ML_2

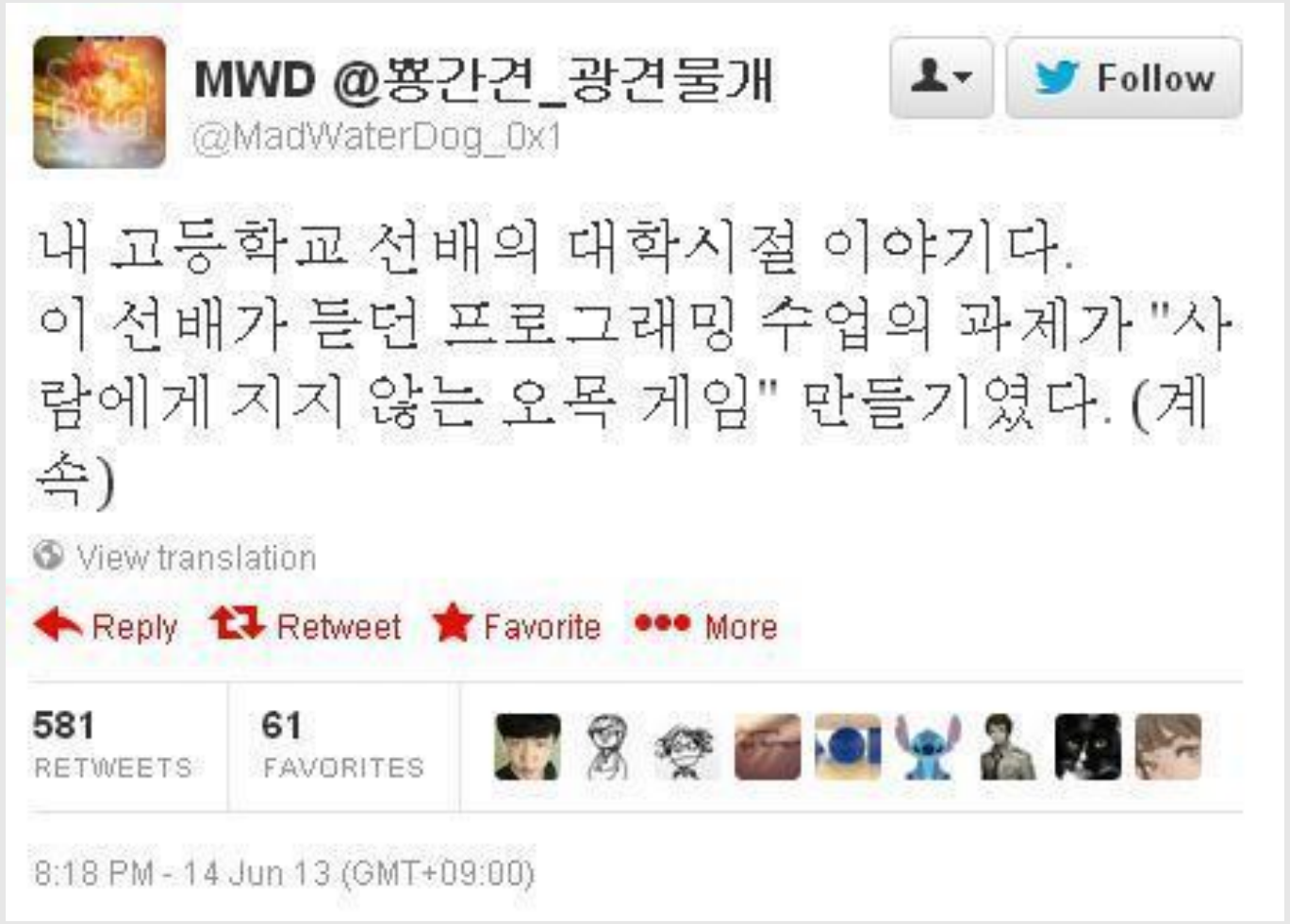
Team BMS

INDEX

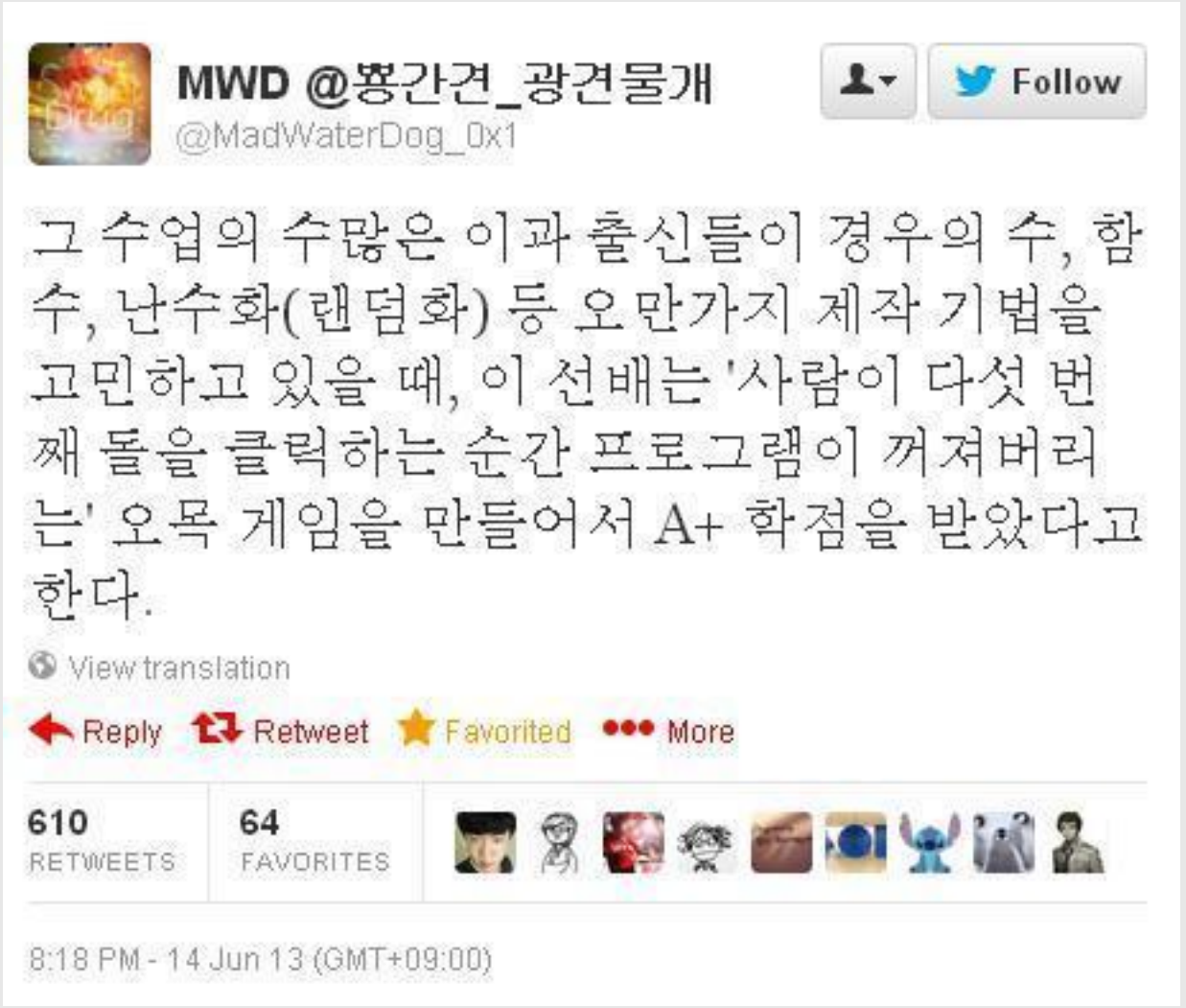
ML Study
2 Week

Ice Breaking

Ice Breaking

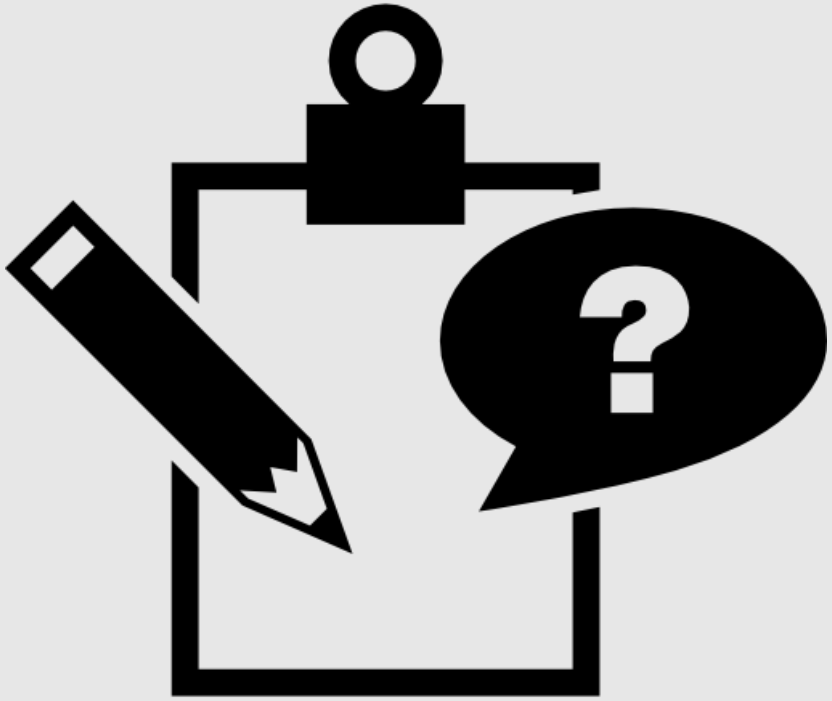


Ice Breaking



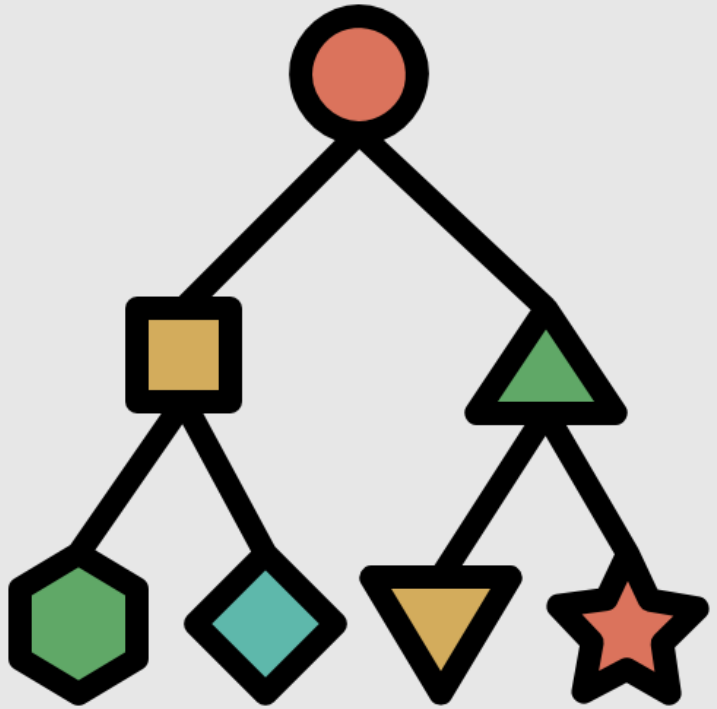
01 I. Ice Breaking

Before Start



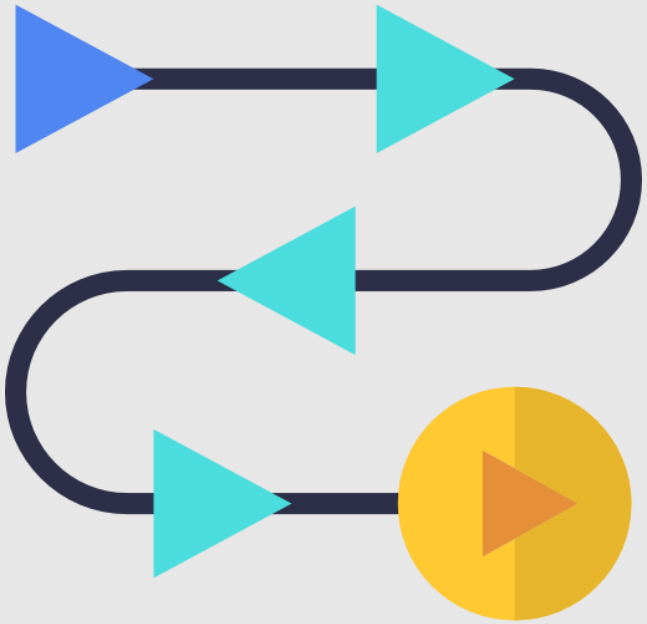
Classification

Classification



Classification이란?

