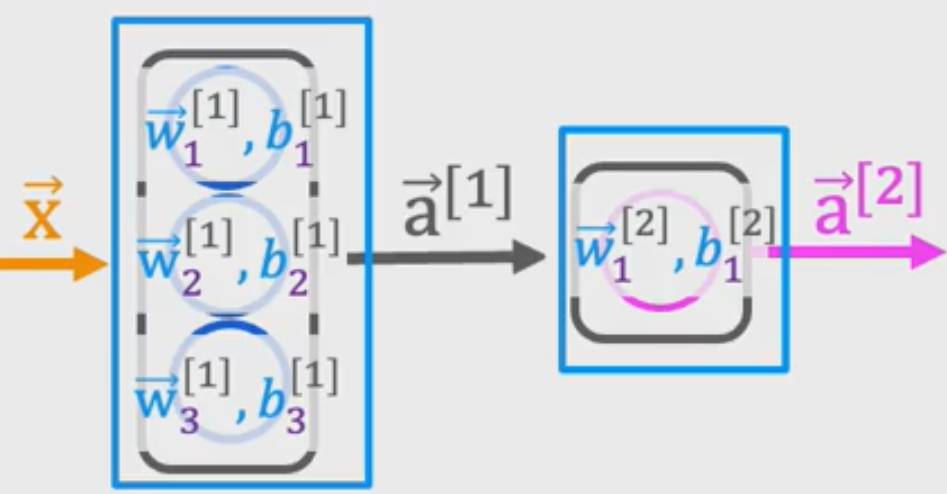
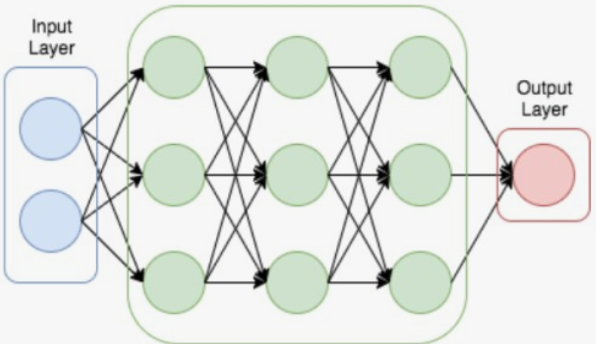
@ Neural Network

# Neural Networks

* Input(data) -> Computation(layers) : Grouped -> Output
* 이전 layer의 Output은 뒤에 올 layer의 Input이 된다. (chain)
* Each layer의 Output을 Vectorization해야 다른 layer에서 이용 가능함
* Forward Propagation : 정방향으로 나가는 성질 때문에 갖는 성질



# TensorFlow

* Useful tool sets for Neural Network building
* ex. Keras, ‘Utility, Optimizer, Regularization’ func
* ‘자동 미분’을 지원해 Back-propagation Algorithm을 효율적으로 구현 가능함.

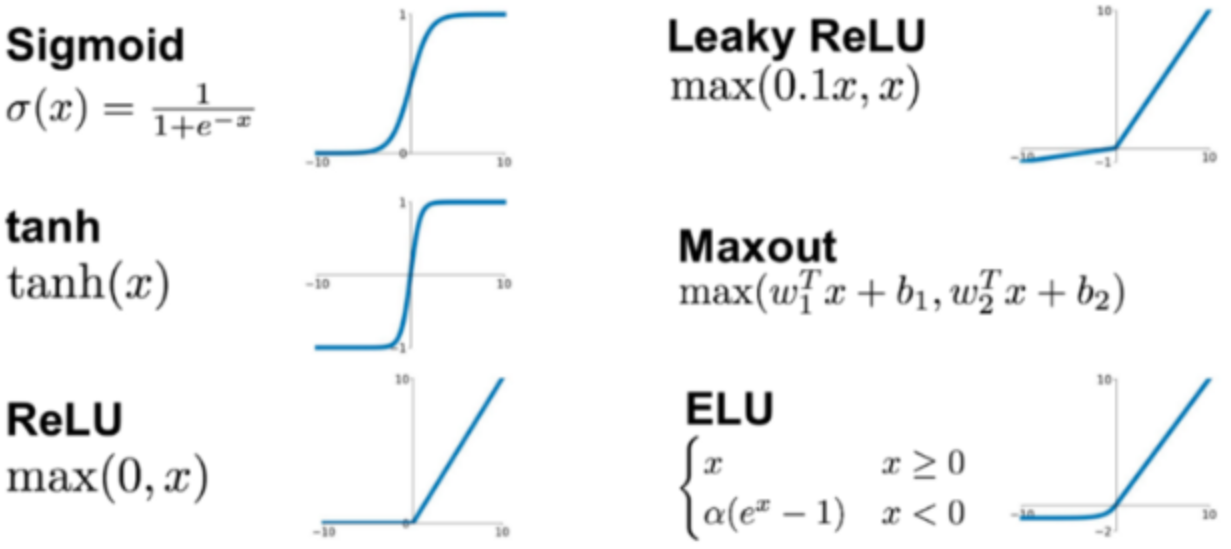
# Model Training progress with TensorFlow

