

# PNU STAT - h2o tutorial (12.27)

대부분 이렇게 합니다!

Tool이 너무 많은 시대

[h2o.ai Overview](#)

[h2o architecture](#)

[h2o를 이제 다루어보아요](#)

[h2o 설치하려면](#)

[데이터의 형태를 바꾸어줘야합니다](#)

[모형 어떻게 fit?](#)

[모형 최적화 및 성능 평가](#)

[새로운 관측치에 대한 예측값 계산](#)

[Performance measurement](#)

[Lift](#)

[Grid Search & CV](#)

[AutoML](#)







[Deep Learning\(Neural Networks\)](#)

[Auto Encoder](#)

[Principal Components Analysis](#)

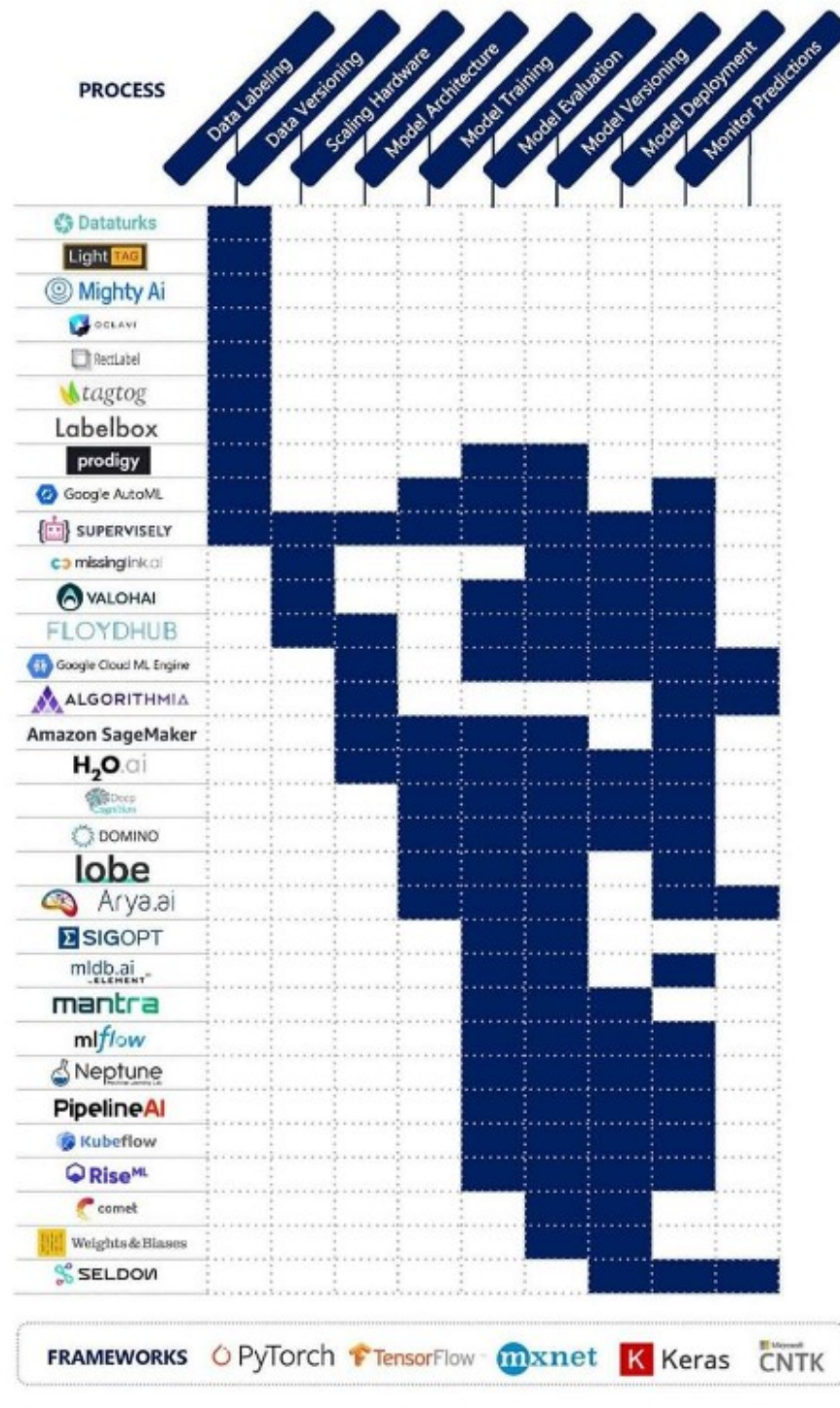
[REF](#)