Train & Test Split Tip

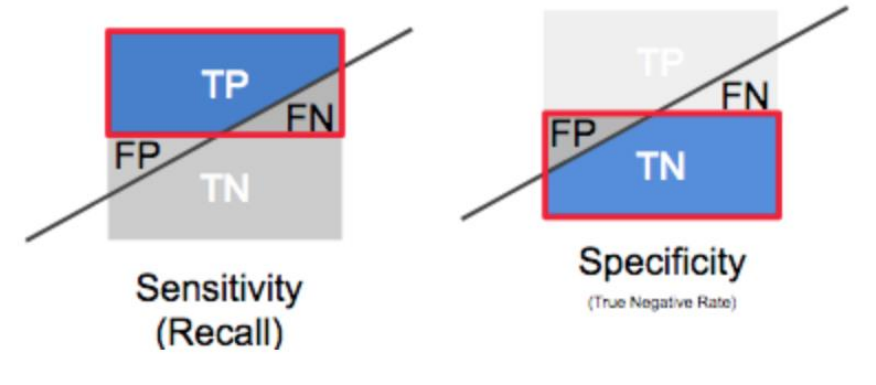


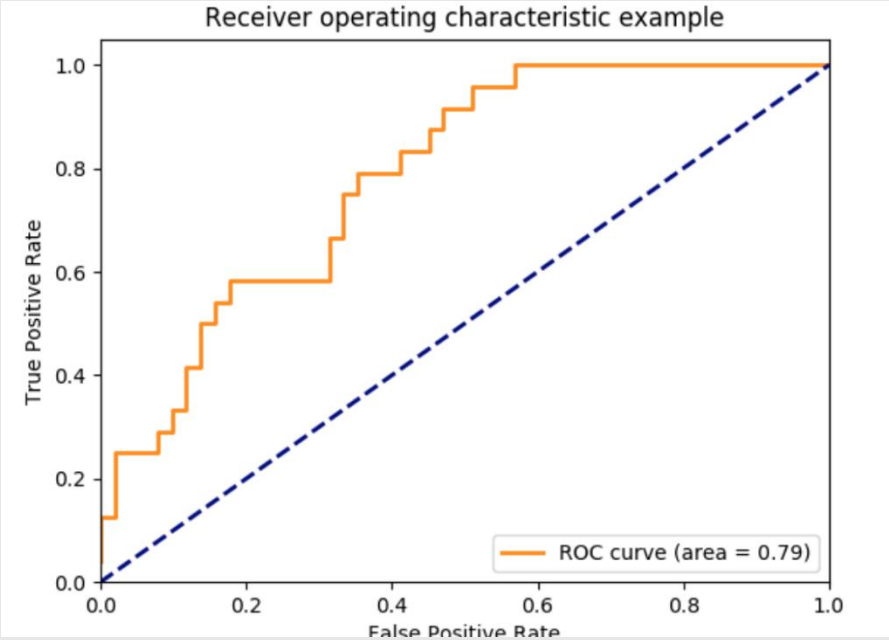
자료형에 따른 Train Test분할 방법을 다르게 할 필요가 있음

Threshold

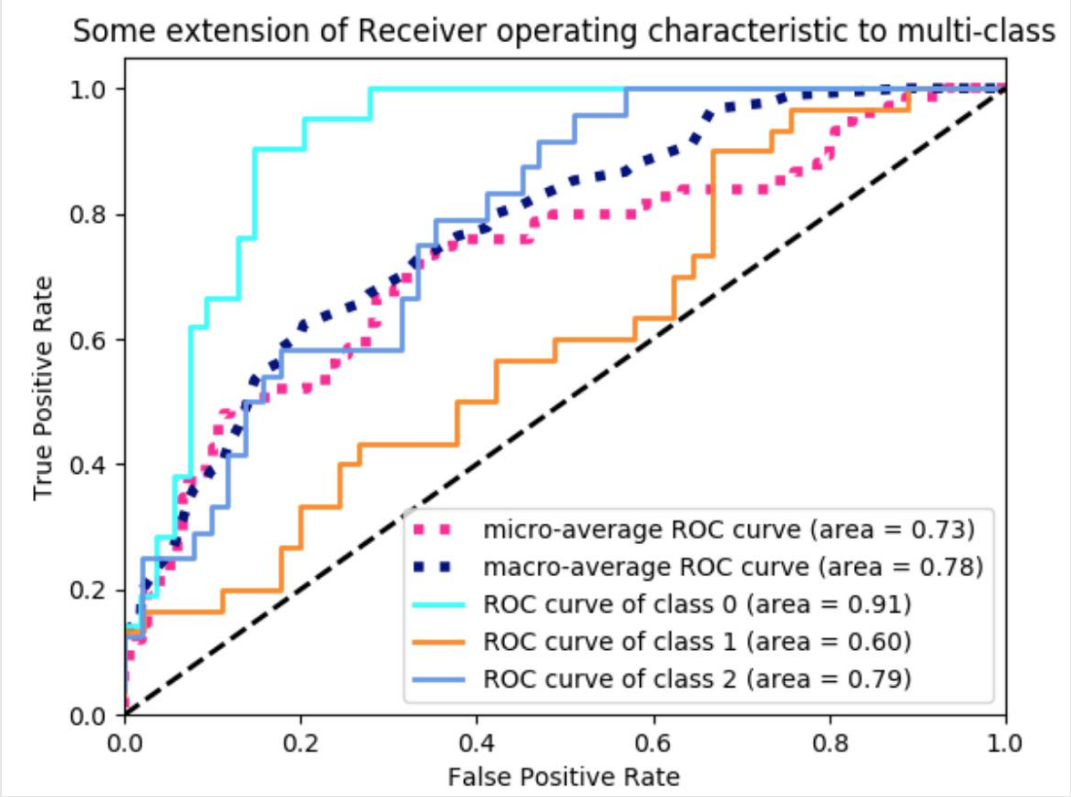
Threshold란?

ROC Curve


TP
FN
FP
TN
Sensitivity
(Recall)
TP
FN
FP
TN
Specificity
(True Negative Rate)


Receiver operating characteristic example
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.79)

ROC Curve


Some extension of Receiver operating characteristic to multi-class
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 0.73)
macro-average ROC curve (area = 0.78)
ROC curve of class 0 (area = 0.91)
ROC curve of class 1 (area = 0.60)
ROC curve of class 2 (area = 0.79)

Multiclass Classifier

OvA (One-Versus-All)

➔ 여러 Target에 대하여
하나의 모델이 필요

➔ Ex) Random Forest, Naïve Bayes

OvO (One-Versus-One)

➔ N개의 Target에 대하여
N×(N-1)/2개의 모델이 필요

➔ Ex) SVM, Logistic Regression

Train Error First!

Train Error

: 과소적합 확인 및
데이터를 다시 전처리

Validation Error

: 과적합 확인 및
하이퍼 파라미터 튜닝

Test Error

: 모형의 최종
성능 확인

Deep Learning Example

```
[59]  cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits = y_conv, labels = y_))
      train_step = tf.train.AdamOptimizer(0.0001).minimize(cost)

      correct_prediction = tf.equal(tf.argmax(y_conv, 1), tf.argmax(y_, 1))
      accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
      pred = y_conv

      sess = tf.Session()
      sess.run(tf.global_variables_initializer())
      before_time = time.time()

      for i in range(501):
          randix = random.sample(range(len(trainimg_loaded_2)), 1000)
          batch_xs = trainimg_loaded_2[randix]
          batch_ys = trainlabel_loaded_2[randix]

          if i % 100 == 0:
              train_accuracy = sess.run(accuracy, feed_dict ={x : batch_xs, y_ : batch_ys, keep_prob : 1.0})
              print("step {}, training accuracy {}".format(i, train_accuracy))
              spend_min = round((time.time() - before_time)/ 60)
              print("{}min spend".format(spend_min))
              before_time = time.time()

          sess.run(train_step, feed_dict = {x : batch_xs, y_ : batch_ys, keep_prob : 0.7})

      test_accuracy = np.mean([sess.run(accuracy,
                                        feed_dict = {x : testimg_loaded,
                                                     y_ : testlabel_loaded, keep_prob: 1.0}) for i in range(10)])

      print("test accuracy: {}".format(test_accuracy))
```

```
step 0, training accuracy 0.20399999618530273
0min spend
step 100, training accuracy 0.5550000071525574
0min spend
step 200, training accuracy 0.7879999876022339
0min spend
step 300, training accuracy 0.9070000052452087
0min spend
step 400, training accuracy 0.9639999866485596
0min spend
step 500, training accuracy 0.9860000014305115
0min spend
test accuracy: 0.7404580116271973
```

Deep Learning Example

```
[89]  from sklearn.metrics import confusion_matrix

      cf = confusion_matrix(y_true = real, y_pred = pred)
      print(cf)
      print("Accuray is {}".format(round((1 - (len(false_index) / len(pred))) * 100, 2)))
```

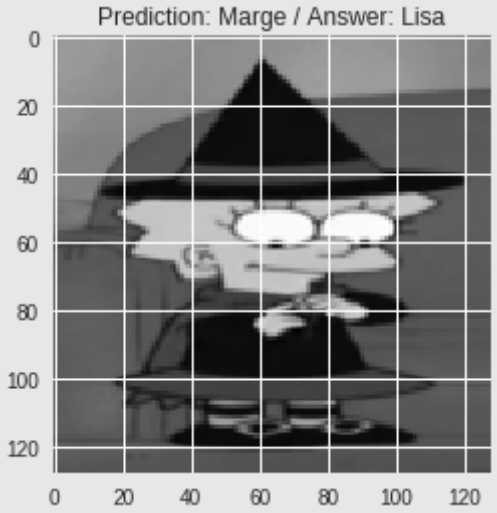
```
[[31  1  4  6]
 [ 7 24  1  4]
 [ 2  3 12  0]
 [ 4  2  0 30]]
Accuray is 74.05
```

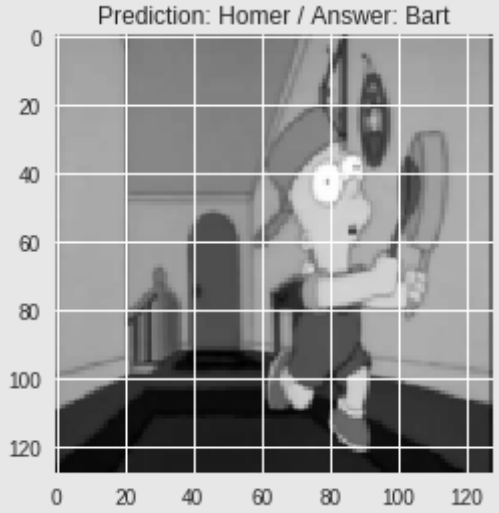
```
[81]  from sklearn.metrics import *

      print(classification_report(real, pred, target_names = categories_loaded))
```

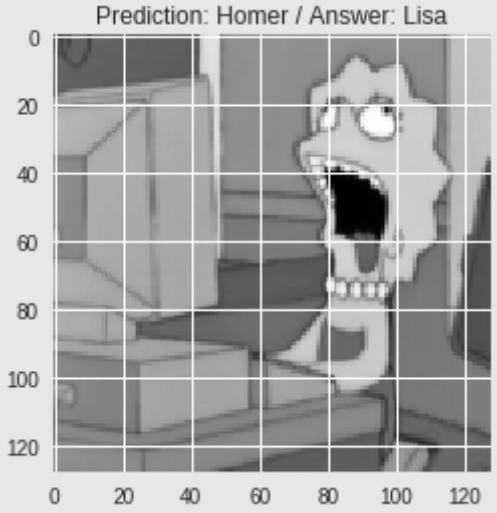
```
              precision    recall  f1-score   support

        Bart       0.70      0.74      0.72        42
       Homer       0.80      0.67      0.73        36
       Marge       0.71      0.71      0.71        17
        Lisa       0.75      0.83      0.79        36

avg / total       0.74      0.74      0.74       131
```

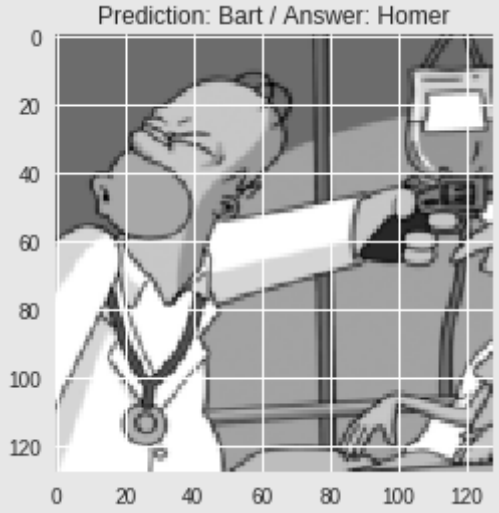
Deep Learning Example

```
import random

figure_count = 5
randidx = random.sample(false_index, figure_count)
for i in randidx:
  currimg = np.reshape(testimg_loaded[i], (64, 64))
  plt.imshow(currimg, cmap = "gray")
  pred_a = categories_loaded[pred[i]]
  real_a = categories_loaded[real[i]]
  plt.title("Prediction: {} / Answer: {}".format(pred_a, real_a))
  plt.show()
```



