CMPUT 567: Machine Learning 2

Keven Qiu
Instructor: Martha White

1

The generalization error for a given f, with squared error is

$$E_{x,y}[(f(X)-Y)^2] = \underbrace{E[(f(X)-f^*(X))^2]}_{\text{reducible error}} + \underbrace{E[(f^*(X)-Y)^2]}_{\text{irreducible error}}$$

where $f^*(x) = E[Y|x]$ and the irreducible error is $E_x[\text{Var}(Y|X)]$.

Bias and variance is about the function class $\mathcal{F}$ (and algorithm, i.e. objective and optimizer).

$$f_{\mathcal{D}} = \text{Alg}(\mathcal{D})$$

where $\mathcal{D} \sim \underbrace{p(x_1,y_1)p(x_2,y_2)\cdots p(x_n,y_n)}_{p(\mathcal{D})}$. Then

$$E_{\mathcal{D}}[GE(f_{\mathcal{D}})] = \underbrace{E_{\mathcal{D}}[\text{reducible error } (f_{\mathcal{D}})]}_{\text{bias}^2 + \text{variance}} + \text{irreducible error}$$

Note there are two variances: the irreducible error and the variance of our function f.

For one x:

$$\begin{aligned} E_{\mathcal{D}}[(f_{\mathcal{D}}(x)-f^*(x))^2] &= E_{\mathcal{D}}[(\underbrace{f_{\mathcal{D}}(x)-E_{\mathcal{D}}[f_{\mathcal{D}}(x)]}_{a} + \underbrace{E_{\mathcal{D}}[f_{\mathcal{D}}(x)]-f^*(x)}_{b})^2] \\ &= \underbrace{b^2}_{\text{bias}^2} + \underbrace{a^2}_{\text{variance}} + 2\underbrace{E_{\mathcal{D}}[f_{\mathcal{D}}(x)-E_{\mathcal{D}}[f_{\mathcal{D}}(x)]]}_{=0}b \\ &= (E_{\mathcal{D}}[f_{\mathcal{D}}(x)]-f^*(x))^2 + \underset{\mathcal{D}}{\text{Var}}(f_{\mathcal{D}}(x)) \end{aligned}$$

If we can represent f^* (high model complexity), then $E_{\mathcal{D}}[f_{\mathcal{D}}(x)] \approx f^*(x)$ for all x.

1.1 Bias and Variance for Linear Regression

With nonlinear representation Φ, we have by SVD

$$\Phi = U\Sigma V^T \in \mathbb{R}^{n\times p}$$

where $U = [u_1,\dots,u_n]$, with left singular vectors $u_j \in \mathbb{R}^{n\times 1}$ and $V = [v_1,\dots,v_p]$, with right singular vectors $v_j \in \mathbb{R}^{p\times 1}$, and $\Sigma \in \mathbb{R}^{n\times p}$.

We consider thin SVD, where we consider only the first p rows/columns.

$$\Sigma = \begin{bmatrix} \sigma_1 & \cdots & 0 \\ 0 & \ddots & 0 \\ 0 & \cdots & \sigma_p \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}, \Sigma_p = diag(\sigma_1, \dots, \sigma_p), U_p = [u_1, \dots, u_p]$$

since $U\Sigma = U_p\Sigma_p$. Note that $\Sigma^T\Sigma = \Sigma_p^2$.

$$\begin{aligned} \Phi^T\Phi &= V\Sigma^T U^T U \Sigma V^T \\ &= V\Sigma^T\Sigma V^T \\ &= V\Sigma_p^2 V^T \\ \mathbf{w} &= (\Phi^T\Phi)^{-1}\Phi^T y \\ &= V\Sigma_p^{-2}V^T V\Sigma_p U_p^T y \\ &= V\Sigma_p^{-1}U_p^T y \\ &= \sum_{j=1}^{p} \frac{u_j^T y}{\sigma_j} v_j \end{aligned}$$

Note that for this, all singular values must be nonzero/positive. $\Phi^t = V\Sigma_p^{-1}U_p^T$ is known as the pseudoinverse of Φ.