## 대부분 이렇게 합니다!

 <b>Data Load</b> <ul style="list-style-type: none"><li>• Data Load</li><li>• Data Integration (Merge)</li></ul>	 <b>Exploratory Data Analysis</b> <ul style="list-style-type: none"><li>• Basic Statistics</li><li>• Initial Data Pre-processing</li><li>• Initial Modeling</li><li>• EDA using Caret</li><li>• <u>Uni-variate</u></li><li>• Relation to Y</li></ul>	 <b>Data Pre-Processing</b> <ul style="list-style-type: none"><li>• Scaling</li><li>• Imputation</li><li>• Feature Transformation</li><li>• Dimension Reduction</li><li>• Feature Selection</li><li>• Sampling</li></ul>
 <b>Model Development</b> <ul style="list-style-type: none"><li>• Single Model</li><li>• Grid Search</li><li>• Parallel Computing</li></ul>	 <b>Predict &amp; Performance Check</b> <ul style="list-style-type: none"><li>• Predict</li><li>• Performance Metric - ROC, Lift</li><li>• Business Performance</li></ul>	 <b>Deploy</b> <ul style="list-style-type: none"><li>• API Development</li><li>• <u>Mojo</u> / <u>Pojo</u> (H2O)</li></ul>

## Tool이 너무 많은 시대

# DEEP LEARNING TOOLS

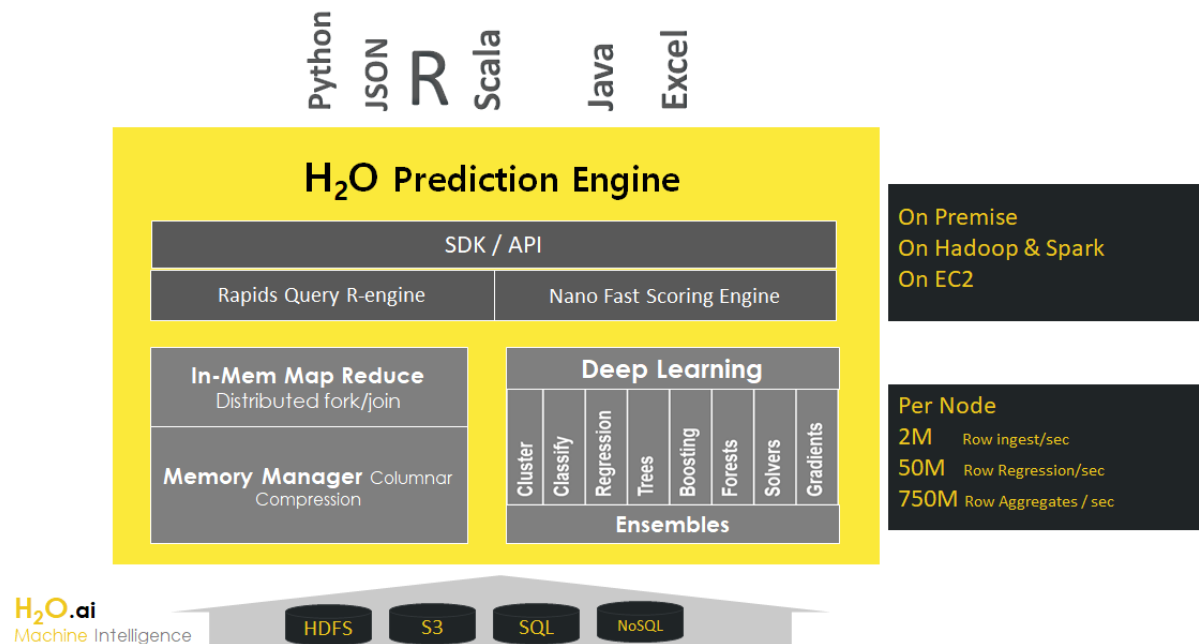