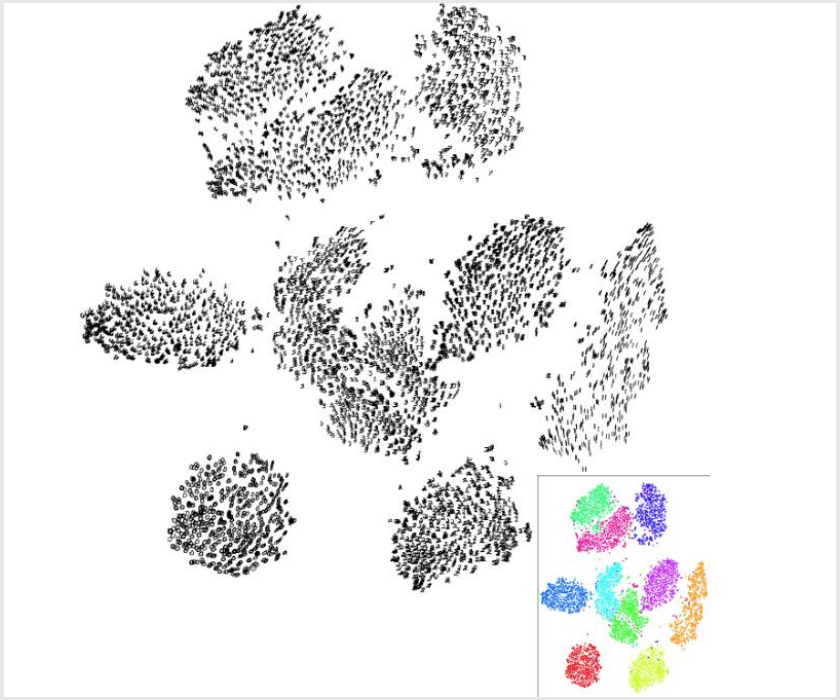




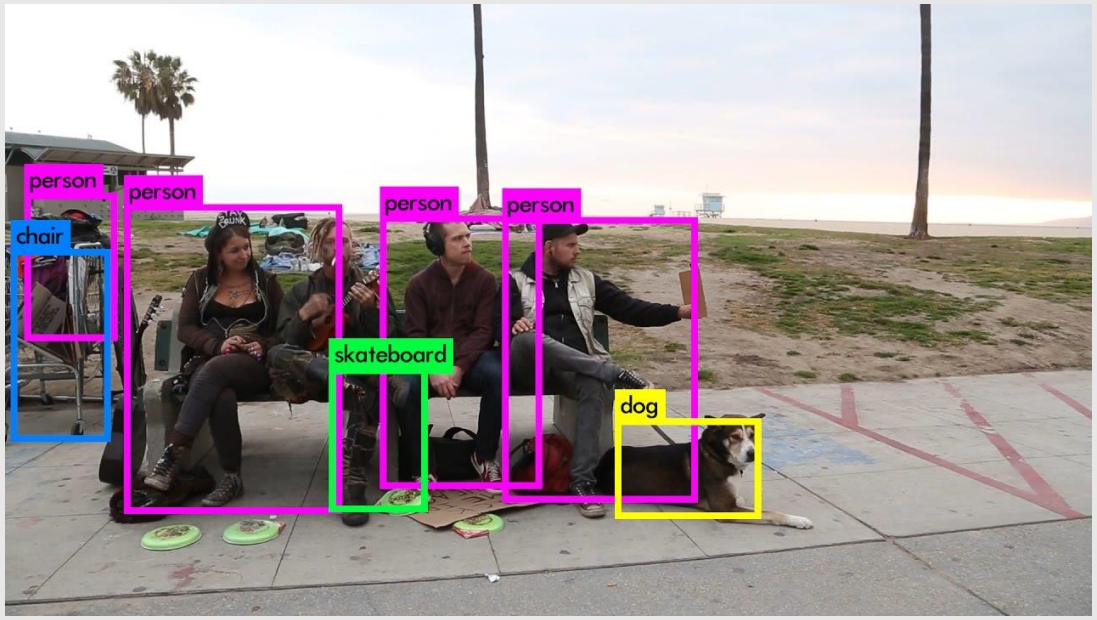


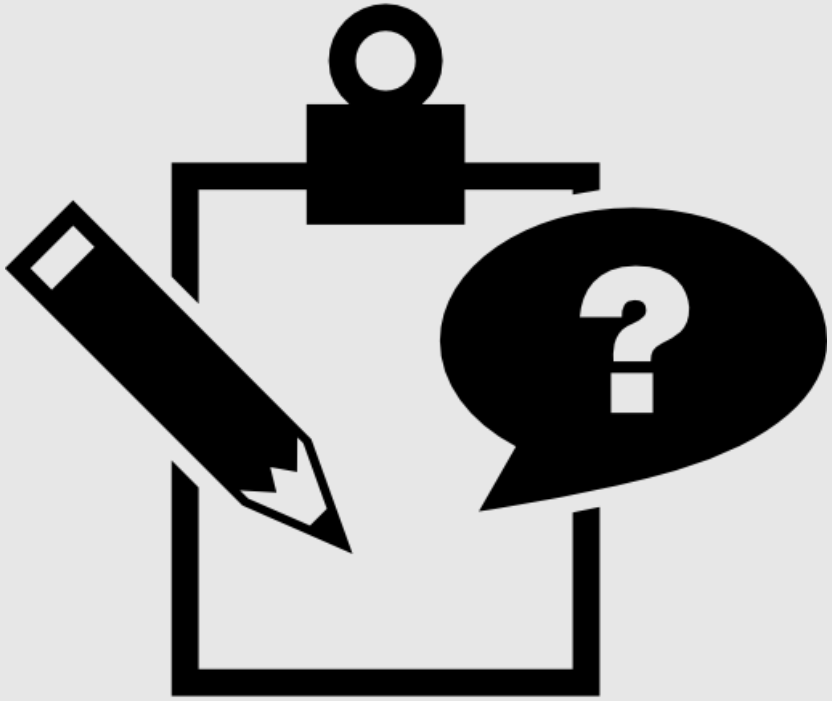
Multi-class & Multi-label

Multi-class



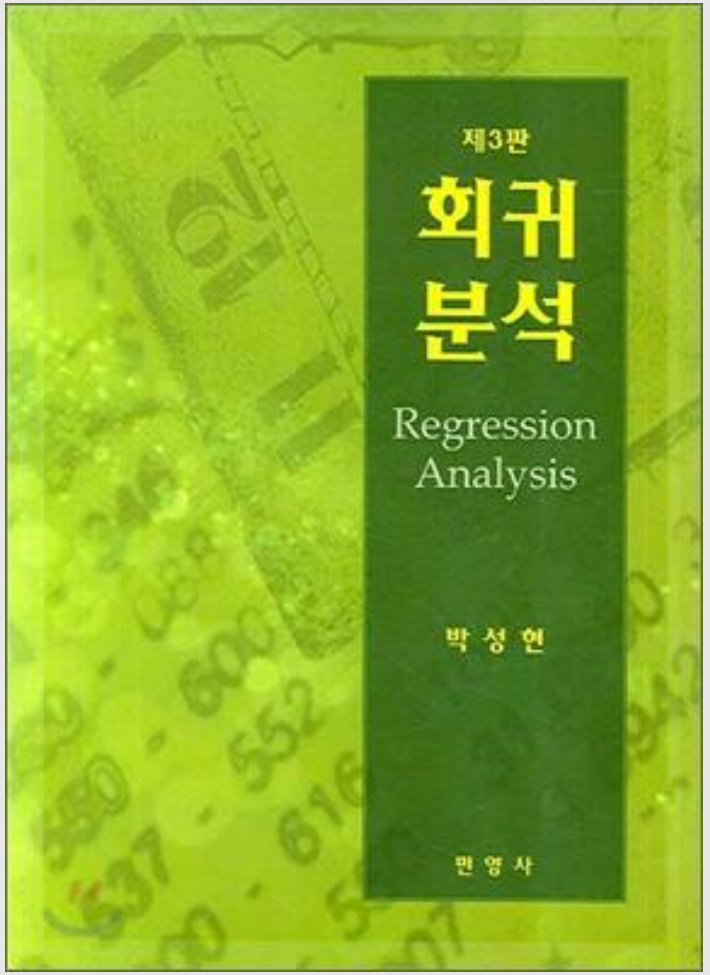
Multi-label
(Multioutput-Multiclass)





Training Models

Regression Analysis Class



Regression Analysis

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

➔$\hat{y} = h_\theta(x) = \theta^T \cdot x$

➔RMSE를 최소화하는 θ를 찾아야 함

➔$\mathrm{MSE}(X, h_\theta) = MSE(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(\theta^T \cdot x^{(i)} - y^{(i)}\right)^2$

➔ θ를 찾는 해석적인 방법: 정규방정식(Normal Equation)

➔Normal Equation: $\hat{\theta} = (X^T \cdot X)^{-1} \cdot X^T \cdot y$

Normal Equation & Gradient Descent

경사 하강법과 마찬가지로 예측알고리즘에 해당한다.
경사 하강법이 수학적 최적화 알고리즘으로서 적절한 학습비율(learning rate)를 설정해야 하고 많은 연산량이 필요하지만 정규방정식에는 그와 같은 단점이 없다는 장점이 있다. 하지만 정규방정식은 행렬 연산에 기반하기 때문에 피쳐의 개수가 엄청나게 많을 경우 연산이 느려지는 것을 피할 수 없다.
하지만 경사 하강법은 아무리 많은 피쳐가 존재하더라도 일정한 시간 내에 해법을 찾는 것이 가능하다. 그러므로 예측 알고리즘을 선택할 때 있어 피쳐의 개수에 따라 알맞은 것을 선택하여야 한다.

〈https://ko.wikipedia.org/wiki/%EC%A0%95%EA%B7%9C%EB%B0%A9%EC%A0%95%EC%8B%9D〉

Complexity Theory

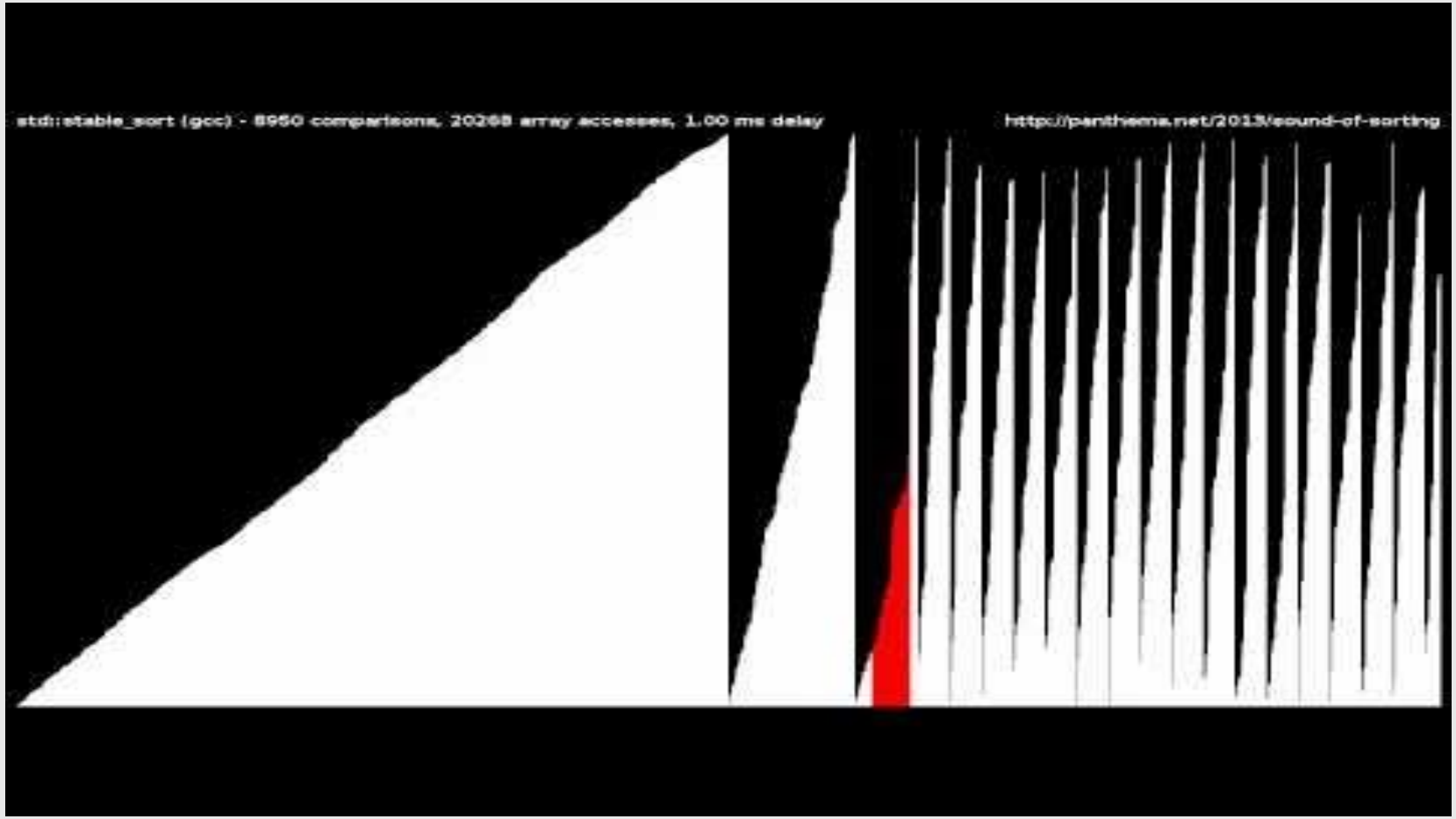
계산 복잡도 이론(Computational complexity theory)

컴퓨터 과학에서 계산 이론의 분야로, 계산 문제를 푸는 알고리즘을 복잡도에 따라 분류하여 문제의 모임을 구성하는 방법을 연구한다. 이 때 알고리즘의 수행은 실제 컴퓨터가 할 수 있지만, 평가하는 데에는 튜링 기계와 관련이 있는 정량화된 방법을 사용한다.

➔ 대다수 대문자 O로 표기한다. (ex: $\mathcal{O}(n^2)$)

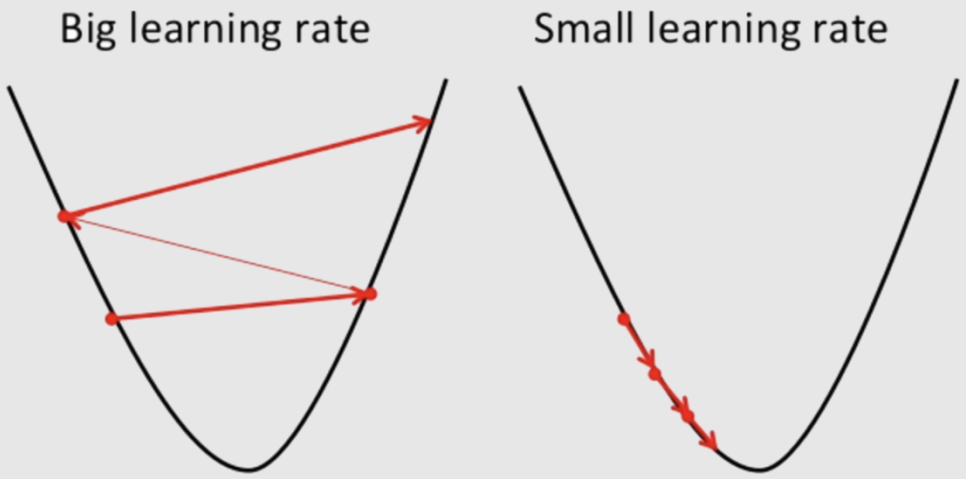
〈https://ko.wikipedia.org/wiki/%EA%B3%84%EC%82%B0_%EB%B3%B5%EC%9E%A1%EB%8F%84_%EC%9D%B4%EB%A1%A0〉

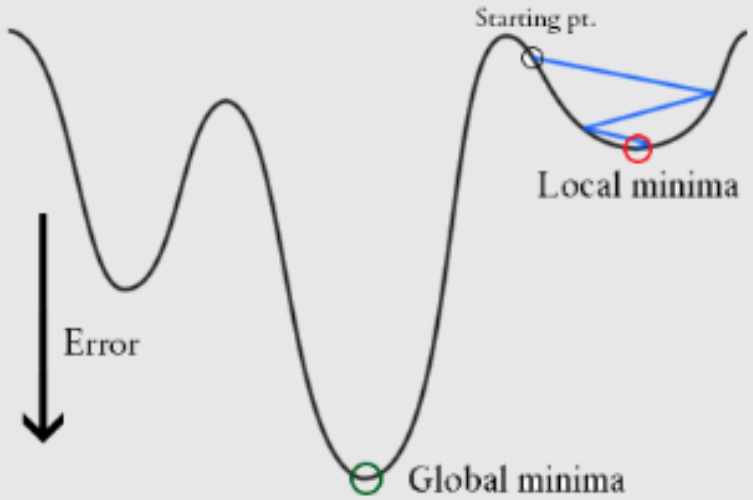
Complexity Theory



〈https://www.youtube.com/watch?time_continue=25&v=kPRA0W1kECg〉

Gradient Descent's Learning Rate





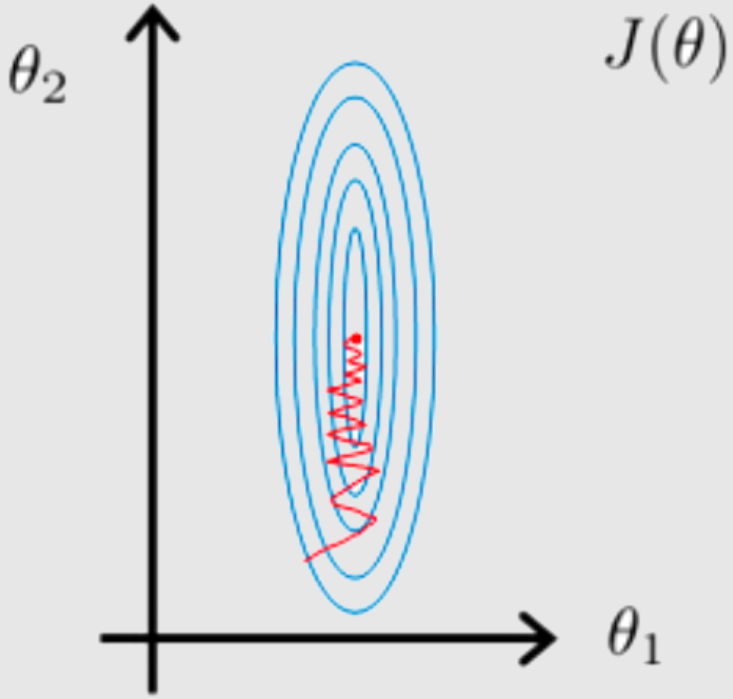
Gradient Descent Scaling

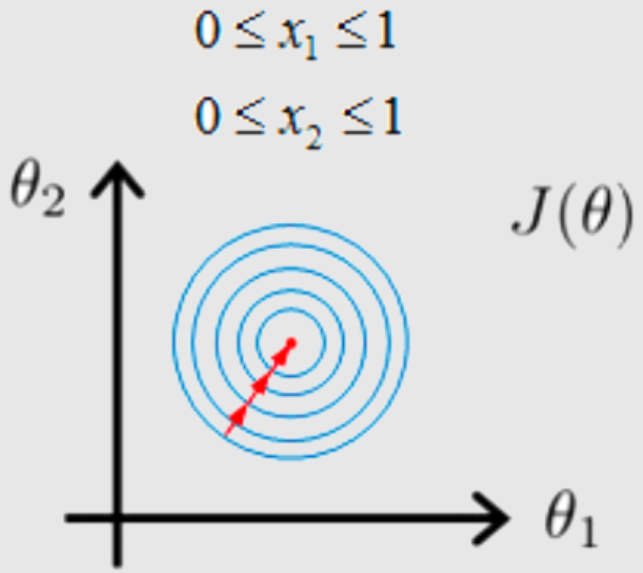
Essentially, scaling the inputs (through mean normalization, or z-score) gives the error surface a more spherical shape, where it would otherwise be a very high curvature ellipse. Since gradient descent is curvature-ignorant, having an error surface with high curvature will mean that we take many steps which aren't necessarily in the optimal direction. When we scale the inputs, we reduce the curvature, which makes methods that ignore curvature (like gradient descent) work much better. When the error surface is circular (spherical), the gradient points right at the minimum, so learning is easy.

(입력을 스케일링 (평균 정규화 또는 z-스코어를 통해)하면 오류 표면이 보다 더 구형 원이 되는 모양을 갖게 됩니다. 그래디언트 디센트는 경사하강 방향을 모르는 것이므로 높은 곡률을 갖는 오류 표면을 갖는 것은 우리가 반드시 최적의 방향으로 있지는 않은 많은 단계를 취한다는 것을 의미합니다. **입력의 크기를 조정할 때 곡률을 줄여** 곡률을 무시하는 방법(예: 그래디언트 디센트)을 훨씬 잘 수행합니다. 오차 표면이 원형 (구형) 일 때, 그래디언트는 최소로 오른쪽을 가리키므로 학습이 더욱 쉽습니다.)

