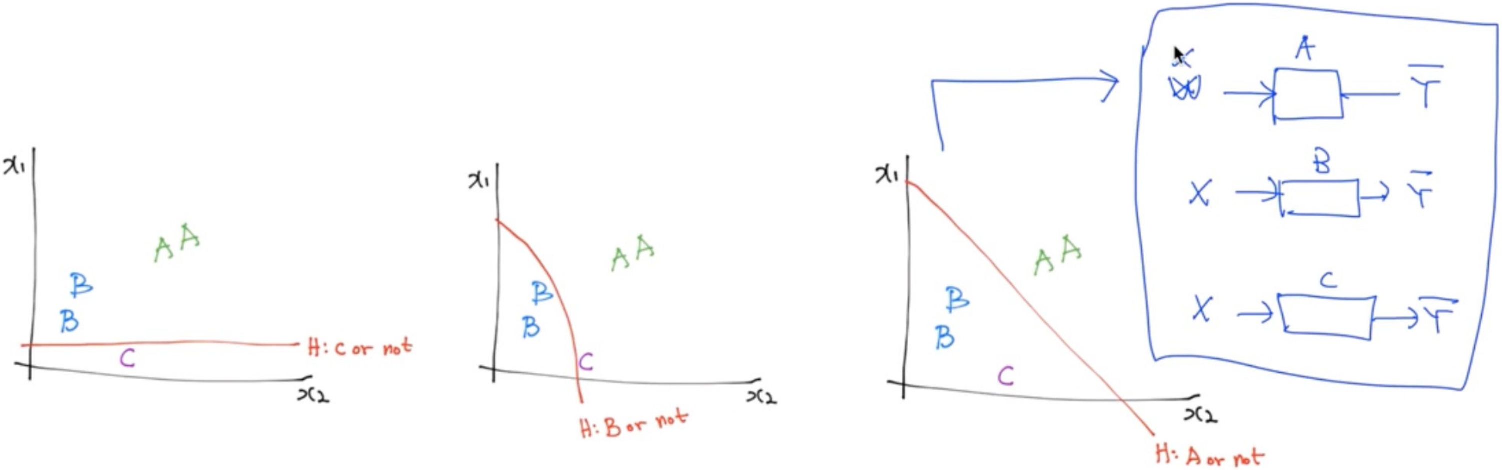
2주차 머신러닝 발표

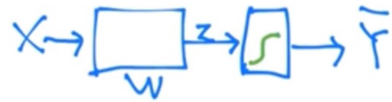
김찬원

Multinomial classification

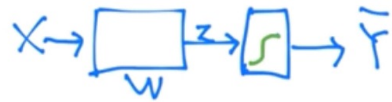


Multinomial classification

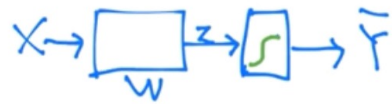
$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 \end{bmatrix}$$



