# Statistic 4

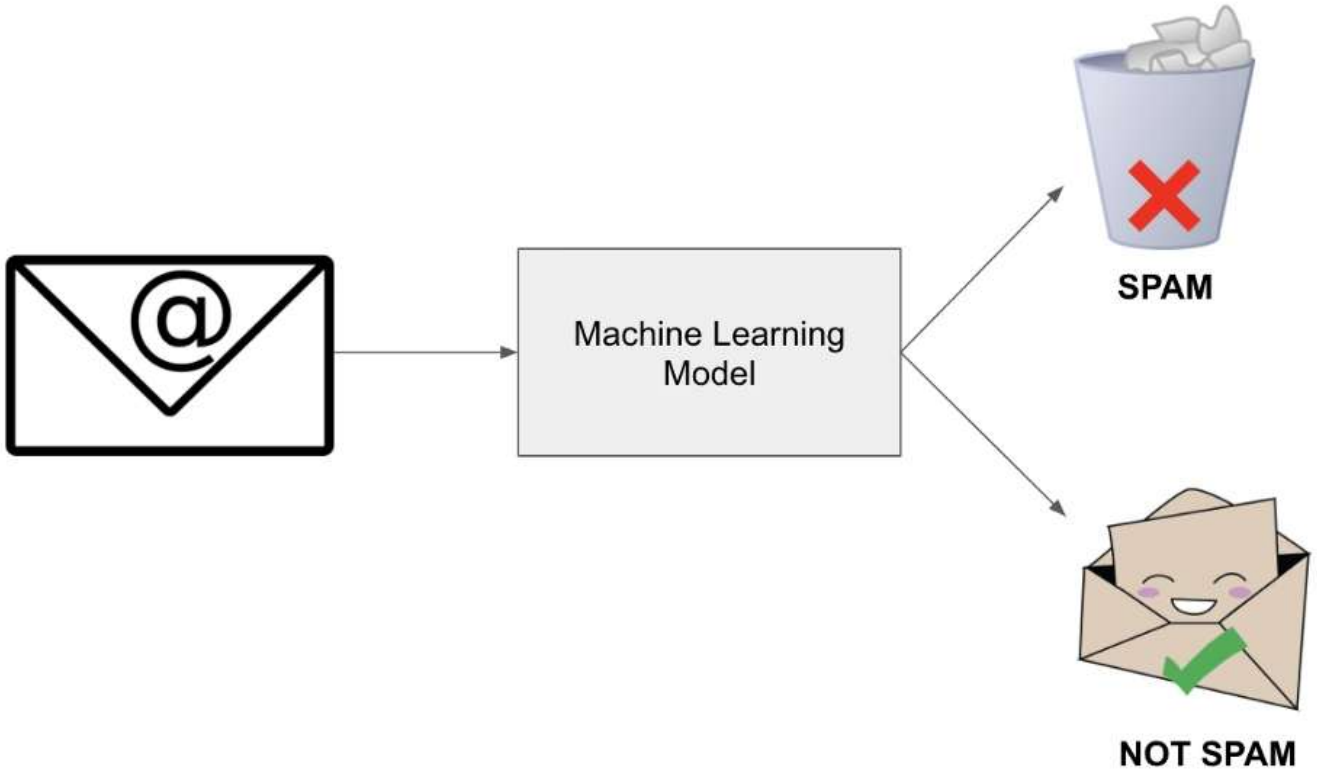
-Supervised learning2
logistic
ridge, lasso,
Regression (multinomial, glm)
decision tree, svm, smoothing(kernel), bayes

