Statistic 3

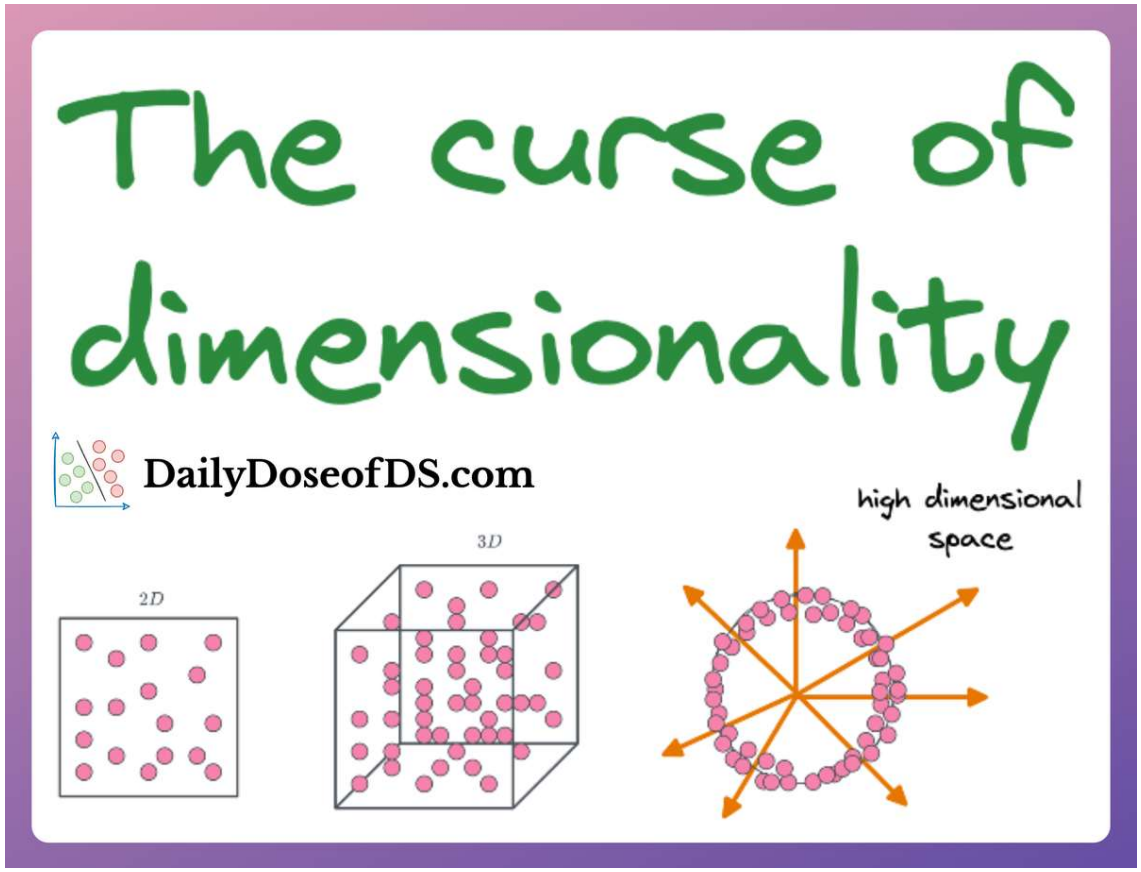
-Dimension reduction
Dimension reduction (pca, cca, pcoa, lda)

-Supervised learning
k-fold validation
bootstrapping

cf: Scikit (python)
Classifier background + over-fitting
Linear

Dimension reduction

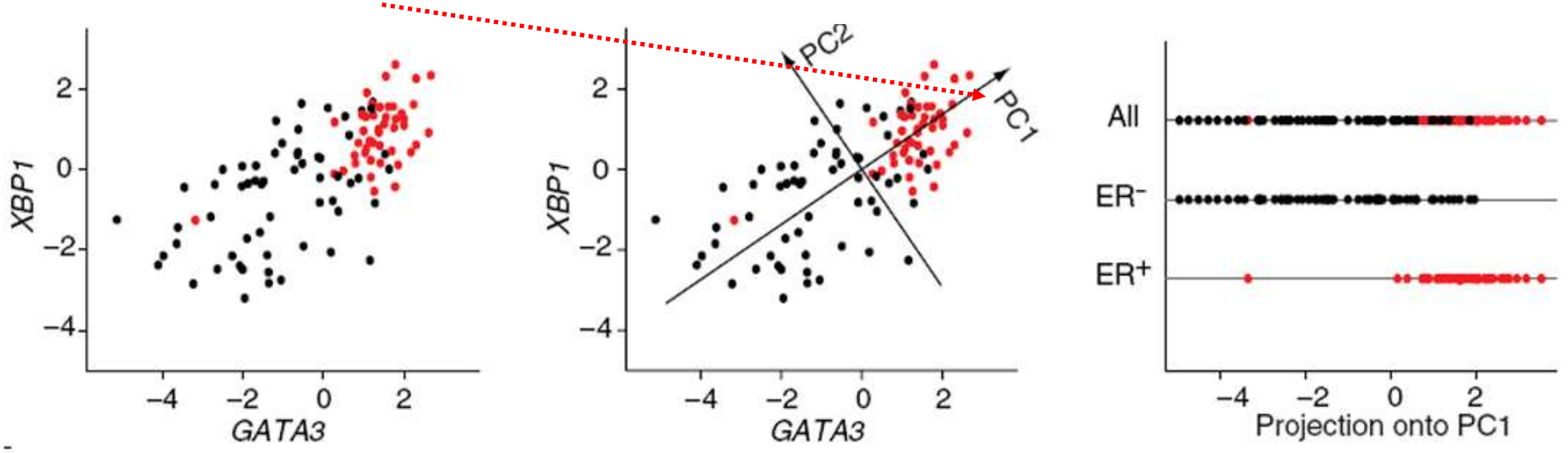
Curse of dimension



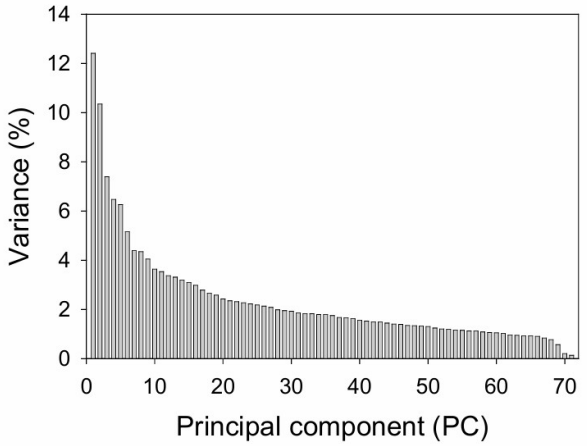
Ex: gene feature

PCA (principal component analysis)

- Find a PC axis to maximize the variance of the data → Distribute samples to maximize the variance



- Number of PC == Number of features or <= sample size
→ Usually, we have more features (~20k)
→ Each PCs: explain the variance of the data
→ Using only a few PCs: Dimension reduction
Advantage: among confounding effects, noise, we can solely capture the biological difference
(or meaningful information)
+ data representation & visualization (20K genes → 2 PCs)



PCOA (Principal Coordinate Analysis)

-Similarity-based

Distance matrix: D

$$D^{(2)} = (d_{ij}^2)$$

$$J = I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$$

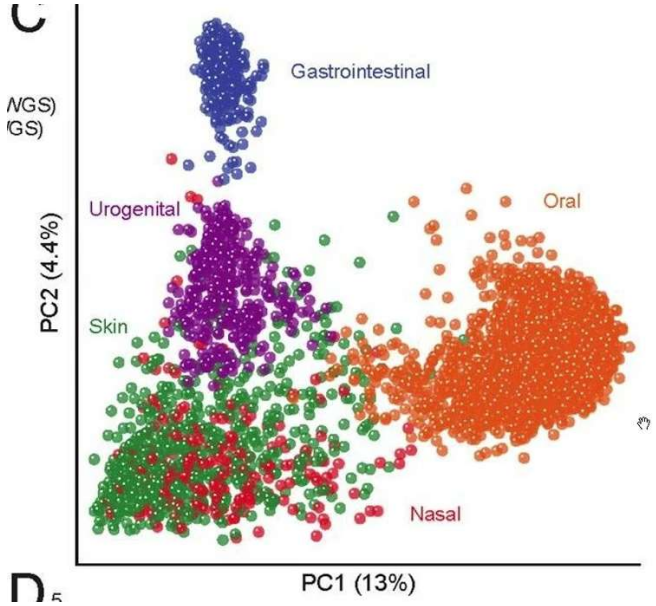
$$B = -\frac{1}{2} J\, D^{(2)}\, J$$

$$B = XX^\top = V\Lambda V^\top$$

$$X = V\Lambda^{1/2}$$

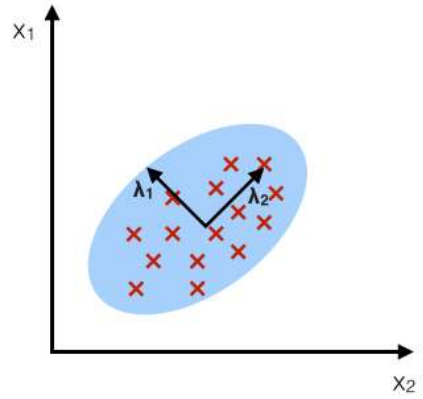
PCA (Eigendecomposition)

Microbiome: species similarity

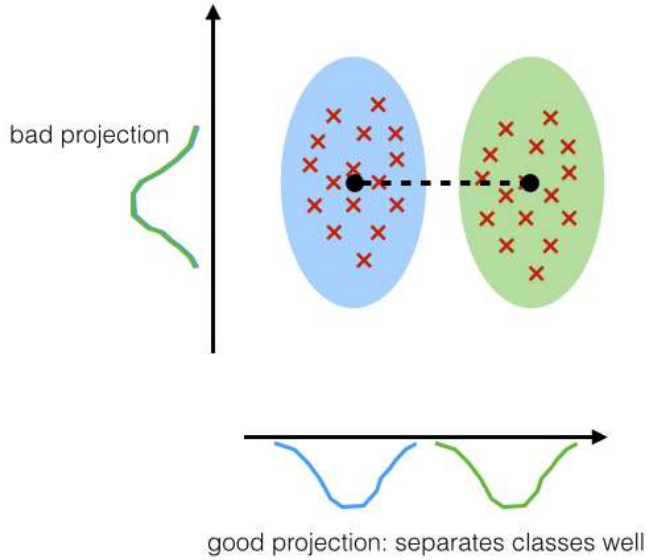


LDA (Linear Discriminant Analysis)

PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation



-Maximize the variance across memberships
-Minimize the variance within each membership

CCA (Canonical Correlation Analysis)

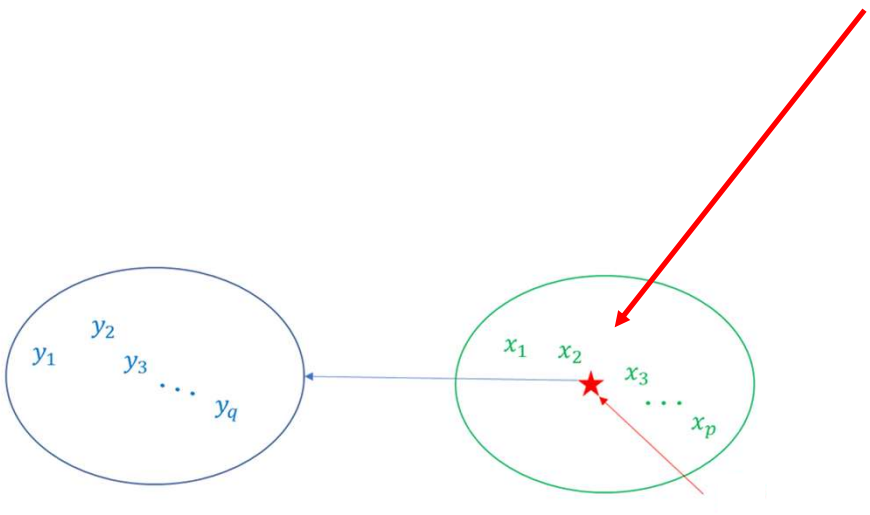
*Canonical Correlation Analysis (CCA)

$$\bar{x} = \sum a_i x_i$$

$$\bar{y} = \sum b_j y_j$$

- x,y: gene expression for each group
- Linear combination for each group

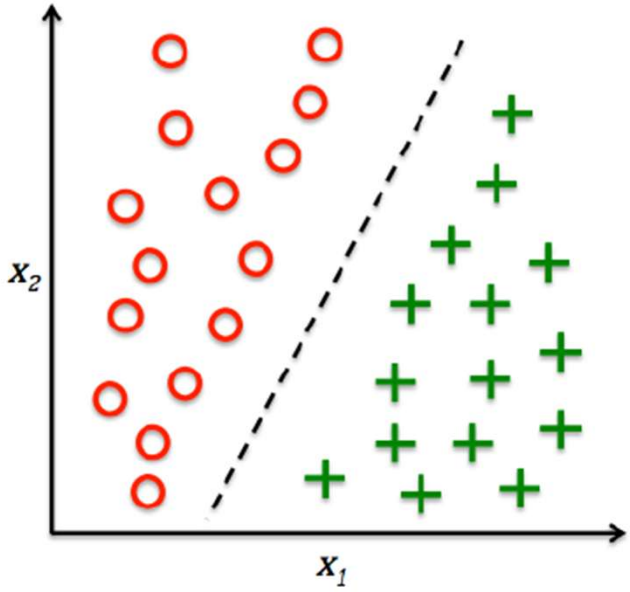
→ Maximize the correlation between $\bar{x}$ & $\bar{y}$



- **Classification for predicting class labels**

  - **Goal:**

    - To predict the categorical class labels of new instances based on past observations
    - Class labels are discrete, unordered values : can be understood as *group membership* of instances



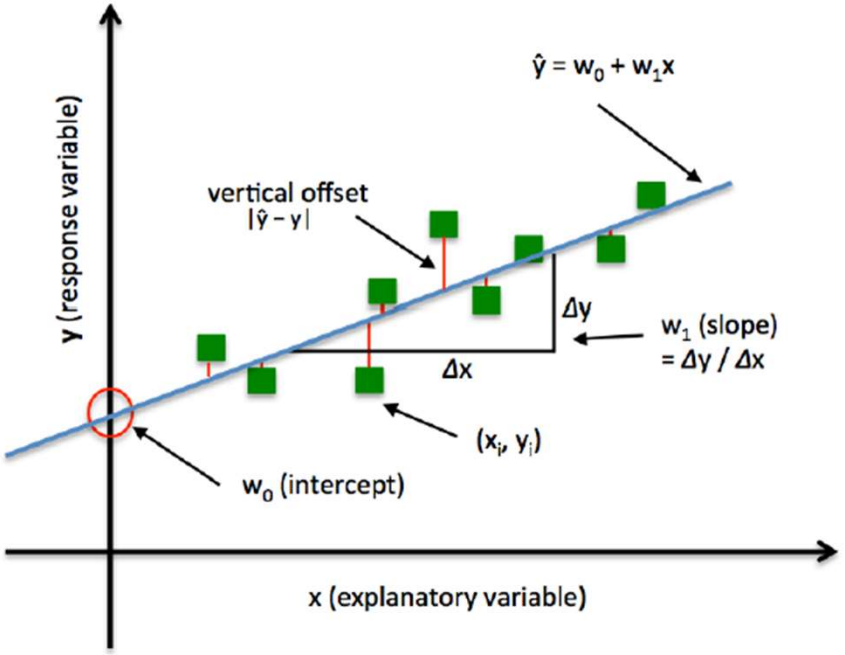
  - **Types of classification**

    - **Binary classification**: distinguish between two possible classes (e.g. spam and non-spam emails)
    - **Multi-class classification**: distinguish amongst multiple classes (e.g. handwritten digits from 0 to 9)

Supervised learning

- **Regression for predicting continuous outcomes**

  - To **predict continuous outcome**
  - We try to find a **relationship** between ***dependent variable*** (response variable) and of one or more ***independent variables*** (explanatory variables).
  - The **parameters** are estimated as to give a "***best fit***" of the data. Most commonly the best fit is evaluated by using the "*least squares*" method.
  - Regression could be *linear regressions* or *non-linear regressions*.



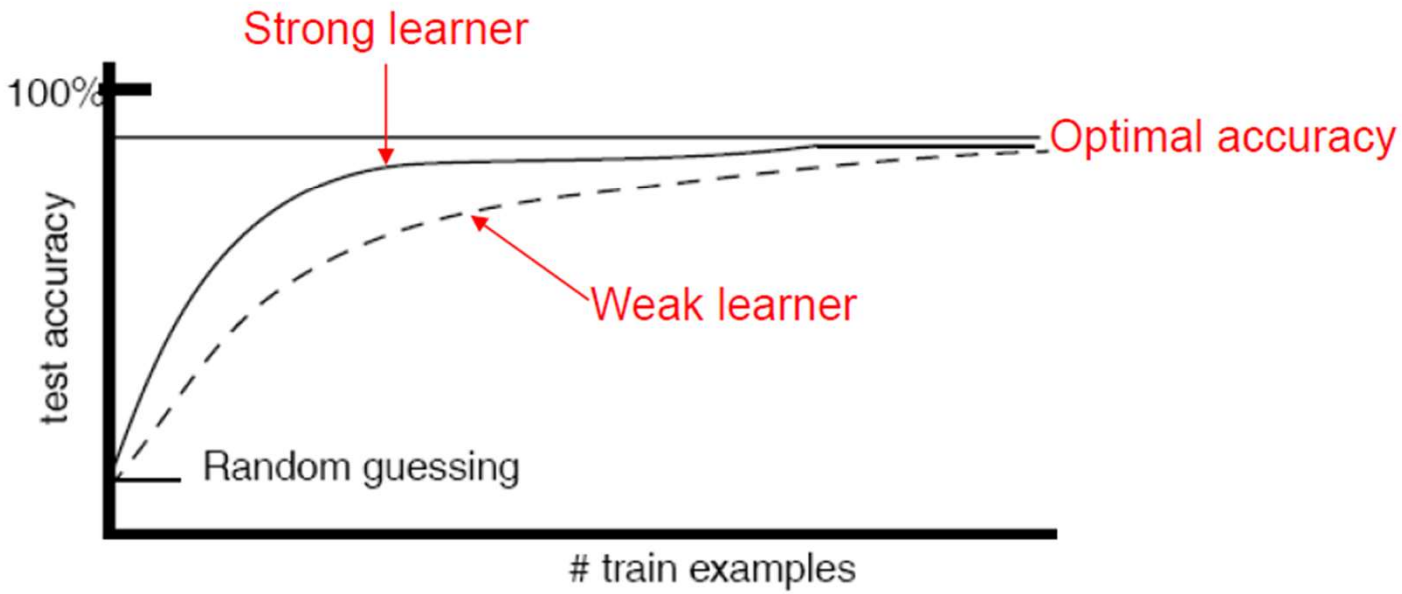
  - **What if one independent variable for linear model is not enough?** We may improve ability to predict by **bringing in additional independent variables**.
  - In linear regression, the model specification of the dependent variable is a ***linear combination of the parameters***. For example, if y is dependent variable;
  - **Multivariate (multiple independent variables) linear regression** model

    $y = w_o + w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_N x_N$

    where ***w*** is weight **parameter**.

❖ The All Important Training Set

- The quality of the classifier depends on the quality of training set.
  - (1) The size of training set (the bigger, the better)
  - (2) The accuracy of training set (the less noisy, the better)
  - (3) The existence of any systematic bias (the less bias, the better)
- Performance of many methods will converge close to optimal after sufficient examples.



❖ How to Measure Classifier performance

- **Confusion matrix** (or **Contingency matrix**): a ***visualization tool*** typically used **in supervised learning**.
- One benefit of a confusion matrix is that it is easy to see if the classification system is **confusing** two (or more) classes.