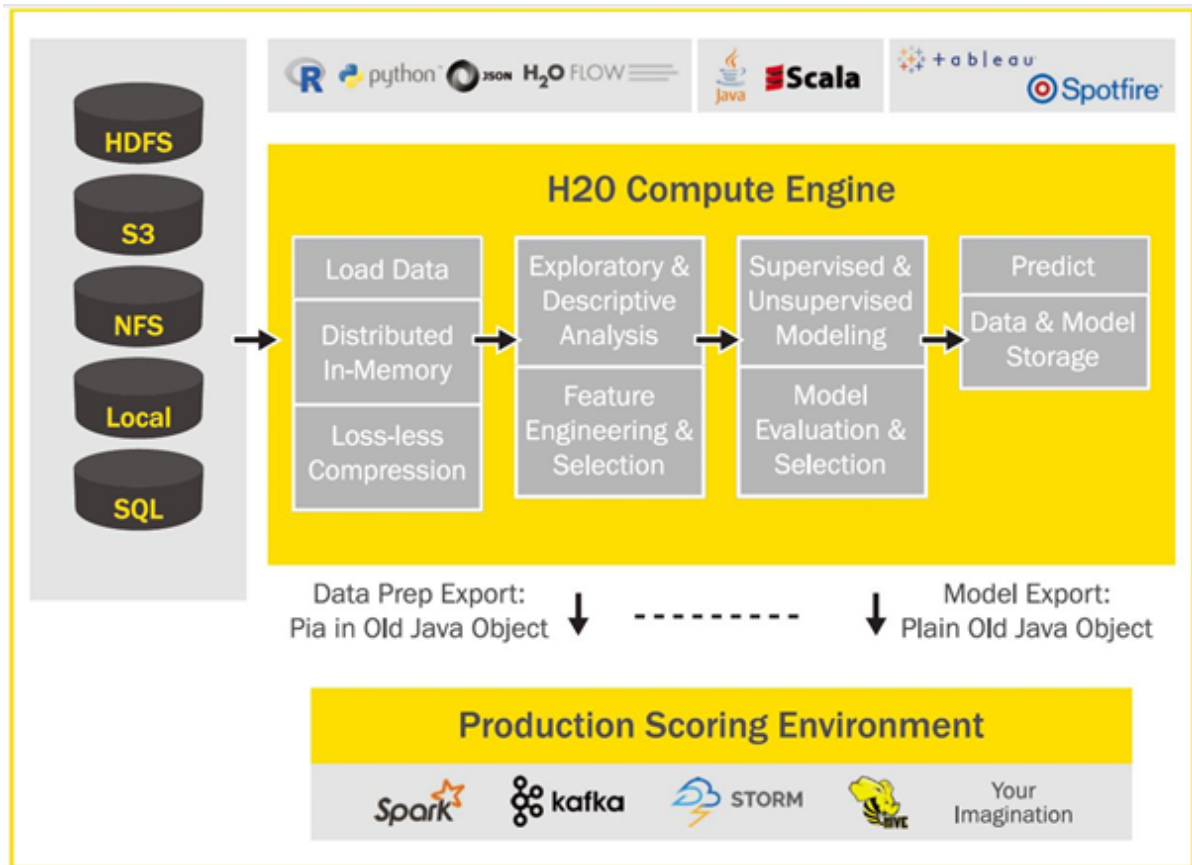
## h2o.ai Overview

- **Founded:** 2011 venture-backed, debuted in 2012
- **Product:** H2O open source in-memory prediction engine
- **Team:** 30
- **HQ:** Mountain View, CA
- **SriSatish Ambati** – CEO & Co-founder (Founder Platfora, DataStax; Azul)
- **Cliff Click** – CTO & Co-founder (Creator Hotspot, Azul, Sun, Motorola, HP)
- **Tom Kraljevic** – VP of Engineering (CTO & Founder Luminix, Azul, Chromatic)

## h2o architecture

- 머신러닝 방법론 중심의 알고리즘 framework
- Spark 활용 분산처리, GPU 병렬처리 등의 다양하게 지원
- DAI(Driverless AI) 등 비즈니스 수요 창출





## h2o를 이제 다루어보아요

### h2o 설치하려면

- java 1.7~1.8 필요함(java 기반의 어플리케이션으로 R과 별도로 구동합니다)
- 오라클설치

#### Java SE Development Kit 8 Downloads

<https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

### 데이터의 형태를 바꾸어줘야합니다

- `as.h2o()`를 활용해서 object를 h2o 형태로 변환해주어야 합니다

## 모형 어떻게 fit?

- h2o로 변환된 데이터를 활용하여 모형을 적합합니다
  - `h2o.randomForest()`
  - `h2o.gbm()`
  - `h2o.xgboost`

## 모형 최적화 및 성능 평가

- 옵션을 활용한 각 모형 모수 변경 가능
  - `h2o.grid()`
- 주요 모수는 코드 보면서 이야기해요!

## 새로운 관측치에 대한 예측값 계산

- `h2o.predict()`

## Performance measurement

분석의 목표치와 measurement를 정해야합니다.