-Ensemble learning

# Supervised learning2

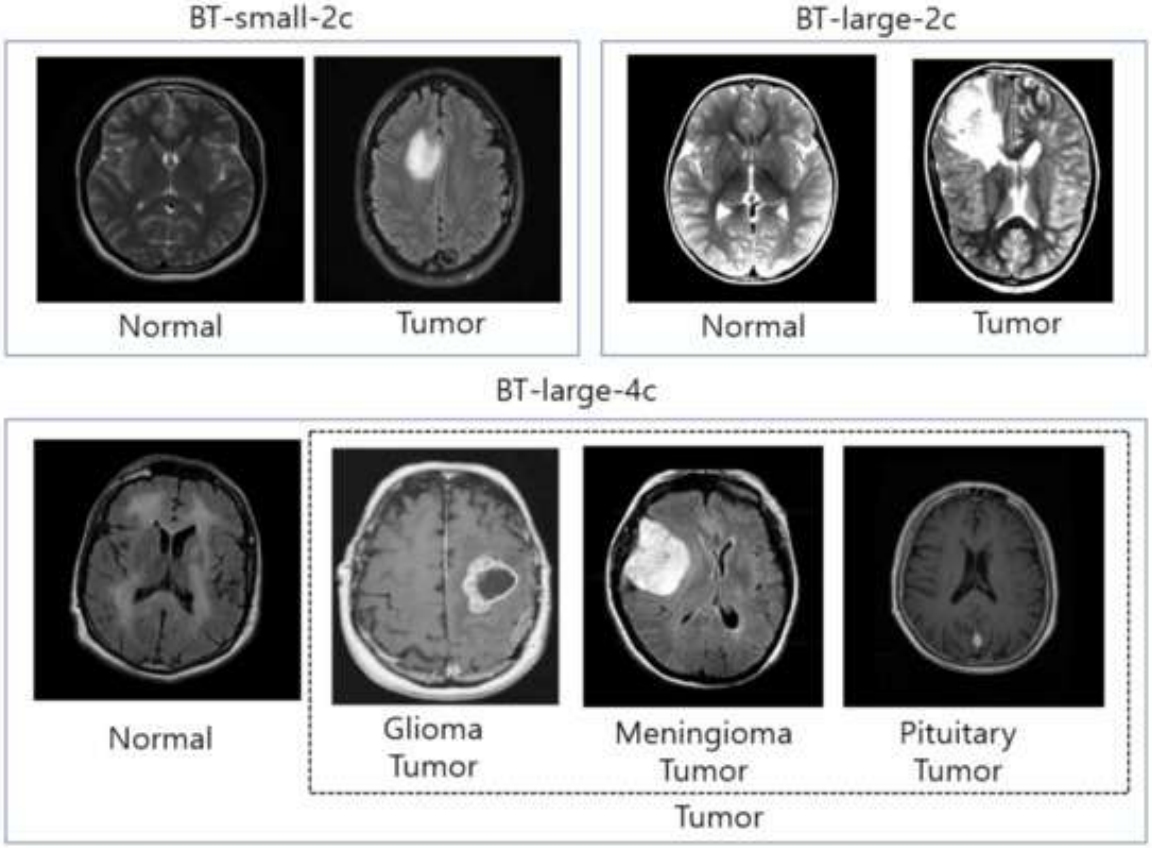
# Logistic regression

- Binary classification problem



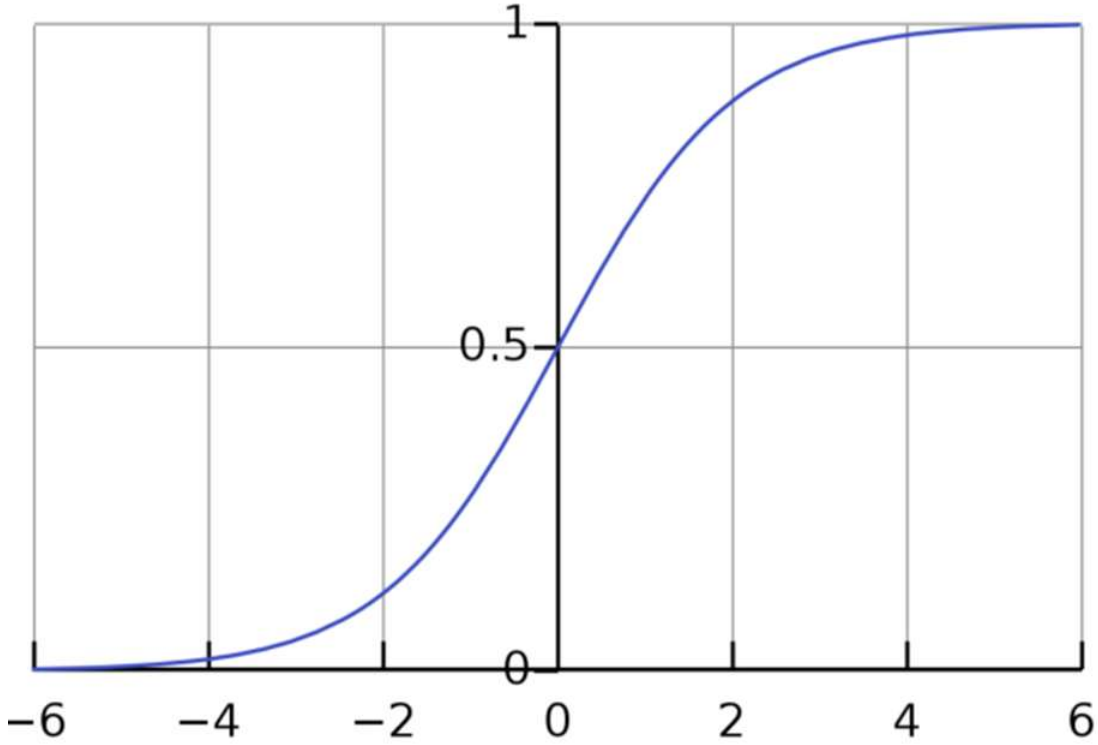
# Logistic regression

- Binary classification problem

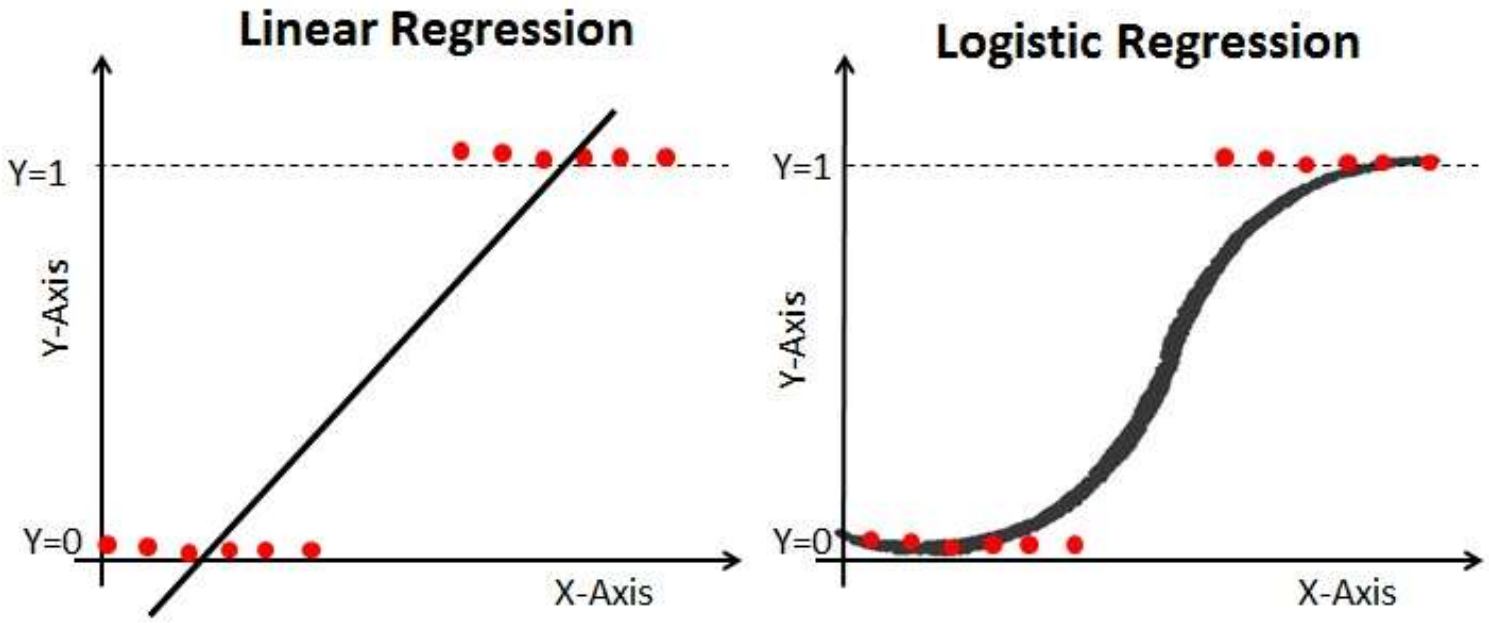


# Logistic regression

- $H(x) = Wx + b$
- $H(x) = G(Wx + b)$
- $G(z) = \frac{1}{1+e^{-z}}$



# Logistic regression vs linear regression




- Red data points

-Linear regression: gradual difference

-Logistic regression: dramatic difference →each to capture

# Logistic regression

- $H(x) = Wx + b$
- $H(x) = G(Wx + b)$
- $H(x) = \frac{1}{1+e^{WX}}$

# Logistic regression

- $H(x) = Wx + b$
- $H(x) = G(Wx + b)$
- $H(x) = \frac{1}{1+e^{WX}}$

- Cost function as linear regression??
- $cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}\left(H(x^i) - y^i\right)^2$

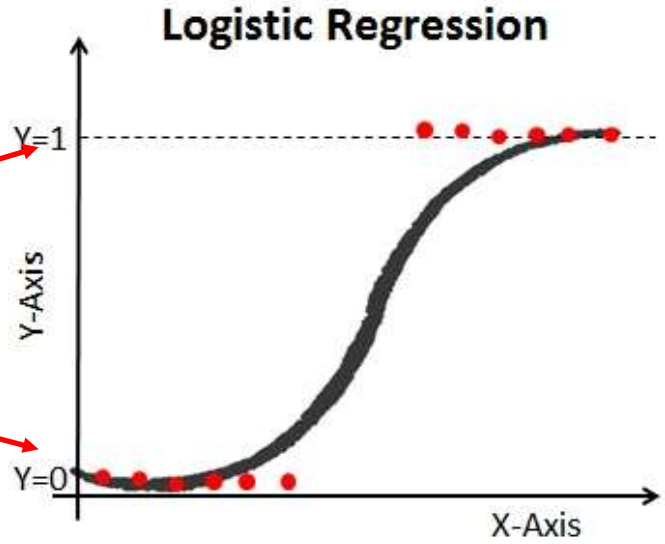
- Difference between two probability distribution

→ $H(P, Q) = -\sum P(x)\log(Q(x))$

# Logistic regression

- $cost(W) = \frac{1}{m}\sum_{i=1}^{m} c\left(H(x^i), y^i\right)$

- $c(H(x), y) = \begin{cases} -\log\left(H(x)\right): & y = 1 \\ -\log\left(1 - H(x)\right): & y = 0 \end{cases}$

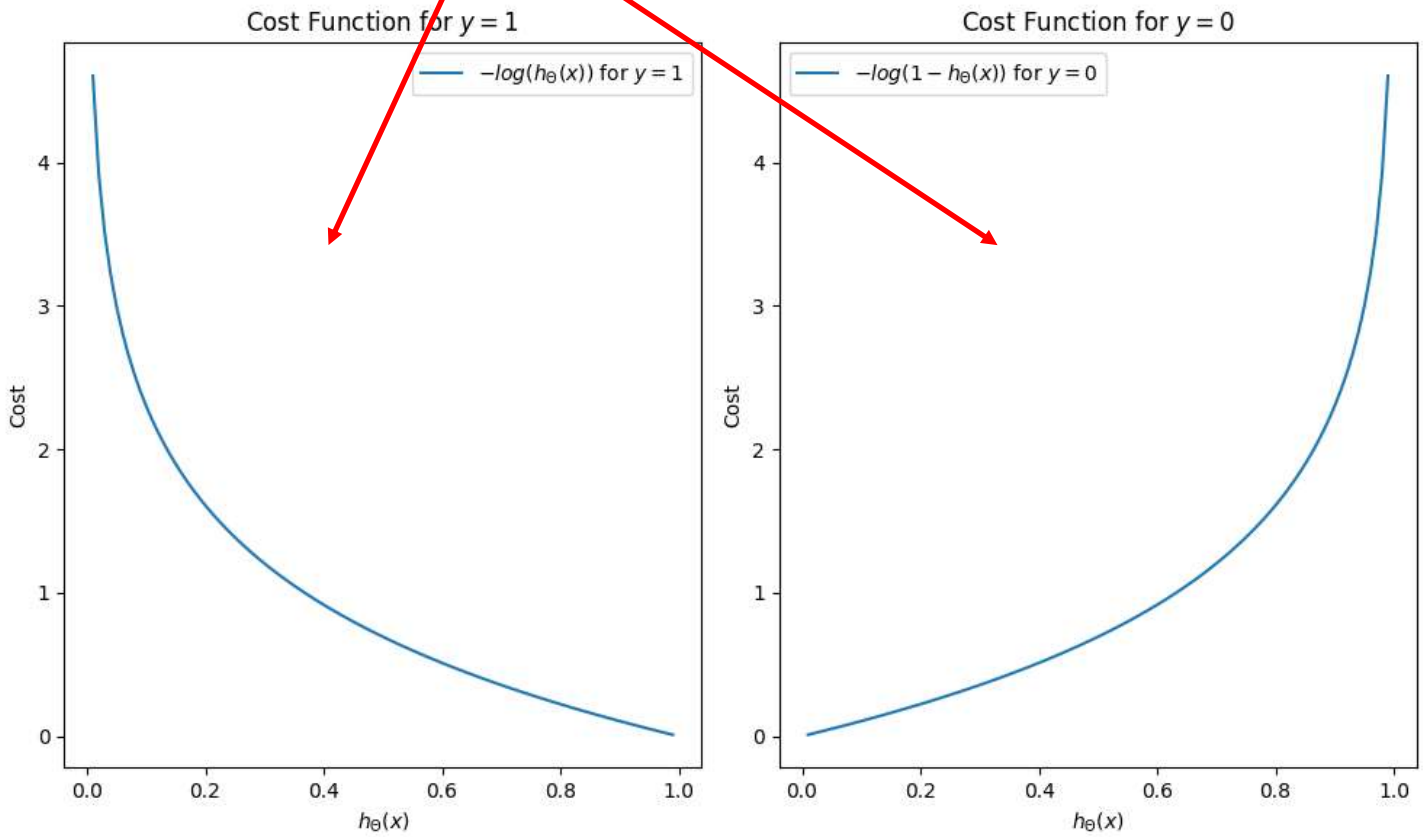



Q → y

- $c(H(x), y) = -ylog\left(H(x)\right) - (1 - y)\log(1 - H(x))$

# Logistic regression

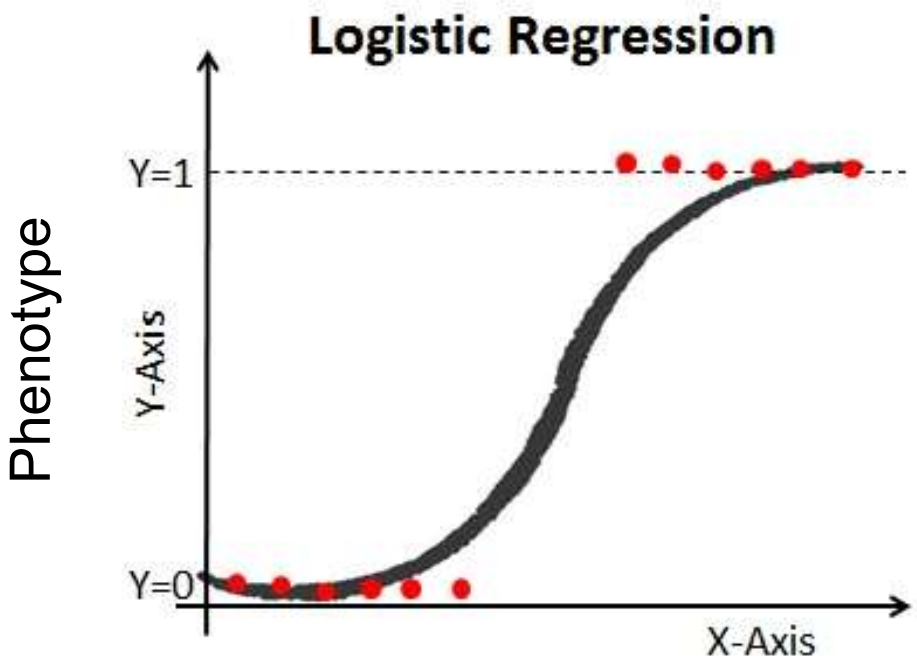
- $c(H(x), y) = \begin{cases} -\log(H(x)) : & y = 1 \\ -\log(1 - H(x)) : & y = 0 \end{cases}$



# Logistic regression

- Phenotype = Logistic regression (gn1 expression)

Ex): Cell type, drug-resistance, …




Gn1 expression → marker gene?