〈https://www.quora.com/Why-does-mean-normalization-help-in-gradient-descent〉

Gradient Descent Scaling





Batch Gradient Descent

$$\frac{\partial}{\partial\theta_j}MSE(\theta) = \frac{2}{m}\sum_{i-1}^{m}\left(\theta^T \cdot x^{(i)} - y^{(i)}\right)x_j^{(i)}$$

$$\nabla_\theta MSE(\theta) = \begin{pmatrix} \frac{\partial}{\partial\theta_0}MSE(\theta) \\ \frac{\partial}{\partial\theta_1}MSE(\theta) \\ \vdots \\ \frac{\partial}{\partial\theta_n}MSE(\theta) \end{pmatrix} = \frac{2}{m}X^T \cdot (X \cdot \theta - y)$$

- 매 스텝마다 훈련 데이터 전체를 사용한 위의 식 계산 ➔ 매우 많은 연산
- 하지만 특성 수에 영향을 받지 않으므로 때에 따라서 정규방정식보다 빠르기도 하다.

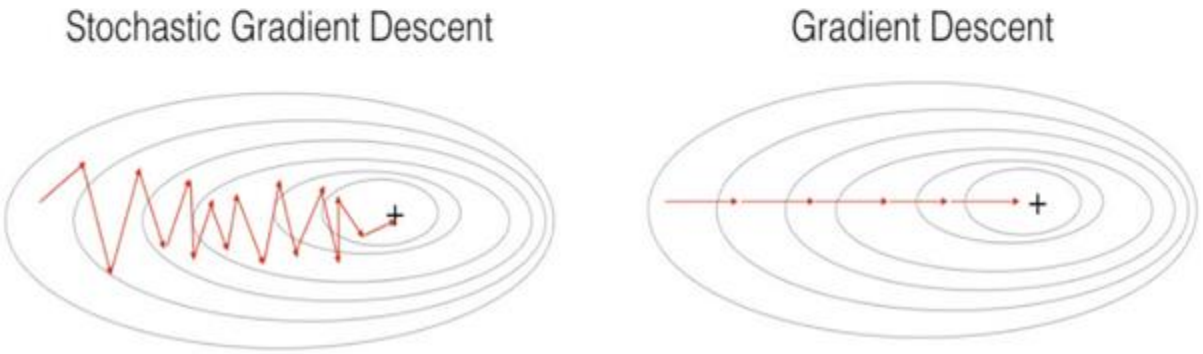
Stochastic Gradient Descent

SGD (Stochastic Gradient Descent)이란?

배치 크기가 1인 경사하강법 알고리즘 입니다.

즉, 확률적 경사하강법은 데이터 세트에서 무작위로 균일하게 선택한 하나의 예를 의존하여 각 단계의 예측 경사를 계산합니다.

< 최소값을 찾는 과정 >



Stochastic Gradient Descent

Pros

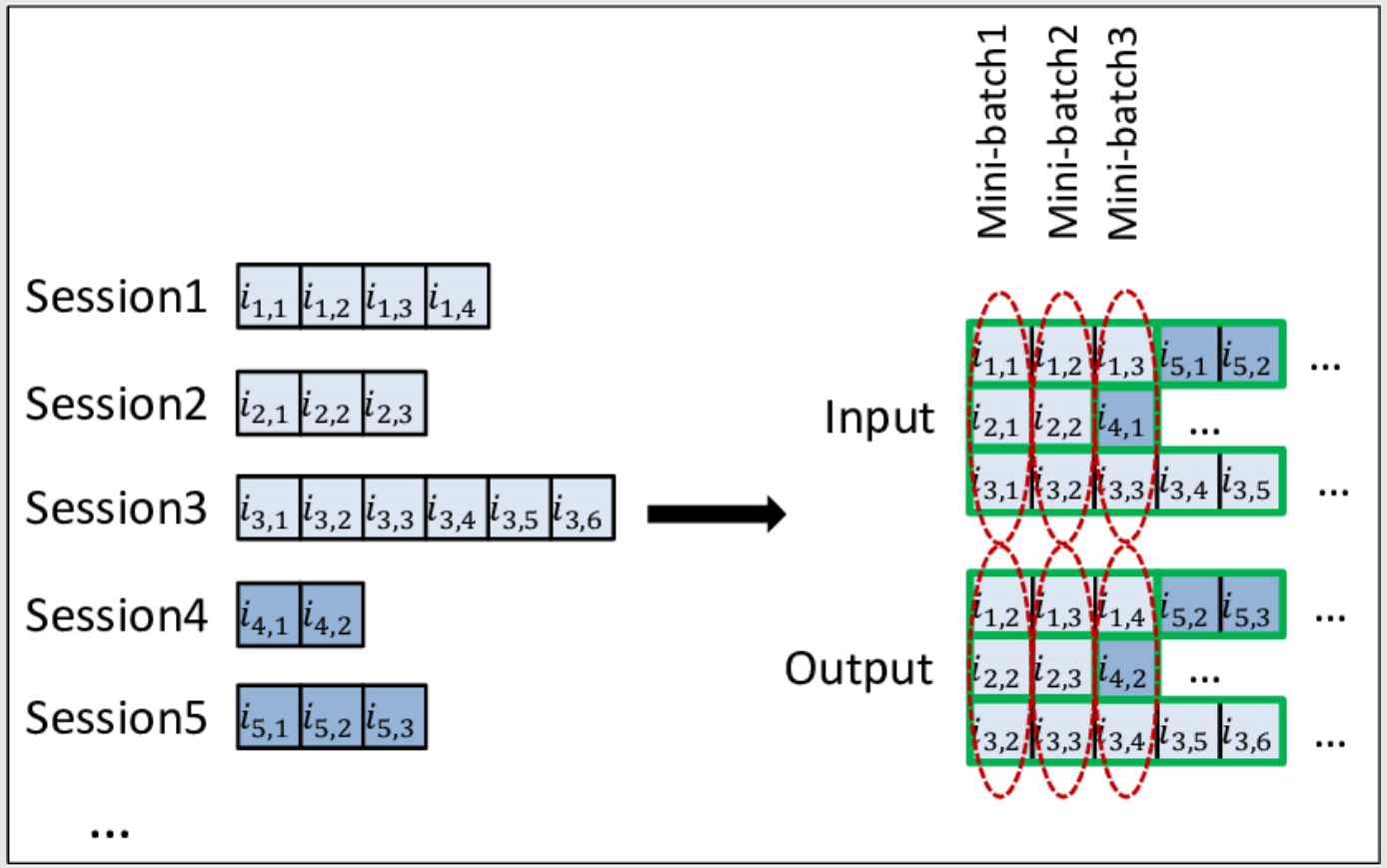
- 적은 메모리 사용
- 무작위성이 추가되어 다양성 증가
- Local Minima 탈출이 용이함

Cons

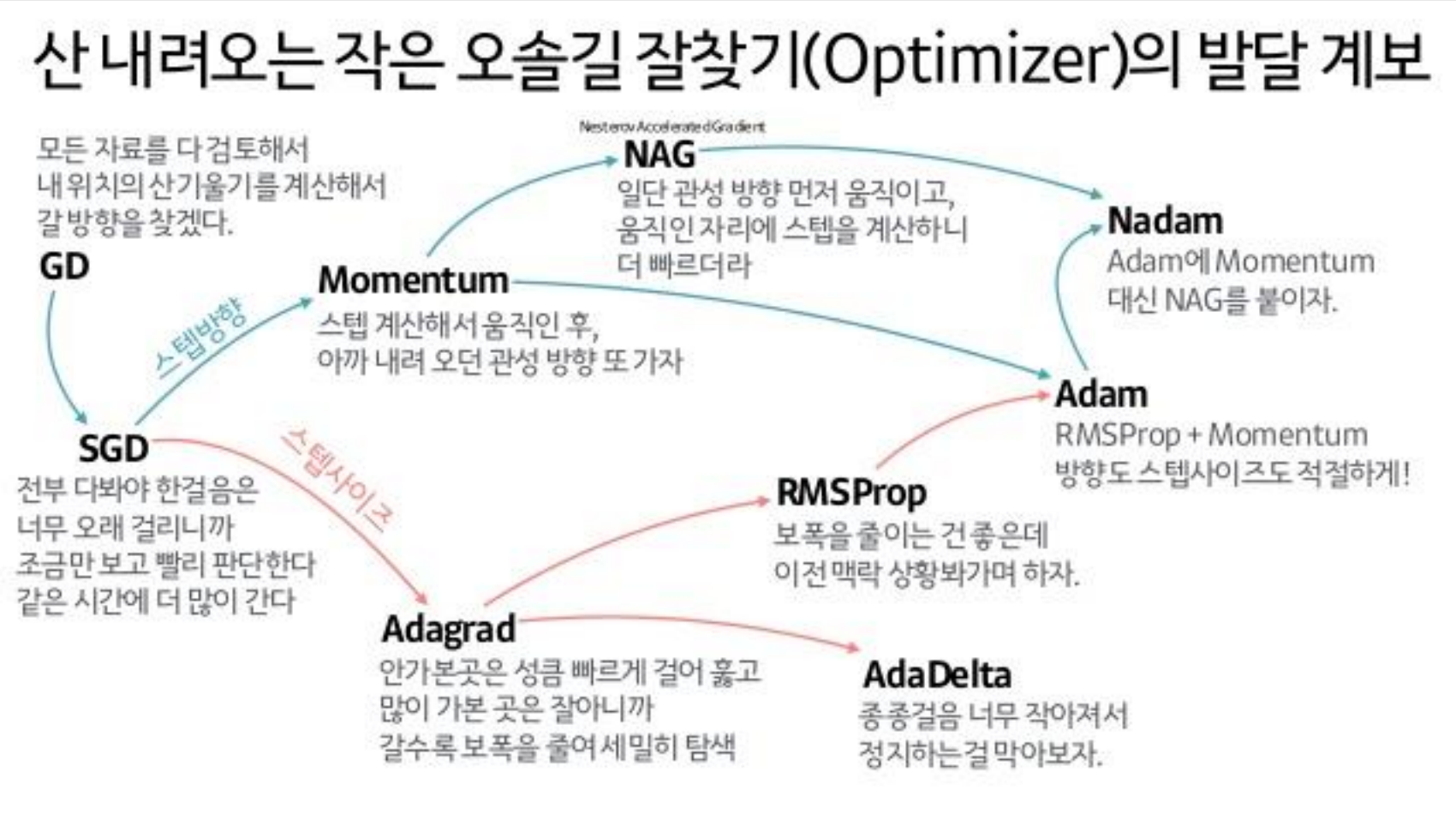
- 노이즈에 취약
- 무작위성이 추가되어 불안정성 증가
- Global Minima의 접근이 힘듬

머신러닝 분야의 Trend 중 하나
➔ 장단점이 있는 알고리즘은 절충안이 꼭 나옴

Mini-batch Gradient Descent



Optimizer

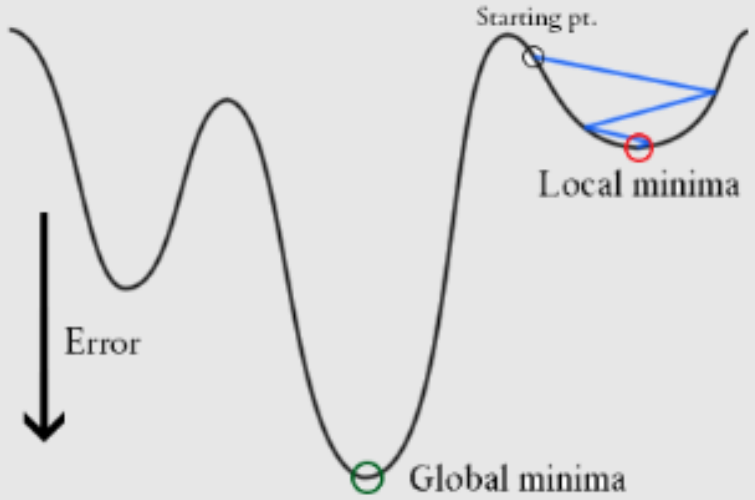


〈https://www.slideshare.net/yongho/ss-79607172〉

Gradient Descent Comparison

Algorithm	Big m	Outer Memory	Big n	Hyper Paremeter	Scaling Requirement
Normal Equation	Fast	No	Slow	0	No
Gradient Descent	Slow	No	Fast	2	Yes
SGD	Fast	Yes	Fast	≥2	Yes
Mini-batch	Fast	Yes	Fast	≥2	Yes

Initial Point Setting



초기값이 중요하지 않을까?

Initial Point Setting

따라서 weight의 초기화를 잘 시키는 것이 중요하다. 우선 모든 weight의 값은 0을 가지게 되면 안 된다. 앞에서 예시를 들었지만 0을 가지게 되면 입력 값의 데이터가 무시될 수 있기 때문이다. 그리고 더욱 좋은 초기화를 시키기는 방법으로 RBM(Restricted Boatman Machine)이 존재한다. weight를 조절하는 방법으로 Forward한 방법과 Backward한 방법을 비교하는 방법이다. 처음 input 값을 통해 결과 값을 만들어내고 결과 값을 반대로 계산해서 input 값을 만들어낸다. 두 개의 input 값을 비교해서 같게 만들어 주는 weight 값을 찾아 학습을 하는 것이다. 이는 두 개의 layer 사이의 weight 값을 비교하게 되어 학습을 하므로 계속 layer를 넘어가면서 학습을 하게 된다. 그러면 이렇게 만들어진 weight 값을 초기 값으로 가지게 하여 Neural Network 모델을 학습하게 되는 것이다.