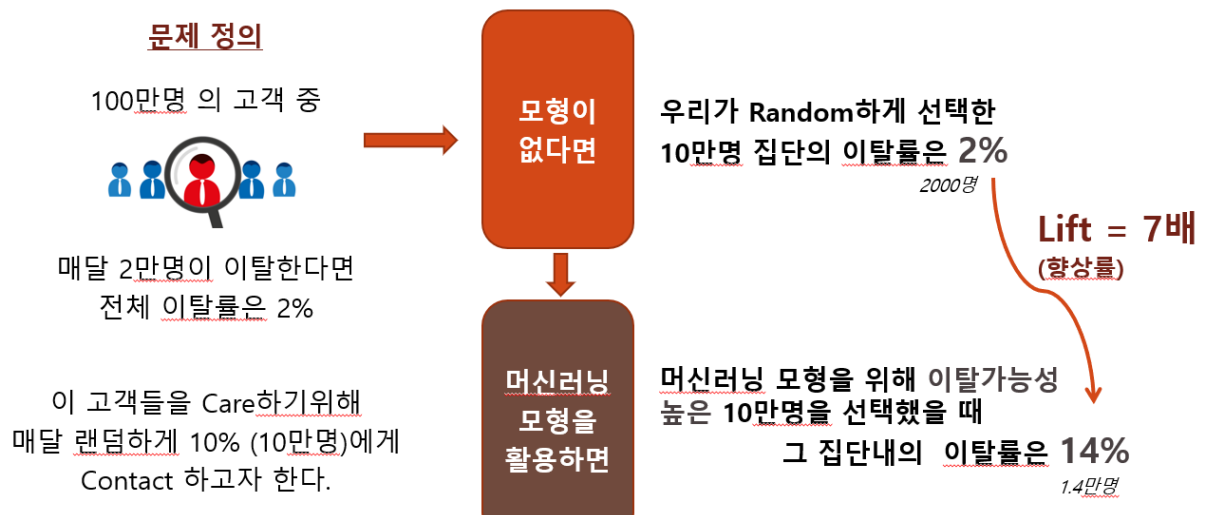
performance measurement는 학교에서 잘 가르쳐 줄 내용들입니다.

- Regression
  - MAE
  - MAPE
  - MSE
  - RMSE
- Classification
  - Logloss
  - accuracy

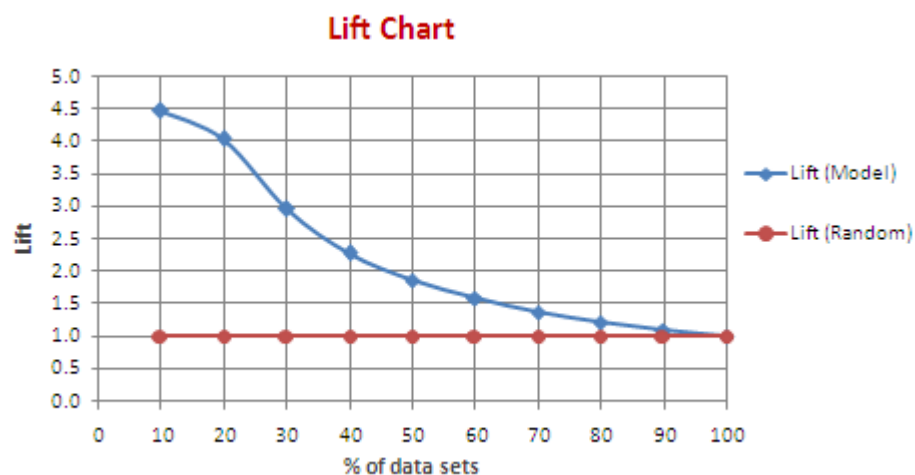
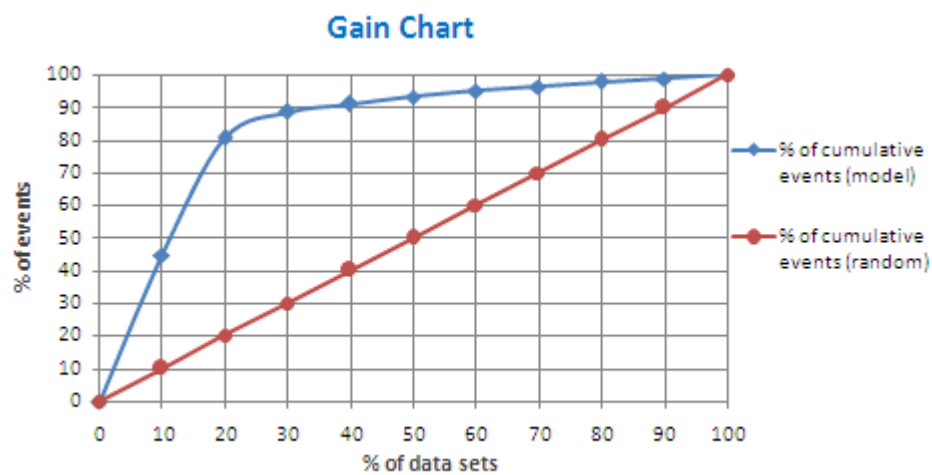
- auc
- precision
- recall
- f1score

## Lift

모형이 없을 경우, 대비 모델을 사용할 때 얼마나 더 개선 되는가?