Confusion Matrix for two class prediction

		Predicted class: P	Predicted class: N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Confusion Matrix for three class prediction

Actual class \ ***Predicted class***	A	B	C	Total
A	88	10	2	100
B	14	40	6	60
C	18	10	12	40
Total	120	60	20	200

All off-diagonal elements are errors.

ACC = (88+40+12)/200 = 0.7

Suppose P is Malignant tumor (M) and N is Benign tumor (B).

TP: Actual M → Predicted M
- It's about detection of malignant tumors
- Higher is better

FP: Actual B → Predicted M
- It's about false alarm
- Lower is better

Two kinds of error
- FP (Type I error)
- FN (Type II error)

$$\text{Accuracy} = \frac{Correct}{ALL} = \frac{TP + TN}{ALL} = 1 - error$$

Accuracy which *measure only proportion of correct predictions is not useful when the two classes are of very different sizes*. For example, assigning every object to the larger set achieves a high proportion of correct predictions, but is not generally a useful classification.

- **Measure Classifier performance with consideration of error cost**
  - In case we have *associated score (e.g., probability) of class values*, we do not want to use simple accuracy.
  - We may **define classes by any score threshold of choice**. *Therefore, simply using accuracy results can be misleading.* We measure classifier performance with consideration of ***error cost***.
  - For example, we want to show how the number of correctly classified positive examples (*True positives rate: TPR*) varies with the number of incorrectly classified negative examples (*False positive rate: FPR*).

Rank	*Pr*	Class	Accuracy
1	0.99	Positive	
2	0.91	Positive	*1*
3	0.89	Positive	
4	0.85	Negative	
5	0.82	Positive	*0.8*
6	0.79	Positive	
7	0.77	Negative	
8	0.75	Positive	
9	0.72	Negative	
10	0.70	Positive	*0.7*

$$\textit{Recall, Sensitivity, TPR} = \frac{TP}{P} = \frac{TP}{TP+FN}$$

$$\textit{Selectivity, Specificity, TNR} = \frac{TN}{N} = \frac{TN}{TN+FP}$$

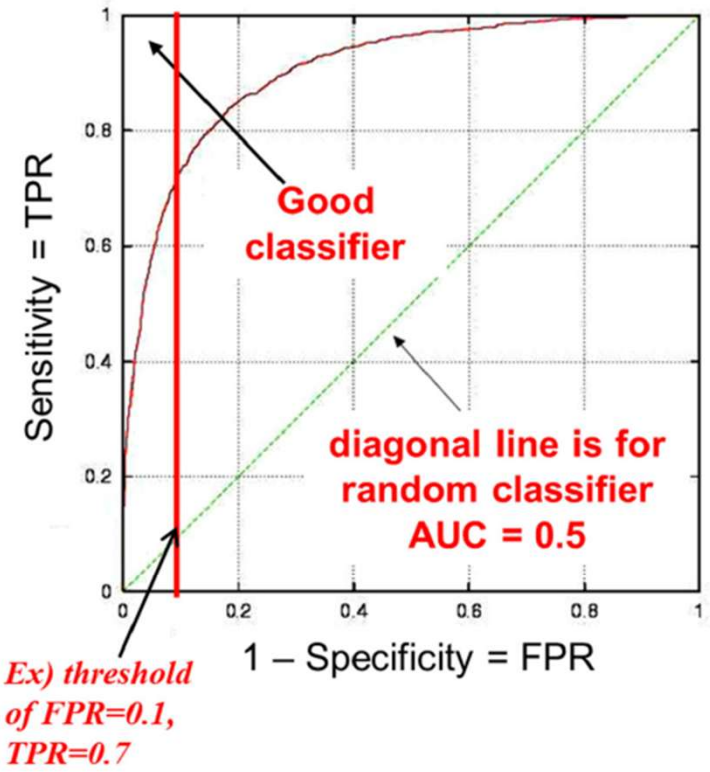
$$\textit{1 – Specificity = FPR} = \frac{FP}{N} = \frac{FP}{FP+TN}$$

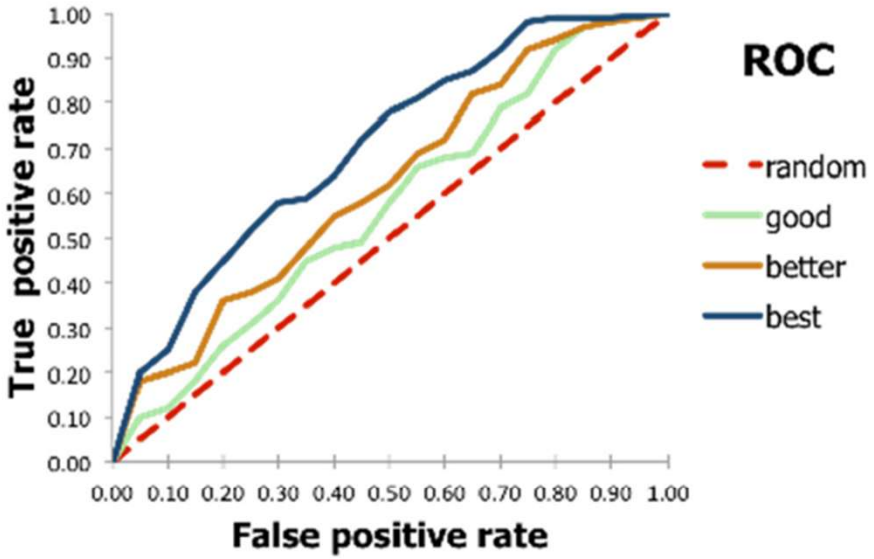
$$\textit{Positive predictive value, PPV} = \frac{TP}{TP+FP}$$

$$\textit{Negative predictive value, NPV} = \frac{TN}{TN+FN}$$

❖ Receiver Operating characteristic (ROC) curve

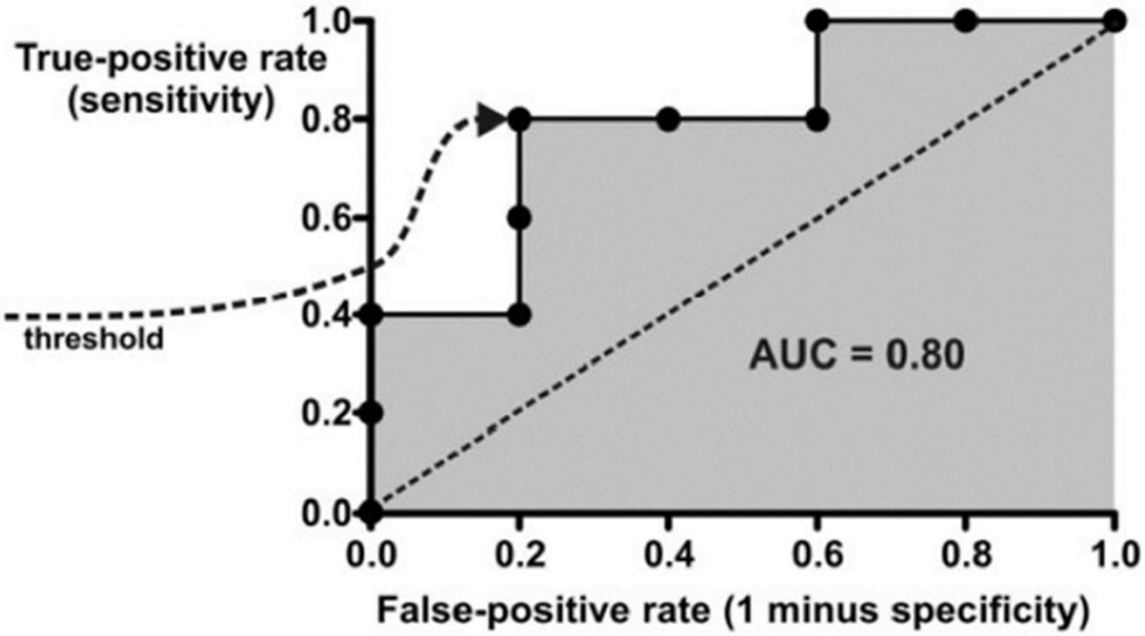
- A plot that illustrates the diagnostic performance of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting TPR against FPR at various threshold settings.





- ROC curve can be summarized as Area Under the Curve (AUC)
- AUC = 1 for a perfect classifier
- AUC > 0.7 is generally considered as a good classifier

#	Real class	$p(x)$
1	+	0.9
2	+	0.8
3	−	0.7
4	+	0.6
5	+	**0.5**
6	−	0.4
7	−	0.3
8	+	0.2
9	−	0.1
10	−	0.05



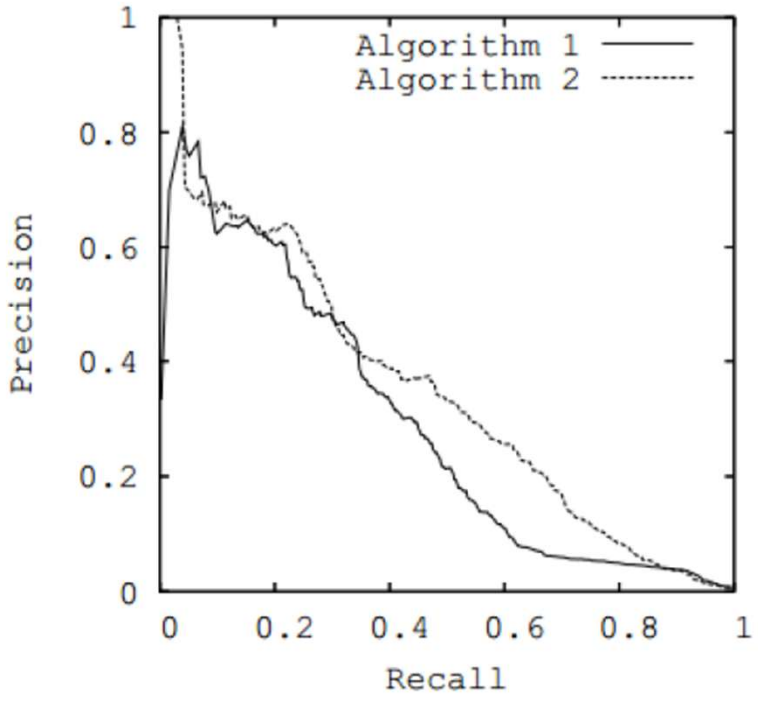
❖ PR (Precision-Recall) curve

- A diagnostic plot for performance of a binary classifier system as its discrimination threshold is varied, created by plotting Precision against Recall (= TPR) at various threshold settings.