If Φ has lower rank $r < p$, i.e. $\sigma_1, \dots, \sigma_r > 0, \sigma_{r+1}, \dots, \sigma_p = 0$, then $\Phi^t = V_r\Sigma_r^{-1}U_r^T$ where $V_r = [v_1, \dots, v_r]$.

Notice that **w** is sensitive to small singular values since we divide by them. Reasons why σ_j can be small:

- linearly correlated features (one dimension might not have anything)
- insufficient data

1.2 ℓ_2-Regularization

For linear regression, the MLE is $p(y|x) = \mathcal{N}(\phi w, \sigma^2)$. MAP with Gaussian prior is $-\ln p(w_j) =$ constant $+ \frac{\lambda}{2\sigma^2}w_j^2$.

$$c(w) = \frac{1}{2}\left\|\Phi w - y\right\|_2^2 + \frac{\lambda}{2}||w||_2^2 \implies (\Phi^T\Phi + \lambda I)w = \Phi^T y \implies w = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T y$$

Using SVD $\Phi^T\Phi + \lambda I = V(\Sigma_p^2 + \lambda I)V^T$,

$$w_{MAP} = V(\Sigma_p^2 + \lambda I)^{-1}U_p^T y = \sum_{j=1}^{p} \frac{\sigma_j}{\sigma_j^2 + \lambda}(u_j^T y)v_j$$

As long as large λ, the fraction does not blow up like it did without ℓ_2-regularization.

1.3 Better Generalization Error

Realizable setting: $y \sim \mathcal{N}(\Phi w^*, \sigma^2)$ for some w^* and $f^* \in \mathcal{F}$ (f^* is in function class).

Let $w(\mathcal{D})$ be our estimator.

$$MSE(w(\mathcal{D})) = E_{\mathcal{D}}\left[\sum_{j=1}^{p}(w_j(\mathcal{D}) - w_j^*)^2\right] = \sum_{j=1}^{p}(E[w_j(\mathcal{D})] - w_j)^2 + \sum_{j=1}^{p}\text{Var}(w_j(\mathcal{D}))$$

Notice $y = \phi w^* + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. For MLE,

$$\begin{aligned} E[w_{MLE}(\mathcal{D})] &= E[(\Phi^T\Phi)^{-1}\Phi^T y] \\ &= E[(\Phi^T\Phi)^{-1}\Phi^T(\Phi w^* + \varepsilon)] \\ &= E[(\Phi^T\Phi)^{-1}(\Phi^T\Phi)w^*] + \underbrace{E[(\Phi^T\Phi)^{-1}\Phi^T]\,E[\varepsilon]}_{=0} \\ &= w^* \end{aligned}$$

This is unbiased as it equals w^*. For MLE variance,

$$\sum_{j=1}^{p}\text{Var}(w_{MLE}(\mathcal{D})) = \sigma^2 E\left[\sum_{j=1}^{p}\sigma_j^{-2}\right]$$

where first σ^2 is variance, not singular value. This can be very big if σ_p is often small.

For MAP,

$$E[w_{MAP}(\mathcal{D})] = E[(\Phi^T\Phi + \lambda I)^{-1}\Phi^T(\Phi w^* + \varepsilon)] \neq w^*$$

so this is biased as λ increases.

$$\sum_{j=1}^{p}\text{Var}(w_{MAP}(\mathcal{D})) = \sigma^2 E\left[\sum_{j=1}^{p}\frac{\sigma_j^2}{(\sigma_j^2 + \lambda)^2}\right]$$

This is not very big if λ is large.

Non-realizable: $y = \mathcal{N}(f^*(x), \sigma^2)$, but $f^* \notin \mathcal{F}$. w_{MLE} is also biased.

Note about n: We can stick a $\frac{1}{n}$ in front of $\Phi^T\Phi$ to act like an average over outerproducts. Note $\frac{1}{n}\Phi^T\Phi = E[\Phi^T\Phi]$.