Multinomial classification

- $H(x) = \frac{1}{1+e^{WX}}$
- $Softmax(\hat{y}_i) = \frac{e^{\hat{y}_i}}{\sum e^{\hat{y}_i}}$

| | Binary | Multinomial |
|---|---|---|
| Hypothesis | $Sigmoid(WX)$ | $Softmax(WX)$ |
| Cost | $-ylog(H(x)) - (1-y)log(1-H(x))$ | $\sum -ylog(H(X))$ |

# Linear & Logisitc regression

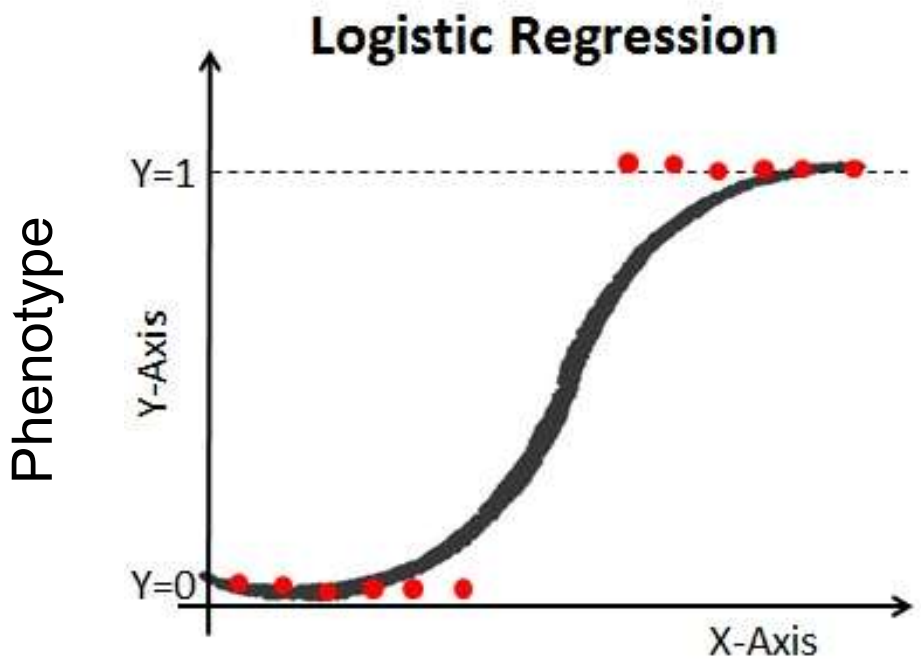
- Linear regression
- One variable: $H(x) = Wx$
- Multinomial: $H(X) =$XW
- Cost function: $\frac{1}{m}\sum_{i=1}^{m}\left(Wx^i - y^i\right)^2$ (MSE: mean-square error)

Logistic regression
- Binary classification: Sigmoid(WX)
- Multinomial: Softmax(WX)
- Cost function: $\sum -ylog(H(x))$ → Cross entropy in the future

# Logistic regression

- Phenotype = Logistic regression (gn1, gn2, gn3 expression)

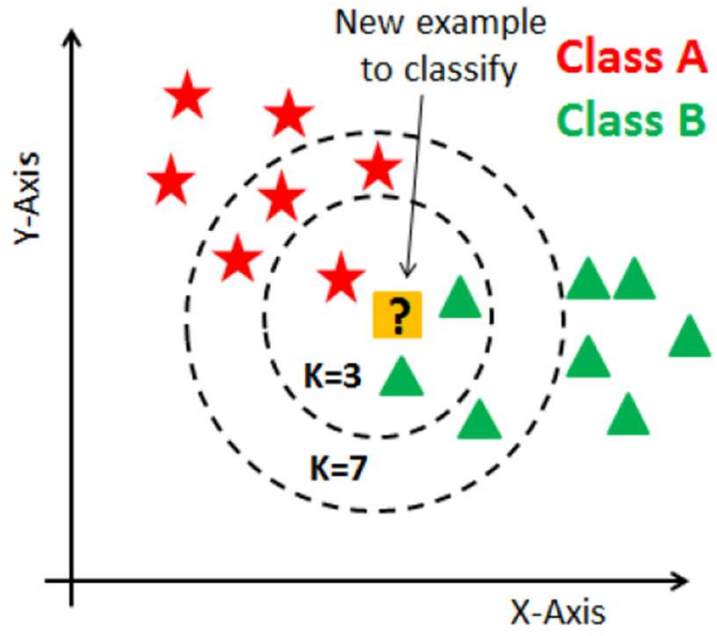
Ex): Cell type, drug-resistance, …




Gn1, gn2, gn3 expression → marker gene?

## ❖ *K*-nearest neighbor classifier (KNN)

- KNN doesn't learn a discriminative function the training data: **lazy learner**
- Learning is performed by memorizing the training dataset: **Instance-based learner**
- Calculate the distance between every training examples.
- **Choose the number *k* of neighbors**.
- Find the *k* nearest neighbors of the sample we want to classify.
- Assign the class label **by majority voting**.
- Voting multiple neighbors helps increase resistance to noise. Odd value of *k* normally used to avoid ties.

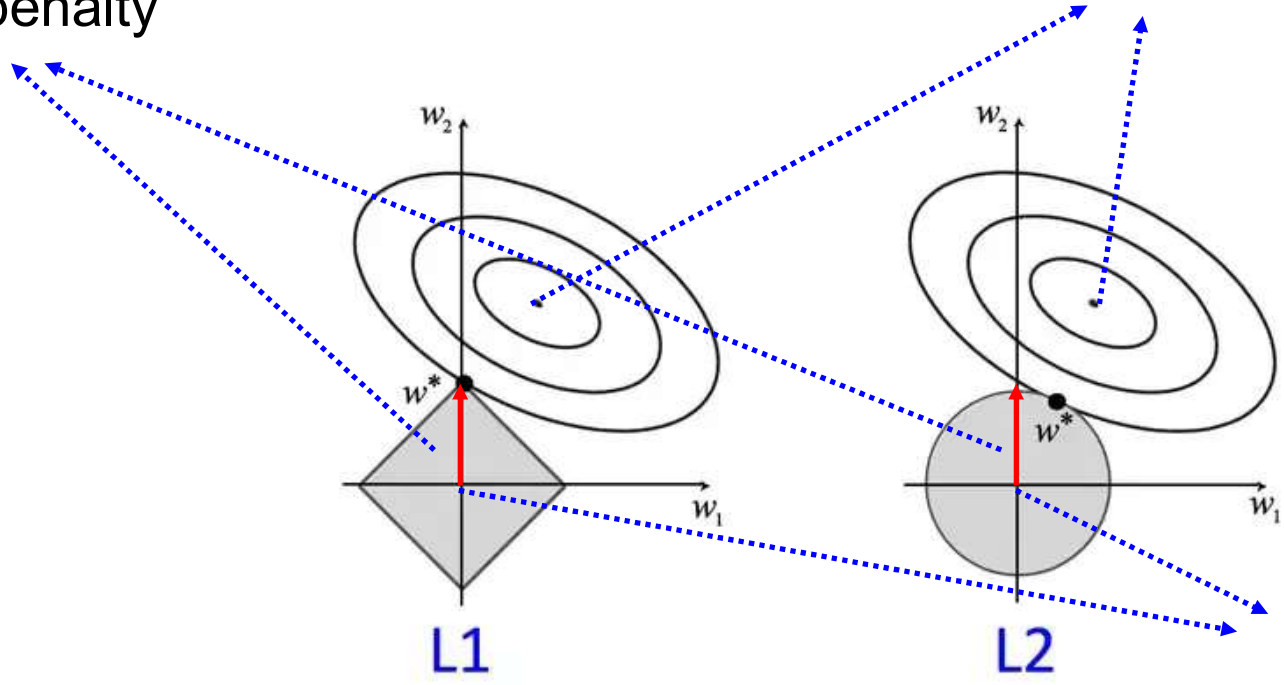



- KNN classifier has been applied to many bioinformatics problems such as prediction of specific kinds of protein, protein modification, protein secondary structure, and etc.

# Regularization

Minimum loss for the test set
→ Can lead to over-fitting

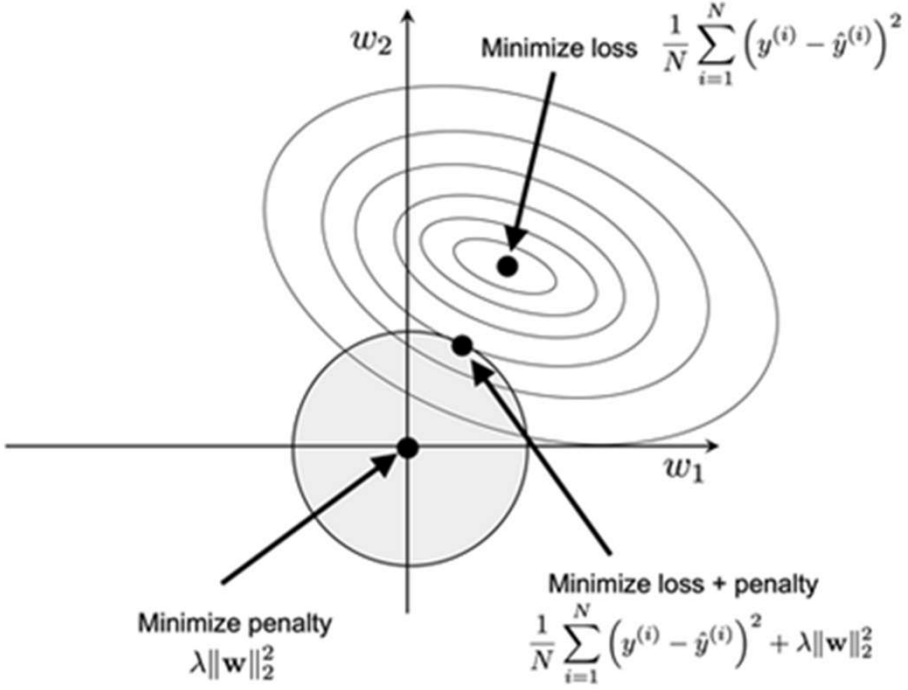
-$\lambda$: penalty




Minimum regularization

-Regularization prevents the “weight” from increasing too much

# Regularization




-Regularization prevents the “weight” from increasing too much
→ Now this model can also predict the data from “left-bottom”

## Linear regression → penalty to avoid over-fitting

-$H(x)=Wx+b$

## Ridge (L2) regression

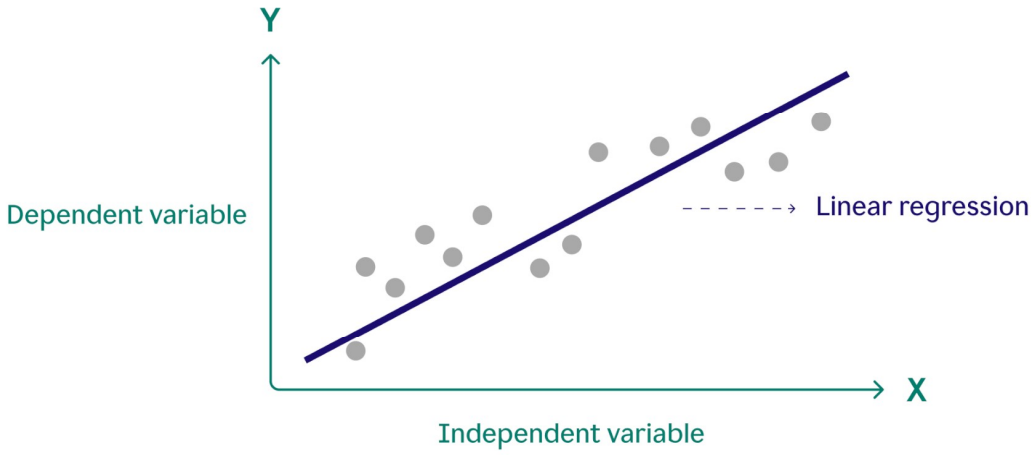
-L2-normalization of each parameter (W) from linear regression

