UCLA QCBio Collaboratory

# Machine Learning with Python

Wenbin Guo
Bioinformatics IDP, UCLA
wbguo@ucla.edu

2023 Winter

UCLA Bioinformatics

Thanks Dr. Seyoon Ko for some code materials

# Day 3: Unsupervised learning

Wenbin Guo
Bioinformatics IDP, UCLA
wbguo@ucla.edu
2023 Winter



Thanks Dr. Seyoon Ko for some code materials

# Agenda

- Day 1: Introduction to machine learning
  - Some key concepts in machine learning
  - Jupyter notebook and some packages usage
- Day 2: Supervised learning
  - Classification
  - Regression
  - Regularization
- Day 3: Unsupervised learning
  - Dimension reduction
  - Clustering

# Notation of the slides

- `Code or Pseudo-Code` chunk starts with "➢", e.g.
  - ➢ `print("Hello world!")`

- Link is underlined

- Important terminology is in **bold** font

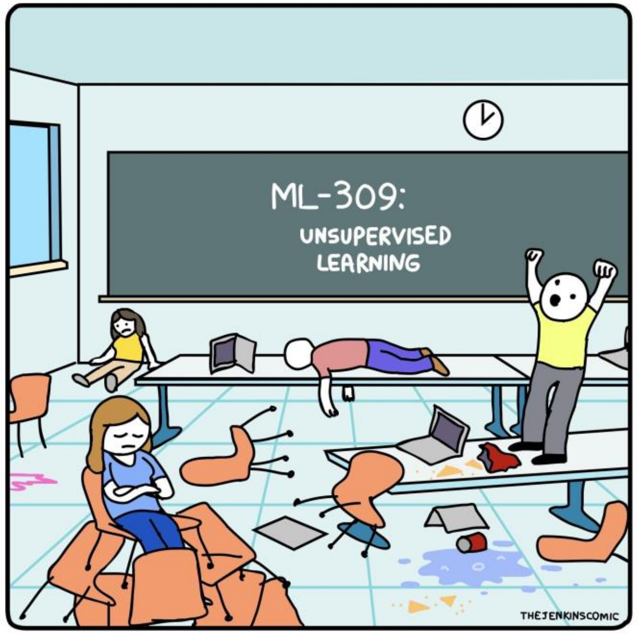
# Overview

## Time

- 3-hour workshop (45min + 45min + 30min + practice/Q&A)

## Topics

- ❑ Regression
- ❑ Regularization
- ❑ Dimension reduction
- ❑ Clustering



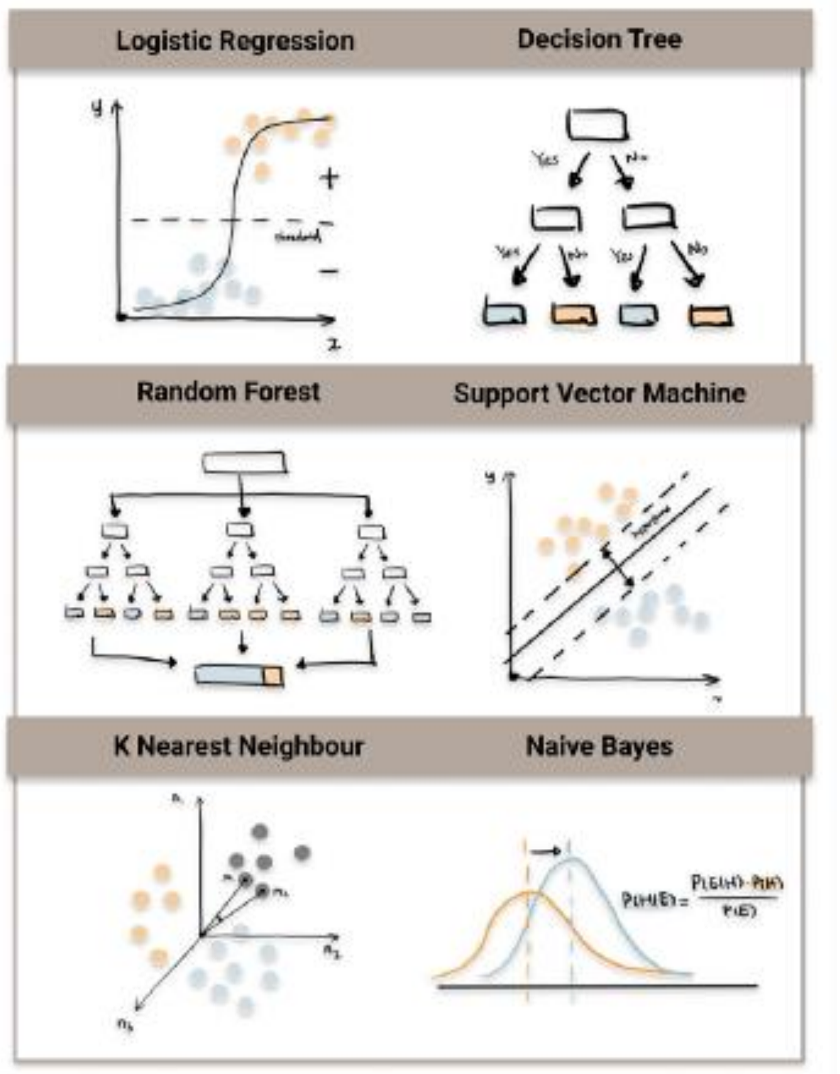



https://medium.com/the-21st-century/machine-learning-a-strategy-to-learn-and-understand-chapter-3-9daaad4afc55

