1/ $J(x) = \|Ax - y\|_2^2 + \lambda \|x\|_2^2$ $\quad A \in \mathbb{R}^{N\times D}, x \in \mathbb{R}^D$

$= (Ax-y)^T(Ax-y) + \lambda x^T x$

$$g(x) = \frac{\partial J}{\partial x} = \frac{\partial\left(x^T A^T A x - 2x^T A^T y + y^T y + \lambda x^T x\right)}{\partial x}$$

$$= 2A^T A x - 2A^T y + 2\lambda x$$

For exact line search, the optimal step size for n-th step:

$$\mu_n^* = \underset{\mu \geq 0}{\operatorname{argmin}}\ J(x_{n-1} - \mu g(x_{n-1})) \quad (*)$$

Consider $J(x_{n-1} - \mu g(x_{n-1}))$

$$= \|A(x_{n-1} - \mu g(x_{n-1})) - y\|_2^2 + \lambda \|x_{n-1} - \mu g(x_{n-1})\|_2^2$$

- $\|A(x_{n-1} - \mu g(x_{n-1})) - y\|_2^2$

$= \|Ax_{n-1} - \mu A g(x_{n-1}) - y\|_2^2$

$= (Ax_{n-1} - \mu Ag(x_{n-1}) - y)^T (Ax_{n-1} - \mu Ag(x_{n-1}) - y)$

$= x_{n-1}^T A^T A x_{n-1} - x_{n-1}^T A^T \mu A g(x_{n-1}) - x_{n-1}^T A^T y$

$- \mu g^T(x_{n-1}) A^T A x_{n-1} + \mu g^T(x_{n-1}) A^T \mu A g(x_{n-1})$

$+ \mu g^T(x_{n-1}) A^T y - y^T A x_{n-1} + y^T \mu A g(x_{n-1}) + y^T y$

By denoting $D = A^T A$, we have:

$\|A(x_{n-1} - \mu g(x_{n-1})) - y\|_2^2$

$= x_{n-1}^T D x_{n-1} - \mu x_{n-1}^T D g(x_{n-1}) - x_{n-1}^T A^T y$

$- \mu g^T(x_{n-1}) D x_{n-1} + \mu^2 g^T(x_{n-1}) D g(x_{n-1})$

$+ \mu g^T(x_{n-1}) A^T y - y^T A x_{n-1} + \mu y^T A g(x_{n-1}) + y^T y$

$$= \mu^2 g^T(x_{n-1}) D g(x_{n-1})$$
$$+ \mu \left[- x_{n-1}^T D g(x_{n-1}) - g^T(x_{n-1}) D x_{n-1} + g^T(x_{n-1}) A^T y + y^T A g(x_{n-1}) \right] + C_1$$

with $C_1 = x_{n-1}^T D x_{n-1} - x_{n-1}^T A^T y - y^T A x_{n-1} + y^T y$

(The reason why do not write C explicitly is we will get the gradient with respect to μ, therefore we only concern terms having μ).

- $\lambda \| x_{n-1} - \mu g(x_{n-1}) \|_2^2$

$$= \lambda (x_{n-1} - \mu g(x_{n-1}))^T (x_{n-1} - \mu g(x_{n-1}))$$
$$= \lambda (x_{n-1}^T x_{n-1} - x_{n-1}^T \mu g(x_{n-1}) - \mu g^T(x_{n-1}) x_{n-1} + \mu^2 g^T(x_{n-1}) g(x_{n-1}))$$
$$= \mu^2 g^T(x_{n-1}) g(x_{n-1}) \lambda - 2\mu x_{n-1}^T g(x_{n-1}) \lambda + C_2$$

with $C_2 = \lambda x_{n-1}^T x_{n-1}$

The objective function in (*) become:

$$J(\mu) = \mu^2 \left[g^T(x_{n-1}) D g(x_{n-1}) + \lambda g^T(x_{n-1}) g(x_{n-1}) \right]$$
$$+ \mu \left[-2 x_{n-1}^T D g(x_{n-1}) + 2 g^T(x_{n-1}) A^T y - 2 \lambda x_{n-1}^T g(x_{n-1}) \right] + C_1 + C_2$$

$J(\mu)$ is quadratic function wrt μ, therefore we are able to solve (*) by stationary condition, then check the found μ^* satisfy greater than equal 0 or not.

$$\frac{\partial J(\mu)}{\partial \mu} = 2\mu \left[g^T(x_{n-1}) D g(x_{n-1}) + \lambda g^T(x_{n-1}) g(x_{n-1}) \right]$$
$$- 2 \left[x_{n-1}^T D g(x_{n-1}) - g^T(x_{n-1}) A^T y + \lambda x_{n-1}^T g(x_{n-1}) \right]$$