## Lasso

-L1-normalization of each parameter (W) from linear regression

## Elastic net regression

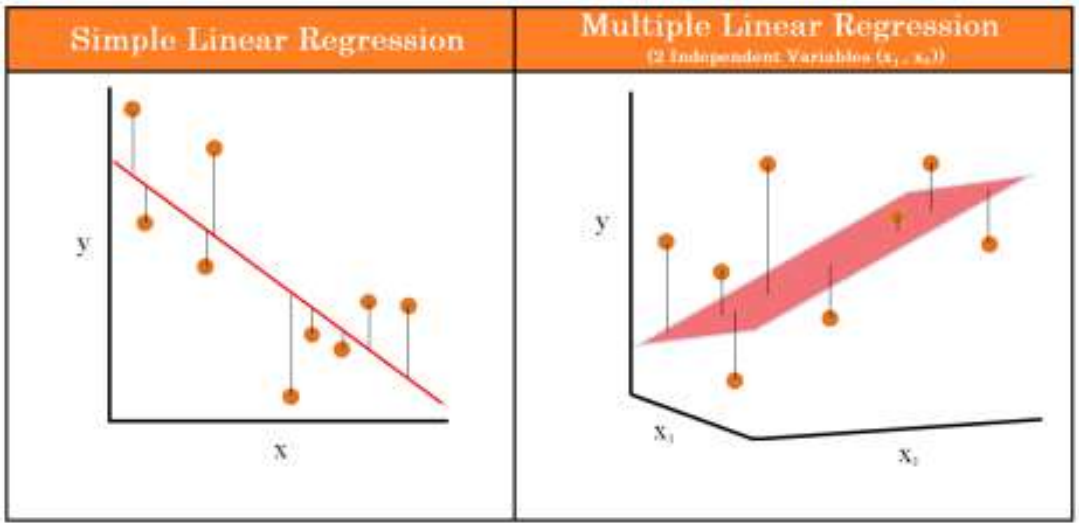
-Combined Ridge + Lasso



# Multiple linear regression

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_p X_{ip} + \epsilon_i$$

-Multiple variables with different weight




# Multinomial logistic regression

$$P(Y = j) = \frac{\mathrm{Exp}(\beta_j' x_i)}{1 + \sum_{k=1}^{J} \mathrm{Exp}(\beta_k' x_i)} \text{ for } j = 1, \ldots, J \text{ and}$$

# Generalized linear model

-g function (link function)
→E(y): expectation of dependent variable
G(E(y)) ~ linear regression

$$g(E(y_i)) = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}$$

m = E(y)

| | |
|---|---|
| Normal | Identity : $g(m) = m$ |
| Gamma | Negative Inverse : $g(m) = -\frac{1}{m}$<br>Log : $g(m) = ln(m)$ |
| Poisson | Log : $g(m) = ln(m)$ |
| Binomial | Logit : $g(m) = ln(\frac{m}{1-m})$ |

Example)
Count data: non-negative → log function
Skewed data: gamma or log
Binary data → binomial distribution

# Generalized linear model

-Fixed effect vs random effect

$$Y = \beta_0 + \beta_1 \text{Age} + \beta_2 \text{Treatment} + u_{\text{Hospital}} + \epsilon$$

β: fixed effect for (co)variables (Age, Treatment)
u: random effect for variable (hospital, group, school ?)

Ex) Score = β*studytime

| | |
|---|---|
| 학생 A | +5 |
| 학생 B | +5 |
| 학생 C | +5 |
| 학생 D | +5 |

β0: fixed intercept
e: residual (error term)

# Generalized linear model

-Fixed effect vs random effect

$$Y = \beta_0 + \beta_1 \text{Age} + \beta_2 \text{Treatment} + u_{\text{Hospital}} + \epsilon$$

β: fixed effect for (co)variables (Age, Treatment)
u: random effect for variable (hospital, group, school ?)
→ Different schools have different learning rate
→ Rather than predicting each β for each school,

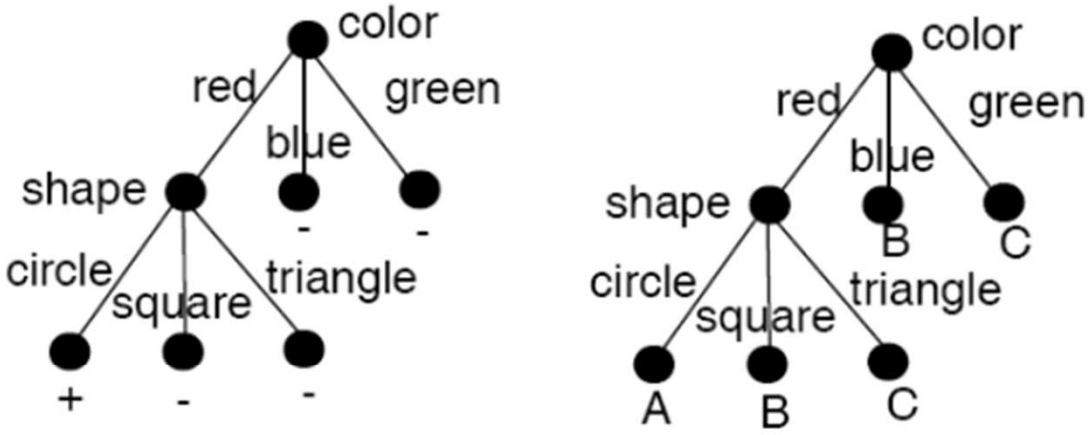
$$b_j \sim N(0, \sigma_b^2)$$

σ: deviation of effect (for each school)
→ Each school has different effect
Higher σ → higher variation across schools

β0: fixed intercept
e: residual (error term)

## ❖ Decision Tree

- A machine learning model that **breaks down the data with a series of questions**.
- In particular, we ask **questions *most* helpful for dividing different classes**.
- We need to **find the feature *most* helpful for dividing the data**.
- Each *internal* ***node*** of decision tree denotes a *test on a* ***feature***, each ***branch*** represents the ***outcome*** *of the test*, ***leaf nodes*** represent ***classes*** *or* ***class distributions***.

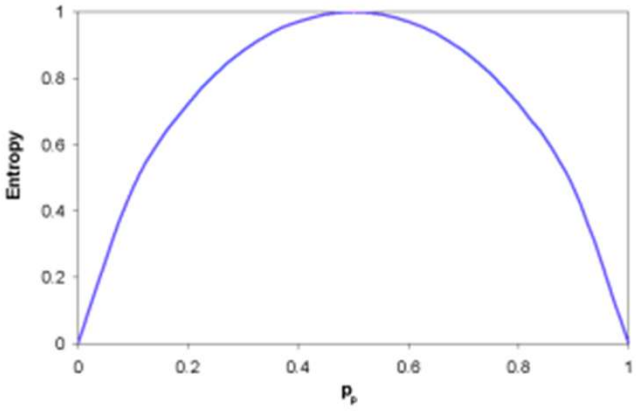



Classification rules can be rewritten as follows:

**red** and **circle** is **A**.
**blue** is **B**.
**red** and **square** is **B**.
**green** is **C**.
**red** and **triangle** is **C**.

- **Information Gain**
  - **Entropy** (impurity, disorder) of a set of examples, S, relative to a binary classification is:

$$Entropy(S) = -p_+ log_2(p_+) - p_- log_2(p_-)$$




  - For multiple classes problems with $c$ classes, entropy generalizes to:

$$Entropy(S) = \sum_{i=1}^{c} - p_i \log_2(p_i)$$

where $p_i$ is proportion of class $i$ examples in $S$.

  - **Information gain** of a feature is the **expected reduction in entropy** caused **by partitioning on this attribute**:

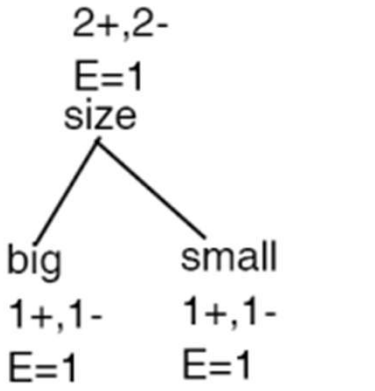
$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Where $S_v$ is the subset of $S$ for which attribute A has value $v$ and the entropy of the partitioned data is calculated by weighting the entropy of each partition by it's size relative to the original set.

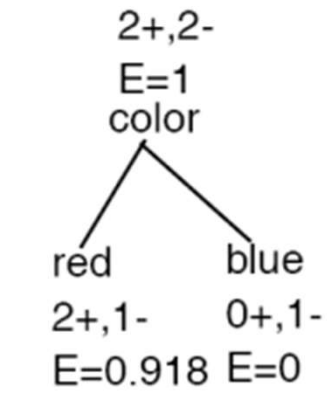
- The lower the impurity at the child nodes, the larger the information gain.