# Summary –Day2

## Classification:

- ❑ 9 different classification algorithms
    - Logistic regression
    - KNN
    - Naïve bayes
    - Support vector machine
    - Decision tree
    - Random forest
    - Adaboost
    - Gradient boosting
    - Neural network



# Summary –Day2

## Classification:

- ❑ 9 different classification algorithms
- ❑ Cross entropy loss function

True probability distribution
(one-hot)

$$H(p,q) = -\sum_{x \in \text{classes}} p(x) \log q(x)$$

Your model's predicted
probability distribution

# Summary –Day2

## Classification:

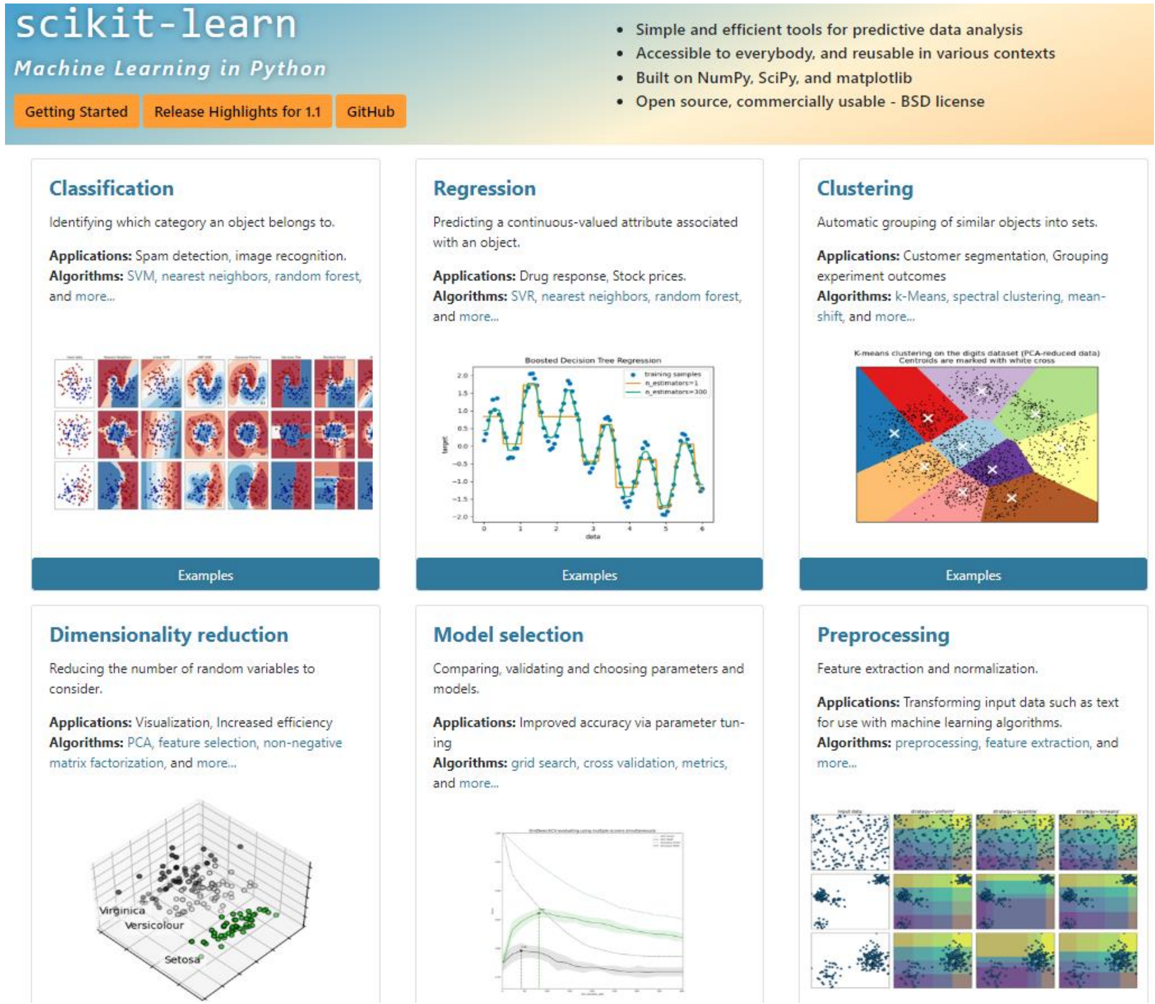
- ❑ 9 different classification algorithms
- ❑ Cross entropy loss function
- ❑ Performance measure
  - Accuracy
  - Confusion matrix
  - Precision & Recall
  - ROC, AUC, PR-ROC