Decile	n	res	survRatio	random	lift	cumRes	cumRandom	cumLift
<fct>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	26	26	1	10.1	2.58	26	10.1	2.58
2	25	23	0.92	9.70	2.37	49	19.8	2.48
3	30	24	0.8	11.6	2.06	73	31.4	2.32
4	26	8	0.308	10.1	0.793	81	41.5	1.95
5	27	7	0.259	10.5	0.668	88	52.0	1.69
6	27	2	0.0741	10.5	0.191	90	62.5	1.44
7	26	3	0.115	10.1	0.297	93	72.6	1.28
8	26	6	0.231	10.1	0.595	99	82.7	1.20
9	28	2	0.0714	10.9	0.184	101	93.5	1.08
10	27	3	0.111	10.5	0.286	104	104.	1.00



## Grid Search & CV



code

## AutoML

- 실제 머신러닝 모델의 Hyperparameter 설정이 매우 피곤한 작업
- 잘못 설정할 경우 이전 모델보다 더 나은 결과의 모델을 얻기 어려움

### No free lunch rule .....pitfalls of Data Scientists' machine learning

- activation function
- hidden layers & number of neurons
- loss function
- Learning rate
- regularization
- nolds
- Train-test ratio
- cross\_validation\_predictions
- weights\_column
- train\_samples\_per\_iteration
- Seed
- mini\_batch\_size
- The depth of a decision tree, number of trees in a forest,
- degree of regularization to prevent over-fitting



#### Manual parameter tuning

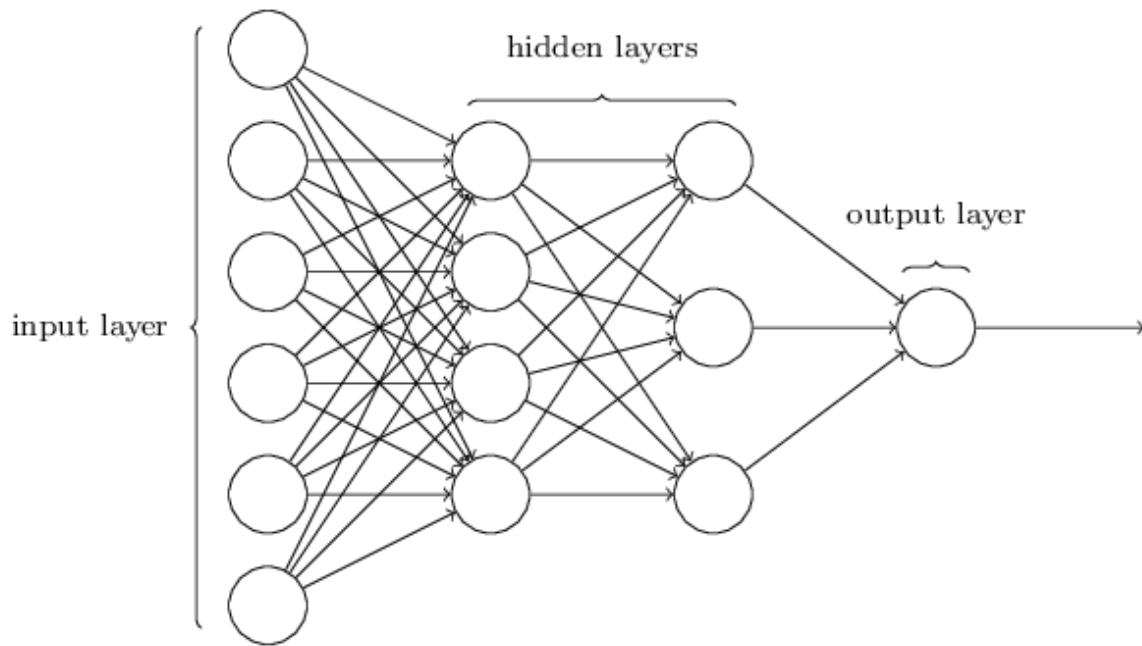
- ✓ ML은 정교한 방식에 의한 최적화 계산 방법론에 기반하고 있으며, 패키지의 default setting의 경우 그 문제에 대한 특이점을 반영하기 어려워 모델적합 Performance가 낮아지게 됨

