3. Linear Neural Networks

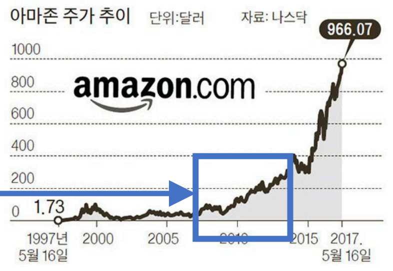
Junggwon Shin, shinjunggwon@gmail.com

Summary for Dive Into Deep Learning, https://d2l.ai/chapter_preface/index.html

A method for modeling the relationship between one or more **independent variables** and **a dependent variable**

Assumptions on Linear Regression

- All data are independently and identically distributed
  - All data are randomly sampled from the same distribution independently
  - Data not following IID: time-series data
- Linear relationship between the independent variables $\mathbf{x}$ and the dependent variable y
  - **Inductive bias** (a bias required for generalization) for the model
- Well-behaved **noise** (following a Gaussian distribution)
  - **Not** expect to find a real-world dataset exactly equals the linear relationship due to factors such as measurement error
  - Thus, incorporating a noise term to account for such errors.



Model

$$\text{price} = w_{\text{area}} \cdot \text{area} + w_{\text{age}} \cdot \text{age} + b.$$

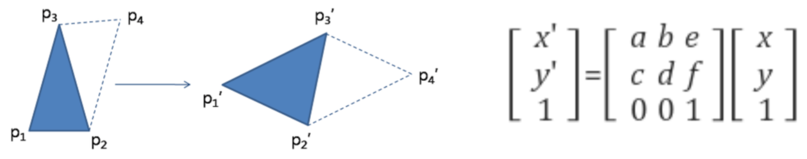
dependent variable y (output) independent variables $\mathbf{x}$ (input)

$$\hat{y} = w_1 x_1 + \ldots + w_d x_d + b.$$

expressing compactly using a dot product

$$\hat{y} = \mathbf{w}^\top \mathbf{x} + b.$$

- $\mathbf{w}$: weight
- b: bias (also called as offset or intercept)
- Strictly speaking, linear regression is an **affine transformation**
  - a kind of space transformation conserving linearity and parallelism including rotation, reflection, scaling, and others
- **Goal**: Choosing weights $\mathbf{w}$ and bias b which optimally fit the given set of pairs between input and output data



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

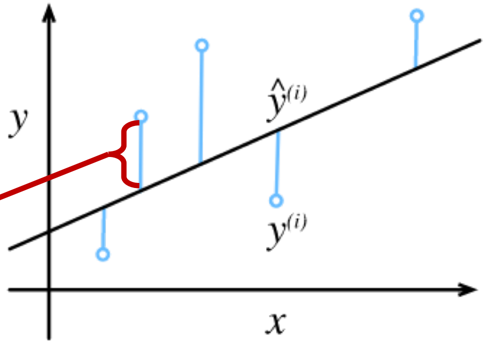
Affine transformation

Elements to find optimal parameter **w**, b

- Loss function: a quality measure for some given model
- Stochastic Gradient Descent: a procedure for updating the model to improve its quality

Loss function

- Quantifying the distance between the real and predicted value of the target
  - Non-negative number where smaller values are better and perfect predictions incur a loss of 0
- Squared error
  - prediction for an data sample i, $\hat{y}^{(i)}$
  - sample target (true label) i, $y^{(i)}$



$$l^{(i)}(\mathbf{w}, b) = \frac{1}{2}\left(\hat{y}^{(i)} - y^{(i)}\right)^2.$$

Loss for dataset of n examples

$$L(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n} l^{(i)}(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}\left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right)^2.$$

Find optimal parameter that minimize
the total loss across all training examples

$$\mathbf{w}^*, b^* = \underset{\mathbf{w},b}{\operatorname{argmin}}\ L(\mathbf{w}, b).$$

Normal Equation

- Analytic solution for the simple liner regression problem

$$\mathbb{y} = X\Theta + \boxed{\mathbb{e}}\ \text{error}$$

$$\mathbb{e} = \mathbb{y} - X\Theta \xrightarrow[\text{squared error}]{\text{Minimize}}$$

$$\sum_{j=1}^{n} \epsilon_j^2 = \mathbb{e}^T\mathbb{e} = (\mathbb{y} - X\Theta)^T(\mathbb{y} - X\Theta)$$
$$= \mathbb{y}^T\mathbb{y} - 2\Theta^T X^T \mathbb{y} + \Theta^T X^T X \Theta$$