| | | Predicted Class | | |
|---|---|---|---|---|
| | | **Positive** | **Negative** | |
| **Actual Class** | **Positive** | True Positive (TP) | False Negative (FN) **Type II Error** | **Sensitivity** $\frac{TP}{(TP+FN)}$ |
| | **Negative** | False Positive (FP) **Type I Error** | True Negative (TN) | **Specificity** $\frac{TN}{(TN+FP)}$ |
| | | **Precision** $\frac{TP}{(TP+FP)}$ | **Negative Predictive Value** $\frac{TN}{(TN+FN)}$ | **Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$ |

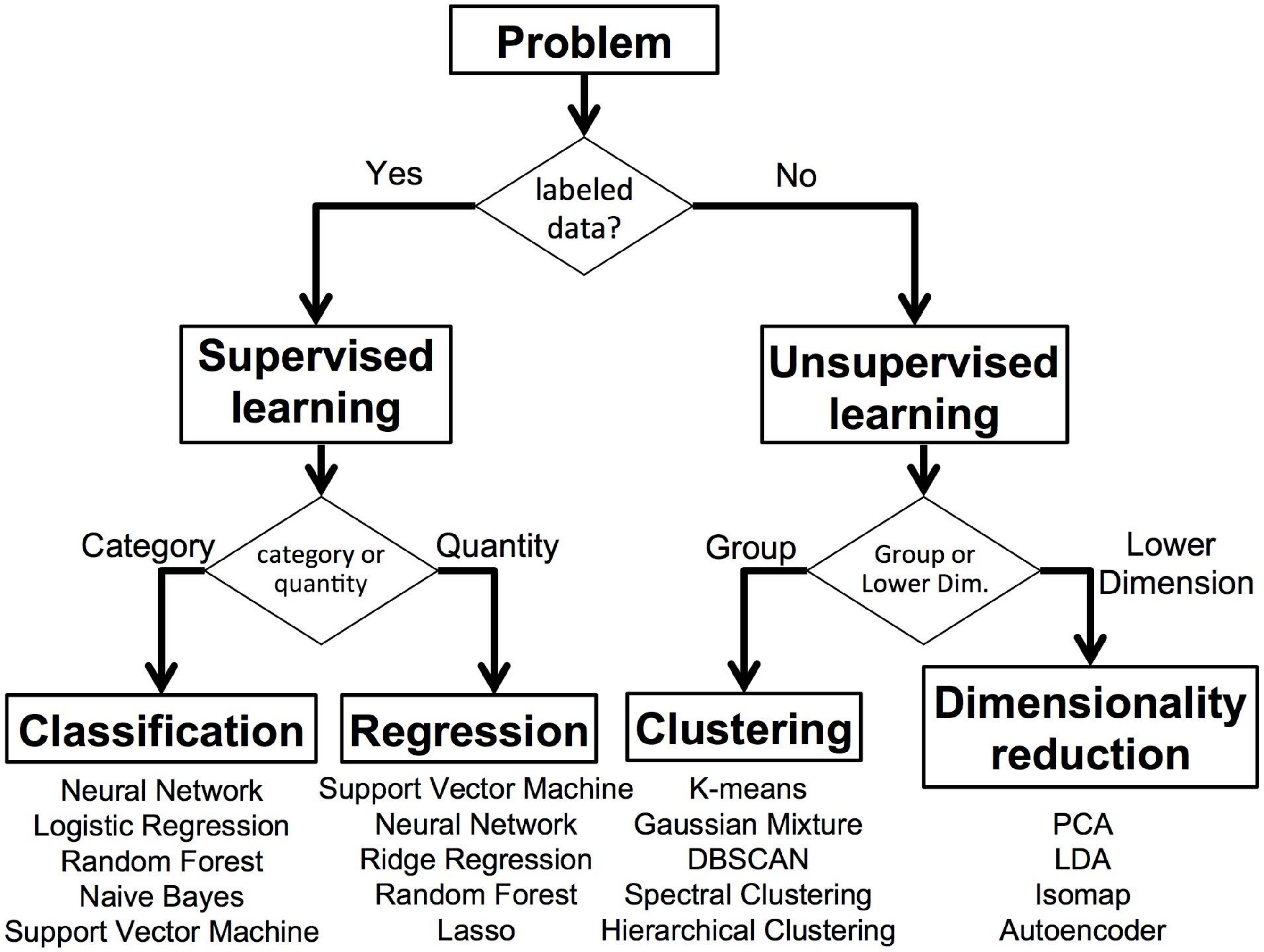
# Summary –Day2

## Classification:

- ❑ 9 different classification algorithms
- ❑ Cross entropy loss function
- ❑ Performance measure
- ❑ Practice
  - Training-test-validation set construction
  - Performance measure calculation
  - ROC curve, decision boundary
  - Linearly non-separable example
  - Parameter tuning (overfitting/underfitting)



