \varepsilon$

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)

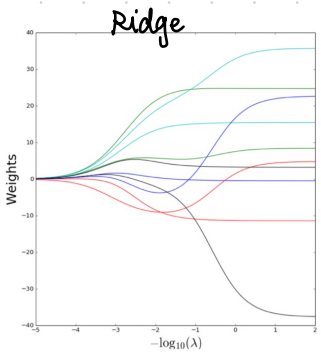
$$\nabla_w \mathcal{L}_{ridge} = 2(X^TX)w - 2X^Ty + 2\lambda w$$
$$= 2[(X^TX + \lambda I_D)w - X^Ty]$$
$$\Rightarrow w_{ridge} = (X^TX + \lambda I_D)^{-1} X^Ty$$

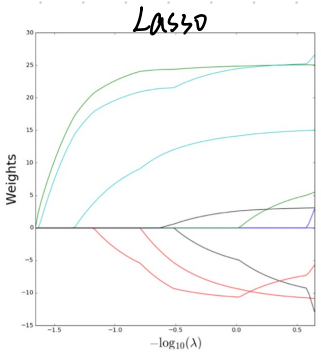
Lasso Regression Objective $\mathcal{L}_{ridge}(w) = (Xw-y)^T(Xw-y) + \lambda \sum_{i=1}^{D} |w_i|$

(*) 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

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

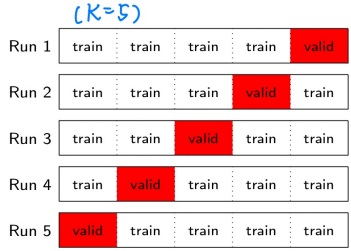
```
# model selection
from sklearn.moel_selection import GridSearchCV

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

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

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

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

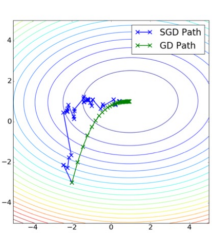
Sub-gradient Descent: range between the left and right derivatives

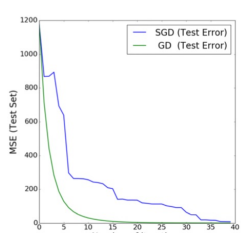
Stochastic Gradient Descent (SGD)

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