$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 \end{bmatrix}$$

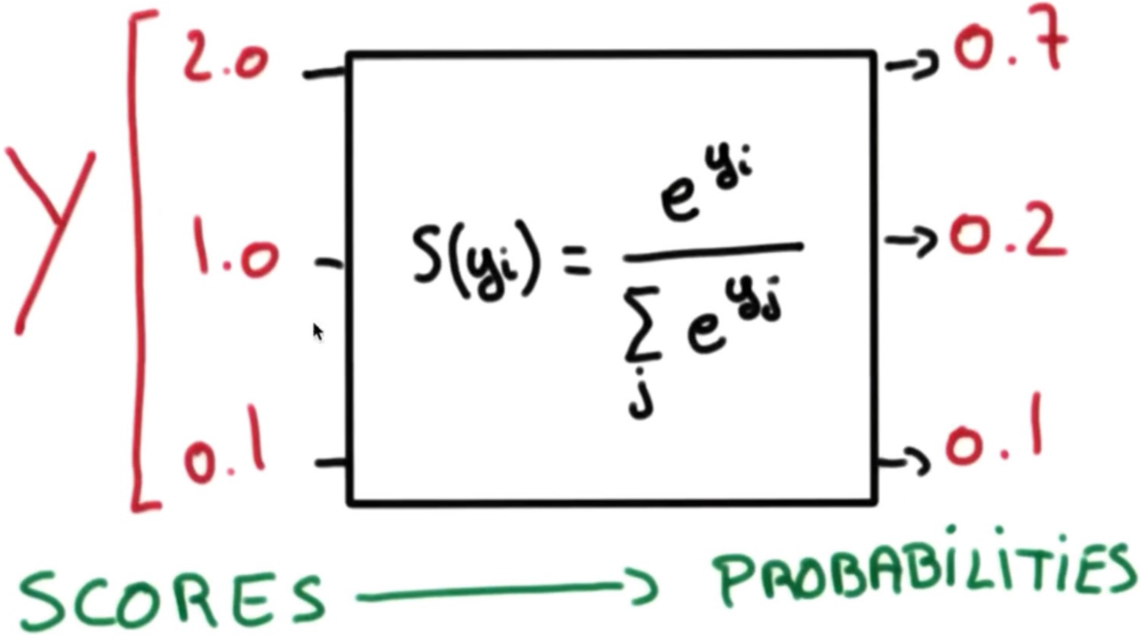


$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 \end{bmatrix}$$



$$\begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ w_{B1} & w_{B2} & w_{B3} \\ w_{C1} & w_{C2} & w_{C3} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{A1}x_1 + w_{A2}x_2 + w_{A3}x_3 \\ w_{B1}x_1 + w_{B2}x_2 + w_{B3}x_3 \\ w_{C1}x_1 + w_{C2}x_2 + w_{C3}x_3 \end{bmatrix} = \begin{bmatrix} \bar{y}_A \\ \bar{y}_B \\ \bar{y}_C \end{bmatrix} \quad \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix}$$

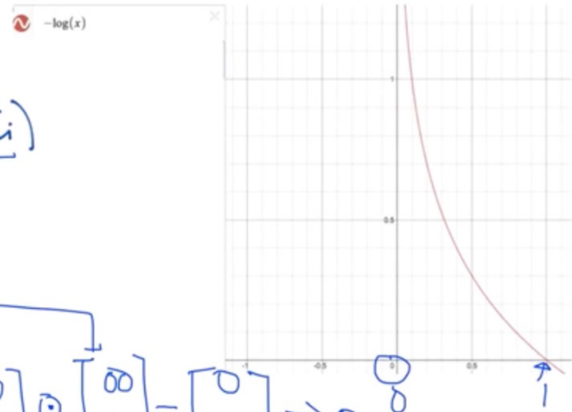
SOFTMAX



Cross-entropy cost function

$D(S, L)$

$$= -\sum_i L_i \log(S_i) \qquad -\sum_i L_i \log(\bar{y}_i) = \sum_i \underline{L_i} * \underline{-\log(\bar{y}_i)}$$



$$\underline{Y} = L = \begin{array}{c} A \\ B \end{array}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \underline{B}$$

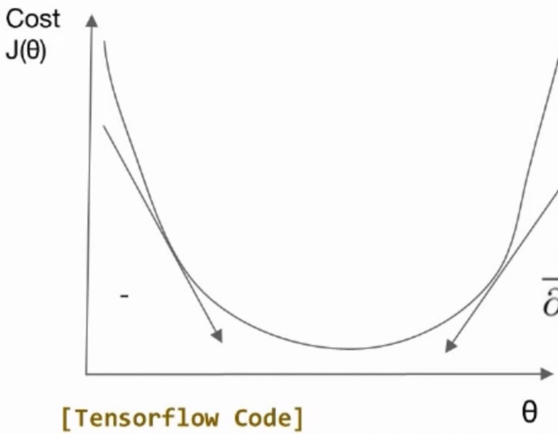
$$\bar{Y} = \begin{array}{c} A \\ B \end{array}\begin{bmatrix} 0 \\ 1 \end{bmatrix} B \;(\text{ok}), \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} \odot -\log \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \odot \begin{bmatrix} \infty \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow 0$$

$$\bar{Y} = \begin{array}{c} A \\ B \end{array}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = A \;(X), \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} \odot -\log \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \odot \begin{bmatrix} 0 \\ \infty \end{bmatrix}$$

총 Loss : $\mathcal{L} = \frac{1}{n} \sum_i D(S(Wx_i, b), L_i)$

Learning rate

Gradient



$$\frac{\partial}{\partial\theta_j}$$

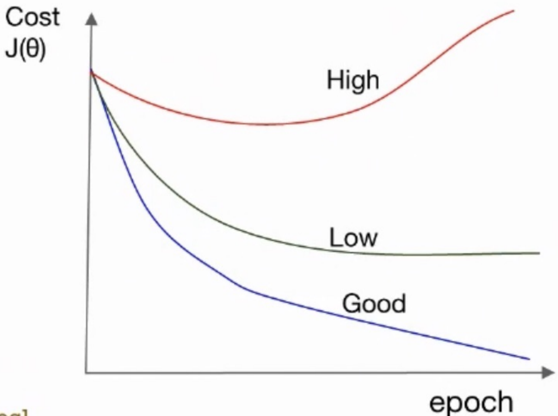
Repeat $\{ \theta_j := \theta_j - \alpha \frac{\partial}{\partial\theta_j} J(\theta) \}$