Find optimal weight

$$\frac{\partial(\mathbb{e}^T\mathbb{e})}{\partial\Theta} = -2X^T\mathbb{y} + 2X^T X\Theta = \mathbf{0}$$

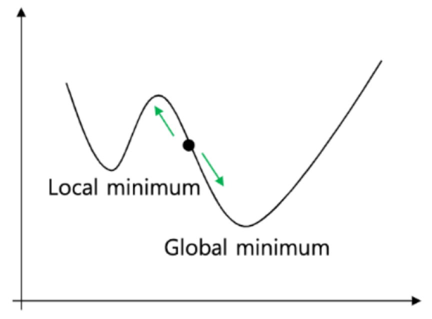
$$X^T X\mathbf{\Theta} = X^T\mathbb{y} \Rightarrow \hat{\mathbf{\Theta}} = (X^T X)^{-1} X^T \mathbb{y}$$

Limitations of Normal Equation

- Hard to calculate a inverse matrix for large-scale data or features, O($n^{2.376}$)
- No inverse matrix can be existed if collinearity between features exists (singular matrix)

Gradient Descent

- Iteratively reducing the error by updating the parameters in the direction that incrementally lowers the loss function
- Taking the derivative of the loss function, which is an average of the losses computed on every single example in the dataset
- ***Stochastic Gradient Descent*** $\longrightarrow$ for $i = 1$ to n : { $\theta_j^{t+1} = \theta_j^t - \alpha(\Theta^T \mathbb{x}_i - y_i)x_i^{(j)}$ for every j }
    - Updating parameters for a gradient of every sampled data
    - Pros: fast calculation speed, small memory requirement
    - Cons: non-stable (largely fluctuating) learning procedure, higher possibility to fall into local minimum
- ***Batch Gradient Descent*** $\longrightarrow$ $\theta_j^{t+1} = \theta_j^t - \alpha \sum_{i=1}^{n} (\Theta^T \mathbb{x}_i - y_i)\, x_i^{(j)}$ for every j
    - update parameters for the mean of calculated gradients for all data
    - Pros: converged into global minimum
    - Cons: low calculation speed, large memory requirement
- ***Mini Batch Gradient Descent***
    - update parameters for the mean of calculated gradients for a **subset** of given data



Mini Batch Gradient Descent

- Initializing the values of the model parameters, typically at random
- Randomly sampling a minibatch $\mathcal{B}$ consisting of a fixed number of training examples
  - $|\mathcal{B}|$ represents the number of examples in each minibatch
- Updating the parameters in the direction of the negative gradient
  - ∂_i denotes the partial derivative of parameter I
  - η denotes the learning rate

$$\begin{aligned}\mathbf{w} &\leftarrow \mathbf{w} - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \partial_{\mathbf{w}} l^{(i)}(\mathbf{w}, b) = \mathbf{w} - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \mathbf{x}^{(i)}\left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right),\\ b &\leftarrow b - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \partial_b l^{(i)}(\mathbf{w}, b) = b - \frac{\eta}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right).\end{aligned}$$

Hyperparameter Tuning

- These parameters that are **tunable** but **not updated in the training loop**
- Typically adjust hyperparameters based on the results of the training loop as assessed on a separate **validation dataset**
- $|\mathcal{B}|$ and η for the simple linear regression model

Stopping Training

- Finishing training after certain number of iterations or until some stopping criteria met
- The trained parameters **will not exact minimizers** of the loss because it cannot achieve it exactly in a finite number of steps

Mini Batch Gradient Descent

- Initializing the values of the model parameters, typically at random
- Randomly sampling a minibatch $\mathcal{B}$ consisting of a fixed number of training examples
  - $|\mathcal{B}|$ represents the number of examples in each minibatch
- Updating the parameters in the direction of the negative gradient
  - ∂_i denotes the partial derivative of parameter I
  - η denotes the learning rate