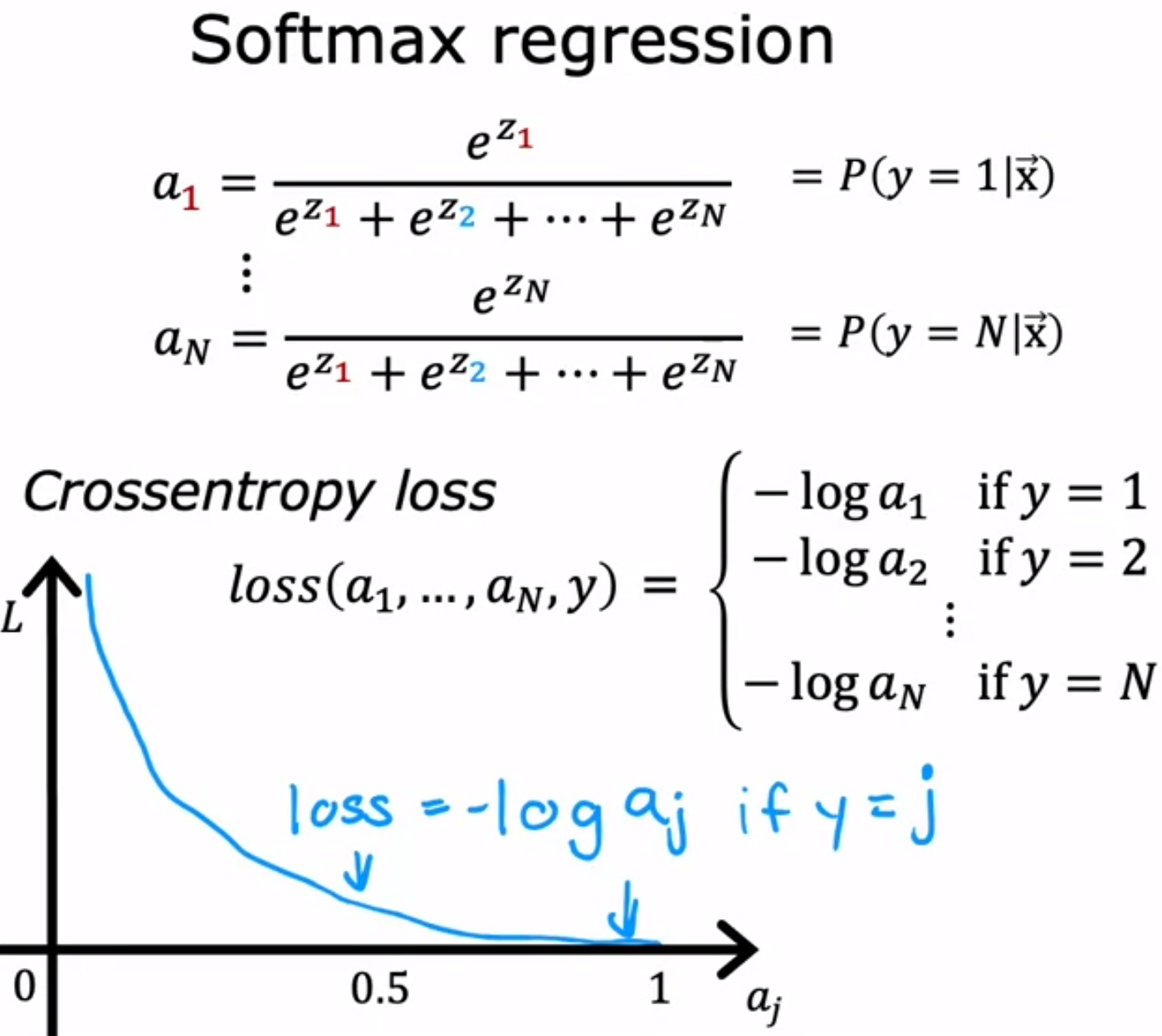
1. Specify how to compute(Activate func) Output
2. Specify f(cost, loss)
3. Train data : goal is minimize cost, loss