〈http://copycode.tistory.com/173〉

Initial Point Setting

Xiavier(Glorot), 2010

- ReLu 등장 후에 Glorot이 2010년에 제안한 방법으로 vanshing gradient 문제를 해결하기 위해 만들어짐(9)
- **특별한 분야가 아닐 경우 대부분 Glorot 을 사용함**
- **Input 과 output neuron의 수에 기반해서 초기화의 스케일을 정함**
- Initialization weight 가 0일 경우 각 레이어를 통과하면서 signal이 쪼그러 들어 0이 되고 반대로 weight가 너무 크다면 signal은 너무도 커져버린다는 점에 주목한 방법
- Uniform distribution 에서
  - x = sqrt(σ / (in + out))
- Normal distribution 에서
  - sqrt(2. / (in + out))

He, 2015

- Glorot과 유사하지만 **Neuron의 out size**를 고려하진 않음
- **ReLu가 0 이하의 신호를 제거하기 때문에 분산을 두배 주어 분산을 유지한다는 의도**
- Uniform distribution 에서
  - [-limit, limit] 안에서의 uniform distribution
  - Limit = limit is sqrt(σ / fan_in)
- Normal distribution 에서
  - center가 0 이고 stddev = sqrt(2 / fan_in) 인 정규분포

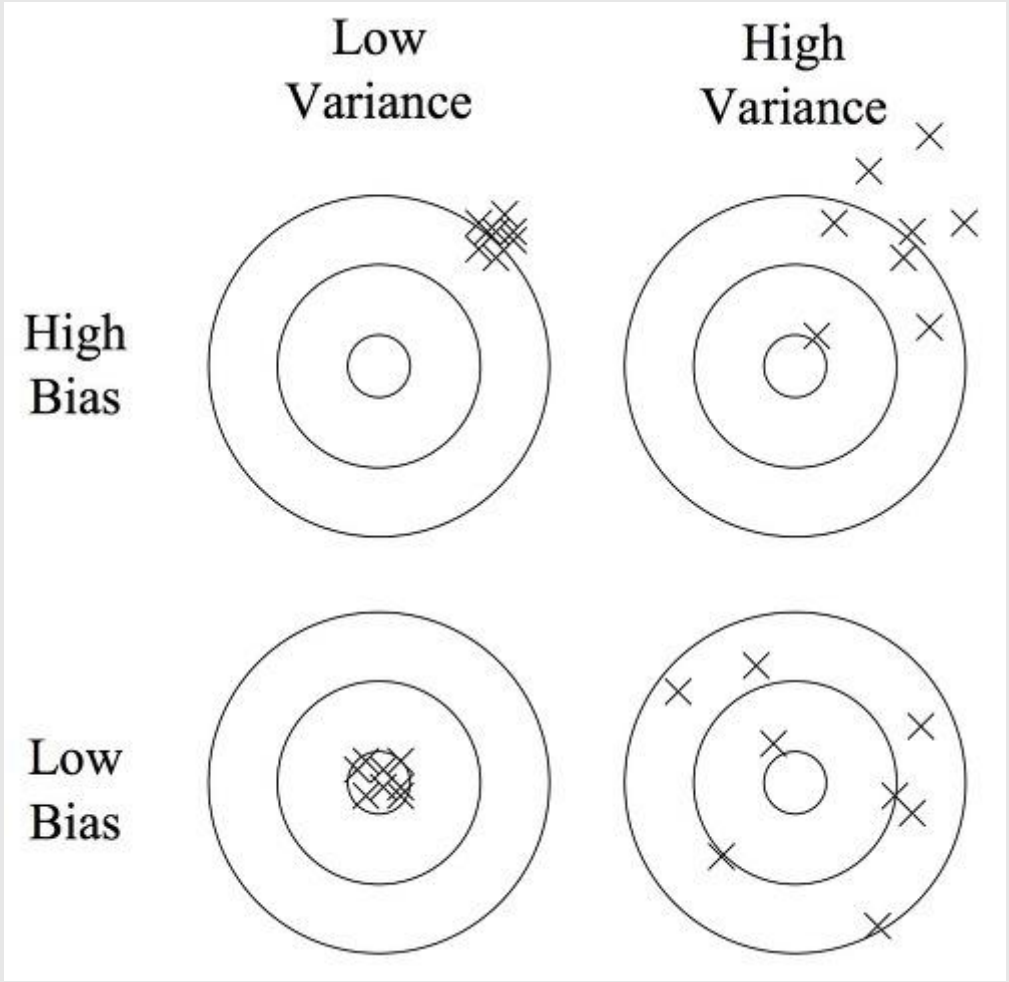
- 논문만 따져보면 사실 이론적으로 둘 다 유사
  (둘 다 초기 파라미터 분포에 대한 좋은 Variance를 찾는 것이 목적)
- 논문에서 Xavier는 Uniform Distribution에,
  He는 Gaussian Distribution에 사용된다 했지만 어디에 사용하여도 무방

〈https://flonelin.wordpress.com/2018/01/28/weight-initalizer-%EC%A2%85%EB%A5%98/〉

Training Models Tips

- 모델이 훈련 데이터에 과소적합되어 있다면 훈련 샘플을 더 추가해도 효과가 없습니다. 더 복잡한 모델을 사용하거나 더 나은 특성을 선택해야 합니다.
- 과대적합 모델을 개선하는 한 가지 방법은 검증오차가 훈련오차에 근접할 때 까지 더 많은 훈련 데이터를 추가하는 것 입니다.

Bias/Variance Trade-off



〈https://www.quora.com/What-is-the-best-way-to-explain-the-bias-variance-trade-off-in-layman%E2%80%99s-terms〉

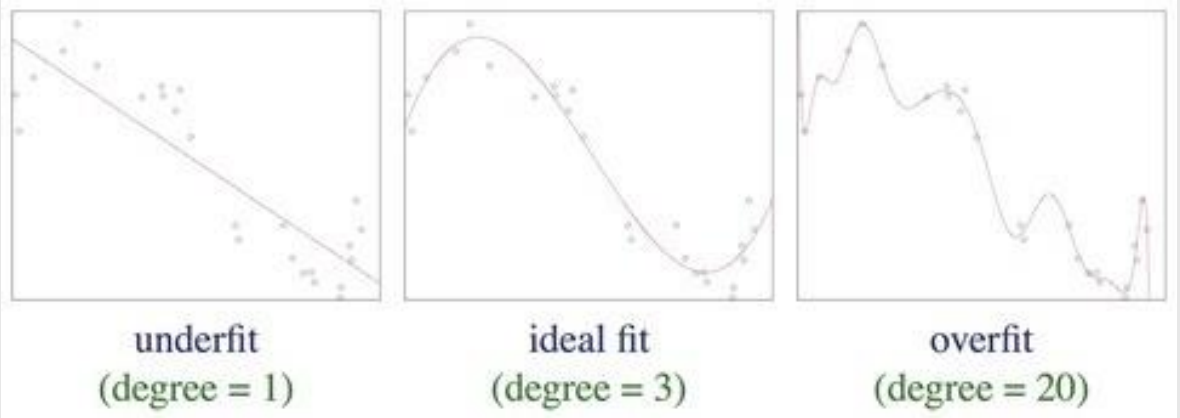
Bias/Variance Trade-off

$$Error(X) = Noise(X) + Bias(X) + Variance(X)$$

➔Noise는 데이터가 가지는 본질적인 한계치이기 때문에 Irreducible Error라고도 불림

➔Bias는 데이터 내의 모든 정보를 고려하지 않음으로 인해 지속적으로 잘못된 것들을 학습하는 경향 (Under-fitting)

➔Variance는 데이터 내에 있는 오류는 잘 잡아내는 Highly Flexible Models에 데이터를 적합 시키면서 실제 현상과 관련 없는 것들까지 학습하는 경향 (Over-fitting)



http://bywords.tistory.com/entry/%EB%B2%88%EC%97%AD-%EC%9C%A0%EC%B9%98%EC%9B%90%EC%83%9D%EB%8F%84-%EC%9D%B4%ED%95%B4%ED%95%A0-%EC%88%98-%EC%9E%88%EB%8A%94-biasvariance-tradeoff

Bias/Variance Trade-off

즉, 이상적인 모델은 데이터의 규칙성을 잘 잡아내어 정확하면서도 다른 데이터가 들어왔을 때도 잘 일반화할 수 있는 모델일 것이다

(degree=3). 하지만, 실제 상황에서는 두 가지를 동시에 만족하는 것은 거의 불가능하다. 따라서 **트레이닝 데이터가 아닌 실제 데이터에서 좋은 성능을 내기 위해 이런 tradeoff는 반드시 생길 수 밖에 없으며 이는 bias-variance trade-off 라고 불린다.**

http://bywords.tistory.com/entry/%EB%B2%88%EC%97%AD-%EC%9C%A0%EC%B9%98%EC%9B%90%EC%83%9D%EB%8F%84-%EC%9D%B4%ED%95%B4%ED%95%A0-%EC%88%98-%EC%9E%88%EB%8A%94-biasvariance-tradeoff

Regularization

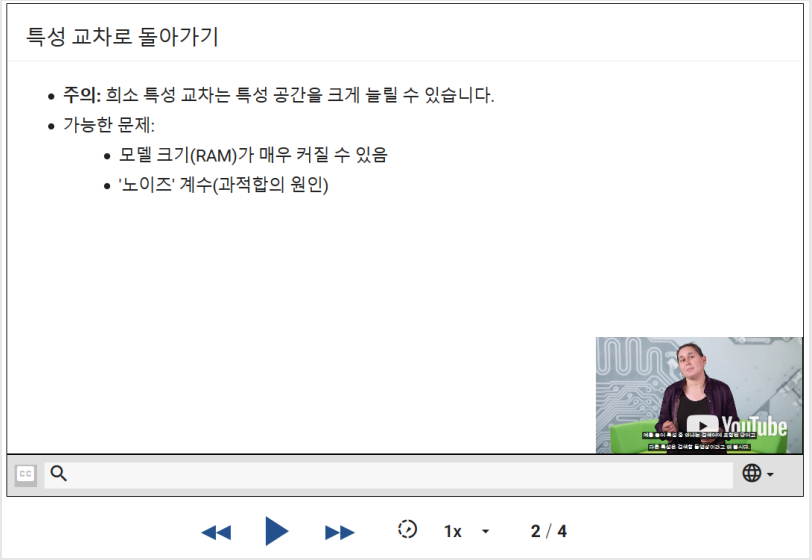
> Everything should be made simple as possible, but not simpler
> – Albert Einstein

우리가 했던 것은 'simpler'였고, 모두가 할 수 있는 것입니다. 하지만 이번 시간에는 'making it simple'에 대해 알아봅시다. 이것이 우리가 정규화를 통해 코드를 최적화하려고 하는 이유입니다.

〈https://brunch.co.kr/@itschloe1/11〉

Regularization

- 과적합이 되었다는 것은 상황에 따라 다르겠으나 대다수의 경우 모형이 과하게 복잡함
- 모형이 복잡하면 RAM사용량이 증가함
- 또한 노이즈에 민감하게 반응하며 다중공선성의 문제도 생길 수 있음
- 이러한 문제 해결을 위해 Regularization Term을 사용



https://developers.google.com/machine-learning/crash-course/regularization-for-sparsity/video-lecture?hl=ko

Ridge & Lasso Regression

Overview

Let's look at the equations. In ordinary least squares, we solve to minimize the following cost function:

$$\text{Cost} = (y - X\beta)^T (y - X\beta)$$

This term is the RSS, residual sum of squares. In ridge regression we instead solve:

$$\text{Cost} = (y - X\beta)^T (y - X\beta) + \lambda \beta^T \beta$$

The $\lambda \beta^T \beta$ term is an L2 norm.

In lasso regression we instead solve:

$$\text{Cost} = (y - X\beta)^T (y - X\beta) + \lambda |\beta|$$

The $\lambda |\beta|$ term is an L1 norm.

〈https://www.kaggle.com/residentmario/l1-norms-versus-l2-norms〉

Norm?

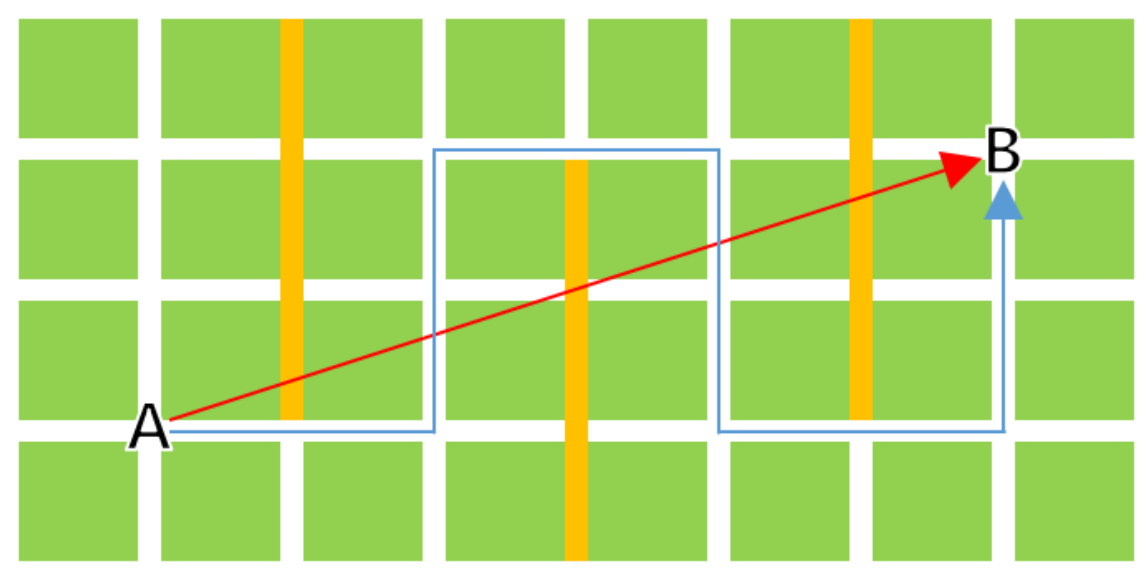
V를 $\mathbb{F}$ 상에서의 벡터공간이라고 하자. $||\cdot|| : V \to \mathbb{F}$ 가 $\mathbf{u}, \mathbf{v} \in V$와 $k \in \mathbb{F}$ 에 대해서 다음 세 조건을 만족시키면 $||\cdot||$ 을 V **상에서의 놈**이라고 정의한다.

(1) 정부호 : $||\mathbf{u}|| \geq 0$ 이고 $\mathbf{u} = 0 \iff ||\mathbf{u}|| = 0$

(2) 동질성 : $||k\mathbf{u}|| = |k|||\mathbf{u}||$

(3) 삼각부등식 : $||\mathbf{u} + \mathbf{v}|| \leq ||\mathbf{v}|| + ||\mathbf{u}||$

〈http://freshrimpsushi.tistory.com/257〉



〈http://freshrimpsushi.tistory.com/257〉

Norm?

Going a bit further, we define $||x||_p$ as a "p-norm". Given x, a vector with i components, a p-norm is defined as:

$$||x||_p = \left(\sum_i |x_i|^p \right)^{1/p}$$

〈https://www.kaggle.com/residentmario/l1-norms-versus-l2-norms〉

L2 Norm ➔ Euclidean Norm

$$||x||_2 = \left(\sum_i x_i^2 \right)^{1/2} = \sqrt{x_1^2 + x_2^2 + \cdots + x_i^2}$$

L1 Norm ➔ Manhattan Norm

$$||x||_1 = \sum_i |x_i| = |x_1| + |x_2| + \cdots + |x_i|$$

L1 Norm vs L2 Norm

Robustness: L1 > L2

Robustness is defined as resistance to outliers in a dataset. The more able a model is to ignore extreme values in the data, the more robust it is.

The L1 norm is more robust than the L2 norm, for fairly obvious reasons: the L2 norm squares values, so it increases the cost of outliers exponentially; the L1 norm only takes the absolute value, so it considers them linearly.

Stability: L2 > L1

Stability is defined as resistance to horizontal adjustments. This is the perpendicular opposite of robustness.

Computational difficulty: L2 > L1

L2 has a closed form solution because it's a square of a thing. L1 does not have a closed form solution because it is a non-differenciable piecewise function, as it involves an absolute value. For this reason, L1 is computationally more expensive, as we can't solve it in terms of matrix math, and most rely on approximations (in the lasso case, coordinate descent).

Sparsity: L1 > L2

Sparsity is the property of having coefficients which are highly significant: very near 0 or very not near 0. In theory, the coefficients very near 0 can later be eliminated.

Feature selection is a further-involved form of sparsity: instead of shrinking coefficients near to 0, feature selection is taking them to exactly 0, and hence excluding certain features from the model entirely. Feature selection is a technique moreso than a property: you can do feature selection as an additional step after running a highly sparse model. But lasso regression is interesting in that it features inbuilt feature selection, while ridge regression is just very sparse.

That about covers the high-level properties of L2 and L1 norms and regularizers. Hopefully you can see how these properties are exactly the same ones exposed in ridge and lasso regression!

Elastic Net

머신러닝 분야의 Trend 중 하나
➔ 장단점이 있는 알고리즘은 절충안이 꼭 나옴

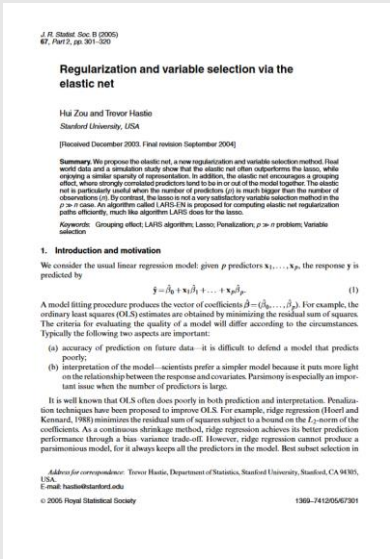
Elastic Net 회귀 모형

Elastic Net 회귀 모형은 가중치의 절대값의 합과 제곱합을 동시에 제약 조건으로 가지는 모형이다.

$$\text{cost} = \sum e_i^2 + \lambda_1 \sum |w_i| + \lambda_2 \sum w_i^2$$

λ_1, λ_2 두 개의 하이퍼 모수를 가진다.