# Types of machine learning






https://pythonnumericalmethods.berkeley.edu/notebooks/chapter25.01-Concept-of-Machine-Learning.html

# Supervised learning

- Training data with n ***samples*** of ***features*** **x** and ***label*** y
- Learn a function class f(**x**) to describe y based on **x**

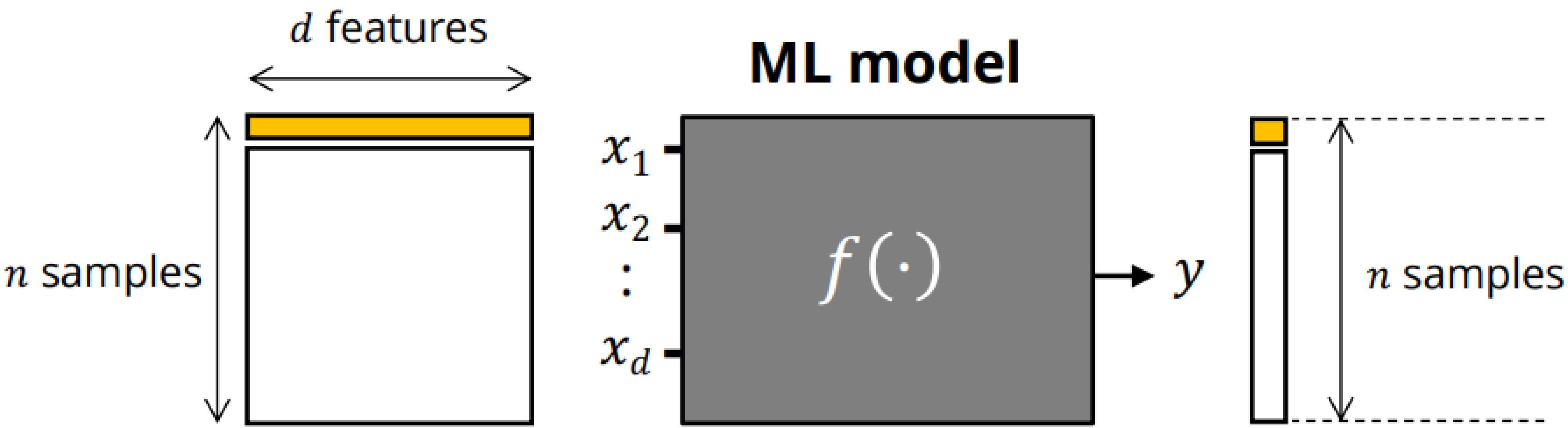


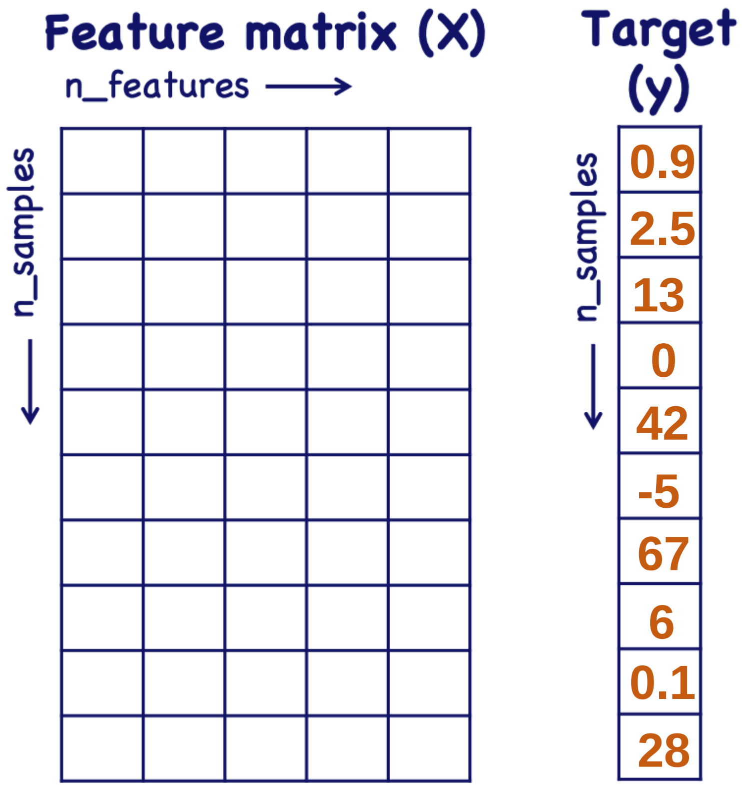



Figure adopted from CSEP 590B Explainable AI

# Different choice of f() for regression tasks

- Linear regression
- Polynomial regression
- SVR
- Tree-based regression
- Boosting
- Neural Network