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

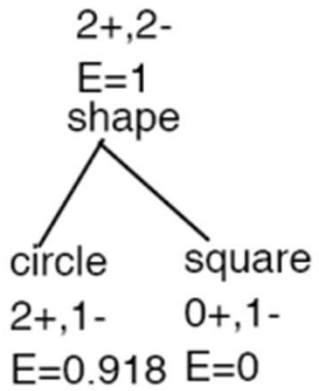
- Example:
  big, red, circle: +
  small, red, circle: +
  small, red, square: -
  big, blue, circle: -




Gain=1- ((.5)1 + (.5)1)
= 0




Gain=1- ((3/4)0.918 +
(1/4)0)
=0.311



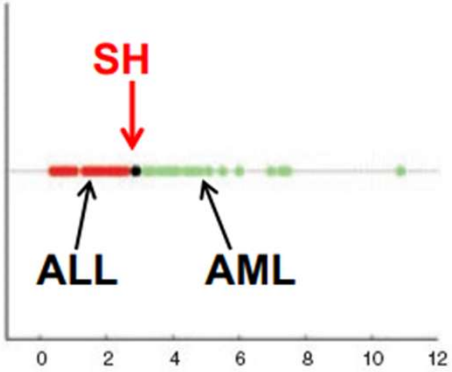

Gain=1- ((3/4)0.918 +
(1/4)0)
=0.311

# Support Vector Machine (SVM)

1. Separating hyperplane
2. Maximum-margin hyperplane
3. Support vector machine (SVM)
4. Soft margin
5. Kernel function

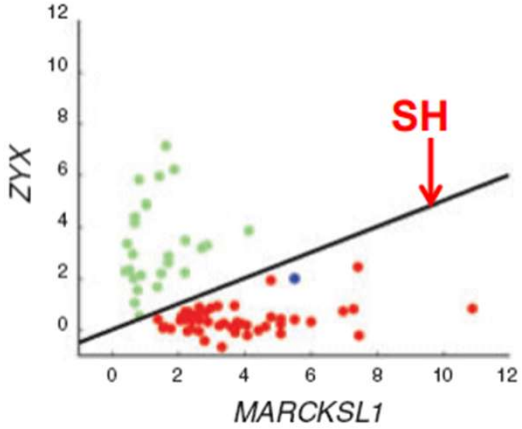
# Support Vector Machine (SVM)

**(1) Separating hyperplane (SH)**



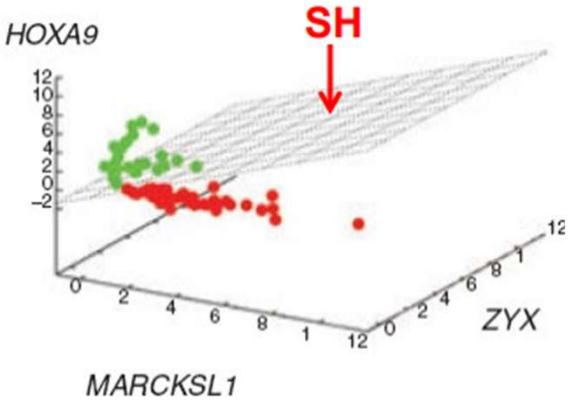

1D data space.

SH is a point (0 D).




2D data space.

SH is a line (1 D).



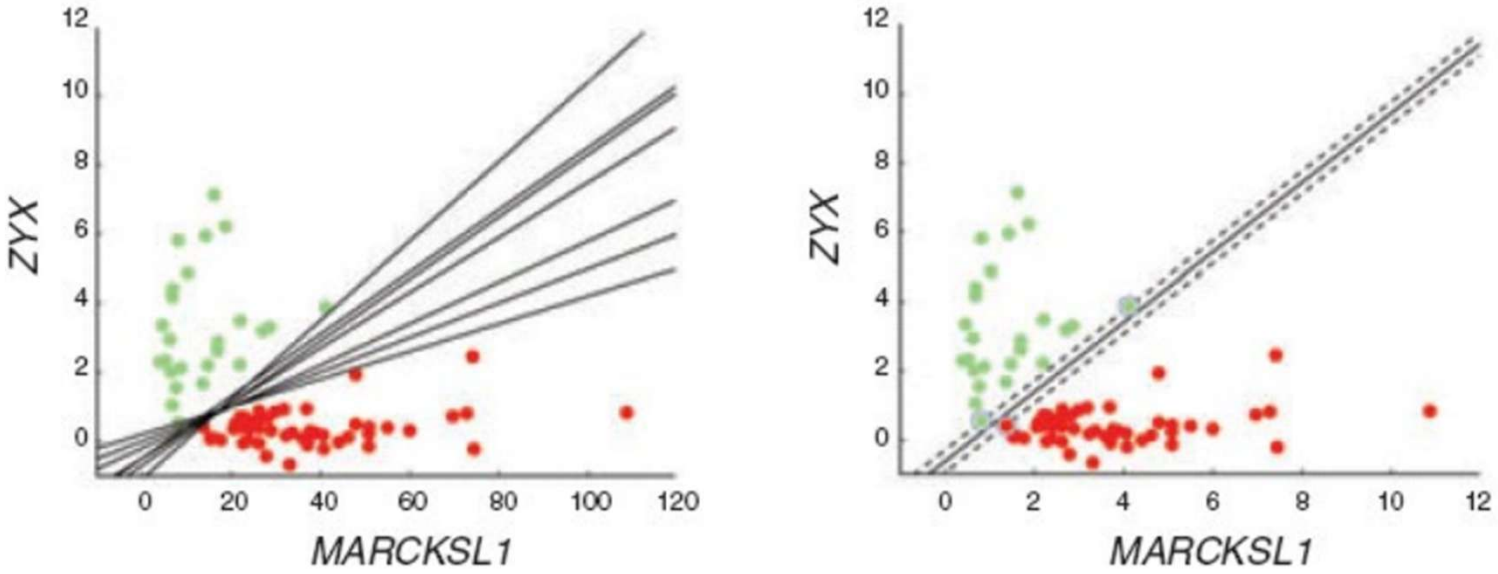

3D data space.

SH is a plane (2 D).

- We can extrapolate this procedure mathematically to higher dimensions.

# Support Vector Machine (SVM)

**(2) Maximum-margin hyperplane**

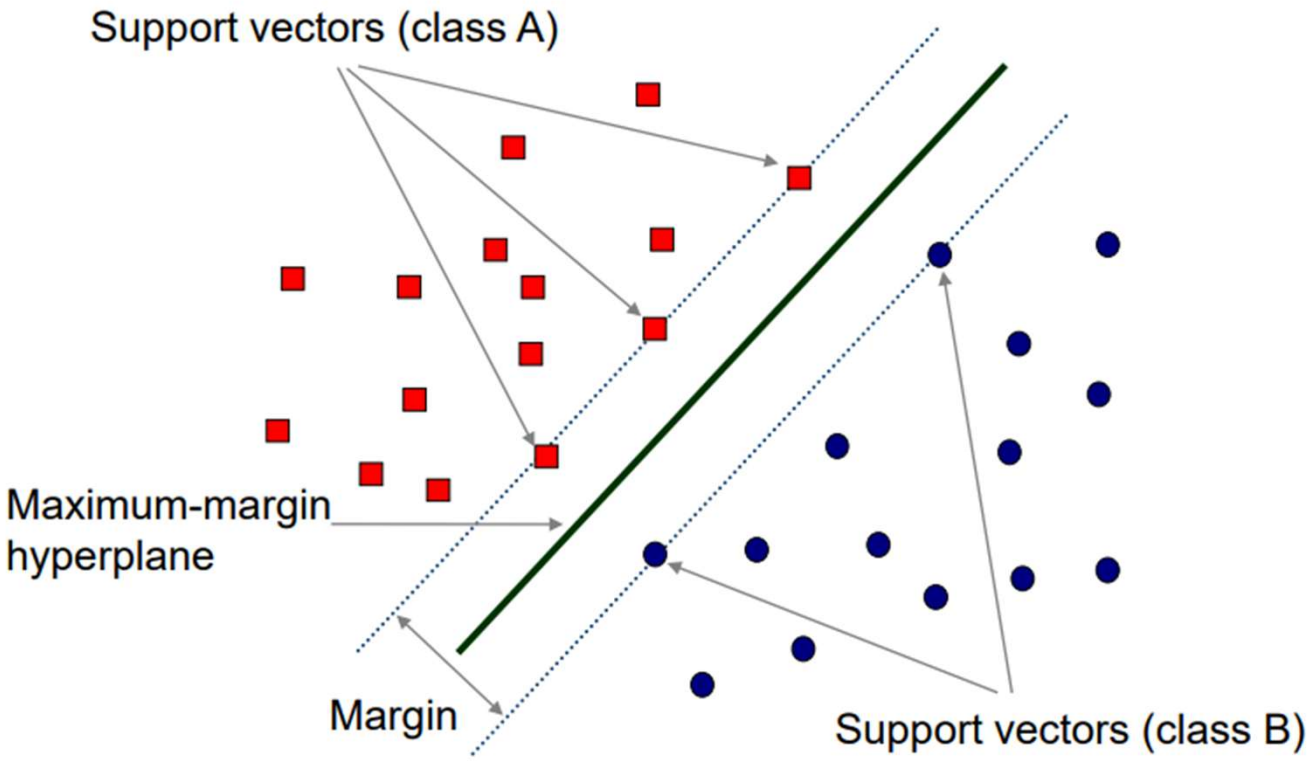



- There are many hyperplanes that can separate ALL and AML. The SVM is different from other hyperplane-based classifications by how the hyperplane is selected.
- The SVM defines the distance from the separating hyperplane to the nearest data point (expression profile) as the margin of the hyperplane to select the maximum-margin separating hyperplane.

# Support Vector Machine (SVM)

**(3) Support Vector Machine (SVM)**

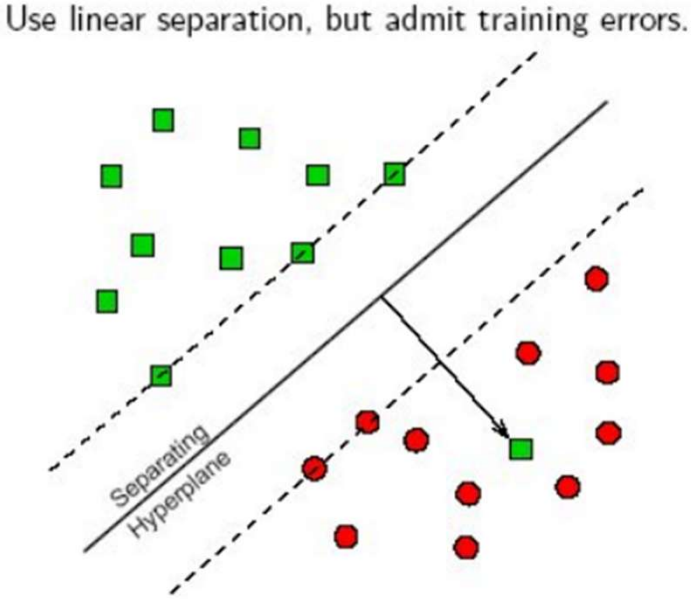
- A set of data points that are nearest to the separating hyperplane (i.e., support vectors) are Support Vector Machine (SVM).



# Support Vector Machine (SVM)

## (4) Soft margin

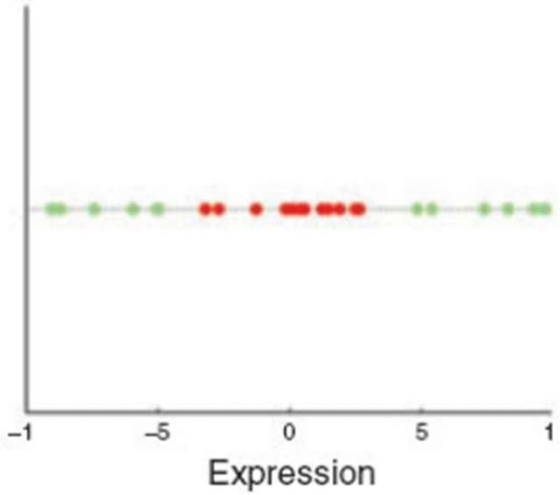
- Ideally an SVM analysis should produce a hyperplane that completely separates the feature vectors into two non-overlapping groups. However, ***perfect separation may not be possible***.
- **To allow some flexibility** in separating the categories, SVM models have a ***cost parameter***, ***C***, that controls the ***trade off between allowing training errors and forcing rigid margins***.