* Model = Sequential ([Dense(..), Dense(..), ..])
* Model is assembly with continued layer

@ Neural Network Training

#Activation Functions

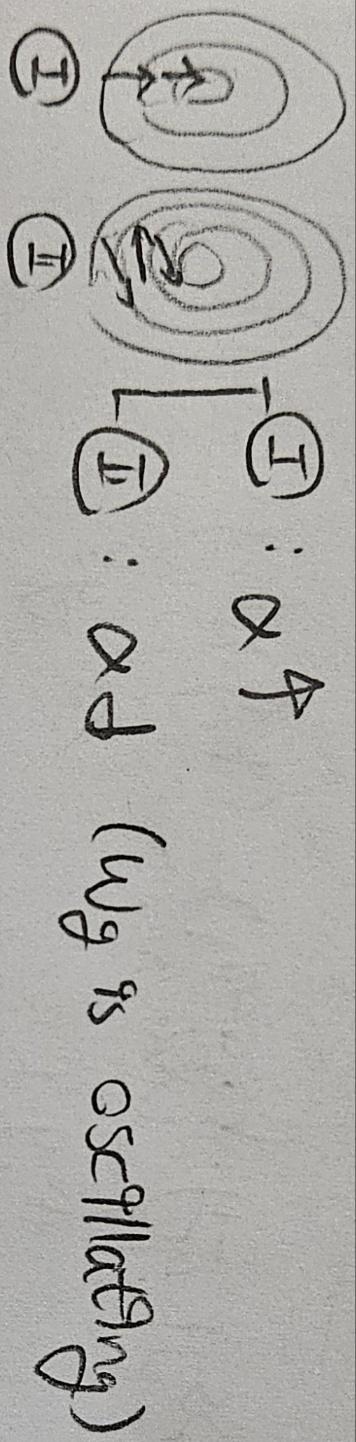
* Select how to return computed output
* Binary Classification : Sigmoid
* Boundary Classification : Linear, non-Linear2
* Value must be higher than zero : ReLU(Rectified Linear Unit) 

# Multi-class Classification : Softmax

* Boundaries can be a mult/noni-linear shape. ex. Mnist
* R) Logistic regression : ‘z=w\*x+b’ (loss func = -y\*log(a1)-(1-y)\*log(1-a1)
* In softmax regression, y has a lot of values. (1, 2, … N)
* So, the softmax regression model’s loss func is cross-entropy loss.

