CSCI 416/516. Linear Regression & optimization

dimensionality

$y = f(x) = \sum_j w_j x_j + b$, whereas $x = (x_1, x_2, \cdots x_j, \cdots, x_D) \in \mathbb{R}^D$

y is linear in x: $y = wx + b$; $y = w^T x + b$.

the decision boundary is linear; could be a line (when only 1 feature) or a hyperplane (when there are D features)

Loss Function: $\mathcal{L}(y, t) = \underbrace{\frac{1}{2}}_{\text{coefficient}} (\underbrace{y - t}_{\text{residual}})^2$

Cost Function: $\mathcal{J}(w, b) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, t_i) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - t_i)^2$

$= \frac{1}{2N} \sum_{i=1}^{N} (w x_i + b - t_i)^2 = \frac{1}{2N} \sum_{i=1}^{N} \left(\sum_{j=1}^{D} (w_j x_j + b) - t_i \right)^2$

$w = (w_1, w_2, \cdots, w_j, \cdots, w_D)$; $x = (x_1, \cdots x_j, \cdots, x_D)$; $y = w^T x + b$

import numpy as np

$\Rightarrow$ y = np.dot(w, x) + b

one feature across all training sample.

$$X = \begin{bmatrix} x^{(1)T} \\ x^{(2)T} \\ x^{(3)T} \end{bmatrix} = \begin{bmatrix} 8 & 0 & 3 & 0 \\ 6 & 4 & 5 & 3 \\ 2 & 5 & -2 & 8 \end{bmatrix}$$

$\dashrightarrow x^{(1)T}$, one training example

entire set of samples

Vectorization: able to calculate the predictions for every single sample in "one go" (matrix manipulation)

$$w^T X + b\mathbf{1} = \begin{bmatrix} w^T x^{(1)} + b \\ \vdots \\ w^T x^{(N)} + b \end{bmatrix} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix} = \underline{\underline{y}}$$

$\shortparallel$

$X w + b\mathbf{1}$

prediction(s) of the whole set of samples, represented by $X = (x^{(1)T} \ldots x^{(N)T})$

$$\sum_{i=1} (y^{(i)} - t^{(i)})^2 = \|y - t\|^2$$

$$\mathcal{J} = \frac{1}{2N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})^2 = \frac{1}{2N} \|y - t\|^2$$

originally, $y = w^T X + b$; but we can integrate the bias b into the weight matrix. together. X has the shape of $\in \mathbb{R}^{N \times (D+1)}$

$$X = N\left\{ \begin{bmatrix} 1 & x^{(1)T} \\ 1 & x^{(2)T} \\ 1 & x^{(3)T} \end{bmatrix} \right. \in \mathbb{R}^{N \times (D+1)} \qquad w = \begin{bmatrix} b \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} \in \mathbb{R}^{D+1}$$

with the dimensionality of 1.

with the dimensionality of D

After the cost function is defined, there are 2 ways to minimize it.

- algebraic: z^* minimizes $f(z) \Rightarrow \forall z \neq z^*,\ f(z) \geq f(z^*)$
- Calculus: find the global minimum for $f(z^*)$, given z^*

$$\frac{\partial y}{\partial w_j} = \frac{\partial}{\partial w_j}(w^T x + b) = \frac{\partial}{\partial w_j}\left(\sum_{j'}^{D} w_{j'} x_{j'} + b\right)$$

$$= \frac{\partial}{\partial w_j}\left(\cancelto{0}{w_1 x_1} + \dots + \cancelto{x_j}{w_j x_j} + \dots + \cancelto{0}{w_D x_D} + \cancelto{0}{b}\right) = x_j.$$

$$\frac{\partial y}{\partial b} = \frac{\partial}{\partial b}(w^T x + b) = \frac{\partial}{\partial b}\left(\cancelto{0}{w_1 x_1} + \dots + \cancelto{0}{x_j x_j} + \dots + \cancelto{0}{w_D x_D} + \cancelto{1}{b}\right) = 1$$

Chain Rule: $\dfrac{\partial \mathcal{L}}{\partial w_j} = \dfrac{\partial \mathcal{L}}{\partial y} \cdot \dfrac{\partial y}{\partial w_j} = \dfrac{\partial}{\partial y}\left(\frac{1}{2}(y - t)^2\right) \cdot \dfrac{\partial y}{\partial w_j}$

$$= (y - t) \cdot x_j$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial y} \cdot \frac{\partial y}{\partial b} = \frac{\partial}{\partial y}\left(\frac{1}{2}(y-t)^2\right) \cdot \frac{\partial y}{\partial b} = (y-t)\cdot 1 = y-t$$

to find the critical pts with respect to w_j and b (our parameters)

$$\frac{\partial \mathcal{J}}{\partial w_j} = \frac{1}{N}\sum_i^N \frac{\partial \mathcal{L}_i}{\partial w_j} = \frac{1}{N}\sum_i^N (y^i - t^i)\, x_j^i = 0$$

$$\frac{\partial \mathcal{J}}{\partial b} = \frac{1}{N}\sum_i^N \frac{\partial \mathcal{L}_i}{\partial b} = \frac{1}{N}\sum_i^N (y^i - t^i)\cdot 1 = 0$$

critical pt, where the derivative is 0.