By setting $\frac{\partial J(\mu)}{\partial \mu} = 0$, we have:

$$\mu_n = \frac{x_{n-1}^T D g(x_{n-1}) - g^T(x_{n-1}) A^T y + \lambda x_{n-1}^T g(x_{n-1})}{g^T(x_{n-1}) D g(x_{n-1}) + \lambda g^T(x_{n-1}) g(x_{n-1})}$$

$$= \frac{x_{n-1}^T (D + \lambda I) g(x_{n-1}) - g^T(x_{n-1}) A^T y}{g^T(x_{n-1}) (D + \lambda I) g(x_{n-1})}.$$

In summary, in the step $n-1$ of Gradient Descent, we got $x_{n-1} \in \mathbb{R}^{D\times 1}$, $g(x_{n-1}) \in \mathbb{R}^{D\times 1}$ by plugging x_{n-1} into the first formula appearing in solution. Then we calculate the optimal step size μ_n^* by:

$$\boxed{\mu_n^* = \frac{x_{n-1}^T (D + \lambda I) g(x_{n-1}) - g^T(x_{n-1}) A^T y}{g^T(x_{n-1}) (D + \lambda I) g(x_{n-1})}}$$

Then update the next parameter x_n by:

$$x_n = x_{n-1} - \mu_n^* g(x_{n-1})$$

$\rightarrow$ The objective function $J(x) = \|Ax - y\|_2^2 + \lambda\|x\|_2^2$ is also called ridge regression by putting additional term $\lambda\|x\|_2^2$ into estimate mean square error ($\|Ax - y\|_2^2$).

$\rightarrow$ By inclusing λ, we are able to handle the linearly multicolumn dependence in data which make A^TA is not invertible ($x = (A^TA)^{-1}Ay$, also known as closed form solution). when we want to find the exact solution in 1 shot.

Also, the characteristic of L2-norm is uniformity which restrict all element x_i to low value and therefore can help reducing overfitting.

<u>End Q.1</u>

a/ $s(x) = [f(x) - \hat{f}(x)]^2$

Data model:

$$y(x) = f(x) + \varepsilon \quad (f \text{ is unknown, deterministic})$$

Trained function $\hat{f}(\cdot)$ that estimates $f(x)$

$\rightarrow$ $\text{Bias}_S(\hat{f}(x)) = f(x) - E_S[\hat{f}(x)]$

$\rightarrow$ $\text{Var}_S(\hat{f}(x)) = E_S[(\hat{f}(x) - E_S[\hat{f}(x)])^2]$

$\rightarrow$ $\text{MSE}_{S,\varepsilon}(\hat{f}_S(x)) = E_{S,\varepsilon}[s(x)]$

$$= E_{S,\varepsilon}[(y(x) - \hat{f}(x))^2]$$

We need to prove

$$\text{MSE}_{S,\varepsilon}(\hat{f}_S(x)) = \text{Bias}^2_S(\hat{f}(x)) + \text{Var}_S(\hat{f}(x))$$

We have:

$$\text{MSE}_{S,\varepsilon} = E[(y-\hat{f})^2] \overset{y=f+\varepsilon}{=\!=} E[(f+\varepsilon-\hat{f})^2]$$

$$= E[(f-\hat{f})^2 + \varepsilon^2 + 2\varepsilon(f-\hat{f})]$$

$$= E\{(f-\hat{f})^2\} + E\{\varepsilon^2\} + 2E\{(f-\hat{f})\varepsilon\}$$

linearity of Expectation operator

We have: $\text{var}(\varepsilon) = E(\varepsilon^2) - E(\varepsilon)^2$

$$\sigma_\varepsilon^2 = E(\varepsilon^2) - 0 \Rightarrow E\{\varepsilon^2\} = \sigma_\varepsilon^2$$

$$E\{(f-\hat{f})\varepsilon\} \underset{\text{independence}}{=} E\{f-\hat{f}\}E\{\varepsilon\} = E\{f-\hat{f}\}\,0 = 0$$