# Neural network with soft-max

* If N(class)=k, k units are in a layer.
* Each unit is treated as a dot product



# Adam(Adaptive Mnist) algorithm

* Optimization for gradient descent
* Adjust learning rate during descent

# CNN (Convolutional Neural Network)

* P of DNN : When DNN uses images, it is flatten without abstract images. Thus, spatial/topological information (like axis, intervals) are lost.
* S as CNN : Remain spatial topological info - Use partial images
* Divide > Faster computation
* Need less training data (less prone to overfitting)

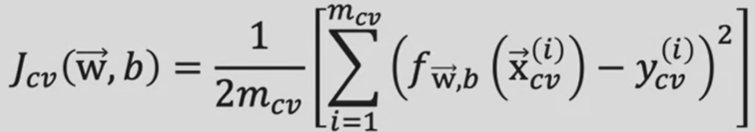
# Back Propagation - Deep Learning

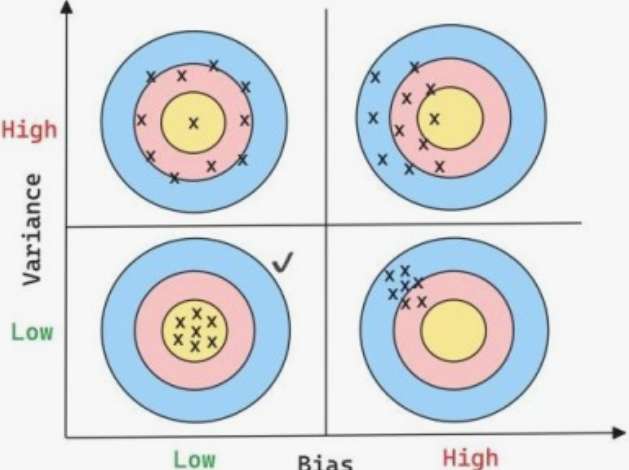
* In Multi-neural network, derivatives are useful.
* Using chain rule, we can assume two derivatives with only one derivatives > Computation process is simplified
* ex. If (dj/dw) = (dj/db)\*(db/dc)\*(dc/dw), Because ‘db/dc’ allows ‘db’, ‘dc’ values, it helps assuming other derivative values.
* Improved computation time : N(Node) + N(parameter)