…




$$\hat{y} = f(x)$$



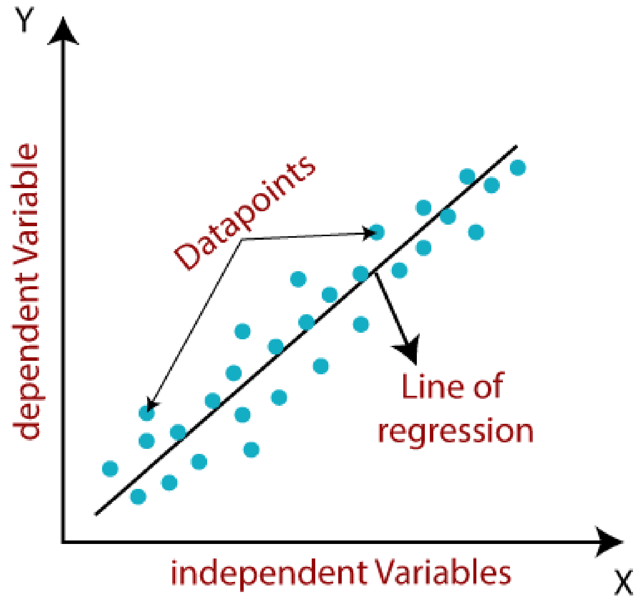
https://pythonnumericalmethods.berkeley.edu/notebooks/chapter25.01-Concept-of-Machine-Learning.html

# Linear regression

- Make prediction by computing the weighted sum of features

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

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x}$$



# Linear regression

- Make prediction by computing the weighted sum of features

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

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x}$$

- Train the model: find a parameter set to best fit the data

- Loss function

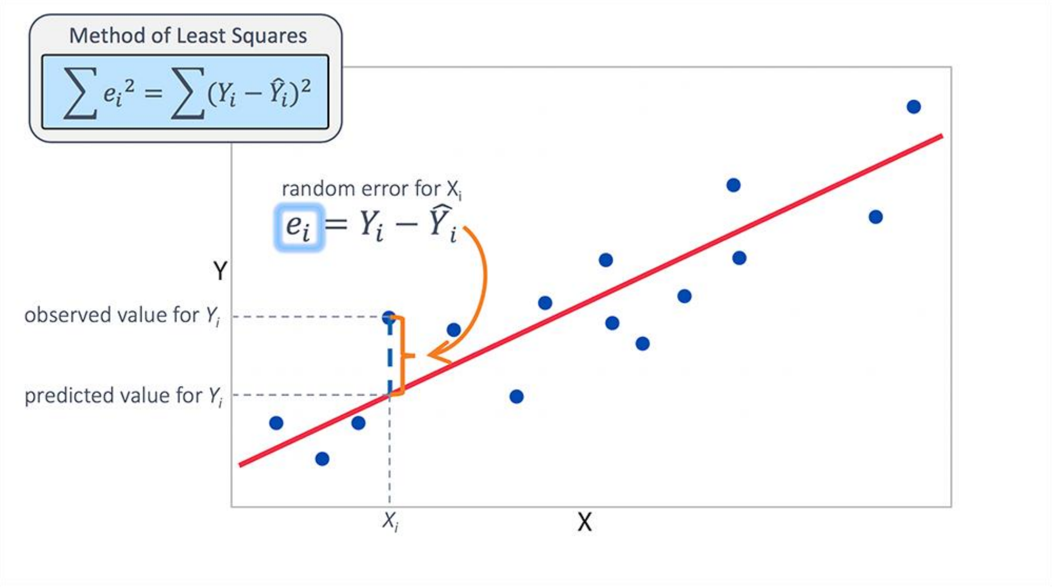
*Equation 4-3. MSE cost function for a Linear Regression model*

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

Sometimes MAE is also used (less sensitive to outliers)

*Equation 2-2. Mean absolute error (MAE)*

$$\mathrm{MAE}\left(\mathbf{X}, h\right) = \frac{1}{m} \sum_{i=1}^{m} \left| h\left(\mathbf{x}^{(i)}\right) - y^{(i)} \right|$$



# Linear regression

- Make prediction by computing the weighted sum of features

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

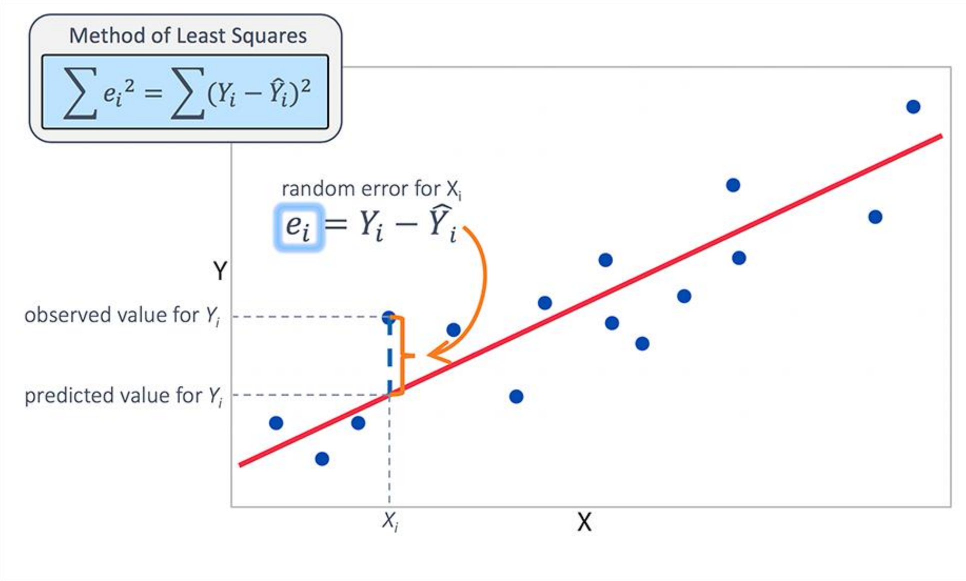
$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x}$$