Without normalization, $\sigma_j^2 \to \infty$ as $n \to \infty$. With normalization $\frac{1}{n}\sigma_j^2 \to$ singular values for $E[\Phi^T\Phi]$.

$\sum_{j=1}^{p}\sigma_j^{-2} \to 0$ as $n \to \infty$, so $\text{Var}(w_{MLE}(\mathcal{D})) \to 0$ as $n \to \infty$.

2

Multiple outputs: $y \in \mathbb{R}^m, \phi(x)W \approx y$.

$$Y = \begin{bmatrix} y_{11} & \cdots & y_{1m} \\ \vdots & \ddots & \vdots \\ y_{n1} & \cdots & y_{nm} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1 & \cdots & \mathbf{y}_m \end{bmatrix}$$

where $\mathbf{y}_i \in \mathbb{R}^n$. So
$$\mathbf{w} = \Phi^t Y = \begin{bmatrix}\Phi^t \mathbf{y}_1 & \cdots & \Phi^t \mathbf{y}_m\end{bmatrix}$$

2.1 Optimization

Find
$$\min_{w \in \mathbb{R}^p} c(w)$$
For example, $c(w) = \frac{1}{n}\sum_{i=1}^n c_i(w)$, linear regression $c_i(w) = \frac{1}{2}(\phi(x_i)w - y_i)^2$ or logistic regression $c_i(w) = -y_i \ln f_w(x_i) - (1-y_i)\ln(1 - f_w(x_i))$.

1. Find a stationary point w_0, $\nabla c(w_0) = 0$.
2. Check type using second derivative test ($c_i'(w_t) < 0, > 0, = 0$).

In p-dimensions, the **Hessian** matrix $H_{c(w_t)} \in \mathbb{R}^{p \times p}$, where
$$H_{c(w)}[i,j] = \frac{\partial^2 c}{\partial w_i \partial w_j}(w)$$

E.g. $c(w) = \frac{1}{2}(w_1 + xw_2 - y)^2$, $\nabla c(w) = \begin{bmatrix}(w_1 + xw_2 - y)\\(w_1 + xw_2 - y)x\end{bmatrix} = \begin{bmatrix}\frac{\partial c}{\partial w_1}(w)\\ \frac{\partial c}{\partial w_2}(w)\end{bmatrix}$.
$$H_{c(w)} = \begin{bmatrix}\frac{\partial}{\partial w_1}\frac{\partial}{\partial w_1}c(w) & \frac{\partial}{\partial w_1}\frac{\partial}{\partial w_2}c(w)\\ \frac{\partial}{\partial w_2}\frac{\partial}{\partial w_1}c(w) & \frac{\partial}{\partial w_2}\frac{\partial}{\partial w_2}c(w)\end{bmatrix} = \begin{bmatrix}1 & x\\ x & x^2\end{bmatrix}$$
For $(x = 0.1, y = 2)$ and $w_0 = (0,0)$,
$$\nabla c(w) = \begin{bmatrix}-2\\-0.2\end{bmatrix}, H_{c(w_0)} = \begin{bmatrix}1 & 0.1\\0.1 & 0.01\end{bmatrix}$$
The eigenvalues of $H_{c(w_0)}$ are $\lambda_1 \approx 1, \lambda_2 = 0$. So we get a concave up in one direction, and flat in the other direction (a taco shape).

E.g. $c(w) = \frac{1}{2}(w_1 + x_1w_2 - y_1)^2 + \frac{1}{2}(w_1 + x_2w_2 - y_2)^2$. At $(x_1 = 0.1, y_1 = 2)$ and $(x_2 = -0.3, y_2 = -1)$,
$$H_{c(w_0)} = \begin{bmatrix}2 & -0.2\\-0.2 & 0.1\end{bmatrix}$$
for $w_0 = (0,0)$. The eigenvalues are $\lambda_1 = 6.5, \lambda_2 = 0.08$.