Definition for gradient ∇: $\nabla f(w) = \left(\frac{\partial f(w)}{\partial w_1}, \dots, \frac{\partial f(w)}{\partial w_D}\right)^T$

the gradient of the first gradient: $\nabla^2 f(w) = \frac{\partial^2 f(w)}{\partial w_{i,j}}$

$\mathcal{L}^2$ or ℓ^2 regularization: $\mathcal{R}(\vec{w}) = \frac{1}{2}\|w\|_2^2 = \frac{1}{2}\sum_j w_j^2$

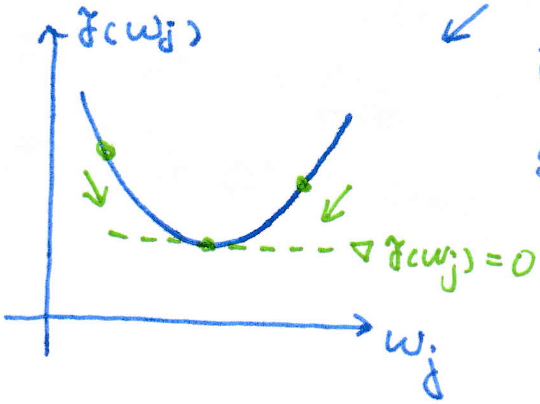
$$\mathcal{J}_{reg} = \mathcal{J}(w) + \lambda \mathcal{R}(\vec{w}) = \mathcal{J}(w) + \frac{\lambda}{2}\sum_j w_j^2 = \frac{1}{N}\sum_i^N \mathcal{L}_i(w) + \frac{\lambda}{2}\sum_j w_j^2$$

$$= \frac{1}{2N}\sum_i^N (y_i - t_i)^2 + \frac{\lambda}{2}\sum_j w_j^2 = \frac{1}{2N}\sum_i^N (\vec{w}^T x_i - t_i)^2 + \frac{\lambda}{2}\sum_j^D w_j^2$$

λ is the hyperparameter here, which is different from the parameters w_j and b.

Gradient Descent.

we have to initialize the parameters, in this figure, w_j, somewhere on the axis. After the initialization, we want to guide w_j's change so that it (gradually) goes to the "global minimum), where $\nabla \mathcal{J}(w_j) = 0$.



$\frac{\partial J}{\partial w_j} > 0 \Rightarrow$ change of J corresponds to : increasing $w_j \Rightarrow$ increasing J

$\frac{\partial J}{\partial w_j} < 0 \Rightarrow$ increasing $w_j \Rightarrow$ decreasing J

gradient descent : always moving against the direction of the gradient

$w_j \leftarrow w_j - \alpha \cdot \frac{\partial J}{\partial w_j}$, whereas α (alpha) is the learning rate, which is also a hyperparameter.

$$\nabla_w J = \frac{\partial J}{\partial w} = \left(\frac{\partial J}{\partial w_1}, \dots, \frac{\partial J}{\partial w_D}\right)$$

$$\frac{\partial J}{\partial w_j} = \frac{1}{N}\sum_i^N \frac{\partial L_i}{\partial w_j} = \frac{1}{N}\sum_i^N (y^i - t^i) x^i$$

$$w_j \leftarrow w_j - \alpha \cdot \frac{1}{N}\sum_i^N (y^i - t^i) x_j^i \; ; \; w \leftarrow w - \frac{\alpha}{N}\sum_i^N (y^i - t^i) x^i$$

Gradient Descent under L_2 norm.

$$w \leftarrow w - \alpha \frac{\partial}{w}(J + \lambda R(w))$$

$$w \leftarrow w - \alpha\left(\frac{\partial J}{\partial w} + \lambda \frac{\partial R(w)}{\partial w}\right)$$

$$w \leftarrow w - \alpha\left(\frac{\partial J}{\partial w} + \frac{\partial \lambda}{\partial w}\frac{1}{2}\|w\|_2^2\right)$$

$$w \leftarrow w - \alpha\left(\frac{\partial J}{\partial w} + \lambda w\right)$$

$$w \leftarrow w - \alpha \cdot \frac{\partial J}{\partial w} - \alpha \cdot \lambda \cdot \cancel{\partial w}$$

$$w \leftarrow \cancel{w}\; w(1 - \alpha \cdot \lambda) - \frac{\partial J}{\partial w} \cdot \alpha.$$

$$\frac{\partial \lambda}{\partial w}\frac{1}{2}\|w\|_2^2$$

$$= \frac{\partial \lambda}{\partial w}\frac{1}{2}\sum w_j^2$$

$$= \left(\frac{\partial \lambda}{\partial w_1} \cdot \frac{1}{2} w_1^2, \frac{\partial \lambda}{\partial w_2} \cdot \frac{1}{2} w_2^2, \dots, \frac{\partial \lambda}{\partial w_D} \cdot \frac{1}{2} w_D^2\right)$$

$$= (\lambda w_1, \lambda w_2, \dots \lambda w_D)$$

$$= \lambda w$$

Stochastic gradient descent

$\mathcal{J}(\theta) = \frac{1}{N}\sum_{i}^{N} \mathcal{L}_i = \frac{1}{N}\sum_{i}^{N} \mathcal{L}(y(x_i, \theta), t_i)$ whereas $\mathcal{L}$ takes t_i and y which takes x_i and θ for computation

instead of $\theta \leftarrow \theta \cdot \lambda \cdot \frac{\partial \mathcal{J}}{\partial \theta}$, we update the paramoter vector $\theta = (\vec{w}, b)$ every time when we train on each sample in the training dataset.