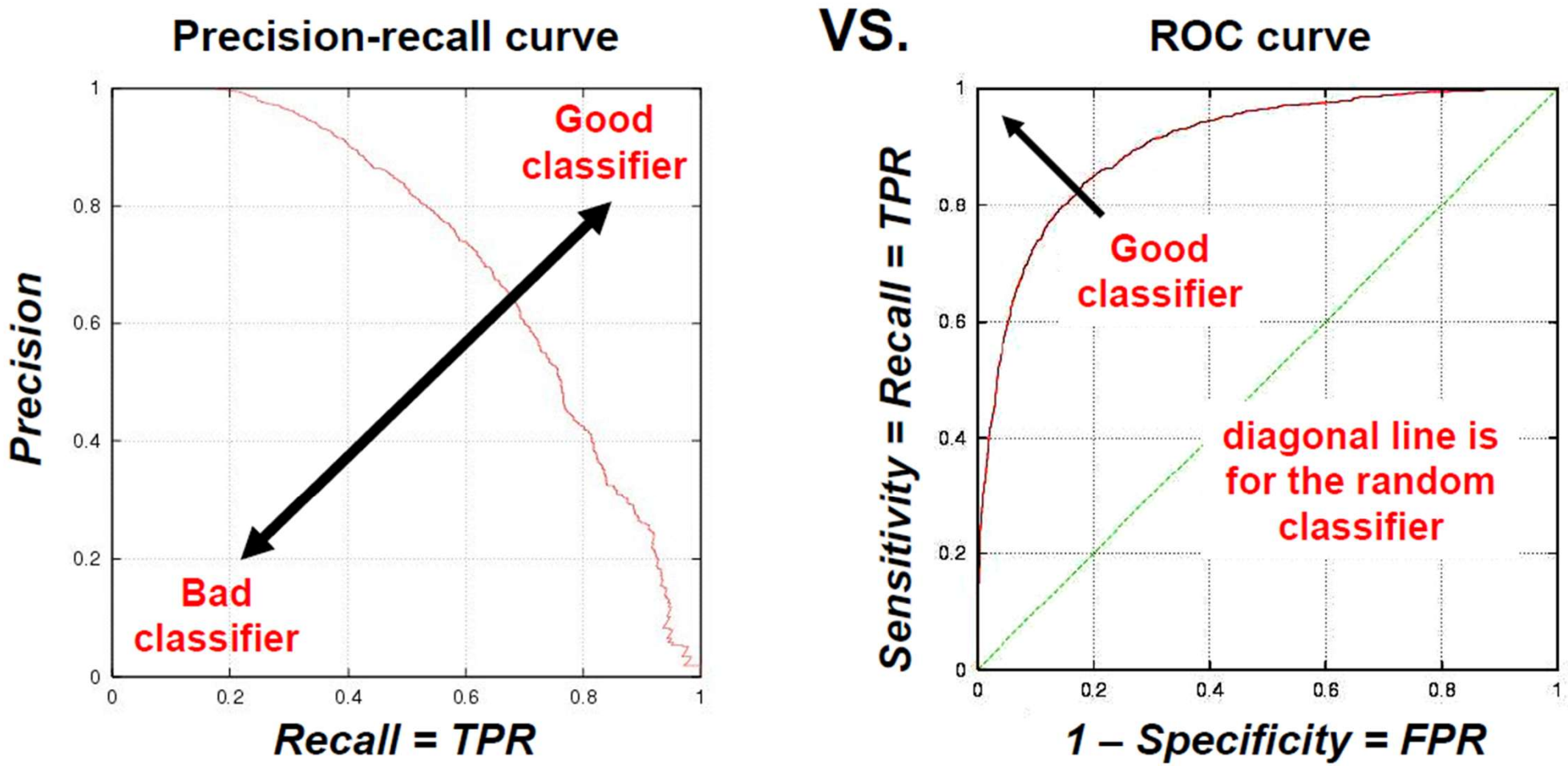
		Predicted class: *P*	Predicted class: *N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

$$\textbf{Precision} = \frac{TP}{Predicted\ P} = \frac{TP}{TP+FP}$$

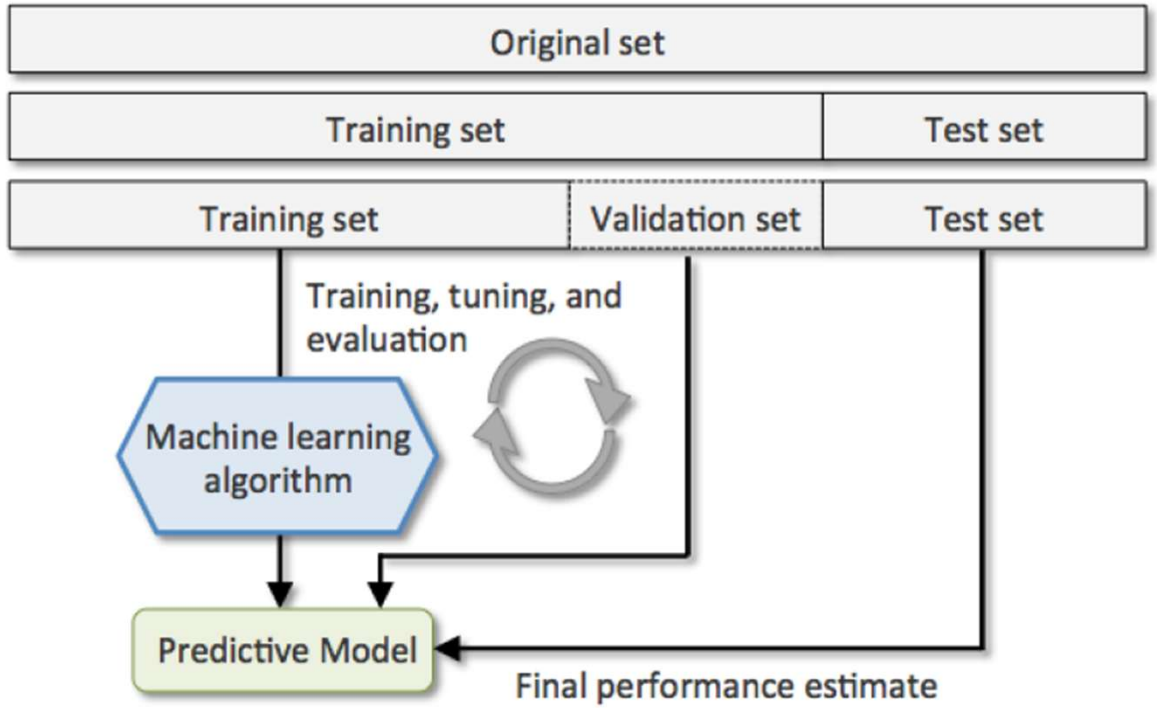
$$\textbf{Recall} = \textbf{TPR} = \frac{TP}{Actual\ P} = \frac{TP}{TP+FN}$$