〈https://datascienceschool.net/view-notebook/83d5e4fff7d64cb2aecfd7e42e1ece5e/〉



J. R. Statist. Soc. B (2005)
67, Part 2, pp. 301–320

Regularization and variable selection via the elastic net

Hui Zou and Trevor Hastie
Stanford University, USA

[Received December 2003. Final revision September 2004]

Summary. We propose the elastic net, a new regularization and variable selection method. Real world data and a simulation study show that the elastic net often outperforms the lasso, while enjoying a similar sparsity of representation. In addition, the elastic net encourages a grouping effect, where strongly correlated predictors tend to be in or out of the model together. The elastic net is particularly useful when the number of predictors (p) is much bigger than the number of observations (n). By contrast, the lasso is not a very satisfactory variable selection method in the $p \gg n$ case. An algorithm called LARS-EN is proposed for computing elastic net regularization paths efficiently, much like algorithm LARS does for the lasso.

Keywords: Grouping effect; LARS algorithm; Lasso; Penalization; $p \gg n$ problem; Variable selection

1. Introduction and motivation

We consider the usual linear regression model: given p predictors $\mathbf{x}_1, \ldots, \mathbf{x}_p$, the response $\mathbf{y}$ is predicted by

$$\hat{\mathbf{y}} = \hat{\beta}_0 + \mathbf{x}_1\hat{\beta}_1 + \ldots + \mathbf{x}_p\hat{\beta}_p. \quad (1)$$

A model fitting procedure produces the vector of coefficients $\hat{\beta} = (\hat{\beta}_0, \ldots, \hat{\beta}_p)$. For example, the ordinary least squares (OLS) estimates are obtained by minimizing the residual sum of squares. The criteria for evaluating the quality of a model will differ according to the circumstances. Typically the following two aspects are important:

(a) accuracy of prediction on future data—it is difficult to defend a model that predicts poorly;
(b) interpretation of the model—scientists prefer a simpler model because it puts more light on the relationship between the response and covariates. Parsimony is especially an important issue when the number of predictors is large.

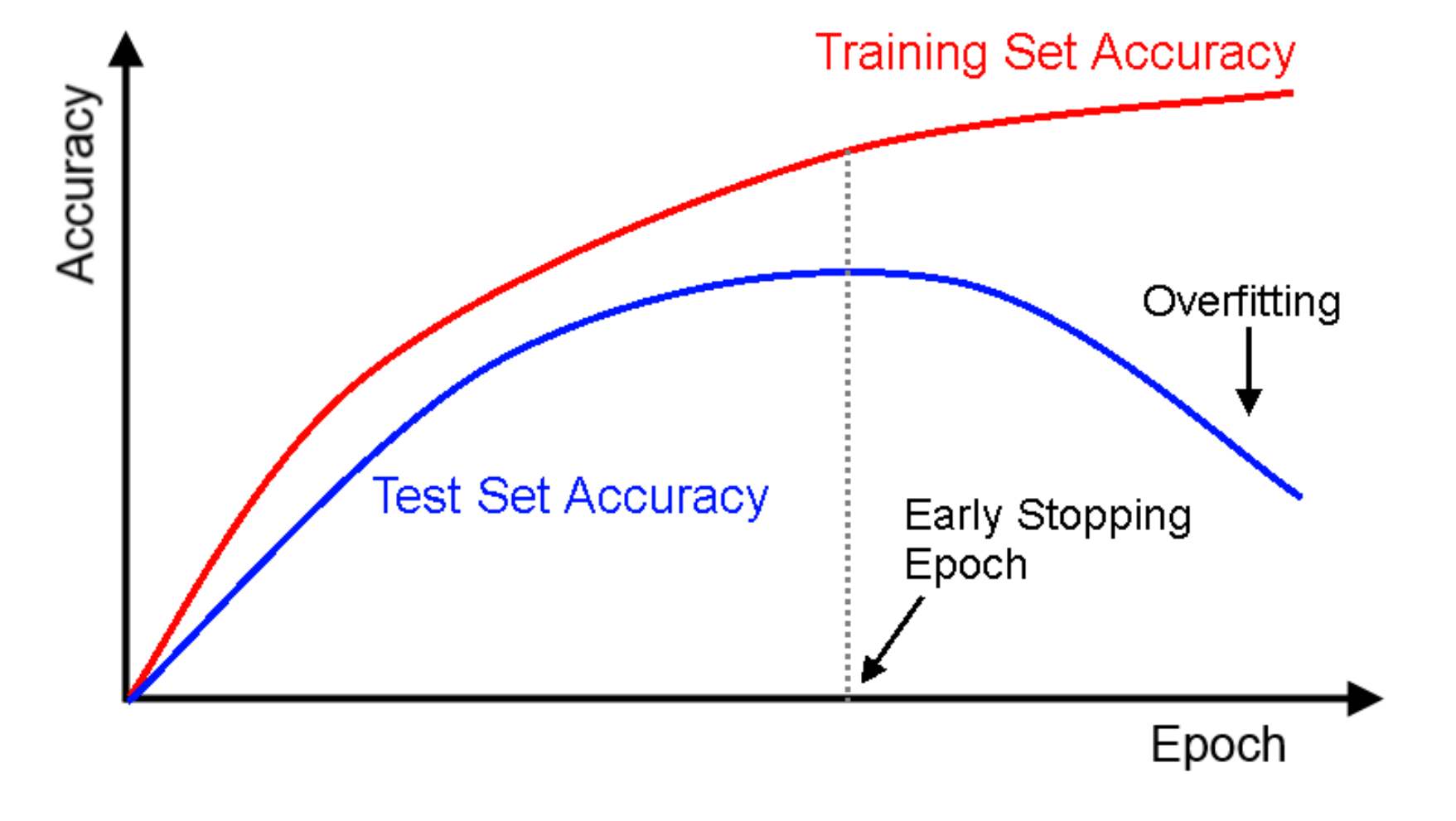
It is well known that OLS often does poorly in both prediction and interpretation. Penalization techniques have been proposed to improve OLS. For example, ridge regression (Hoerl and Kennard, 1988) minimizes the residual sum of squares subject to a bound on the L_2-norm of the coefficients. As a continuous shrinkage method, ridge regression achieves its better prediction performance through a bias–variance trade-off. However, ridge regression cannot produce a parsimonious model, for it always keeps all the predictors in the model. Best subset selection in

Address for correspondence: Trevor Hastie, Department of Statistics, Stanford University, Stanford, CA 94305, USA.
E-mail: hastie@stanford.edu

 1369–7412/05/67301

Hui Zou et al, 2005

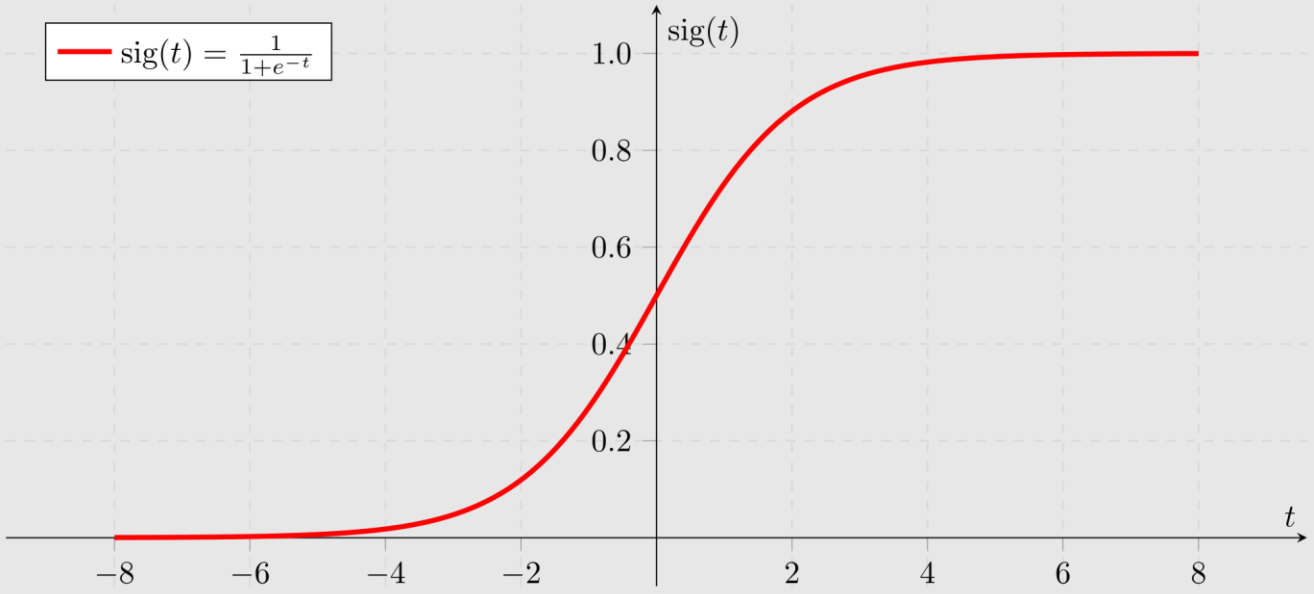
Early Stopping



〈https://deeplearning4j.org/docs/latest/deeplearning4j-nn-early-stopping〉

Logistic Regression

선형 회귀 모델과 같이 로지스틱 회귀 모델은 입력 특성의 가중치 합을 계산
➔ 선형 회귀 처럼 바로 결과를 출력하는 것이 아닌 로지스틱을 출력
(본문 188쪽)



Logistic Regression

추정 [편집]

최대가능도 방법 [편집]

로지스틱 회귀의 계수 추정은 최대가능도 방법을 이용한다.[4]

위의 로지스틱 함수를 바탕으로 가능도(likelihood)를 나타내면 아래 식으로 나타낼 수 있다. 편의를 위해 로지스틱 함수를

$p(y=1|x) = \theta(\beta \cdot X_i)$ 라 하면,

가능도는

$Pr(Y_i = y_i|X_i) = p_i^{y_i}(1-p_i)^{1-y_i} = \theta(y\beta \cdot X_i)$ 이 된다.

이 식을 바탕으로 전체 데이터에 대한 가능도를 표현하면 아래 식과 같다. 이 때 모든 데이터는 독립이어야 한다.

$p(Y|X) = \prod_{i=1}^{N} Pr(Y_i = y_i|X_i)$ (N: 전체 데이터 갯수, y_i, x_i: 데이터 중의 각 항목)

이 식을 최대로 하는 계수 (위의 식에서 β) 을 찾으면, 모델 추정이 완료된다.

최댓값을 찾기 위한 식을 계산의 편의성을 위해 최솟값을 구하는 함수로 나타내기 위해선 log 함수 형태로 고치면 된다.

Negative Log Likelihood: $\mathrm{NLL} = \frac{1}{N}\log(Y|X) = \frac{1}{N}\sum_{i=1}^{N}\log\frac{1}{Pr(Y_i = y_i|X_i)} = -\frac{1}{N}\sum_{i=1}^{N}\theta(y_i\beta \cdot X_i)$

그리고 위의 식은 종속 변수 y의 범위가 [0,1] 사이이므로, 이를 다시 표현하면 아래와 같다.

$$-\frac{1}{N}\sum_{i=1}^{N}\theta(y_i\beta \cdot X_i) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log\frac{1}{\theta(\beta \cdot X_i)} + (1-y_i)\log\frac{1}{1-\theta(\beta \cdot X_i)})$$

〈https://ko.wikipedia.org/wiki/%EB%A1%9C%EC%A7%80%EC%8A%A4%ED%8B%B1_%ED%9A%8C%EA%B7%80〉

Logistic Regression

Appl. Statist. (1992)
41, *No.* 1, *pp.* 191–201

Ridge Estimators in Logistic Regression

By S. LE CESSIE† and J. C. VAN HOUWELINGEN

University of Leiden, The Netherlands

[Received January 1990. Revised November 1990]

SUMMARY

In this paper it is shown how ridge estimators can be used in logistic regression to improve the parameter estimates and to diminish the error made by further predictions. Different ways to choose the unknown ridge parameter are discussed. The main attention focuses on ridge parameters obtained by cross-validation. Three different ways to define the prediction error are considered: classification error, squared error and minus log-likelihood. The use of ridge regression is illustrated by developing a prognostic index for the two-year survival probability of patients with ovarian cancer as a function of their deoxyribonucleic acid (DNA) histogram. In this example, the number of covariates is large compared with the number of observations and modelling without restrictions on the parameters leads to overfitting. Defining a restriction on the parameters, such that neighbouring intervals in the DNA histogram differ only slightly in their influence on the survival, yields ridge-type parameter estimates with reasonable values which can be clinically interpreted. Furthermore the model can predict new observations more accurately.

Keywords: Cross-validation; Deoxyribonucleic acid histogram; Logistic regression; Predictive value; Ridge regression

1. Introduction

In biostatistics, logistic regression is a popular method to model binary data. However, unstable parameter estimates occur when the number of covariates is relatively large or when the covariates are highly correlated. In this paper it is shown how ridge estimators can be combined with logistic regression to improve the model in such situations.

As an example we consider the following clinical problem. For 81 patients with ovarian cancer the deoxyribonucleic acid (DNA) content of about 300 cancer cells was determined by DNA image cytometry. For each patient a histogram of the distribution of the DNA content of the cancer cells was made. The question was how the relation between survival and DNA content of cancer cells could be modelled by using the information on the whole DNA histogram.

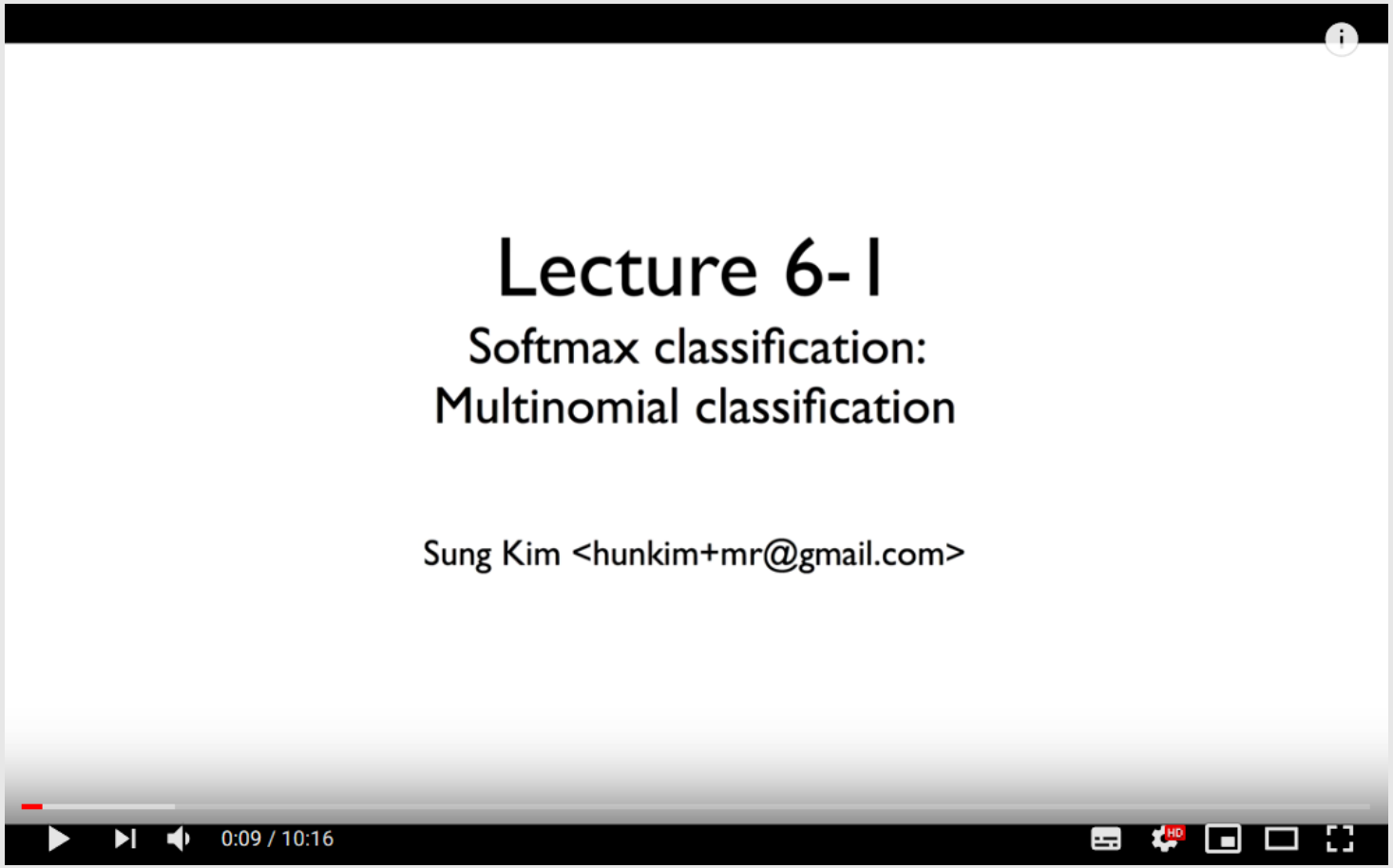
The DNA value expresses the amount of DNA in a cell, where 1*C* corresponds to the amount of DNA in a haploid cell, a cell with one set of 23 chromosomes. To construct a DNA histogram, the range of DNA values is split into 37 classes, with class interval 0.2*C*, except that the first class contains the fraction of cells with DNA values less than 0.9*C* and the last class the fraction of cells with DNA values greater than 7.9*C*. An example of a DNA histogram is given in Fig. 1.