예: H2o.deeplearning 모델개발 시 세팅 해줘야 하는 파라미터

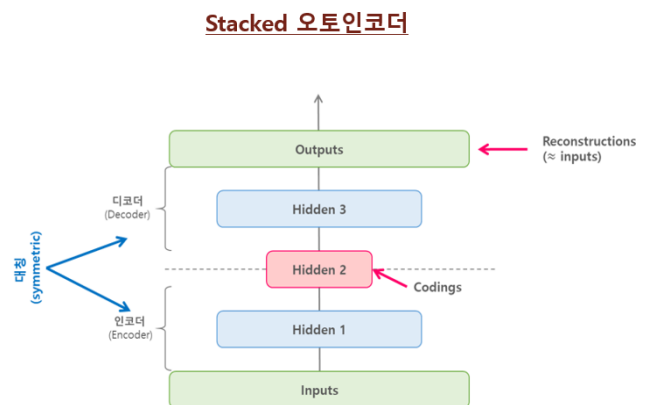
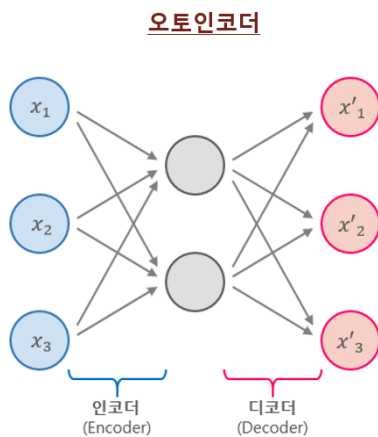
```
h2o.deeplearning(x, y, training_frame, model_id = NULL, validation_frame = NULL, nolds = 0,
keep_cross_validation_predictions = FALSE, keep_cross_validation_fold_assignment = FALSE, fold_assignment =, fold_column =
NULL, ignore_const_cols = TRUE, score_each_iteration = FALSE, weights_column = NULL, offset_column = NULL,
balance_classes = FALSE, class_sampling_factors = NULL, max_after_balance_size = 5, max_hit_ratio_k = 0, checkpoint =
NULL, pretrained_autoencoder = NULL, overwrite_with_best_model = TRUE, use_all_factor_levels = TRUE, standardize = TRUE,
activation =, hidden = c(200, 200), epochs = 10, train_samples_per_iteration = 2, target_ratio_comm_to_comp = 0.05, seed =
-1, adaptive_rate = TRUE, rho = 0.99, epsilon = 1e-08, rate = 0.005, rate_annealing = 1e-06, rate_decay = 1,
momentum_start = 0, momentum_ramp = 1e+06, momentum_stable = 0, nesterov_accelerated_gradient = TRUE,
input_dropout_ratio = 0, hidden_dropout_ratios = NULL, l1 = 0, l2 = 0, max_w2 = 3.4028235e+38, initial_weight_distribution =,
initial_weight_scale = 1, initial_weights = NULL, initial_biases = NULL, loss =, distribution =, quantile_alpha = 0.5,
tweedie_power = 1.5, huber_alpha = 0.9, score_interval = 5, score_training_samples = 10000, score_validation_samples = 0,
score_duty_cycle = 0.1, classification_stop = 0, regression_stop = 1e-06, stopping_rounds = 5, stopping_metric =,
stopping_tolerance = 0, max_runtime_secs = 0, score_validation_sampling =, diagnostics = TRUE, fast_mode = TRUE,
force_load_balance = TRUE, variable_importances = TRUE, replicate_training_data = TRUE, single_node_mode = FALSE,
shuffle_training_data = FALSE, missing_values_handling =, quiet_mode = FALSE, autoencoder = FALSE, sparse = FALSE,
col_major = FALSE, average_activation = 0, sparsity_beta = 0, max_categorical_features = 2147483647, reproducible = FALSE,
export_weights_and_biases = FALSE, mini_batch_size = 1, categorical_encoding =, elastic_averaging = FALSE,
elastic_averaging_moving_rate = 0.9, elastic_averaging_regularization = 0.001, verbose = FALSE)
```