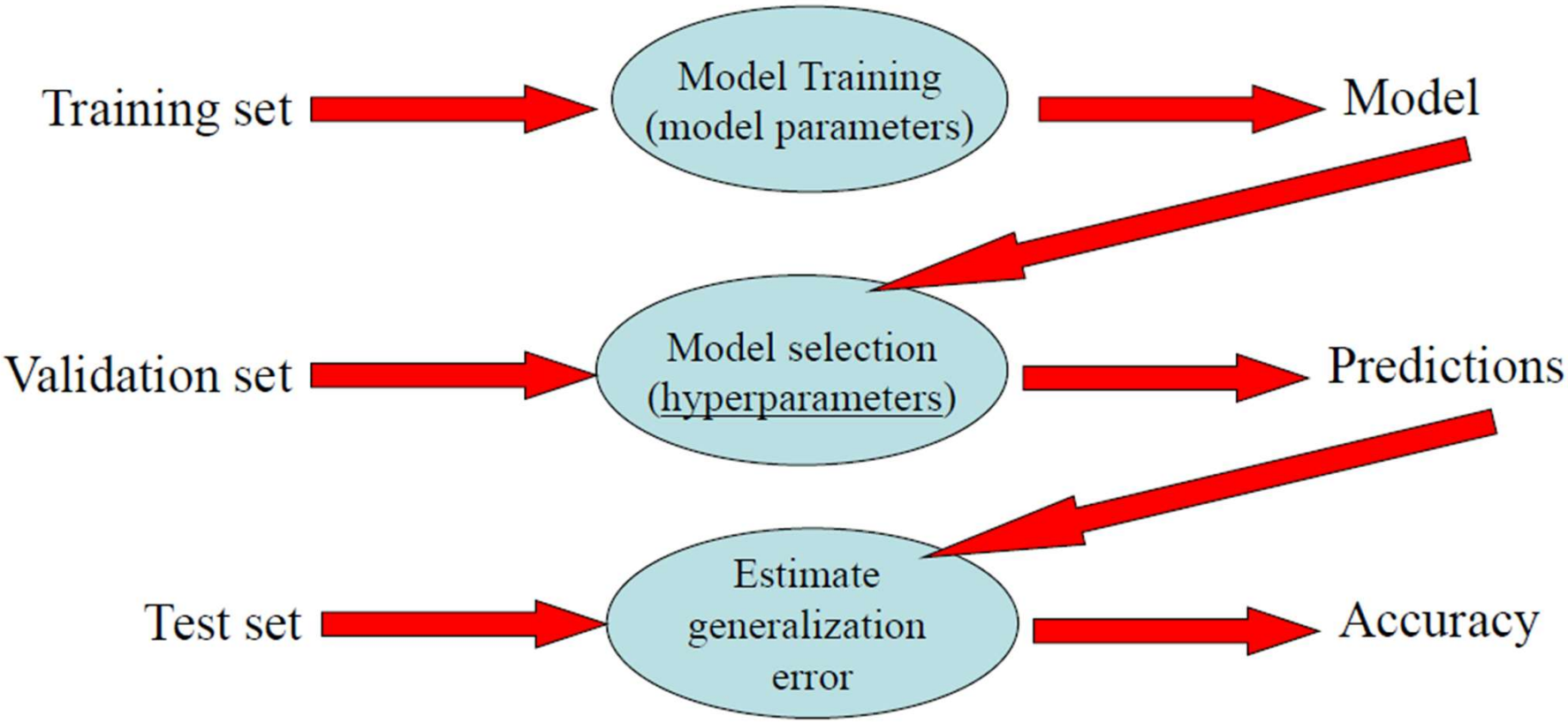
❖ **Relationship between ROC and PR**



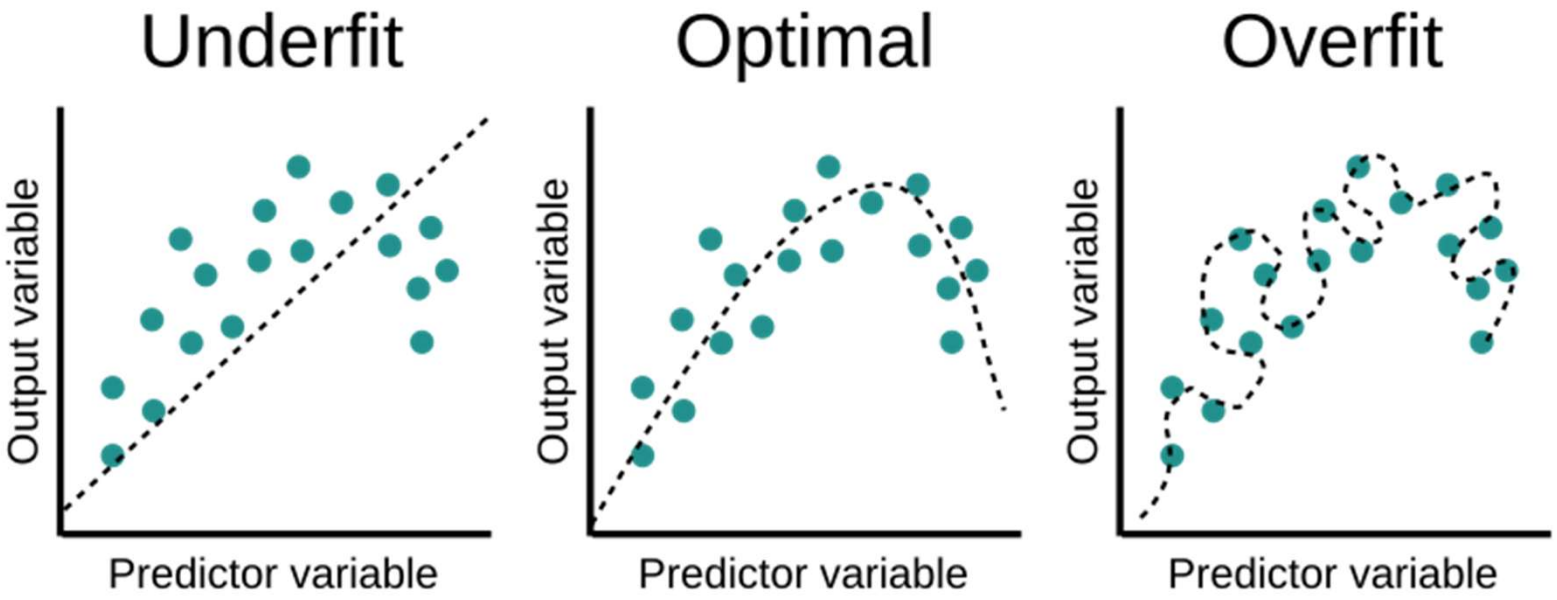
❖ Model Training, Validation, and Test



- Original training set is divided into three subsets.
- **Training set**: used to fit different models (*fit model parameters*)
- **Validation set**: used for model selection (*hyperparameter tuning*)
- **Test set**: used to *estimate model's generalization error*. Since we want to have less biased estimates, test set should be untouched in learning phase.
- In general, Training : Validation : Test = 70% : 10% : 20% of the original set

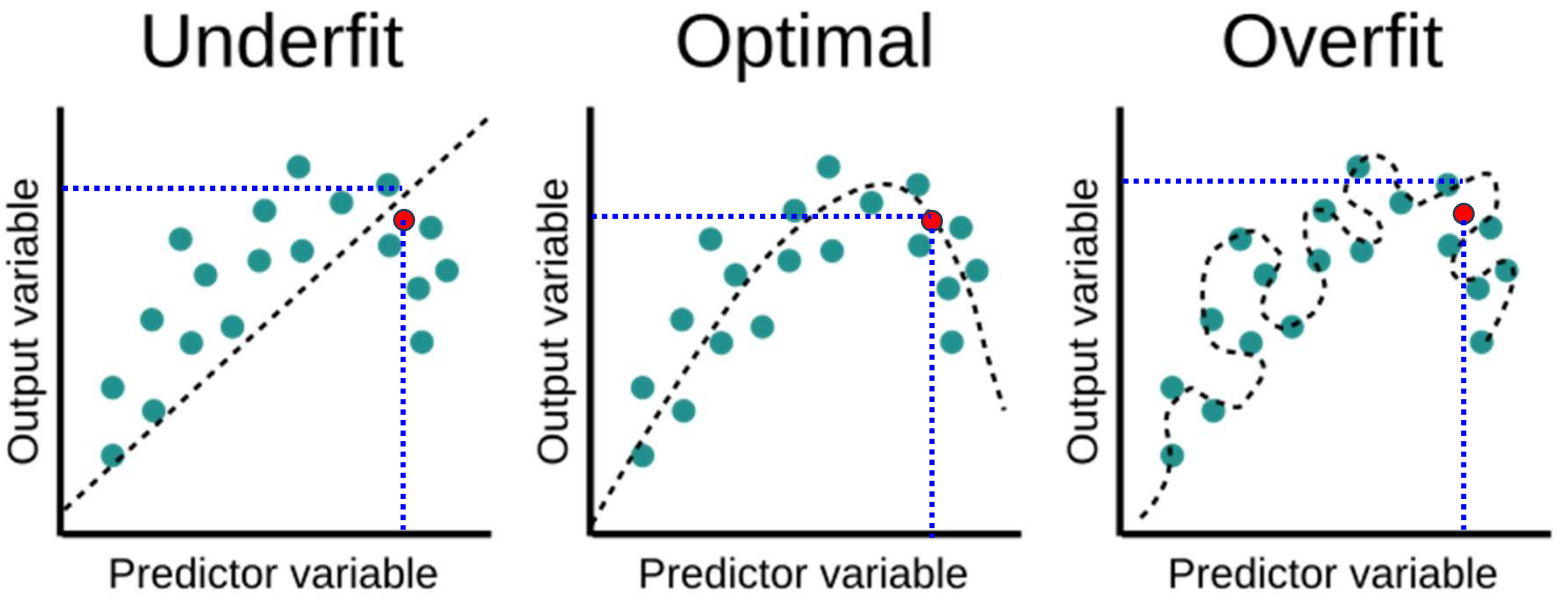

Training set
Model Training
(model parameters)
Model
Validation set
Model selection
(hyperparameters)
Predictions
Test set
Estimate
generalization
error
Accuracy

Over-fitting



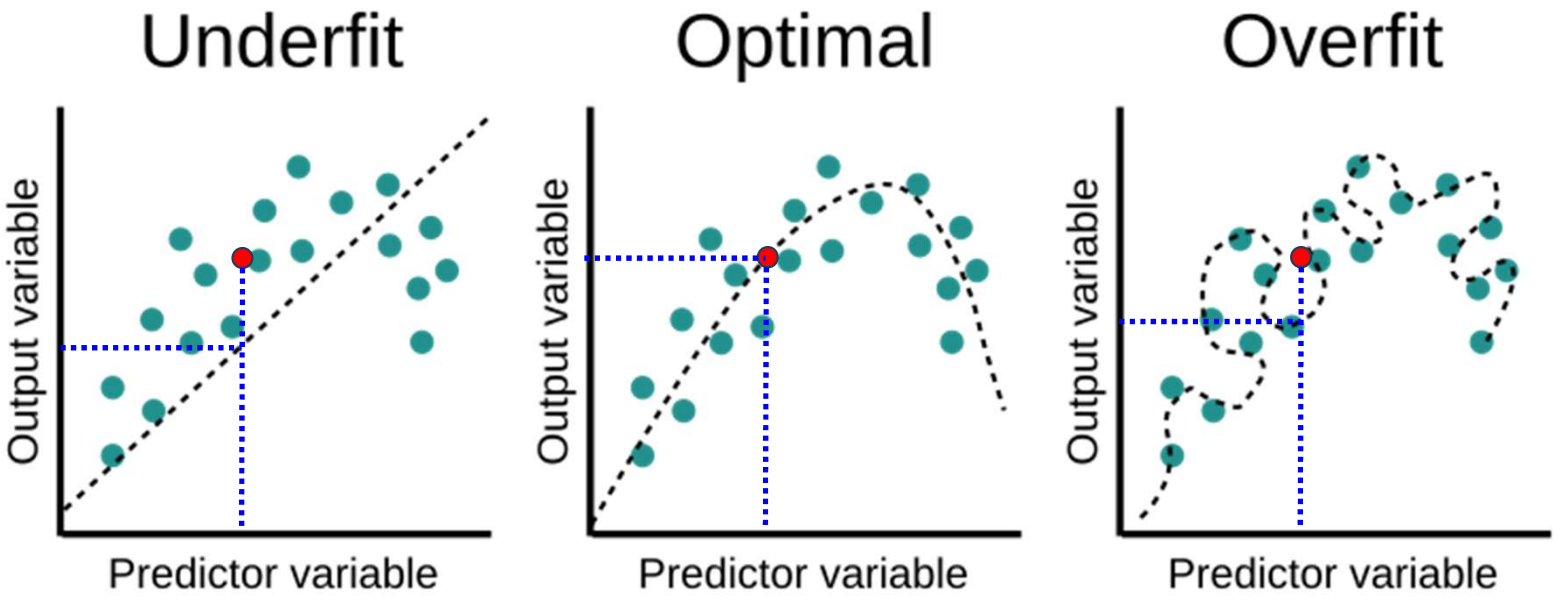
Although over-fitting explains the test set very well,
It cannot be used for “new data”