Better Steps: $H^{-1}_{c(w)}$ is a matrix stepsize. The second-order method is
$$w_{t+1} = w_t - H^{-1}_{c(w_t)}\nabla c(w_t)$$
The scalar version was $w_{t+1} = w_t - \frac{1}{c''(w_t)} \cdot c'(w_t)$.

Using eigenvalue decomposition, $H^{-1}_{c(w_t)}\nabla c(w_t) = U\Lambda^{-1}\underbrace{U^T\nabla c(w_t)}_{\tilde{g}_t}$. This shows that the step is inversely proportional to curvature magnitude (λ_i's). One step size could be bad, since in regular gradient descent uses stepsize η, so Hessian allows us not to step too much on step surfaces and more on flat surfaces.

Computational issues: Computing $H^{-1}_{c(w)}$ is $O(p^2n)$ for linear regression. This is because $H_{c(w)} = \frac{1}{n}\sum_{i=1}^{n} H_{c_i(w)}$, where $H_{c_i(w)}$ takes $O(p^2)$ and you sum n times. Computing the inverse is $O(p^3)$, but for $p < n$, $p^3 < p^2n$, so $O(p^2n)$ overall.

An alternative is to use a vector of stepsizes $\eta_t \in \mathbb{R}^p$. So

$$H^{-1}_{c(w_t)} \approx diag(\eta_{t,1}, \eta_{t,2}, \dots, \eta_{t,p})$$

and the update is

$$w_{t+1} = w_t - \eta_t \overbrace{\cdot}^{\text{element-wise product}} \nabla c(w_t)$$

We do not use a fixed $\eta \in \mathbb{R}^p$. We use vector stepsizes, that change with time.

Definition 2.1: Adagrad

$$\eta_{t,j} = (1 + \bar{g}_{t,j})^{-1/2}$$

where $\bar{g}_{t,j} = \bar{g}_{t-1,j} + g^2_{t,j}$ and $\bar{g}_0 = 0$.

This has nice theory, but in practice, stepsizes shrink too much since we have a monotonic decrease in stepsize.

An intuitive modification is to take an exponential average of $g^2_{t,j}$, which leads to RMSProp.

Definition 2.2: RMSProp

$$v_{t,j} = (1-\beta)g^2_{t,j} + \beta v_{t-1,j}, \beta > 0$$
$$\eta_{t,j} = \frac{\eta}{\sqrt{v_{t,j} + \varepsilon}} = \eta(v_{t,j} + \varepsilon)^{-1/2}$$

Also add **momentum**: with fixed stepsize η,

$$w_{t+1} = w_t - \eta m_{t+1}, m_{t+1} = g_t + \beta_2 m_t, \beta_2 > 0$$

An alternate form is

$$w_{t+1} = w_t - \eta g_t + \beta(w_t - w_{t-1})$$

Definition 2.3: Adam

Puts RMSProp and momentum together with extra modification.

There is a bias correct for $v_{0,j} = 0$. There is an exponential average in momentum $m_{t+1} = (1 - \beta_2)g_t + \beta_2 m_t$. The last modification is

$$\text{Adam's } \frac{1}{\sqrt{v_{t,j}} + \varepsilon} \text{ vs. RMSProp's } \frac{1}{\sqrt{v_{t,j} + \varepsilon}}$$

2.2 Comparisons

Second-order updates takes $O(p^2n)$ per iteration and gradient descent takes $O(pn)$ per iteration.

Definition 2.4: Stochastic Gradient Descent (SGD)

Randomly pick $\mathcal{B}_t \subseteq \{1, \dots, n\}$ with $|\mathcal{B}_t| = b$, the gradient $\hat{g}_t = \frac{1}{b}\sum_{i\in\mathcal{B}_t} \nabla c_i(w_t)$,

$$w_{t+1} = w_t - \eta_t \hat{g}_t$$

The runtime is $O(pb)$.

This is much faster than $O(pn)$ for GD. $\hat{g}_t \approx g_t = \frac{1}{n}\sum_{i=1}^{n} c_i(w_t)$.