## Deep Learning(Neural Networks)

code

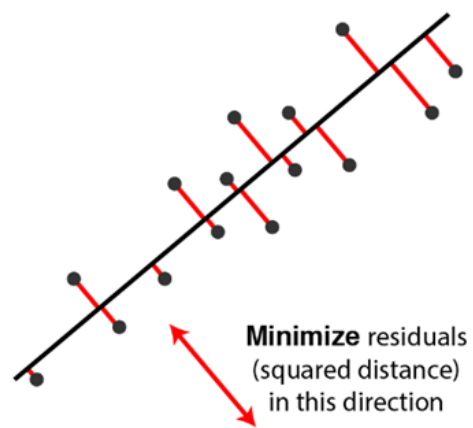
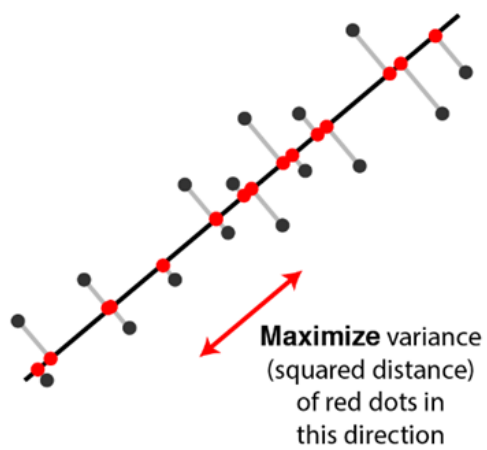
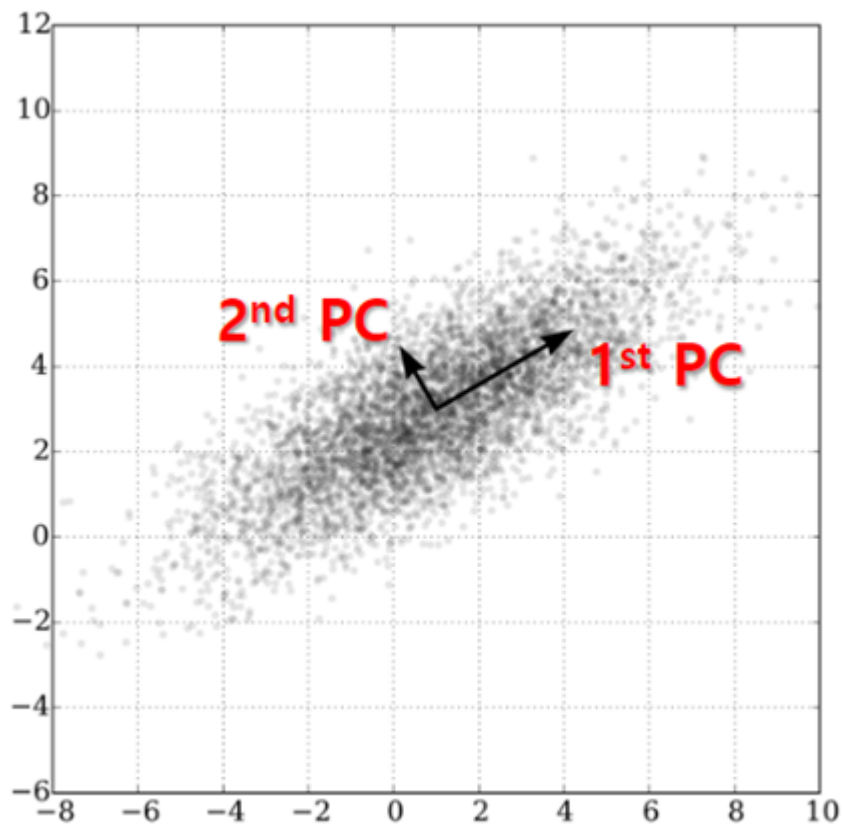