Over-fitting



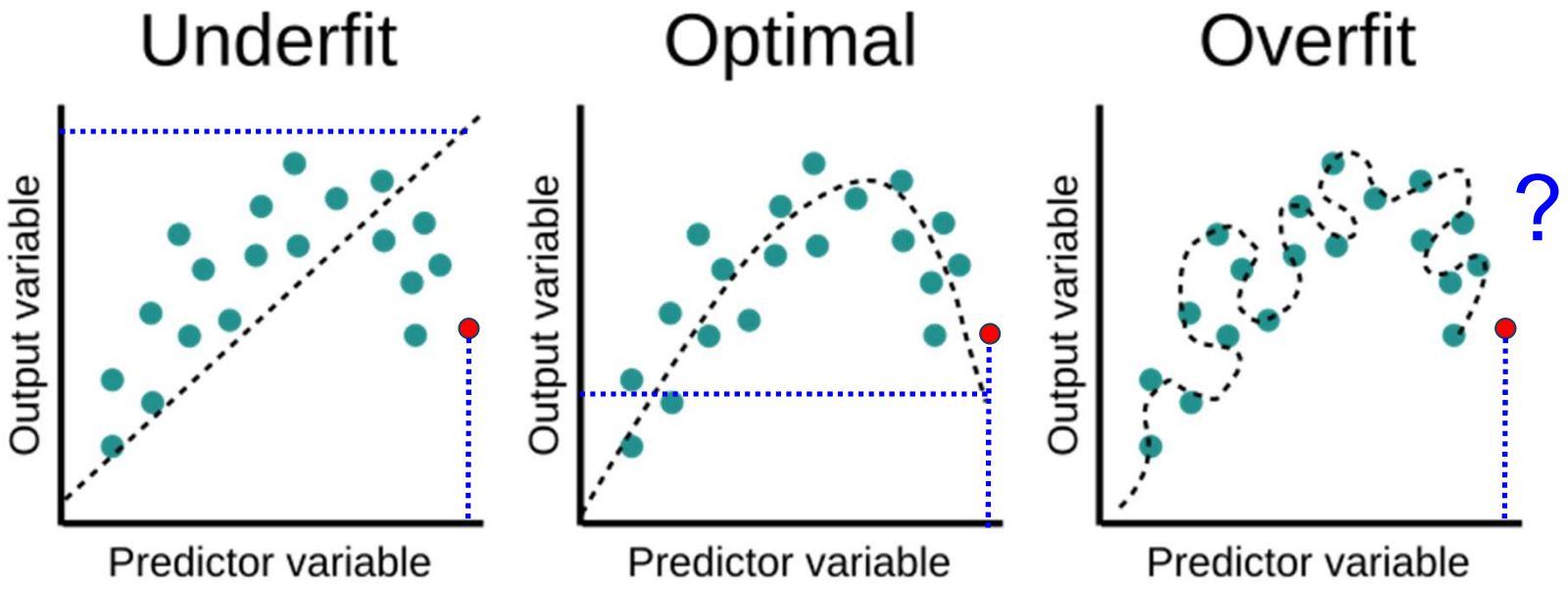
Within calibration range

Over-fitting



Within calibration range

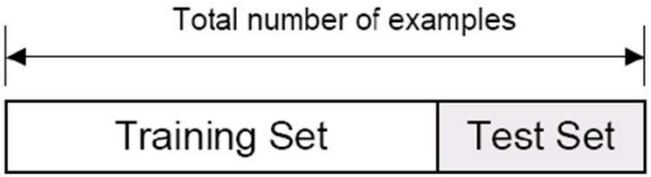
Over-fitting



Outside of calibration range

❖ Methods to avoid over-fitting

1. Holdout method
2. Cross Validation
3. Bootstrap



1. Holdout method: Split dataset into two groups

- **Training set** (includes validation set): used to train model (and model selection)
- **Test set**: used to estimate the error rate of the trained model

▪ **The holdout method has two major drawbacks.**

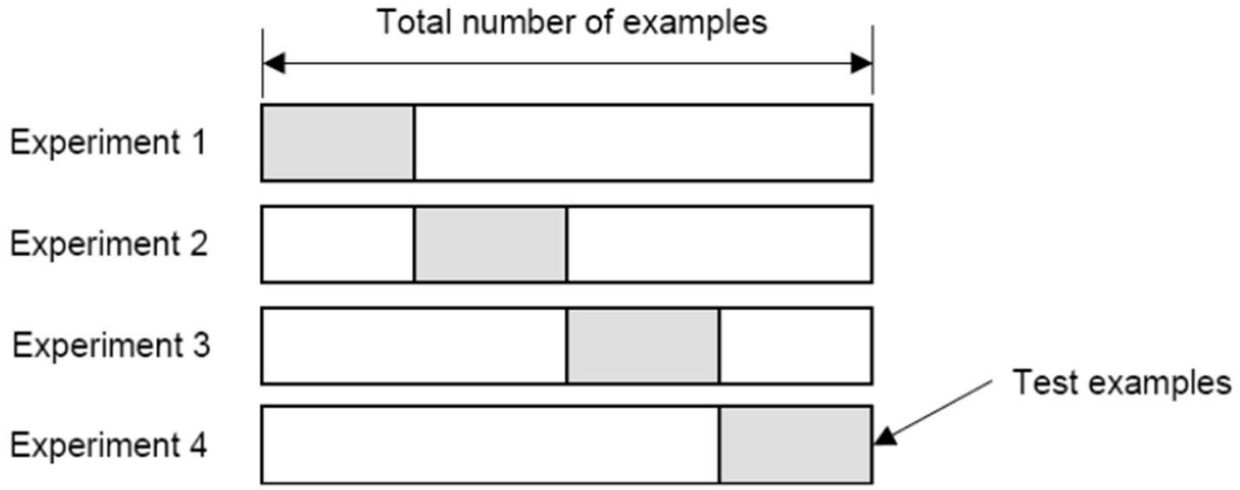
- In problems where we have a sparse dataset we may not be able to afford the "luxury" of setting aside a portion of the dataset for testing.
- Since it is a single train-and-test experiment, the holdout estimate of error rate will be misleading if we happen to get an "unfortunate" split.

❖ The limitations of the holdout can be overcome with a family of ***resampling methods*** at the expense of more computations.

- **Cross Validation** (resampling **without replacement**)
  - k-fold Cross Validation
  - Leave-one-out Cross Validation
- **Bootstrap** (resampling **with replacement**)

2. *k*-fold Cross Validation

- Create a *k*-fold partition of the dataset
- For each of *k* experiments, use $k-1$ folds for training and the remaining one for testing (k = 4 for the below example).



- The advantage of *k*-fold Cross Validation is that all the examples in the dataset are eventually used for both training and testing.
- A common choice for *k*-fold Cross Validation is *k* = 10.
- The true error is estimated as the average error rate.

$$E = \frac{1}{k}\sum_{i=1}^{k} Ei$$

3. Bootstrap

- Based on ***sampling the dataset (original training set) with replacement*** to form a training set (***new training set***). A particular variant is 0.632 bootstrap.
- *A dataset of N examples is sampled N times, with replacement*, to *give another dataset of N examples*. Because some elements in this second dataset will (almost certainly) be repeated, there must be *some* ***instances in the original dataset that have not been picked****: we will use these as* ***test examples***.
- The probability of being picked each time is 1/N, and probability of not being picked is 1-1/N. Therefore, *the* ***chance*** *that a particular example will* ***not be picked*** *for the entire training set* is

$$\left(1-\frac{1}{N}\right)^{N} \approx e^{-1} = 0.368$$

- Thus, for a reasonably large dataset, the **test set** will contain ~**36.8%** of the original examples and the **training set** will contain ~**63.2%** of them. Some examples will be repeated in the training set. (Example: After x1000 bootstrap sampling of the original N=1000 training set, a new training set has 1000 but only ~632 unique samples, a test set has ~368 samples)
- **Training a model** on the new training set and calculating its error over the test set will be a **pessimistic estimate** of the true error rate, **because** the training set, although its size is same as the original training set, **contains only 63% of the unique original training examples**.
- **To compensate** for this, we combine the test set error rate with the resubstitution (**testing using the same dataset used for training**) error. This gives a very **optimistic estimate** of the true error. The final estimate e as follows:

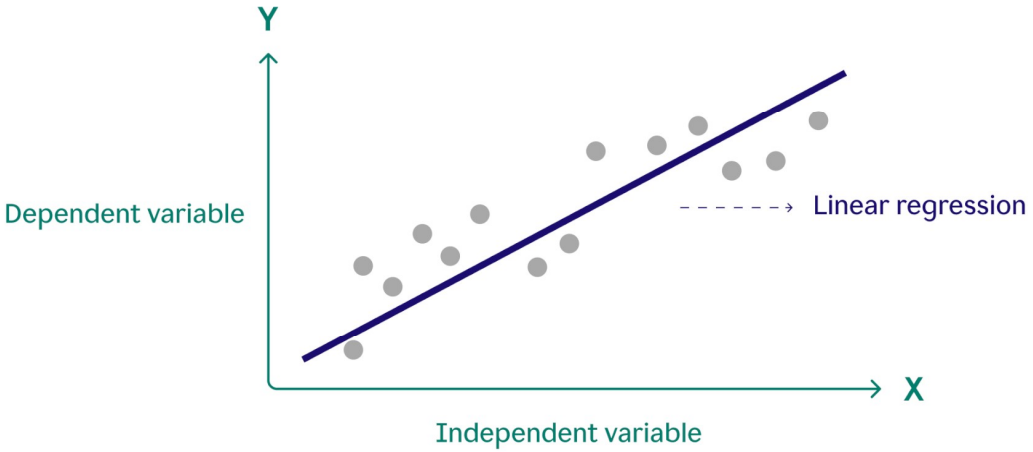
$$e = 0.632 * e_{\text{test\_examples}} + 0.368 * e_{\text{training\_examples}}$$

- Then the whole bootstrap procedure is repeated several times, with different replacement samples for the training set, and the results averaged.
- The bootstrap procedure may be the best way of estimating error for very small datasets.