Therefore:

$$MSE_{S,\varepsilon} = E\{(f-\hat{f})^2\} + \sigma_\varepsilon^2$$

Consider $E\{(f-\hat{f})^2\}$, we have:

$$E\{(f-\hat{f})^2\} = E\{f^2 - 2f\hat{f} + \hat{f}^2\}$$

$$= E\{f^2\} - 2E\{f\hat{f}\} + E\{\hat{f}^2\}$$

Because f is deterministic $\Rightarrow \begin{cases} E\{f^2\} = f^2 \\ E\{f\hat{f}\} = fE\{\hat{f}\} \end{cases}$

$$\Rightarrow E\{(f-\hat{f})^2\} = f^2 - 2fE\{\hat{f}\} + E\{\hat{f}^2\}$$

$$= f^2 + E\{\hat{f}^2\} - 2fE\{\hat{f}\} + E\{\hat{f}\}^2 - E\{\hat{f}\}^2 - 2E\{\hat{f}\}^2 + 2E\{\hat{f}\}^2$$

$$= f^2 + E[\hat{f}^2 + E\{\hat{f}\}^2 - 2E\{\hat{f}\}^2] - 2fE\{\hat{f}\} + E\{\hat{f}\}^2$$

$$= [f^2 - 2fE\{\hat{f}\} + E\{\hat{f}\}^2] + E[\hat{f}^2 + E\{\hat{f}\}^2 - 2E\{\hat{f}\}^2]$$

$$= (f - E\{\hat{f}\})^2 + E\{(f - E\{\hat{f}\})^2\}$$

$$= Bias^2(\hat{f}) + Var(\hat{f})$$

End Q2

3/

- The term "The curse of dimensionality" in non-parametric learning refers to challenges in high-dimensional space (i.e., d grow fast)
- One of parts was introduced is curse of sample size. It means that as the number of dimensions d increases, the fraction of volume considered as neighborhoods would be increase exponentially
- Several challenges:
  + computational complexity: d grows fast → cost for calculating distance between data point increase exponentially.
  + Increased sparsity of data: d ↑ then sparsity ↑ which leads unreliable density estimation, hard to determine the relationships between variables accurately.
  + Overfitting and generalization: d ↑, Pr(overfitting) ↑, therefore need to provide more training data → poor generalization on unseen data.
- Some teniques to mitigate "Curve of Dimensionality"
  + Dimensionality reduction (LDA, PCA) } slow down d
  + Feature selection
  + Regularization (L1 → make model sparsity, L2

4/ Parametric model

$$y = b(x)^T w + \varepsilon \quad , \quad w \in \mathbb{R}^D$$

$\{(y_i, x_i)\}_{i=1}^{N}$: training data.

Prior :

$$w \sim N(m_0, C_0) \quad , \quad m_0 \in \mathbb{R}^D , \quad C_0 \in \mathbb{R}^{D \times D}$$

or $\quad f_*(w) = N(m_0, C_0)$

MAP try to solve :

$$\max_{w} f_*(w \mid y, X)$$

By Bayes rule, we can rewrite $f_*(w \mid y, X)$ as :

$$f_*(w \mid y, X) = \frac{f_*(y, X \mid w) f_*(w)}{f_*(y, X)}$$

$$= \frac{f_*(y \mid X, w) f_*(X \mid w) f_*(w)}{f_*(y, X)} .$$

$$= \frac{f_*(y \mid X, w) f_*(X) f_*(w)}{f_*(y, X)}$$

$$= f_*(y \mid X, w) f_*(w) \, c$$

with $c = \dfrac{f_*(X)}{f_*(y, X)} > 0$

therefore $\max_{w} f_*(w \mid y, X)$ is written equivalently as:

$$\max_{w} f_*(y / X, w) f_*(w)$$

$$\equiv \max_{w} \ln(f_*(y \mid X, w)) + \ln(f_*(w)) \quad (**)$$

By MLE, we found: $\qquad H = [b(x_1), \dots b(x_N)]$

$$f_*(y \mid X, w) = \mathcal{N}(H^T w, V I_N)$$

$c_1 > 0$

$$= c_1 \exp\left\{ -\frac{1}{2} (y - H^T w)^T (V I_N)^{-1} (y - H^T w) \right\}$$

By prior distribution:

$$f_*(w) = \mathcal{N}(m_0, C_0) = c_2 \exp\left\{ -\frac{1}{2} (w - m_0)^T C_0^{-1} (w - m_0) \right\}$$

$c_2 > 0$

$$\Rightarrow \ln(f_*(y \mid X, w)) = \ln(c_1) - \frac{1}{2V} \| y - H^T w \|_2^2$$

$$\Rightarrow \ln(f_*(w)) = \ln(c_2) - \frac{1}{2} (w - m_0)^T C_0^{-1} (w - w_0)$$