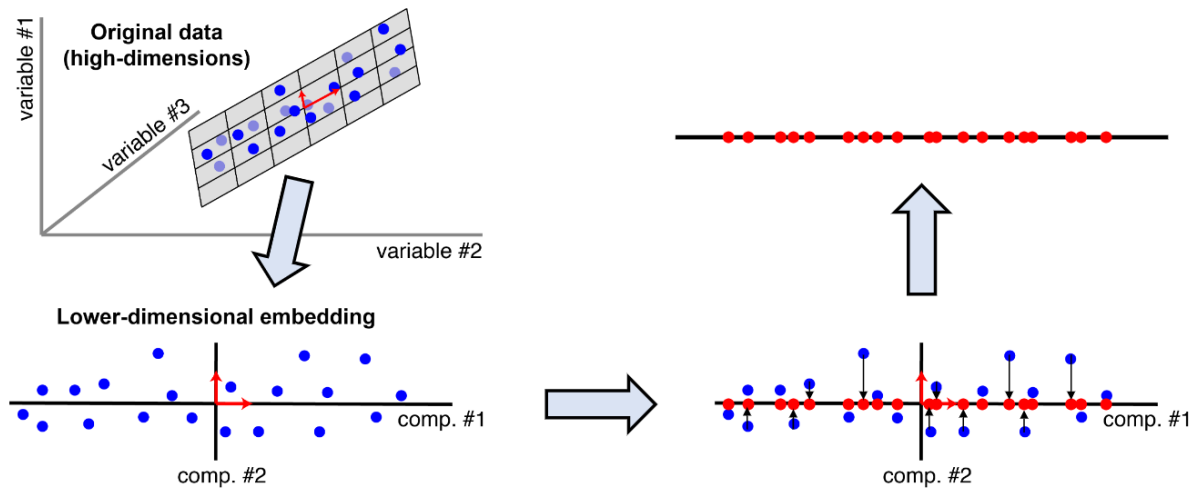
## Auto Encoder



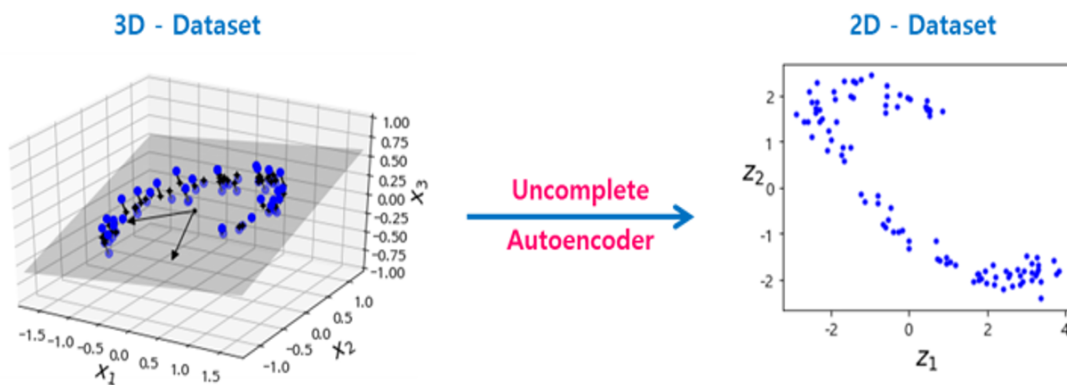
오토인코더(Autoencoder)는 그림과 같이 단순히 입력을 출력으로 복사하는 신경망이다. 어떻게 보면 간단한 신경망처럼 보이지만 네트워크에 여러가지 방법으로 제약을 줌으로써 어려운 신경망으로 만든다. 예를들어 아래 그림처럼 hidden layer의 뉴런 수를 input layer(입력층) 보다 작게해서 데이터를 압축(차원을 축소)한다거나, 입력 데이터에 노이즈(noise)를 추가한 후 원본 입력을 복원할 수 있도록 네트워크를 학습시키는 등 다양한 오토인코더가 있다. 이러한 제약들은 오토인코더가 단순히 입력을 바로 출력으로 복사하지 못하도록 방지하며, 데이터를 효율적으로 표현(representation)하는 방법을 학습하도록 제어한다.

## Principal Components Analysis





### Using Autoencoders like PCA



위에서 살펴본 Undercomplete 오토인코더에서 활성화 함수를 sigmoid, ReLU같은 비선형(non-linear)함수가 아니라 선형(linear) 함수를 사용하고, 손실함수로 MSE(Mean Squared Error)를 사용할 경우에는 PCA라고 볼 수 있다.

## REF

[h2oai/h2o-tutorials](https://github.com/h2oai/h2o-tutorials)

<https://github.com/h2oai/h2o-tutorials>

[H2O Tutorials](http://docs.h2o.ai/h2o-tutorials/latest-stable/)

<http://docs.h2o.ai/h2o-tutorials/latest-stable/>

### H2O 소개 및 간단한 사용법

[https://rstudio-pubs-static.s3.amazonaws.com/359032\\_6d2fa1280f8a40a582c8a40fb46c8c15.html](https://rstudio-pubs-static.s3.amazonaws.com/359032_6d2fa1280f8a40a582c8a40fb46c8c15.html)

### Understand Gain and Lift Charts

<https://www.listendata.com/2014/08/excel-template-gain-and-lift-charts.html>

### GitHub & BitBucket HTML Preview

<https://htmlpreview.github.io/?https://github.com/ledell/sldm4-h2o/blob/master/sldm4-deeplearning-h2o.html>