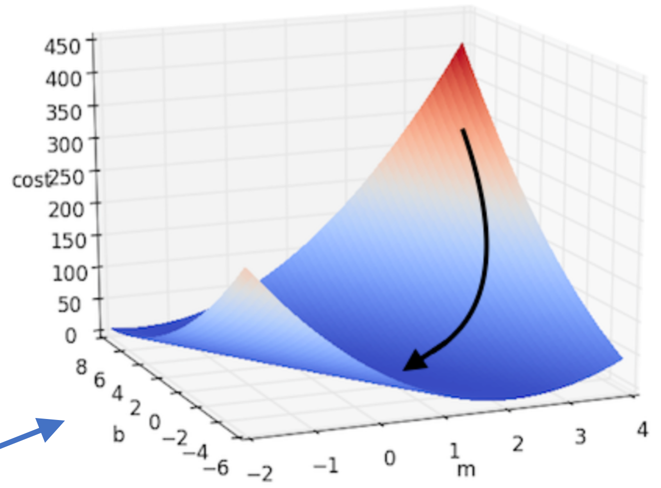
$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \partial_{\mathbf{w}} l^{(i)}(\mathbf{w}, b) = \mathbf{w} - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathbf{x}^{(i)} \left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right),$$

$$b \leftarrow b - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \partial_b l^{(i)}(\mathbf{w}, b) = b - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right).$$

Loss surface of a simple linear regression model

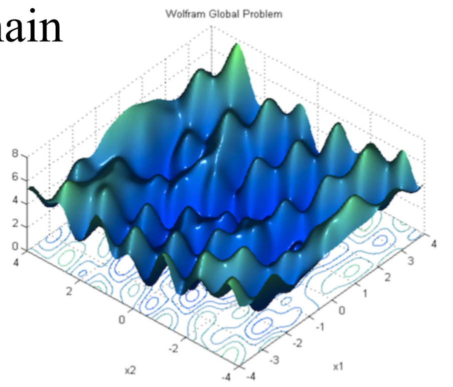
$$m := m - \alpha \cdot \frac{\partial J}{\partial m}$$

$$b := b - \alpha \cdot \frac{\partial J}{\partial b}$$



Loss function of Linear Regression and Deep Neural Networks

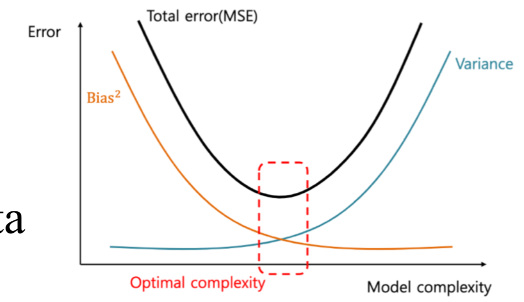
- Linear regression works for a learning problem where there is only one minimum over the entire domain
  - The bowl-shaped loss-function
  - Model with simple inductive bias → model with **high bias error** for complex data
- DNN model contains loss surfaces with many minima
  - The loss function of DNN is not bowl-shaped and not convex (much more complex)
  - Model with complex inductive bias → model with **low bias error** for complex data

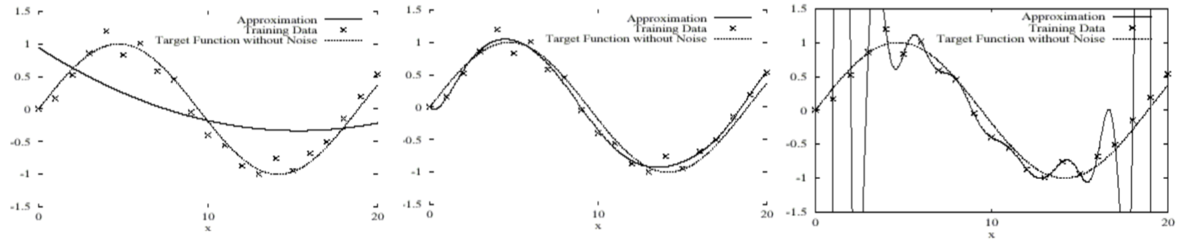


Loss surface of a DNN model

Bias-Variance Trade-off

- Kinds of prediction error
  - Bias error: generated due to erroneous hypothesis on a model
    - high bias = underfitting
  - Variance error: from sensitivity to small fluctuations on different input data
    - high variance = overfitting
  - Irreducible error: noise in data





high bias
(underfitting)

high variance
(overfitting)