(**) become:

$$\max_{w} \ln(c_1) + \ln(c_2) - \frac{1}{2V} \| y - H^T w \|_2^2 - \frac{1}{2} (w - w_0)^T C_0^{-1} (w - m_0)$$

$$\equiv \min_{w} \| y - H^T w \|_2^2 + (w - m_0)^T (V \cdot C_0^{-1}) (w - w_0)$$

$\| y - H^T w \|_2^2$ is quadratic
$(w - m_0)^T (V \Sigma^{-1}) (w - w_0)$ is quadratic $\Bigg\} \Rightarrow$ the objective is convex

By applying stationary condition :

$$\frac{\partial J(w)}{\partial w} = \frac{\partial\left(y^T y + w^T H H^T w - 2 w^T H y + w^T (V C_0^{-1}) w + m_0^T (V C_0^{-1}) m_0 - 2w^T (V C_0^{-1}) m_0\right)}{\partial w}$$

$$= 2 H H^T w - 2 H y + 2 (V C_0^{-1}) w - 2 (V C_0^{-1}) m_0 \overset{set}{=} 0$$

$$\Rightarrow \boxed{w_{MAP} = \left(H H^T + V C_0^{-1} \right)^{-1} \left(H y + V C_0^{-1} m_0 \right)}$$

↑ map solution

The solution in MLE

$$w_{MLE} = (H H^T)^{-1} H y$$

We can see that the MAP work when we have both knowledge about prior distribution and given dataset.

In case of MLE, we only know training data $\{(y_i, x_i)\}_{i=1}^{N}$ while in MAP, we are given prior distribution of w.

→ MAP is generalized version of MLE.

→ If we don't know prior knowledge, we boil down to MLE by considering $C_0 = aI$ with $a \to \infty$

$$w_{MAP} \xrightarrow{C_0^{-1} \to 0} (H H^T)^{-1} H y = w_{MLE}$$

→ If we do not have dataset then we turn into prior distribution maximization

$$H \to 0 \Rightarrow w_{MAP} = (0 + V C_0^{-1})^{-1} (0 + C_0^{-1} m_0 V)$$

$$= (V C_0^{-1})^{-1} (V C_0^{-1}) m_0 = m_0$$

→ Another view of MAP is least square + regularization

$$\min_{x} \underbrace{\| y - H^T x \|}_{\text{least square error}} + \underbrace{(x - m_0)^T (V C_0^{-1}) (x - m_0)}_{\substack{\text{pernalized term} \\ \text{by prior knowledge}}}$$

End Q4

5/ Binary classification with Logistic regression and training data $\{(t_n, x_n)\}_{n=1}^N$

Cross-entropy loss :

$$\boxed{\begin{aligned} L(w) &= -\sum_{n=1}^{N} \left(\ln(y_n) t_n + \ln(1-y_n)(1-t_n) \right) \\ &= -\sum_{n=1}^{N} \left(\ln(\sigma(w^T b(x_n))) t_n + \ln(1-\sigma(w^T b(x_n)))(1-t_n) \right) \end{aligned}}$$

The gradient of $L(w)$:

$$\nabla_w L(w) = \nabla_w - \sum_{n=1}^{N} \left[\ln(\sigma(w^T b(x_n))) t_n + \ln(1-\sigma(w^T b(x_n)))(1-t_n) \right]$$

$$= -\sum_{n=1}^{N} \left(\nabla_w \ln(\sigma(w^T b(x_n))) t_n + \nabla_w \ln(1-\sigma(w^T b(x_n)))(1-t_n) \right)$$

- Consider $\nabla_w \ln(\sigma(w^T b(x_n)))$

$$= \frac{1}{\sigma(w^T b(x_n))} \nabla_w \sigma(w^T b(x_n))$$

$$= \frac{1}{\sigma(w^T b(x_n))} \sigma(w^T b(x_n)) (1-\sigma(w^T b(x_n))) b(x_n)$$

$$= (1-\sigma(w^T b(x_n))) b(x_n)$$

Consider $\nabla_w \ln(1-\sigma(w^T b(x_n)))$

$$= \frac{1}{1-\sigma(w^T b(x_n))} \nabla_w \left(1 - \sigma(w^T b(x_n))\right)$$

$$= \frac{1}{1-\sigma(w^T b(x_n))} \nabla_w \left(-\sigma(w^T b(x_n))\right)$$

$$= \frac{-1}{1-\sigma(w^T b(x_n))} \sigma(w^T b(x_n))\left(1-\sigma(w^T b(x_n))\right) b(x_n)$$