Linear regression
Logistic regression

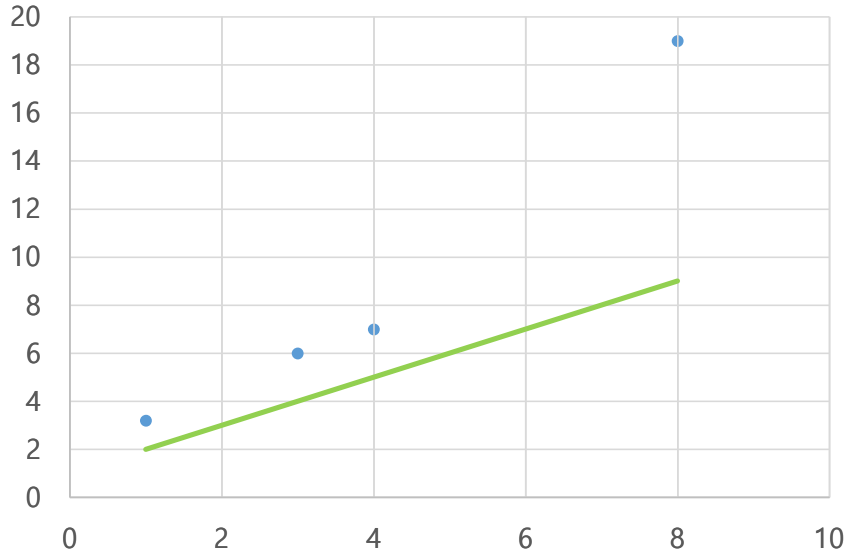
Linear regression

- Hypothesis: $H(x) = Wx + b$
- Cost function: $cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}\left(H(x^i) - y^i\right)^2$
- Gradient descent algorithm



Linear regression

X	Y
1	3.2
3	6
4	7
8	19



- $H(x) = Wx + b$

- Let's start with W=1, b=1
- $cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}\left(H(x^i) - y^i\right)^2$

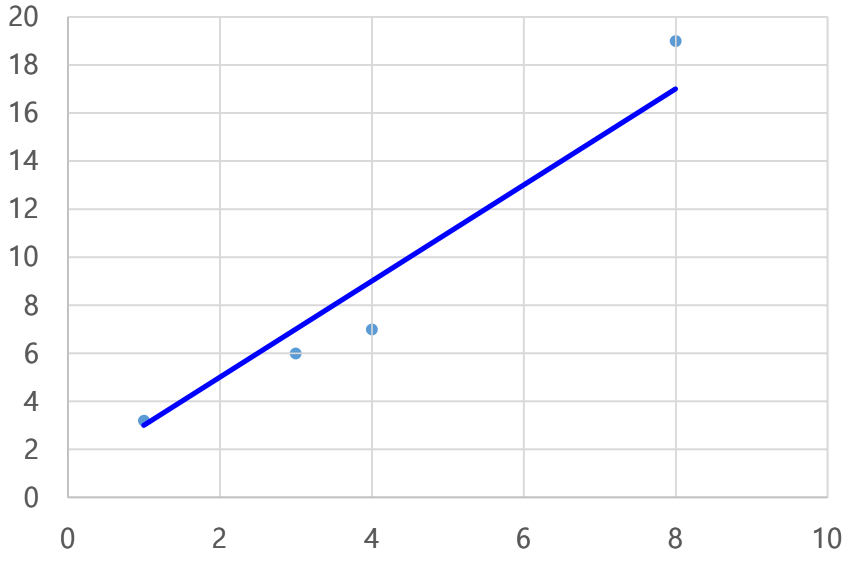
→ $\frac{1}{4}$ (([1*1+1] – 3.2)^2 +
([1*3+1] – 6)^2 + …)

→ cost: $\frac{1}{4}$(1.44+4+4+100)

= 27.36

Linear regression

X	Y
1	3.2
3	6
4	7
8	19



- $H(x) = Wx + b$

- Let's start with W=1, b=1
- $cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}\left(H(x^i) - y^i\right)^2$

→ $\frac{1}{4}$ (([1*1+1] – 3.2)^2 +
([1*3+1] – 6)^2 + …)

→ cost: $\frac{1}{4}$(1.44+4+4+100)

= 27.36

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

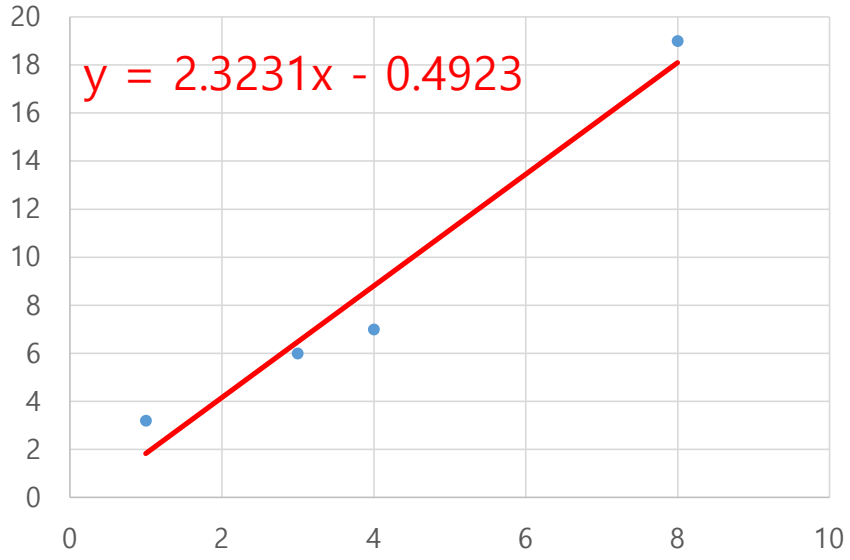
→ $\frac{1}{4}$ (([2*1+1] – 3.2)^2 +
([2*3+1] – 6)^2 + …)

→ cost: $\frac{1}{4}$(0.04+1+4+4)

= 2.26

Linear regression

X	Y
1	3.2
3	6
4	7
8	19



- $H(x) = Wx + b$

- Let's start with W=1, b=1
- $cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}\left(H(x^i) - y^i\right)^2$

→ $\frac{1}{4}$ (([1*1+1] – 3.2)^2 +
([1*3+1] – 6)^2 + …)

→ cost: $\frac{1}{4}$(1.44+4+4+100)

= 27.36

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

→ $\frac{1}{4}$ (([2*1+1] – 3.2)^2 +
([2*3+1] – 6)^2 + …)

→ cost: $\frac{1}{4}$(0.04+1+4+4)

= 2.26

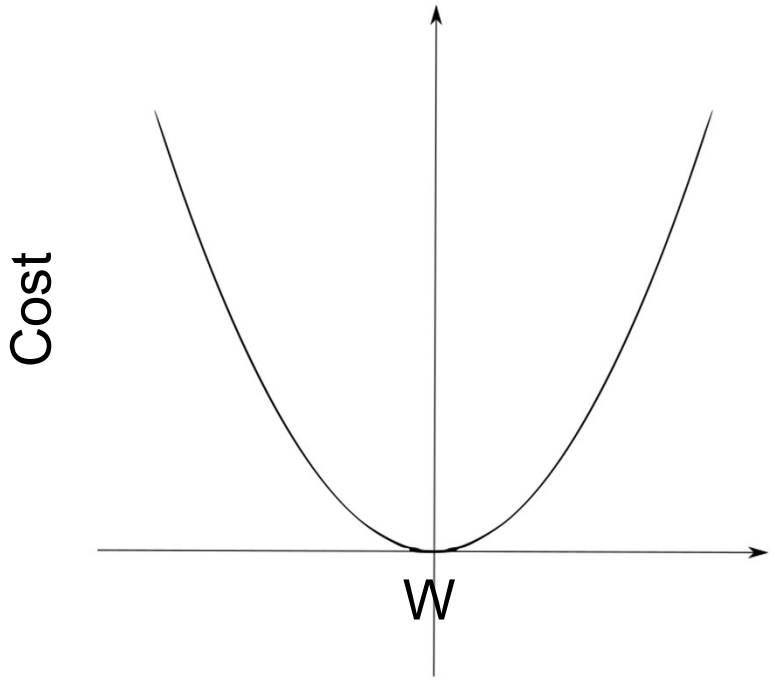
Linear regression

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

→ How to minimize it?

Linear regression

- Simplified hypothesis and cost
- $H(x) = Wx$
- $cost(W) = \frac{1}{m}\sum_{i=1}^{m}\left(H(x^i) - y^i\right)^2$
- $cost(W) = \frac{1}{m}\sum_{i=1}^{m}\left(Wx^i - y^i\right)^2$



Linear regression

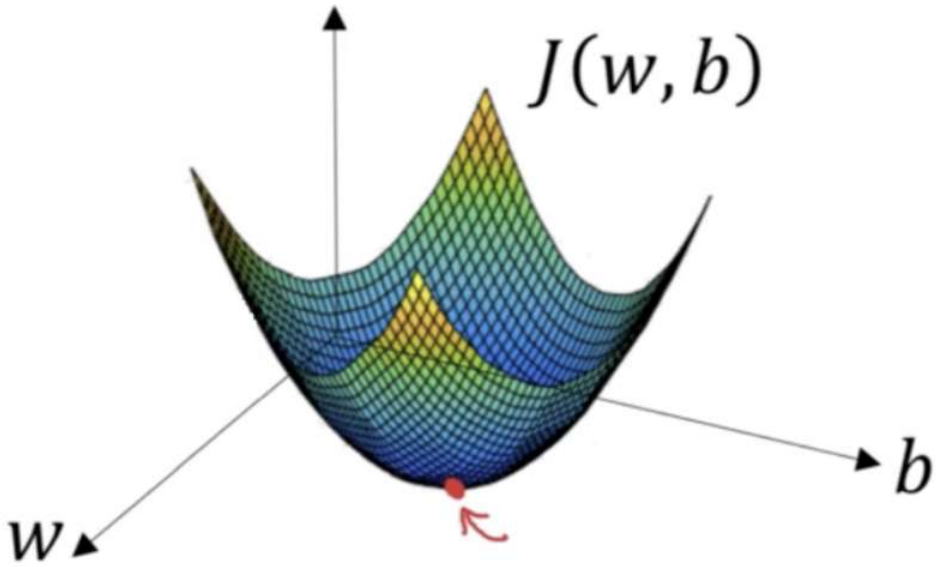
- Gradient descent algorithm
- Minimize cost function
- $Wnew = W$ - $gradient * cost(W)$