- Train the model: find a parameter set to best fit the data

- Normal equation

$$\widehat{\boldsymbol{\theta}} = \left(\mathbf{X}^\intercal \mathbf{X}\right)^{-1} \mathbf{X}^\intercal \mathbf{y}$$

$$\boldsymbol{\theta} = E(\widehat{\boldsymbol{\theta}})$$



# Polynomial regression

- When your data is more complex than a straight line
  - Add powers of a feature as new features

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1 x_2$$

Be aware of the combinatory explosion of features!

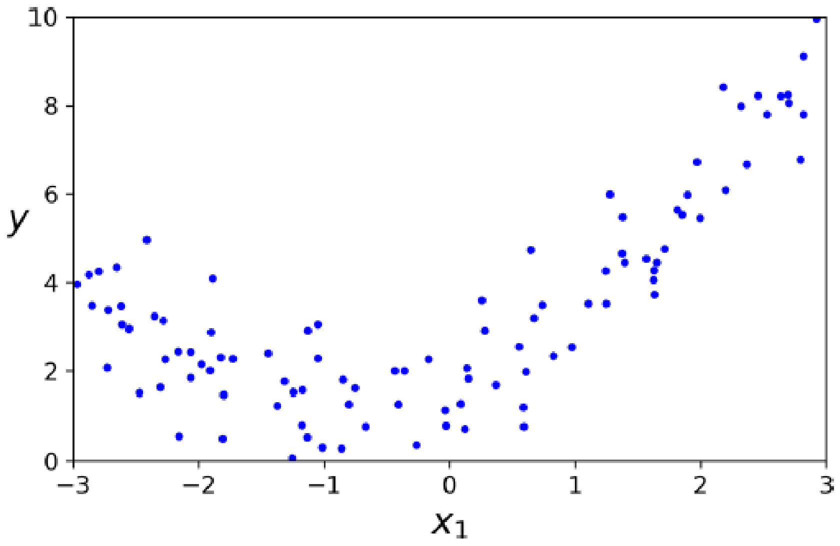
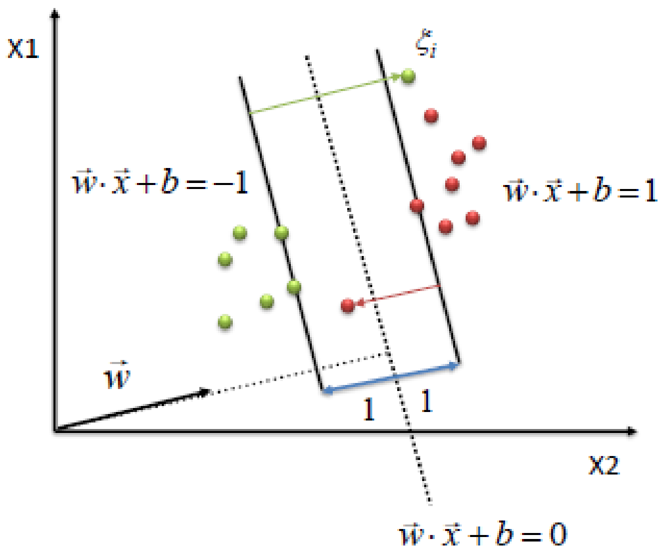



*Figure 4-12. Generated nonlinear and noisy dataset*

# Support vector regression

## SVM classification

fit the largest possible street between two classes while limiting margin violations




Constraint becomes :

$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \ \forall x_i$

$\xi_i \geq 0$

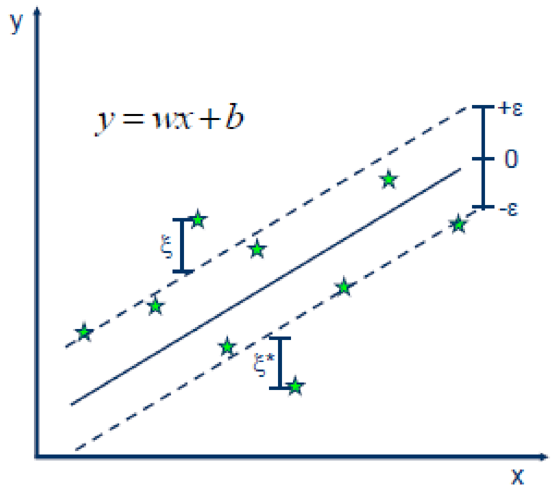
**Objective function**
penalizes for misclassified instances and those within the margin

$$\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

*C* trades-off margin width and misclassifications

## SVM regression

fit as many instances as possible *on* the street while limiting margin violations




- Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$$

- Constraints:

$$y_i - wx_i - b \leq \varepsilon + \xi_i$$
$$wx_i + b - y_i \leq \varepsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0$$

The kernel tricks can be used to solve nonlinear regression problem

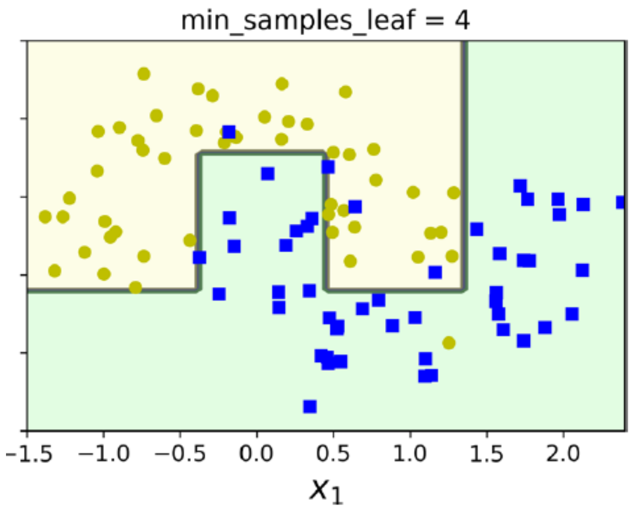


http://www.saedsayad.com/support_vector_machine.htm

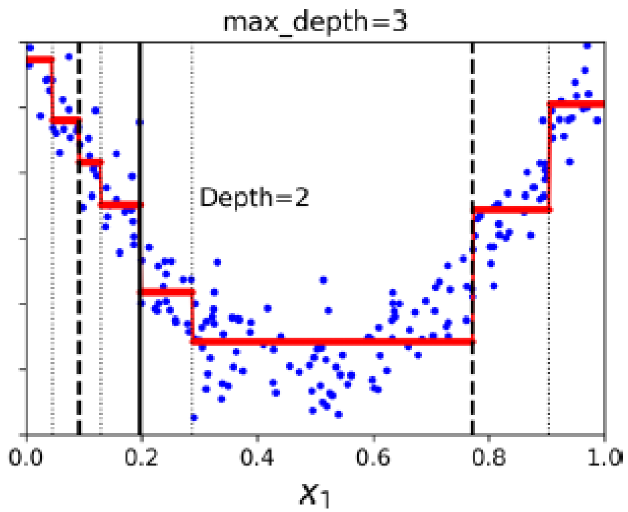
# Tree-based regression

- The predicted value for each region is always the average target value of the instances in that region