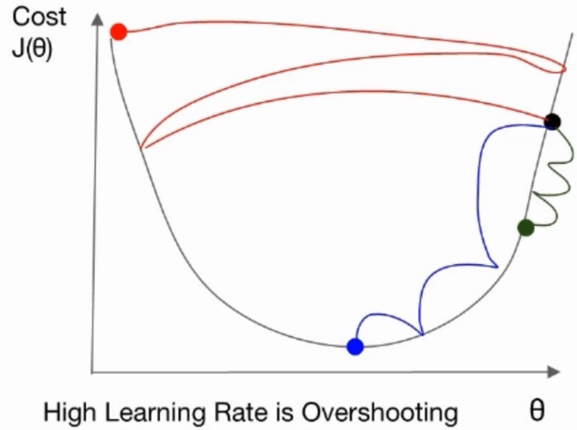
Learning rate is a hyper-parameter that controls how much we are adjusting the weights with respect the loss gradient

```
[Tensorflow Code]
def grad(hypothesis, labels):
   with tf.GradientTape() as tape:
       loss_value = loss_fn(hypothesis, labels)
   return tape.gradient(loss_value, [W,b])
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
optimizer.apply_gradients(grads_and_vars=zip(grads,[W,b]))
```

Learning rate

Good and Bad





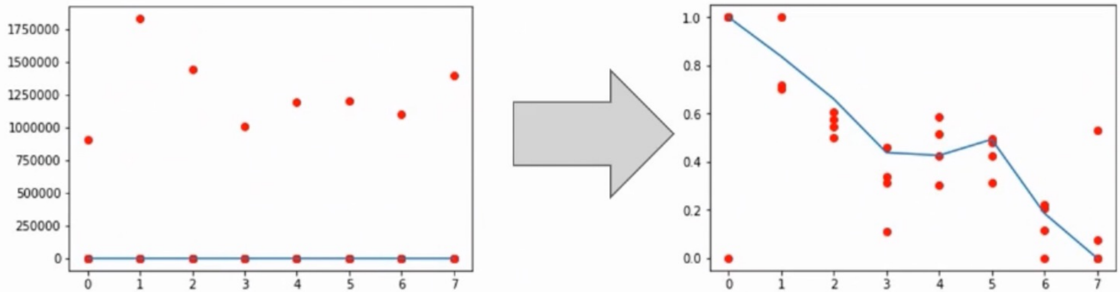
[Train Log]
Iter: 0, Loss: 6.0257, Learning Rate: 0.1000
Iter: 1000, Loss: 0.3723, Learning Rate: 0.0960
Iter: 2000, Loss: 0.2779, Learning Rate: 0.0922
Iter: 3000, Loss: 0.2293, Learning Rate: 0.0885
Iter: 4000, Loss: 0.1977, Learning Rate: 0.0849
Iter: 5000, Loss: 0.1750, Learning Rate: 0.0815

High Learning Rate is Overshooting
Normal Learning Rate is `0.01`
3e-4 is the best learning rate for Adam, hands down (andrej karpathy)

[Tensorflow Code]
```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
```

Data preprocessing

Feature Scaling



Standardization
(Mean Distance)

$$x_{new} = \frac{x - \mu}{\sigma}$$

[Python Code(numpy)]

```
Standardization = (data - np.mean(data)) / sqrt(np.sum(
(data - np.mean(data))^2 ) / np.count(data))
```

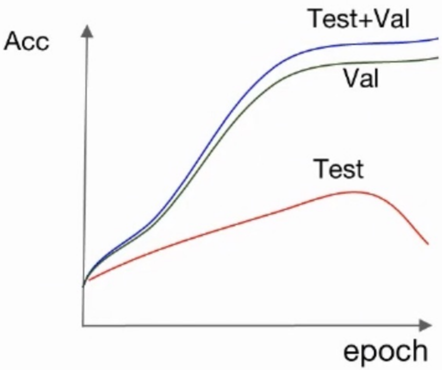
Normalization
(0~1)

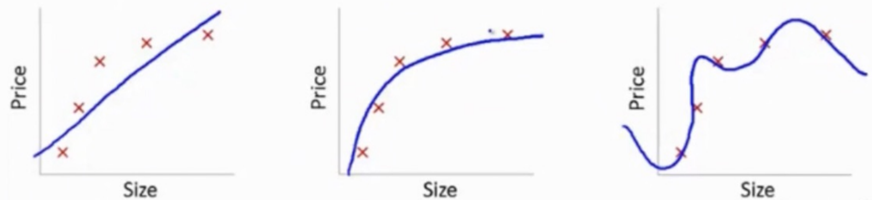
$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

```
Normalization = (data - np.min(data, 0)) / (np.max(data, 0)
- np.min(data, 0))
```

https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-07-3-linear_regression_min_max.ipynb