→ gradient: partial differentiation (assuming it is independent variables)

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

Ex)

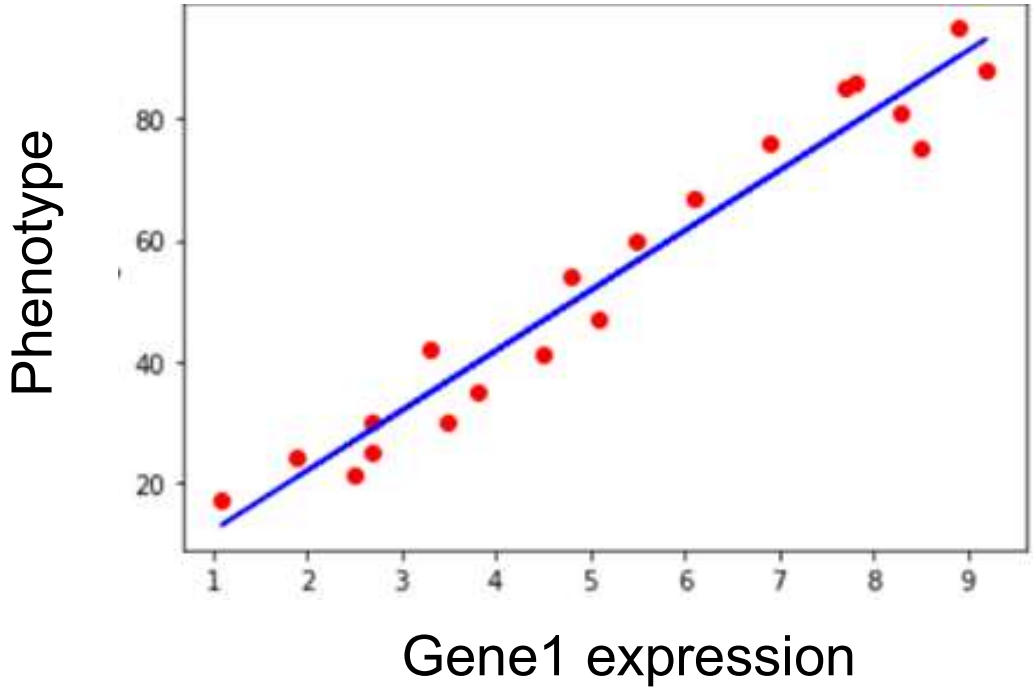
- $cost(W) = \alpha \frac{\partial}{\partial W} \frac{1}{2m} \sum_{i=1}^{m} (Wx^i - y^i)^2$
- $cost(W) = \alpha \frac{1}{2m} \sum_{i=1}^{m} 2(Wx^i - y^i)x^i$
- $W := W - \alpha \frac{1}{m} \sum_{i=1}^{m} (Wx^i - y^i)x^i$



Linear regression

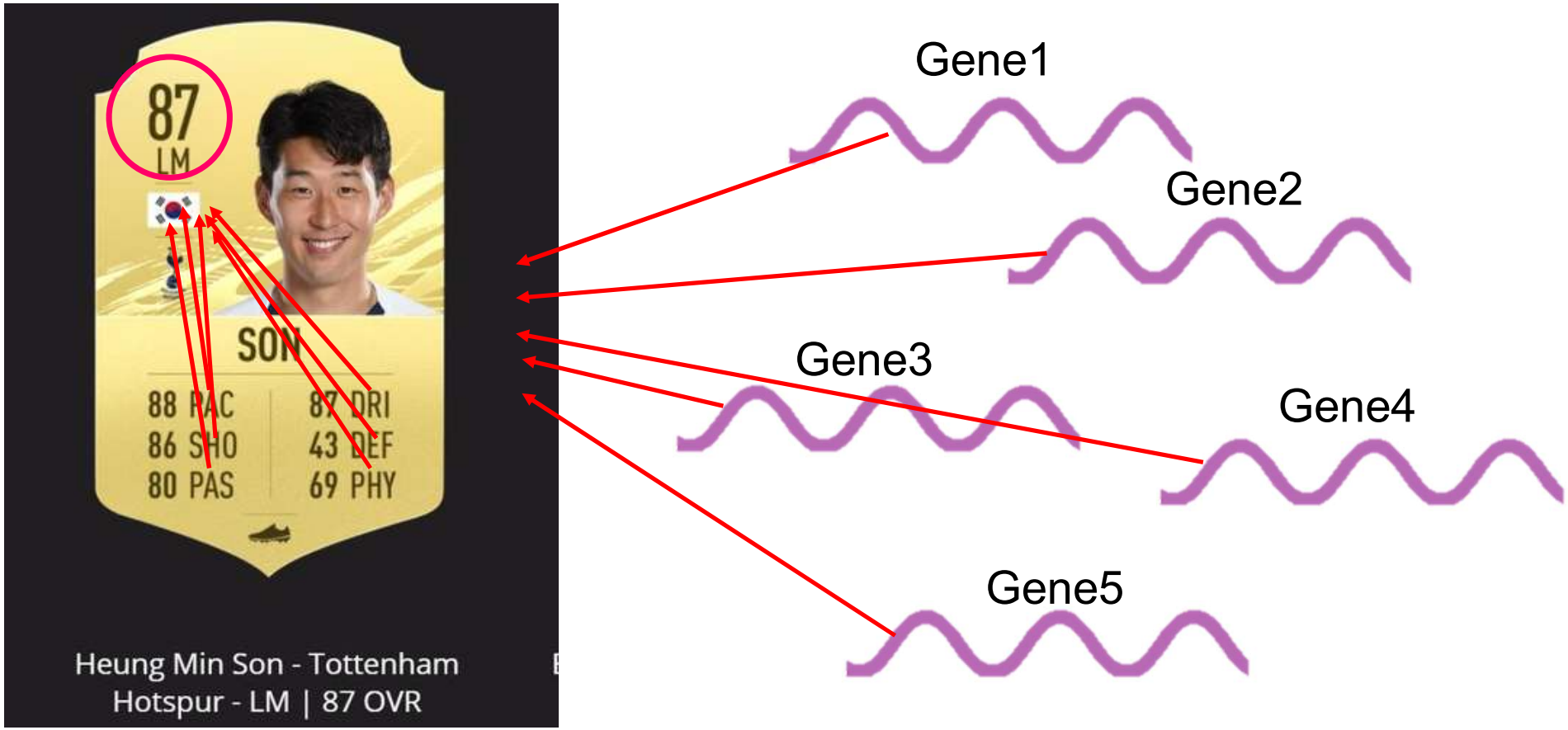
- Phenotype = Linear regression (Gene1 expression)

Ex): Cell size, height, …



Multi-variable Linear regression

- But! What if there are multiple variables?



Multi-variable Linear regression

- $H(x_1, x_2, x_3) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$
- $H(x_1, x_2, x_3 \ \ldots, x_n) = w_1 x_1 + w_2 x_2 + w_3 x_3 + \ \ldots + w_n x_n + b$

Multi-variable Linear regression

- Matrix dot product

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}\begin{bmatrix}7 & 8\\ 9 & 10\\ 11 & 12\end{bmatrix}=\begin{bmatrix}58 & 64\\ 139 & 154\end{bmatrix}$$

$$\begin{bmatrix}\boxed{1 & 2 & 3}\\ 4 & 5 & 6\end{bmatrix}\begin{bmatrix}\boxed{7}\\ 9\\ 11\end{bmatrix}\begin{bmatrix}8\\ 10\\ 12\end{bmatrix}=\begin{bmatrix}\boxed{58} & 64\\ 139 & 154\end{bmatrix}$$ 1*7 + 2*9 + 3*11 = 58

$$\begin{bmatrix}\boxed{1 & 2 & 3}\\ 4 & 5 & 6\end{bmatrix}\begin{bmatrix}7\\ 9\\ 11\end{bmatrix}\begin{bmatrix}\boxed{8}\\ 10\\ 12\end{bmatrix}=\begin{bmatrix}58 & \boxed{64}\\ 139 & 154\end{bmatrix}$$ 1*8 + 2*10 + 3*12 = 64

$$\begin{bmatrix}1 & 2 & 3\\ \boxed{4 & 5 & 6}\end{bmatrix}\begin{bmatrix}\boxed{7}\\ 9\\ 11\end{bmatrix}\begin{bmatrix}8\\ 10\\ 12\end{bmatrix}=\begin{bmatrix}58 & 64\\ \boxed{139} & 154\end{bmatrix}$$ 4*7 + 5*9 + 6*11 = 139

$$\begin{bmatrix}1 & 2 & 3\\ \boxed{4 & 5 & 6}\end{bmatrix}\begin{bmatrix}7\\ 9\\ 11\end{bmatrix}\begin{bmatrix}\boxed{8}\\ 10\\ 12\end{bmatrix}=\begin{bmatrix}58 & 64\\ 139 & \boxed{154}\end{bmatrix}$$ 4*8 + 5*10 + 6*12 = 154

Multi-variable Linear regression

- $H(x) = Wx$

- $(x_1, x_2, x_3) \cdot \begin{matrix} w_1 \\ w_2 \\ w_3 \end{matrix} = w_1 x_1 + w_2 x_2 + w_3 x_3$ → one sample

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\ x_{41}w_1 + x_{42}w_2 + x_{43}w_3 \\ x_{51}w_1 + x_{52}w_2 + x_{53}w_3 \end{pmatrix}$$ → many samples

→ $H(X)$ =XW

Linear regression

- Phenotype = Linear regression (gn1, gn2, gn3, … expression)

Ex): Cell size, height, …