@ Advice for applying machine learning

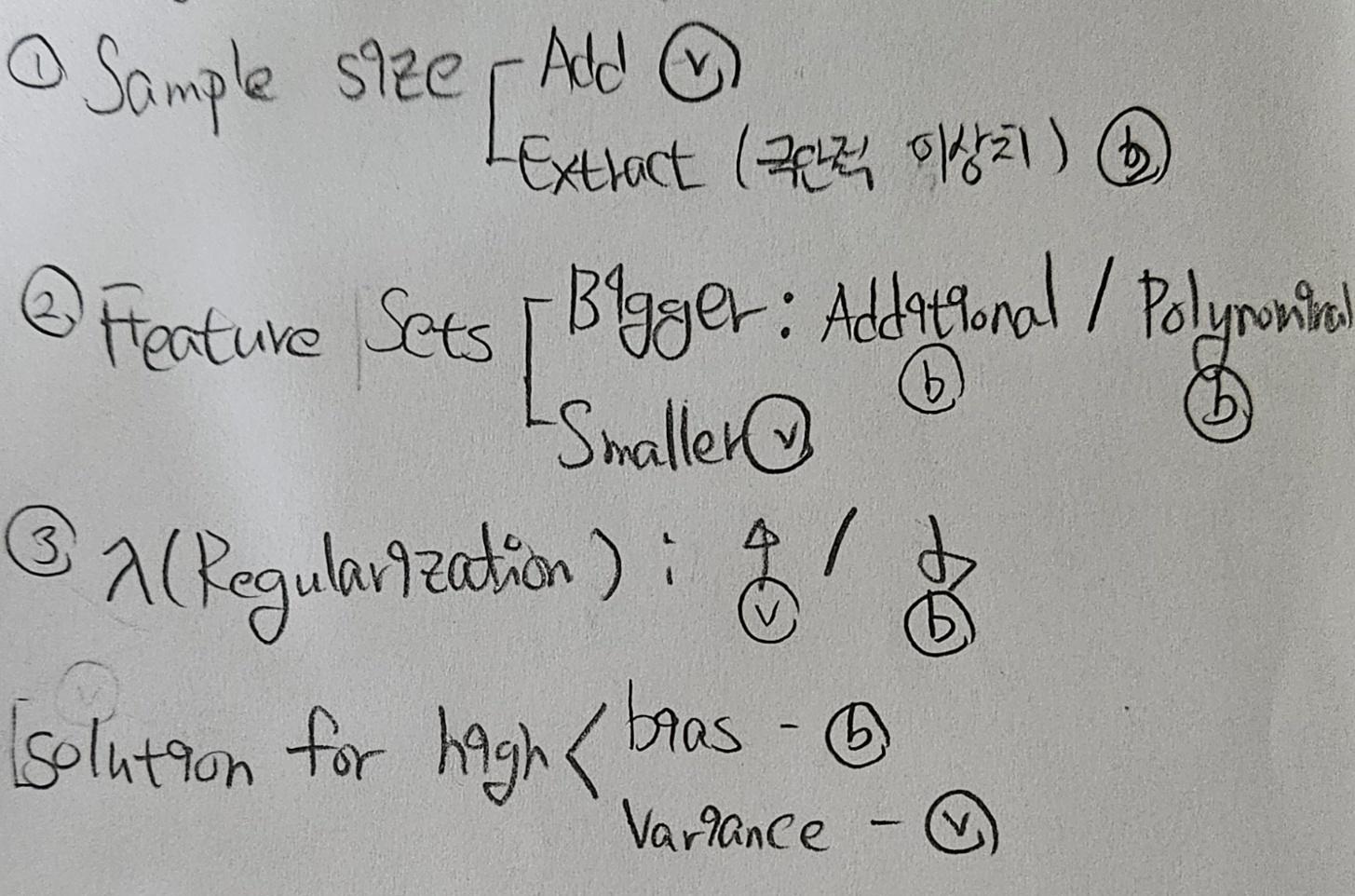
# Advice for applying machine learning

* Adjusting features, learning rate, and size of train can make a better model.
* Once parameters are fit to the training set, the training error can be estimated by cross-validation error



# Algorithm tuning : bias and variance

* If the degree of polynomial is higher, training cost is lower. but, prediction cost can be higher
* Bias : mean of sum(Prediction - Real)
* goal : finding sweet spot between underfitting and overfitting
* Adjusting sample size, feature set, regularization makes a better model.



# Model evaluation

* ex. Confusion matrix, Precision-recall, F1-score
* Accuracy = (TP + TN)/(TP + FP + FN + FN)
* Precision = TP / (TP + FP)
* Recall = TP / (TP + FN)
* F1 score = 2 \* Prediction \* Recall / (Precision + Recall)

@ Decision Trees

# Classification Decision Tree

* Tree when target value is categorical.
* Typical model : CART (Classification And Regression Tree)
* Divide features based on in-purity(Gini Criterion)
* V(train) is high, but V(pred) is low.
* The higher depth should make the purity higher.
* P) NP-complete : Finding optimum boundaries is too hard.
* S) Heuristic way

# Regression Tree

* Tree when target value is continuous(numerical)
* Use binary recursive partitioning process

+) Fit well in the non-linear data. no need to EDA

-) Vulnerable to the overfitting, Pred score can be lower

# Encoding for no binary features

* Computer can’t identify string, text
* Encoding allows computation
* Target encoding : use median value
* One-hot encoding : no-binary(class) -> 0/1/2/..(numeric)

# Tree Ensembles

* P) Single tree is too sensitive to small changes of the data
* S) Sampling with replacement multiple times

# Random Forest Algorithm

* P) CART Tree can make trees which have too high depth. -> Overfitting
* S) Use median value of sample data from train dataset.
* Reputation of sampling makes lower speed of training, but accuracy can be higher.

# Ensemble

* predict based on many models > put together > Define final prediction value

1. Bagging : Sampling without replacement, Divide-and-conquer

* Divide training set into small sets. > Gather all and fit to the model

1. Boosting : Sampling with replacement

* AdaBoost, Gradient Boosting, XGBoost
* XGBoost is highly good at criteria classification and regularization is implied.

# Decision Tree vs Neural Network

1. Decision Tree and Tree Ensemble : work well only at structured data
2. Neural Network

* Work well on both structured and unstructured data.
* V(train) is slower than Tree.
* Converting from single into multiple model is very easy