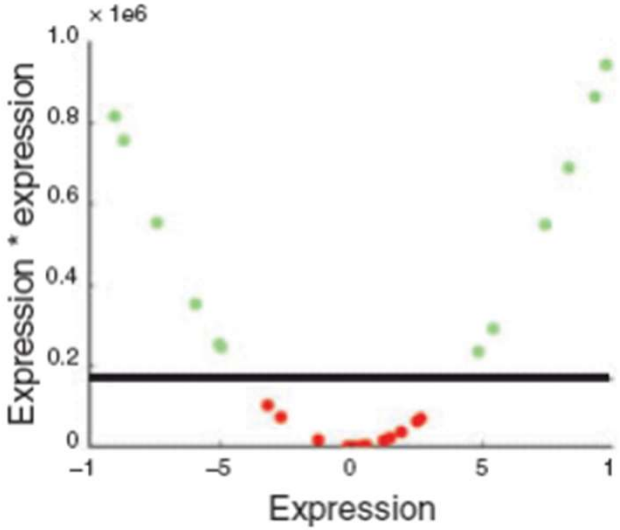

- It creates a ***soft margin*** that ***permits some misclassifications***. Of course, we ***don't want the SVM to allow for too many misclassifications***. Increasing the value of *C* increases the cost of misclassifying points and forces the creation of a more accurate model that may not generalize well.

# Support Vector Machine (SVM)

## (5) Kernel function




No single point can separate the two classes. A soft margin cannot help.




Thus original expression values are simply squared to add an additional dimension to the data.

# Kernel-based smoothing

-Noisy data → smoothing → better modeling
-weighted mean for each data point from neighbors
-kernel function: ex) gaussian
-bandwidth: low → noisy, high → smoother

$$K_{h_\lambda}(X_0, X) = D\left(\frac{\|X - X_0\|}{h_\lambda(X_0)}\right)$$

where:

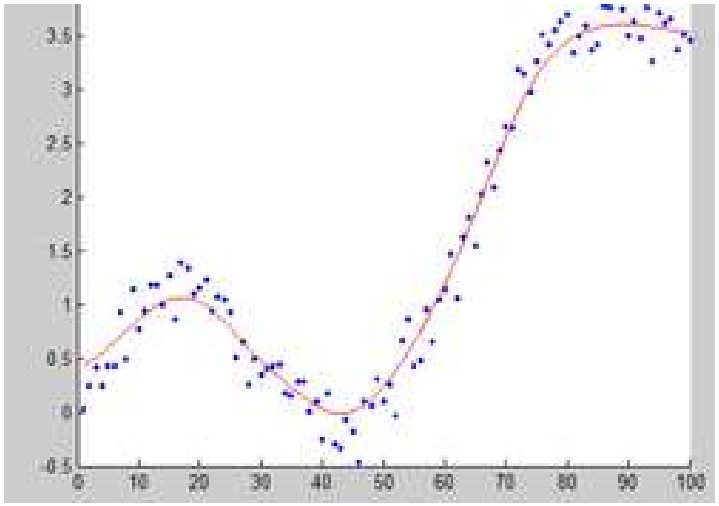
- $X, X_0 \in \mathbb{R}^p$
- $\|\cdot\|$ is the Euclidean norm
- $h_\lambda(X_0)$ is a parameter (kernel radius)

# Kernel-based smoothing

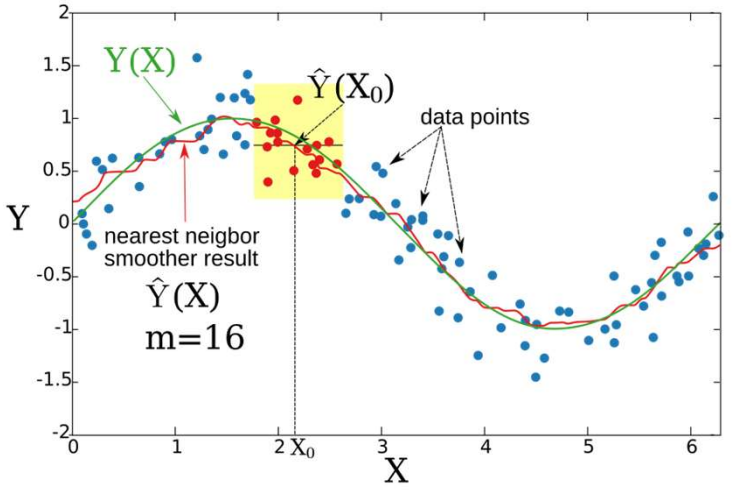
-Gaussian kernel smoother

$$K(x^*, x_i) = \exp\left(-\frac{(x^* - x_i)^2}{2b^2}\right)$$

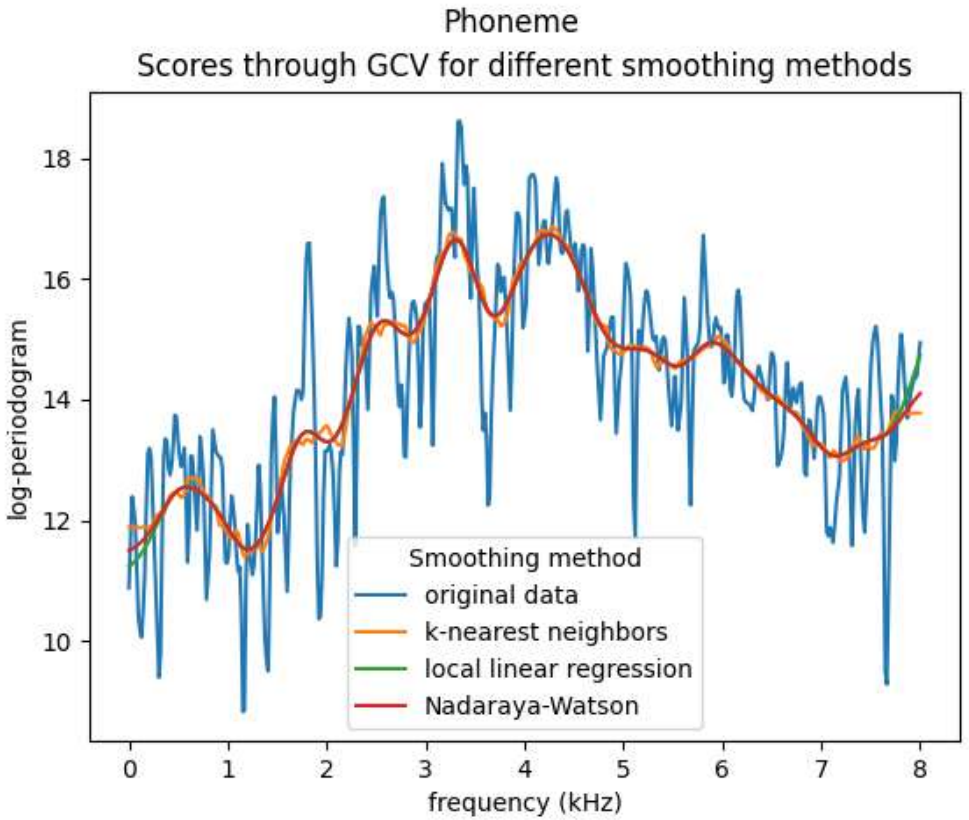
Here, b is the length scale for the input space.

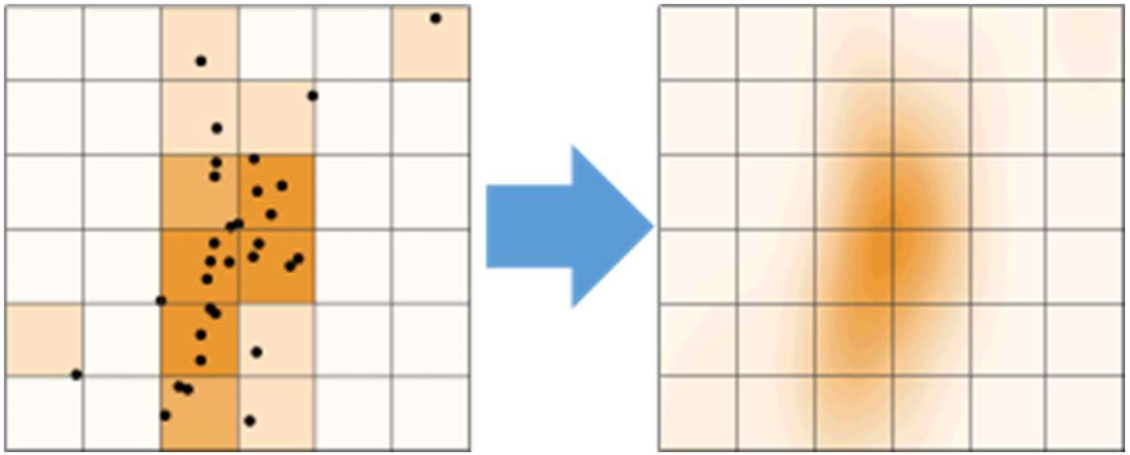



-Nearest neighbor smoother
K-nearest neighbor averaging
Y(x): average value of kNN