$$= -\sigma(w^T b(x_n))\, b(x_n)$$

Therefore:

$$\nabla_w L(w) = -\sum_{n=1}^{N} \left(1-\sigma(w^T b(x_n))\right) b(x_n) t_n - \sigma(w^T b(x_n)) b(x_n)(1-t_n)$$

$$= -\sum_{n=1}^{N} b(x_n)\left[\left(1-\sigma(w^T b(x_n))\right)t_n - \sigma(w^T b(x_n))(1-t_n)\right]$$

$$= -\sum_{n=1}^{N} b(x_n)\left[t_n - t_n\sigma(w^T b(x_n)) - \sigma(w^T b(x_n)) + t_n\sigma(w^T b(x_n))\right]$$

$$= -\sum_{n=1}^{N} b(x_n)\left(t_n - \sigma(w^T b(x_n))\right)$$

Therefore

$$\boxed{g(w) = \nabla_w L(w) = -\sum_{n=1}^{N} b(x_n)\left(t_n - \sigma(w^T b(x_n))\right)}$$

End Q5

6/

Adam algorithm

1/ Initialize W_0 ; $s_0 = 0$; $\Delta_0 = 0$

2/ For $i = 1, 2, \ldots$ until termination criteria holds

3/ Update s_i : $[s_i]_j = \gamma [s_{i-1}]_j + (1-\gamma)[g_i]_j^2$, $\forall j \in [d]$

4/ Bias correction s : $s = s_i \dfrac{1}{1-\gamma^i}$

5/ Update $V_i = \alpha V_{i-1} + (1-\alpha) g_i$

6/ Bias correction V : $V = V_i \dfrac{1}{1-\alpha^i}$

7/ Update parameter W_i : $W_i = W_{i-1} - \mu \left(\mathrm{diag}(s)^{\frac{1}{2}} + \varepsilon I_d\right)^{-1} V$

8/ Return W_i

Potential merits of Adam :

1/ It combines benefits of Momentum SGD and RMSProp
- RMSProp characteristic: adaptively change the learning rate
- Momentum characteristic: accelerate convergence.

2/ It includes bias correction which helps fixing the bias when initializing moment estimates at zero → more accurate estimations, specially in early steps of training.

3/ Handles noisy data and works well with square gradients more effectively compared to Momentum SGD and RMSProp

4/ Can get good performance without too much effort in tuning hyperparameters.

End Q6

7/ K- Class Softmax Logistic Regression and training data $\{(t_n, x_n)\}_{n=1}^{N}$

<u>Cross - entropy loss</u>

$$\boxed{\begin{aligned} L(W) &= -\ln(p(T \mid H, W)) \\ &= -\ln\left(\prod_{n=1}^{N}\prod_{k=1}^{K}[Y(W)]_{k,n}^{[T]_{k,n}}\right) \\ &= -\sum_{n=1}^{N}\sum_{k=1}^{K}[T]_{k,n} \cdot \ln[Y(W)]_{k,n} \\ &\text{with } W \in \mathbb{R}^{D\times K} \end{aligned}}$$

<u>Gradient of $L(W)$</u> $\rightarrow G(W)$

$$\mathbb{R}^{D\times K} \ni G(W) = [g_1(W), \dots, g_K(W)]$$
$$= [H a_1(W), \dots, H a_K(W)]$$

$$= HA(W)$$

$$A(W) = [a_i(W), \dots, a_K(W)] = (Y(W) - T)^T$$

$$\Rightarrow \boxed{G(W) = H(Y(W) - T)^T}$$

with $H = [b(x_1), \dots, b(x_n)]$, $[T]_{:,i} = \{[t_1]_i, \dots, [t_N]_i]^T$

$[Y(W)]_{:,i} = [[y_1(W)]_i, \dots [y_N(W)]_i]^T$

End Q7.

8/

$\{(x_n \in \mathbb{R}^D)\}_{n=1}^N$

L1-PCA formula :

$$\max_{G \in \mathbb{R}^{D\times K},\ G^T G = I_K} \sum_{n=1}^{N} \|G^T x_n\|_1$$

Fixed point iterations algorithm :

1/ Initialize $B \in \{-1, 1\}^{N \times K}$ arbitrarily
2/ Repeat until termination conditions hold
3/ $U, S, V^T \xleftarrow{\text{thin SVD}} (HB)$
4/ $G \longleftarrow UV^T$
5/ $B \longleftarrow \text{sign}(H^T G)$
6/ Return G

End Q8

EE 6363. Advanced Machine Learning
Tuan Van Nguyen
vdp076