Bias–variance decomposition of mean squared error

Target output

Prediction result

$$\mathrm{MSE} = E\left[\left(y-\hat{f}\right)^2\right]$$

zero-mean noise in the target output $\sim N(0,\sigma)$

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

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

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

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

$$= \underbrace{E\left[\left(f-E[\hat{f}]\right)^2\right]}_{\text{Bias}^2}+\underbrace{E[\epsilon^2]}_{\text{Irreducible Error}}+\underbrace{E\left[\left(E[\hat{f}]-\hat{f}\right)^2\right]}_{\text{Variance}}+0+0+0$$

zero-mean

$E[XY] = E[X]E[Y]$
Mean of deviation = 0

Bias^2: Error between true result and expected result from model

Irreducible Error: Variance of the random noise, $E[(\epsilon-0)^2] = \sigma^2$

Variance: Fluctuation (variance) of the prediction results

https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

Statistical Interpretation of Linear Regression

- ***Dose least-squares loss function find actual optimal parameters of a linear regression model?***

MLE for weight parameters of a linear regression model with gaussian noise equals Minimizing MSE of the model

$$y_i = \Theta \mathbb{x}_i + \boxed{\epsilon_i} \xrightarrow{\text{Gaussian noise}} p(\epsilon_i) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{\epsilon_i^2}{2\sigma^2}\right)$$

$$p(y_i|\mathbb{x}_i;\boxed{\theta}) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i-\Theta\mathbb{x}_i)^2}{2\sigma^2}\right)$$

MLE for weight parameter Θ

$$L(\Theta) = L(\Theta; X, Y) = p(Y|X;\Theta)$$

$$L(\Theta) = \boxed{\prod_{i=1}^{n}} p(y_i|\mathbb{x}_i;\Theta) = \prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i-\Theta\mathbb{x}_i)^2}{2\sigma^2}\right)$$

IID assumptions

Log likelihood

$$\boxed{\log L(\Theta)} = \log\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i-\Theta\mathbb{x}_i)^2}{2\sigma^2}\right) = \sum_{i=1}^{n}\log\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i-\Theta\mathbb{x}_i)^2}{2\sigma^2}\right)$$

Maximizing log likelihood

$$= n\log\frac{1}{\sqrt{2\pi}\sigma}\boxed{-\frac{1}{\sigma^2}\frac{1}{2}\sum_{i=1}^{n}(y_i-\Theta^T\mathbb{x}_i)^2}$$

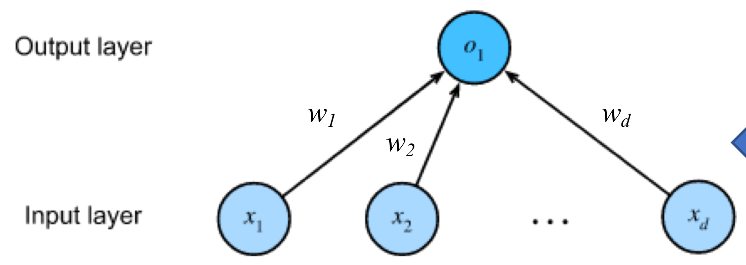
Minimizing Squared error

σ is some fixed constant

Interpretation of LR model in a perspective on Deep Neural Network

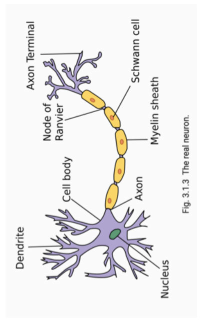
Linear regression model is A kind of fully-connected layer or dense layer

- Every input is connected to every output



Layer representation of a liner regression model

$o_1 = w_1x_1 + w_2x_2 + \ldots + w_dx_d$



axon

nucleus $y = \sum_i x_i w_i + b,$

dendrites

Linear Neural Networks

Further material for following chapters

https://github.com/howawindelu/dive-into-deep-learning/blob/master/week3/week3_1_implementation_lukeshin.ipynb

- 3.1.2. Vectorization for Speed
- 3.2. Linear Regression Implementation from Scratch
- 3.3 Concise Implementation of Linear Regression