# Kernal-based smoothing



# Kernel Density Estimation (KDE)

-Density estimation by kernel function

Kernel function: symmetric by 0, integral = 1, non-negative

Ex) Gaussian, Epanechnikov, uniform

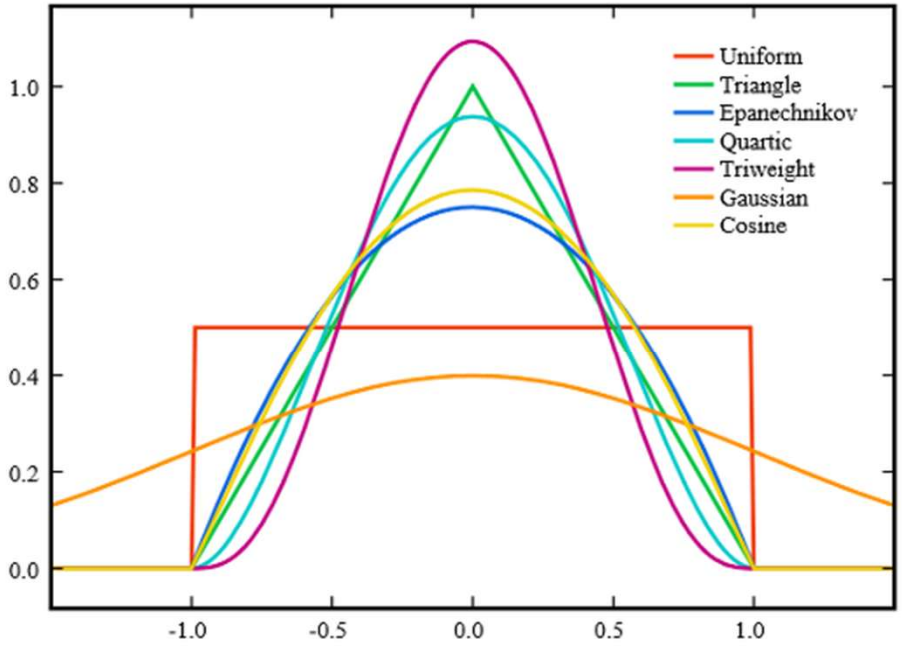
$$\int_{-\infty}^{\infty} K(u)du = 1$$

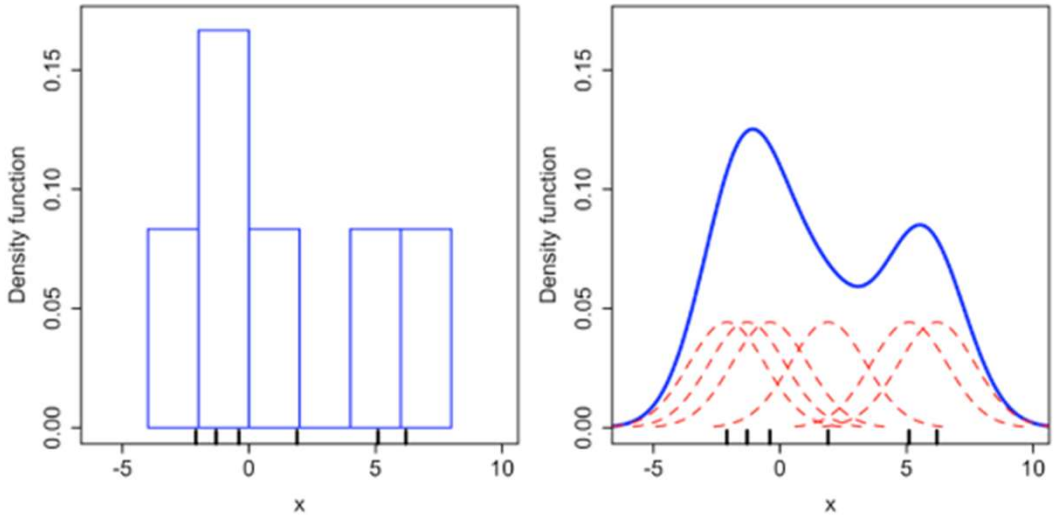
$$K(u) = K(-u),\ K(u) \nleq 0,\ \forall u$$

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h})$$

h (bandwidth): low → sharp shape, high → smooth

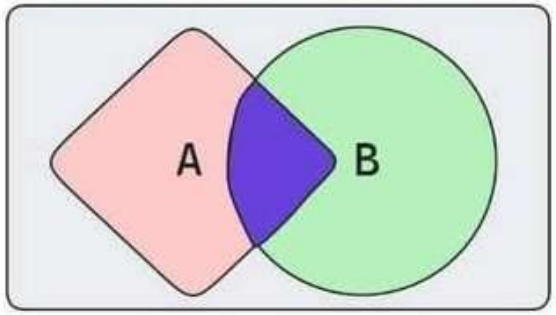
Accumulation of K (kernel) for each data point



# Bayes theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$




$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Conditional probability

**LIKELIHOOD**
The probability of "B" being True, given "A" is True

**PRIOR**
The probability "A" being True. This is the knowledge.

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)}$$

**POSTERIOR**
The probability of "A" being True, given "B" is True

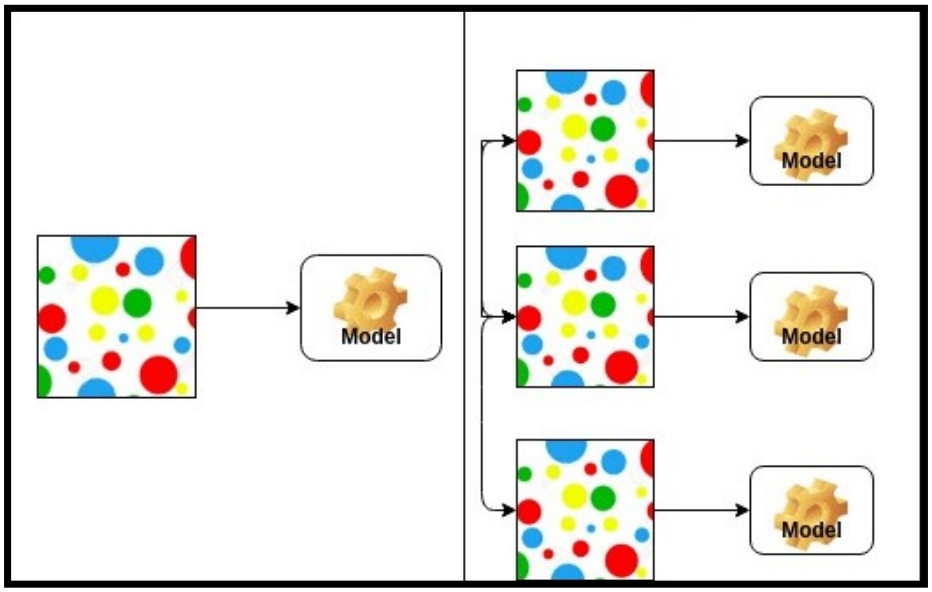
**MARGINALIZATION**
The probability "B" being True.

Likelihood: $\frac{P(L|D)/P(\neg L|D)}{P(L)/P(\neg L)}$

Positive / negative (in the data)
Positive / negative (goldstandard)

# Ensemble learning

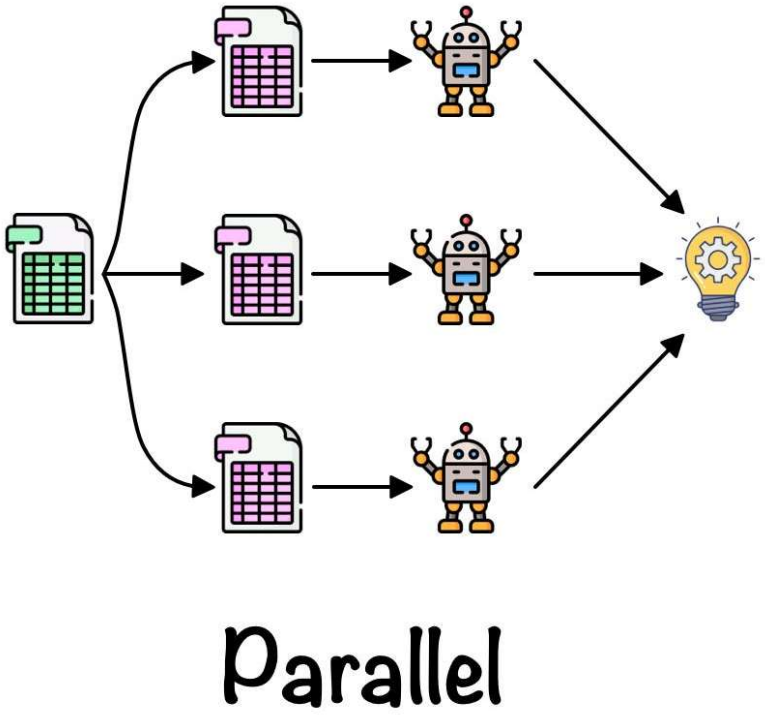
# Ensemble learning




- Single classifier → multiple classifier → Improve performance

# Ensemble learning

- Majority voting




- Single classifier might be over-fitted
- Not robust enough
- Maybe single classifier may work for a certain classifier

-Majority voting
Use various models →

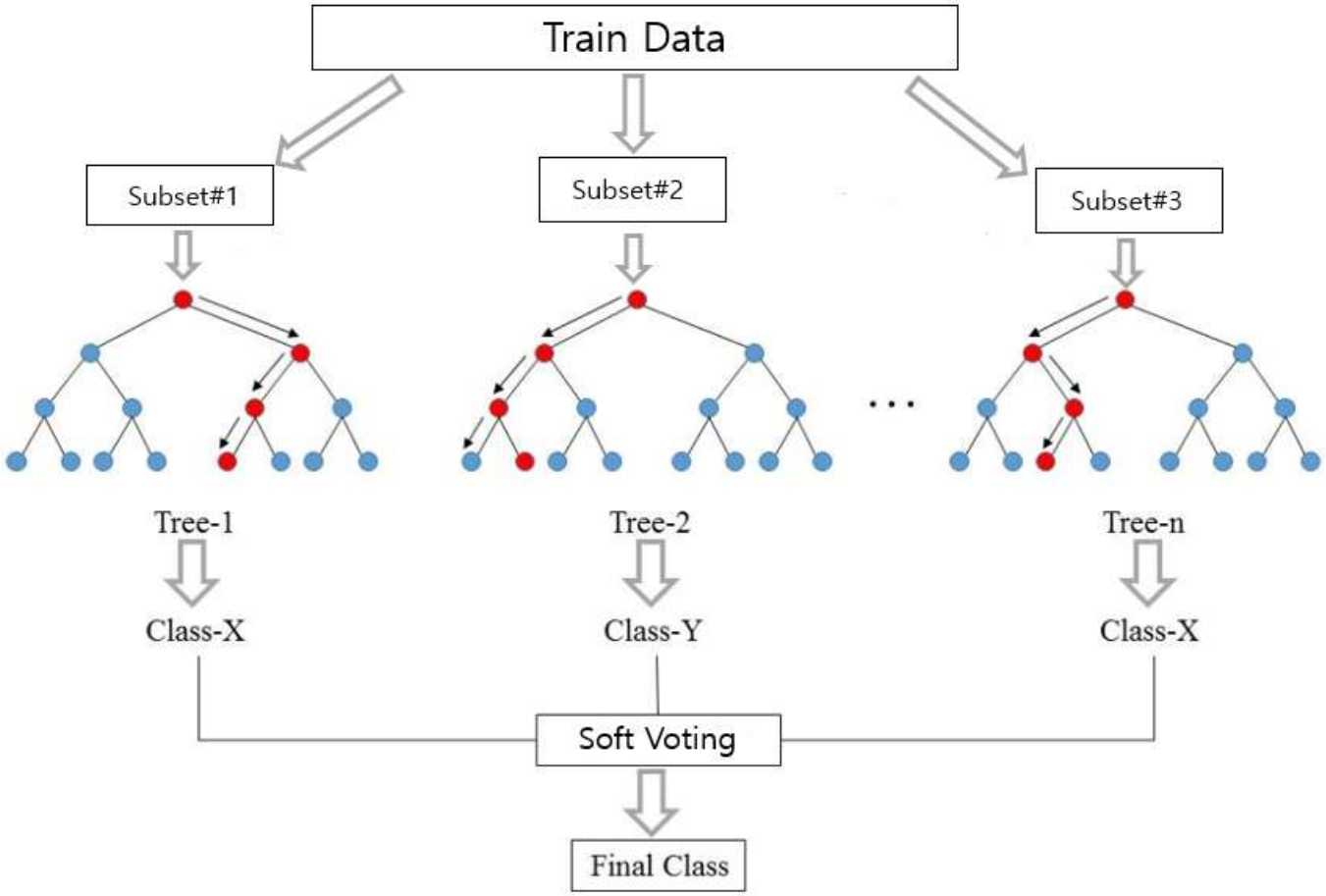
| sample1 | model1 | model2 | model3 | model … |
|---|---|---|---|---|
| Yes | o | o | o | x |
| No | x | x | x | o |

| sample1 | model1 | model2 | model3 | model … |
|---|---|---|---|---|
| Yes | x | x | o | x |
| No | o | o | x | o |