Overfitting





$\theta_0 + \theta_1 x$ | $\theta_0 + \theta_1 x + \theta_2 x^2$ | $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

High bias (underfit) | "Just right" | High variance (overfit)

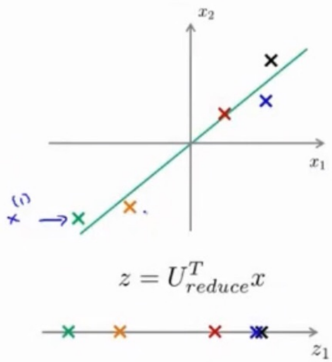
FaceScrub dataset(Aaron Eckhart)

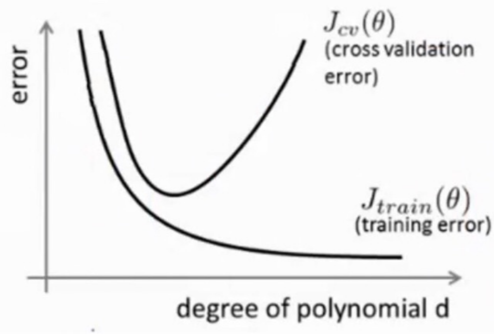
http://www.holehouse.org/mlclass/10_Advice_for_applying_machine_learning.html

Overfitting

Set a features

- Get more training data - more data will actually make a difference, (helps to fix high variance)
- Smaller set of features - dimensionality reduction(PCA) (fixes high variance)
- Add additional features - hypothesis is too simple, make hypothesis more specific (fixes high bias)





1. $h_\theta(x) = \theta_0 + \theta_1 x$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$

$\vdots$

10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10}$

[sklearn Code]

```
from sklearn.decomposition import PCA
pca = decomposition.PCA(n_components=3)
pca.fit(X)
X = pca.transform(X)
```

Overfitting

Regularization (Add term to loss)

Large λ
High bias (underfit)
$\lambda = 10000.\ \theta_1 \approx 0, \theta_2 \approx 0, \ldots$
$h_\theta(x) \approx \theta_0$

Intermediate λ
"Just right"

Small λ
High variance (overfit)
$\lambda \approx 0$

λ-- : fixes high bias (Under fitting)
λ ++ : fixes high variance (overfitting)

Linear regression with regularization

Model: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

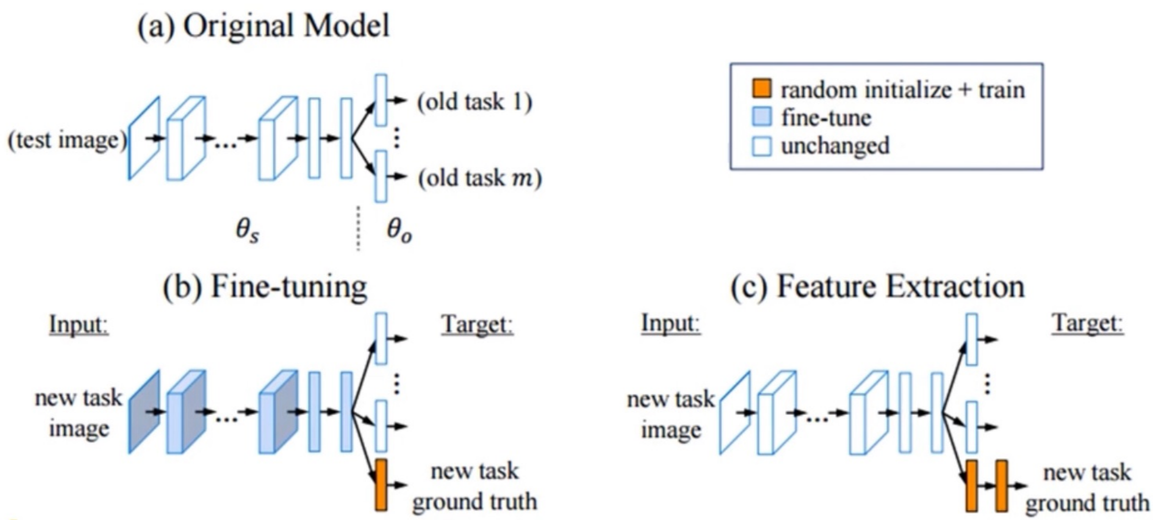
$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

[Tensorflow Code]

```
L2_loss = tf.nn.l2_loss(w) # output = sum(t ** 2) / 2
```

Learning

Fine Tuning / Feature Extraction



[Tensorflow Code]

```
saver = tf.train.import_meta_graph('my-model-1000.meta')
saver.restore(tf.train.latest_checkpoint('./'))
```

Learning without Forgetting : https://arxiv.org/pdf/1606.09282.pdf
Fine tuning : https://goodtogreate.tistory.com/entry/Saving-and-Restoring