†*Address for correspondence*: Department of Medical Statistics, University of Leiden, PO Box 9512, 2300 RA Leiden, The Netherlands.

 0035–9254/92/41191 $2.00

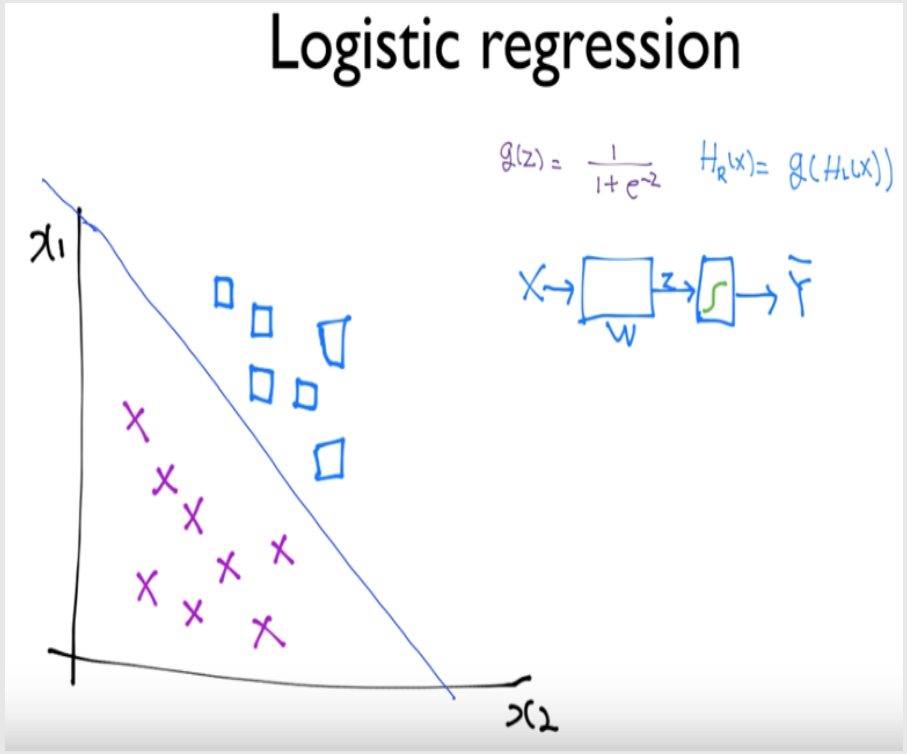
S.LE Cessie et al. 1990

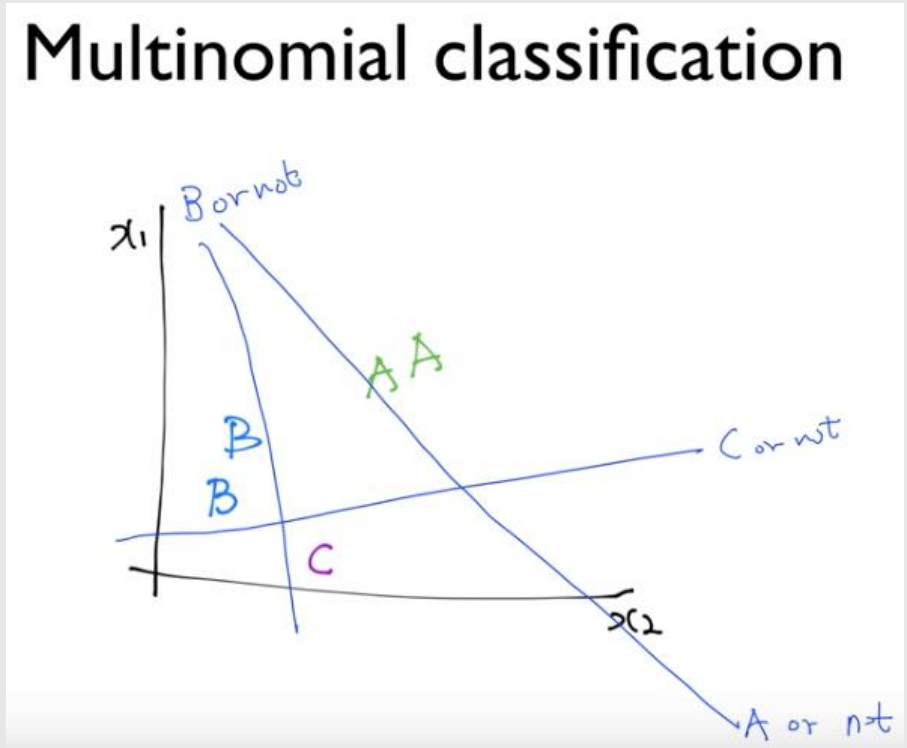
Softmax Classification



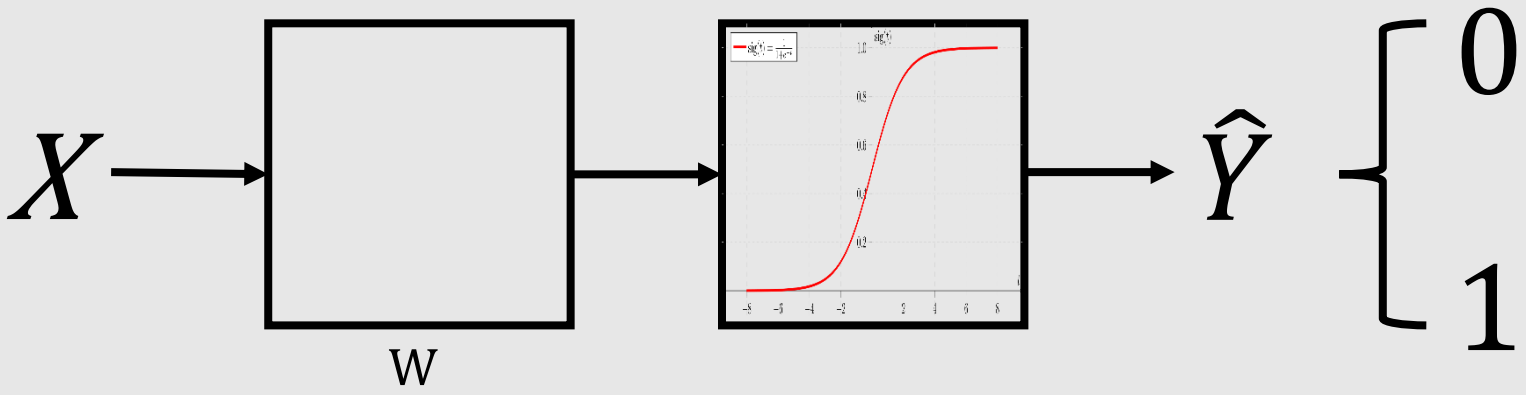
〈https://www.youtube.com/watch?v=MFAnsx1y9ZI〉

Softmax Classification

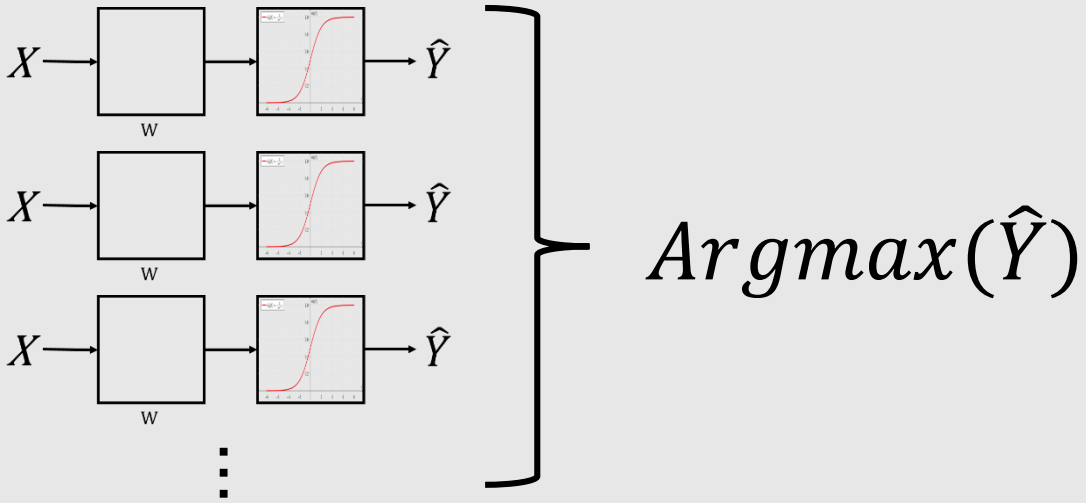




Logistic vs Softmax



Logistic Regression



Softmax Regression

Log Loss vs Cross Entropy

cross-entropy 도출

우선 로지스틱회귀 비용함수를 조금 변형해보자.

$$-(y\log(H(x)) + (1-y)\log(1-H(x)))$$

맞는가? 그렇다면 이번에는 우리 문돌이를 종종 미궁에 빠뜨리는 치환이다. y를 p1, H(x)를 q1, y-1을 p2, 1-H(x)를 q2로
치환해보자 그러면 식은 다음과 같이 표현할 수 있다.

$$-(p_1\log(q_1) + p_2\log(q_2))$$

이 식은 다시 다음과 같이 변형이 가능하고...

$$-(\sum_{i=1}^{2} p_i \log q_i)$$

위 식을 일반화 하면 최종적으로 다음과 같은 식이 나오는데 이 식이 바로 Cross-entropy 식이다.

$$-\sum_{i} p_i \log q_i$$

아마도 제대로 된 cross-entropy의 개념은 이보다 더 깊은 의미가 있고 식의 도출도 더 복잡하겠지만 문돌이가 이해하기에는 이정도가 딱인 듯싶다.

〈http://mazdah.tistory.com/791〉

Odds Ratio

	# of times caught	# of times not caught	Total # of cast
Bassassinator	50 (a)	50 (b)	100 (a+b)
No bait	2 (c)	98 (d)	100 (c+d)

1. **오즈(odds)**

당신이 물고기를 잡을 확률(P) / 물고기를 한 마리도 잡지 못할 확률 (1-P) 으로 물고기를 잡을 확률이 물고기 잡지 못할 확률에 몇 배가 되는 가의 값이 된다. 아래 결과에서 보듯이 Bassassinator을 사용했을때 물고기를 잡을 확률은 잡지 못할 확률에 1배이므로 같다고 할수 있다. 그렇지만 No bait를 한 경우는 물고기를 잡을 확률이 훨씬 작아진다.

Bassaddinator 오즈 (odds) = a/b=50/50 = 1

No bait의 오즈 (odds)= c/d=2/98 = 0.0204

2. 오즈비(odds ratio)

오즈비는 위에서 구한 오즈의 비율이다. 아래에서 보듯이 **당신이** Bassassinator를 사용한 경우 물고기를 잡을 확률비는 no bait를 사용한 경우의 물고기를 잡을 확률비 대비 50배가 높다고 할 수 있다.

Bassassinator vs. no bait 의 오즈비 =1.0/0.02 = 50.

〈https://tip.daum.net/openknow/65887671〉

Odds Ratio

3. 상대위험도(relative risk(chance))

당신이 Bassassinator를 사용하고 물고기를 잡을 확률 = a/(a+b)= 50/100 = **0.50**

당신이 no bait의 경우 물고기를 잡을 확률 = c/(c+d)= 2/100 = **0.02**

상대 위험도 (relative risk) =

Bassassinator를 사용하고 물고기를 잡을 확률 /no bait의 경우 물고기를 잡을 확률 = **0.5/0.02 = 25**

〈https://tip.daum.net/openknow/65887671〉

거의 모든 연구는 실제로 처리(treatment)그룹과 대조(control)그룹의 모집단수를 알 수 없다.
왜냐하면 연구자가 할 수 있는 것은 샘플 수를 조정할 수 있을 뿐이다.

➔ **오즈비(Odds Ratio) 사용**

Odds Ratio

아래의 표는 물고기를 잡은 100명의 낚시꾼을 조사한 결과 이들중 40명이 Bassassinator를 사용하였다. 그리고 100명의 물고기를 한마리도 잡지 못한 낚시꾼을 조사한 결과 단지 이들중 20명이 Bassassinator를 사용한 것으로 조사되었다. 즉 전체 물고기를 잡은 사람수를 구할 수 있는 방법은 없다. 즉, 연구자는 각각의 모집단의 수를 알 수 없다.

	Bassassinator use	No Bassassinator	Total #
Caught fish	40	60	100
Caught nothing	20	80	100

그래서 이때 모집단을 알 수 없기 때문에 대신 오즈비를 사용하게 된다.

오즈 (Bassassinator를 사용하고 물고기를 잡을 확률/ 잡지 못할 확률) = 40/60 = **0.67**
오즈 (No Bassassinator 경우 물고기를 잡을 확률/ 잡지 못할 확률) = 20/80 = **0.25**

따라서,
성공적(물고기를 낚은)인 낚시꾼이 Bassassinator 를 사용하는 비율과
실패한(물고기를 낚지 못한) 낚시꾼의

오즈비(Odds ratio) = 0.67/0.25 = 2.7

이것은 물고기를 잡은 낚시꾼들은 물고기를 낚지 못한 낚시꾼들에 대비해서 Bassassinator를 2.7배 더 사용하는 경향이 있다라고 해석을 해야한다.

흔히 범하는 실수는 Bassassinator를 사용하는 낚시꾼들은 물고기를 낚을 확률이 2배가 높다라고 해석하는 경우가 있는데 이것은 잘 못된 해석이다.