# Ensemble learning

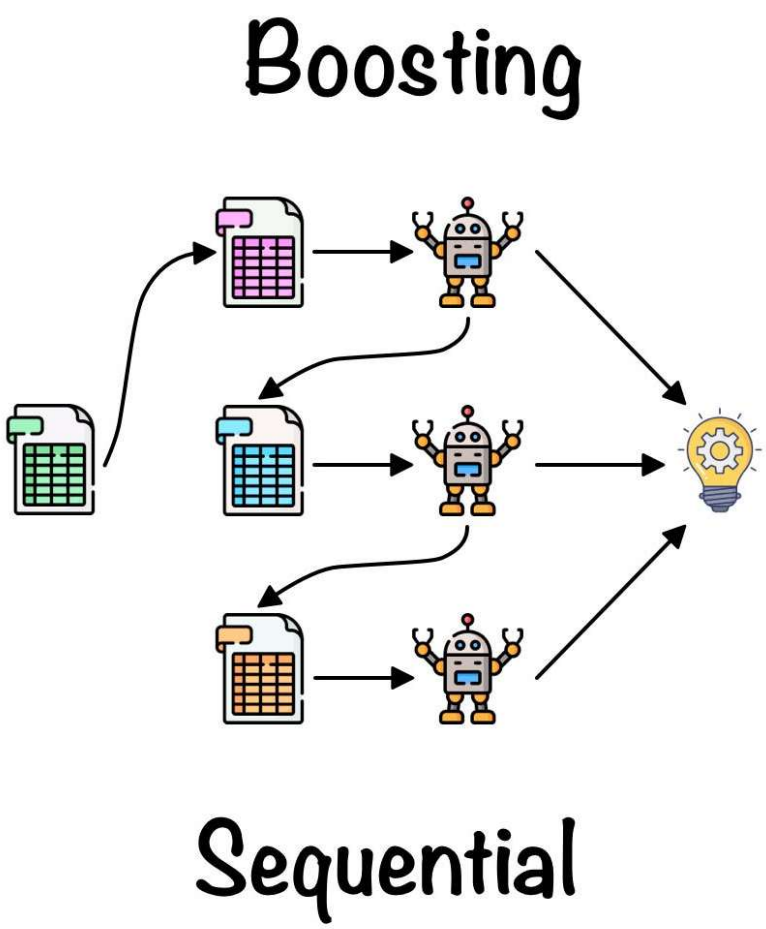
- Bagging (ex: Random forest)

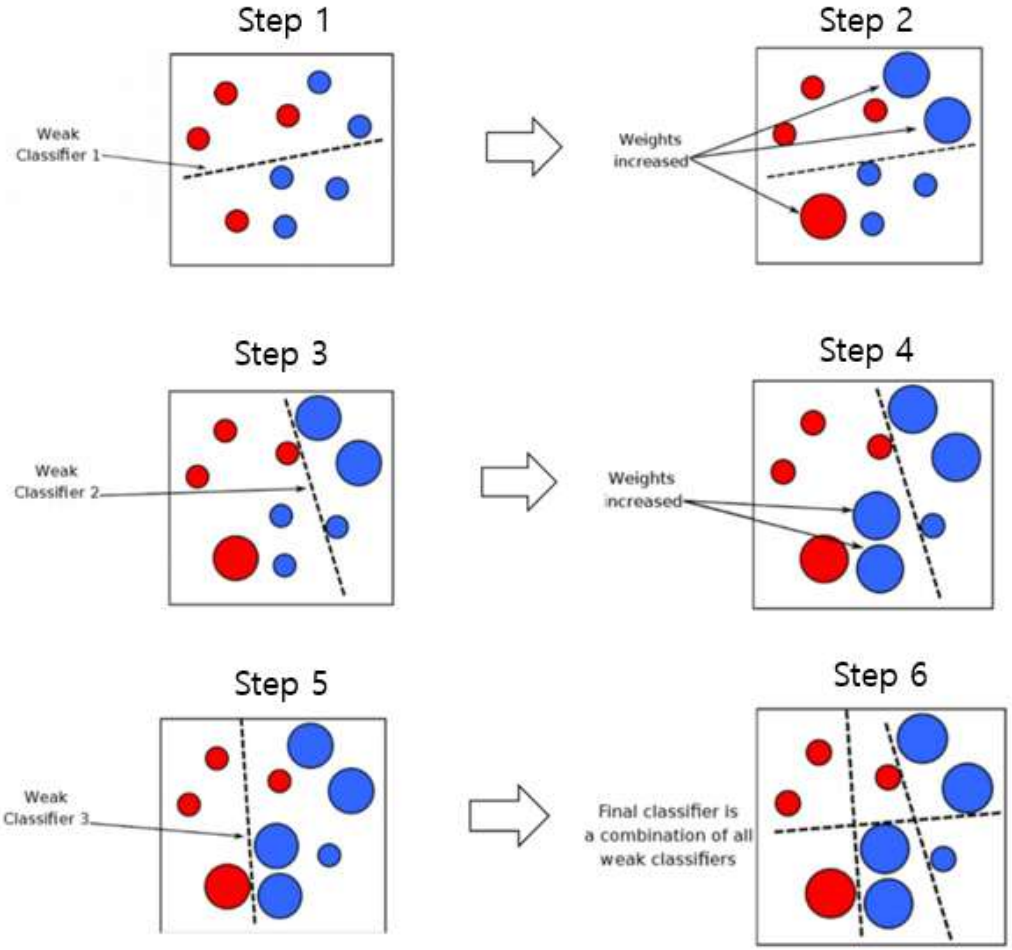
→ Down-sampling → average score of each prediction



# Ensemble learning

- Boosting: update weight for each training (starts with weak classifier)



# Ensemble learning

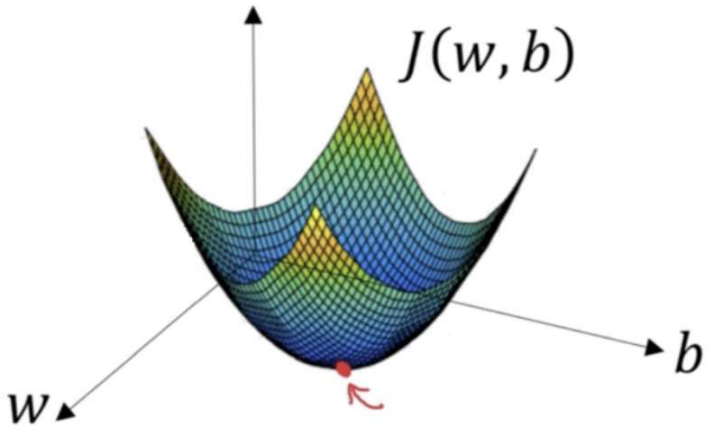
- Boosting: weight update

-AdaBoost

-Gradient Boosting machine: uses gradient

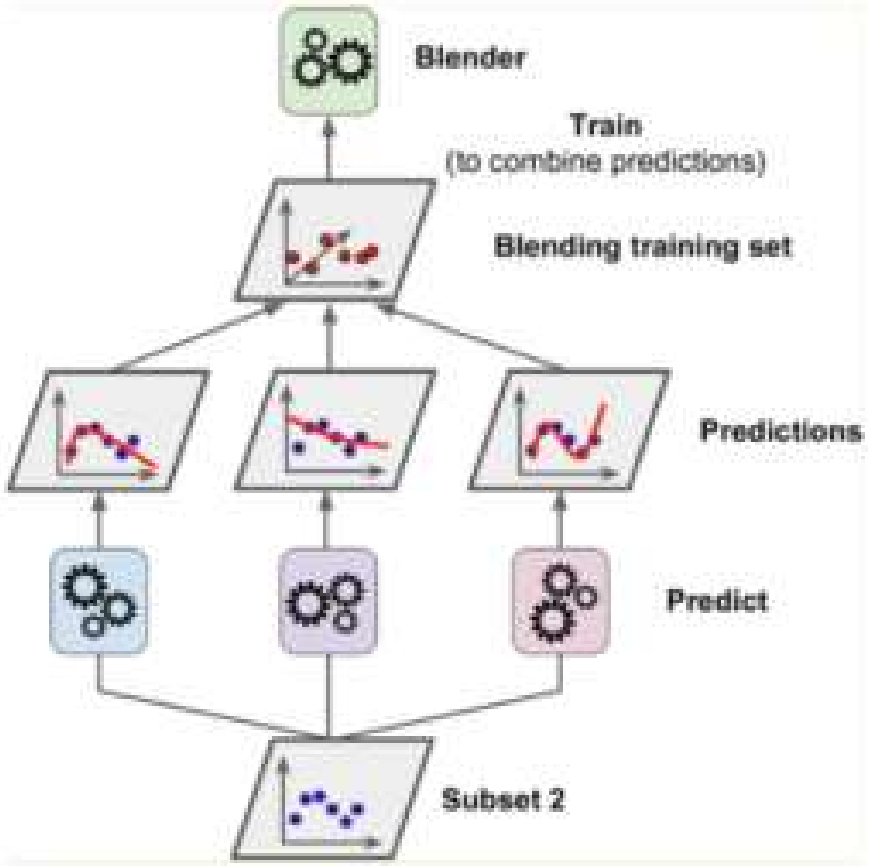
$W := W - \alpha \frac{\partial}{\partial W} * cost(W)$

-XGBoost: faster, improve over-fitting



# Ensemble learning

- Blending




- Training data → multiple classifier
- Score for each prediction
- Use that score as an input for final classifier

| Celltype | Model1 | Model2 | Model3 | Model4 |
|---|---|---|---|---|
| B cell | 3 | 1 | 2 | 6 |
| T cell | 2 | 4 | 1 | 8 |
| Epithelium | 5 | 9 | 5 | 4 |
| Stroma | 0 | 0 | 4 | 2 |

| | Model1_B | Model1_T | ... | Model4_S |
|---|---|---|---|---|
| Input | 3 | 2 | ... | 2 |

Epithelium