〈https://tip.daum.net/openknow/65887671〉

Cross Entropy's linearity

$$P(Y_1|X) = \frac{1}{1+e^{-a}}$$

$$a = \ln\left(\frac{P(X|Y_1)P(Y_1)}{P(X|Y_2)P(Y_2)}\right) \Rightarrow Log\ Odds$$

$if\ P(X|Y_1)P(Y_1) > P(X|Y_2)P(Y_2)\ then\ Class1 \Rightarrow$ a $= 0$ is Decision boundary

정규성,
등분산성 가정

$$a = \ln \frac{\exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right)P(Y_1)}{\exp\left(-\frac{1}{2}(x-\mu_2)^T\Sigma^{-1}(x-\mu_2)\right)P(Y_2)}$$

$$a = \mu_1\Sigma^{-1}x - \mu_2\Sigma^{-1}x - \frac{1}{2}\mu_1^T\Sigma\mu_1 + \frac{1}{2}\mu_2^T\Sigma\mu_2 + \ln\left(\frac{P(Y_1)}{P(Y_2)}\right)$$

$$a = w^Tx + w_0 \Rightarrow \textbf{선형식}$$

Cross Entropy's linearity

- Logistic Regression Model

  - The logistic regression model form is $\log\left(\frac{\pi(x)}{1-\pi(x)}\right) = \alpha + \beta x$

    로지스틱 회귀 모형은 $\log\left(\frac{\pi(x)}{1-\pi(x)}\right) = \alpha + \beta x$이다.

    the random component for the outcomes has a binomial distribution.

    랜덤 구성요소의 결과값은 이항 분포이다.

    The link function is the logit function, symbolized by "logit(π)"

    link function 은 로짓 함수이고, "logit(π)"라고 쓴다.

    Whereas π is restricted to the 0-1 range, the logit can be any real number

    반면에 π는 0 에서 1 사이의 범위를 가지며, 로짓은 아무 숫자나 될 수 있다.

    - so this model does not have the structural problem

    그래서 모형은 구조적 결함을 가지지 않는다.

```
PROC GENMOD data=glm;
    MODEL disease/total=snoring/dist=bin link=logit;
RUN;
```

Analysis Of Maximum Likelihood Parameter Estimates							
Parameter	DF	Estimate	Standard Error	Wald 95% Confidence Limits		Wald Chi-Square	Pr > ChiSq
Intercept	1	-3.8662	0.1662	-4.1920	-3.5405	541.06	<.0001
snoring	1	0.3973	0.0500	0.2993	0.4954	63.12	<.0001
Scale	0	1.0000	0.0000	1.0000	1.0000		

$logit[\hat{\pi}(x)] = -3.87 + 0.40x$

〈범주형 자료분석 필기 자료 中〉

Linear?

선형 회귀 모델이란?

➔파라미터 선형식으로 표현되는 회귀 모델을 의미

➔파라미터를 추정하거나 모델을 해석하기가 비선형 모델에 비해 비교적 쉽기 때문에 데이터를 적절히 변환하거나 도움이 되는 Feature들을 추가하여 선형 모델을 만들 수 있다면 적은 개수의 Feature로 복잡한 모델을 만드는 것 보다 여러 면에서 유리함

선형 회귀 모델의 문제는?

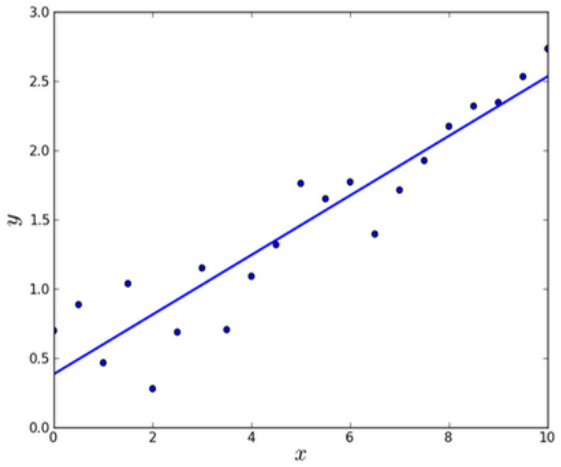
➔모델의 수(파라미터의 개수가 아니라 파라미터의 결합 형태)가 한정되어 유연성이 부족함

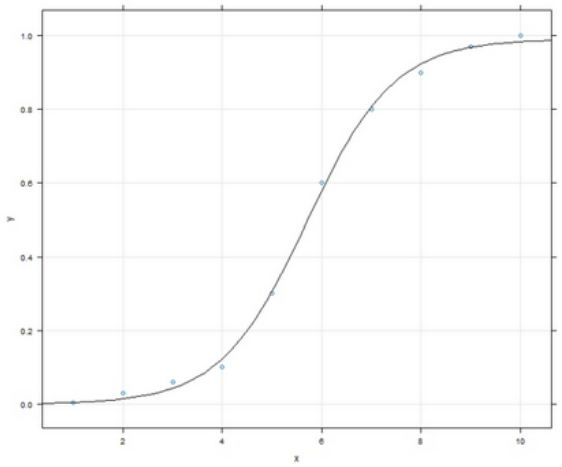
➔복잡한 패턴을 가진 데이터에 취약함

이에 대한 대안이 바로?

➔ Neural Network(뉴럴넷 ≒ 딥러닝)

Linear?





<그림 1> 선형 회귀 모델과 비선형 회귀 모델로 각각 검색했을때 나오는 대표적인 이미지 예

선형 회귀 모델은 '회귀 계수(regression coefficient)를 선형 결합으로 표현할 수 있는 모델'을 말합니다. 즉, 독립 변수가 일차식이냐 이차식이냐 로그 함수식이냐가 아니라 우리가 추정할 대상인 파라미터가 어떻게 생겼느냐의 문제입니다. 가령 아래 함수들은 모두 선형 회귀식입니다.

- $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$
- $y = \beta_0 x^{\beta_1}$
- $y = \frac{e^{\beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3}}{1 + e^{\beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3}}$

〈https://brunch.co.kr/@gimmesilver/18〉

Cross Entropy

어떠한 확률 분포의 Parameter를 추정할 때 가장 만만한 방법이 MLE
➔주어진 데이터로부터 어떠한 Parameter를 선택할 때 주어진 Data가 가장 잘 설명될까?
➔$P(Classes|X,w)$가 최대가 될 것인가?

$$Likelihood = \prod_{n=1}^{N} P(t_n|x_n,w) = \prod_{n=1}^{N} \begin{cases} P(t_n=1|x_n)\ when\ (t_n=1) \\ 1-P(t_n=1|x_n)\ when\ (t_n=0) \end{cases}$$

여기서 $y_n = P(t_n=1|x_n) = P(Y_1|X) = \sigma(w^T x)$

$$Likelihood = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n} \Rightarrow LogLikelihood = \sum_{n=1}^{N} t_n \log(y_n) + (1-t_n)\log(1-y_n)$$

$$Loss = -\sum_{n=1}^{N} t_n \log\big(\sigma(w^T x)\big) + (1-t_n)\log\big(1-\sigma(w^T x)\big)$$

Cross Entropy's Convex

다음 Cost Function이 Convex하다는 것을 증명해야 하는데, 왜냐하면 Convex하게 되면, Unique한 Solution이 존재하게 되고, Gradient 방법을 사용해서 쉽게 해를 구할 수 있기 때문이다.

$$Loss = -\sum_{n=1}^{N} t_n log(\sigma(w^T x)) + (1 - t_n) log(1 - \sigma(w^T x))$$

그 전에 Convex의 조건을 살펴보면,

어떠한 함수를 2차함수로 근사하고자 한다면, Taylor 급수를 사용해서

$$f(x) = f(a) + Df(a)(x - a) + \frac{1}{2}(x - a)^T Hf(a)(x - a)$$

처럼 근사해서 (여기서 D는 미분, H는 2차 미분이다) 마치 우리가 일반적인 2차함수

$$f(x) = ax^2 + bx + c$$

에서 a > 0이면 Convex하듯이 따져보면 되는데,

Multi-variate Function에서는 D는 Vector가 되고 H (Hessian이라 부른다)는 Matrix형태가 된다.

〈https://taeoh-kim.github.io/blog/cross-entropy%EC%9D%98-%EC%A0%95%ED%99%95%ED%95%9C-%ED%99%95%EB%A5%A0%EC%A0%81-%EC%9D%98%EB%AF%B8/〉

Cross Entropy's Convex

저 Hessian이 Positive Definite하게 되면, 일반적인 2차함수에서 a > 0과 같은 의미가 되어, Convex함이 보장되는데,

다음과 같을 때 Positive Definite하다고 한다.

$$x^T H x > 0$$

그 이유는 일반적인 2차함수에서 a > 0 도 조건이지만,

$$ax^2 > 0$$

도 같은 조건이기 때문이다.

아무튼 결국 다시 Cost Function

$$Loss = -\sum_{n=1}^{N} t_n log(\sigma(w^T x)) + (1 - t_n) log(1 - \sigma(w^T x))$$

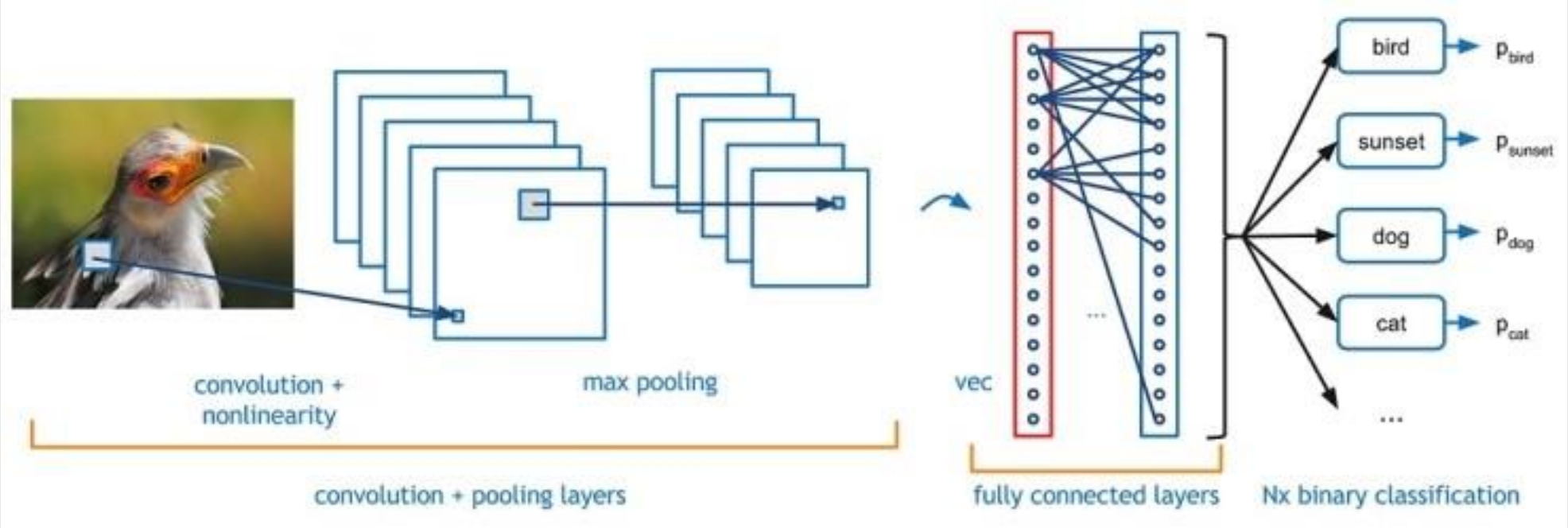
의 Hessian을 구해 보면 다음과 같게 되는데,

$$\sum_{n=1}^{N} y_n (1 - y_n) x_n {x_n}^T$$

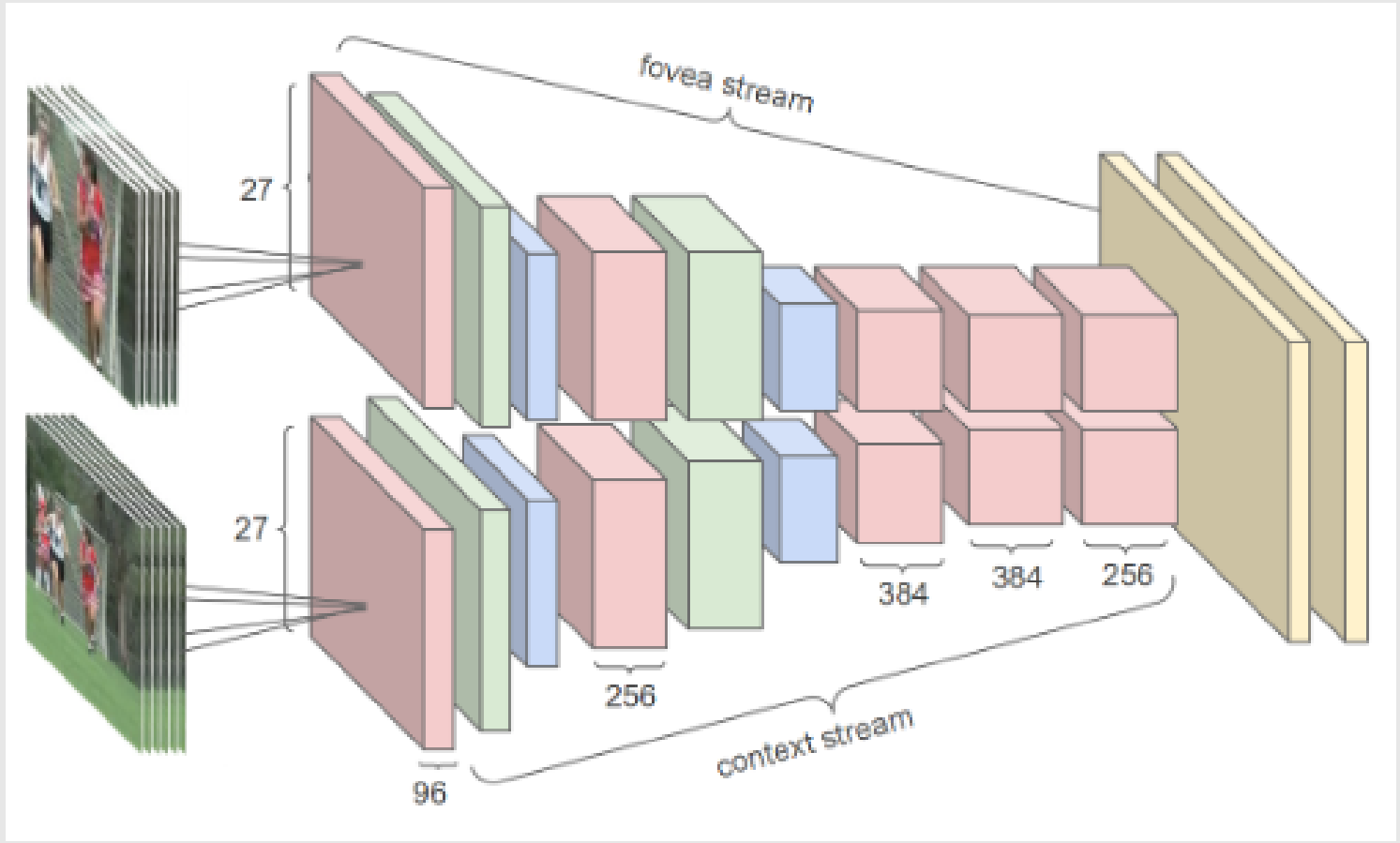
x는 vector이고, y는 Logistic Function의 Output으로서 0에서 1 사이의 값을 가지므로, Positive Definite가 되어 Convex를 보장하게 된다.

〈https://taeoh-kim.github.io/blog/cross-entropy%EC%9D%98-%EC%A0%95%ED%99%95%ED%95%9C-%ED%99%95%EB%A5%A0%EC%A0%81-%EC%9D%98%EB%AF%B8/〉

Hyper Quick Guide for CNN



Cross Entropy Example



Andrej Karpathy et al. 2014

Deeplearning Example

〈https://www.youtube.com/watch?v=PCBTZh41Ris〉


Ideas worth spreading
– TED Talks

고생하셨습니다.