Decision tree classification




Decision tree regression



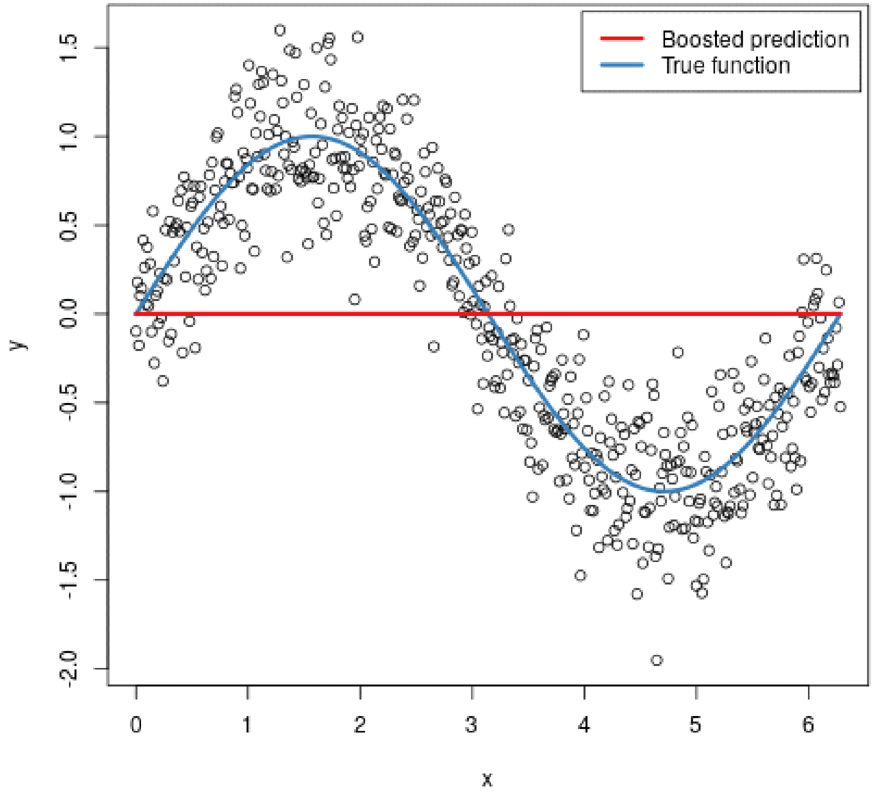

*Equation 6-4. CART cost function for regression*

$$J(k, t_k) = \frac{m_{\text{left}}}{m}\text{MSE}_{\text{left}} + \frac{m_{\text{right}}}{m}\text{MSE}_{\text{right}} \quad \text{where} \begin{cases} \text{MSE}_{\text{node}} = \sum_{i \in \text{node}} \left(\hat{y}_{\text{node}} - y^{(i)}\right)^2 \\ \hat{y}_{\text{node}} = \dfrac{1}{m_{\text{node}}} \sum_{i \in \text{node}} y^{(i)} \end{cases}$$

# Gradient Boosting Machines

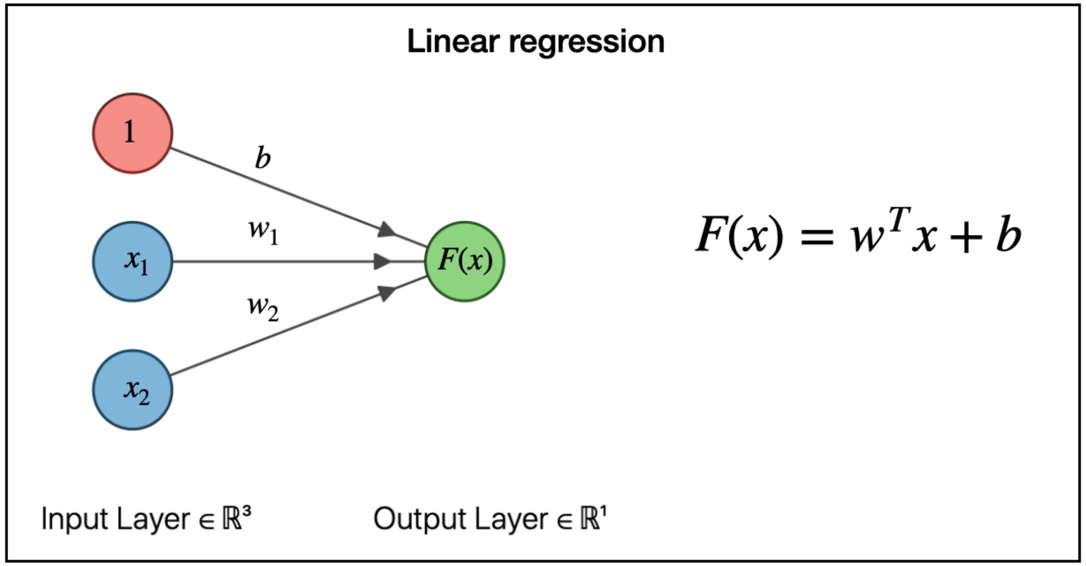
Use decision trees as base-learner, iteratively improves the weak learning model

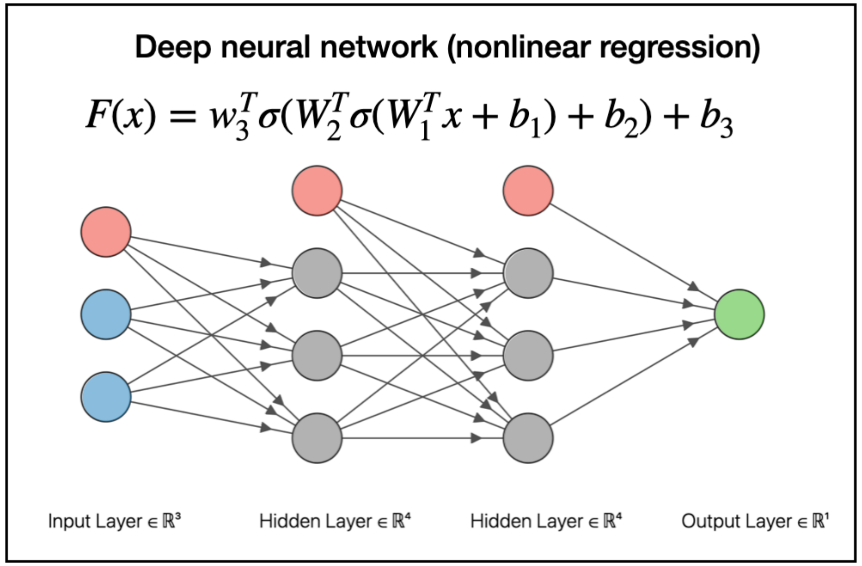
- Fit a decision tree to the data $\hat{y} = F_1(x)$
- Fit the next decision tree to the residuals from previous tree $h_1(x) = y - F_1(x)$
- Add the new tree to the algorithm $F_2(x) = F_1(x) + h_1(x)$
- Repeat this process until some mechanism tells us to stop (e.g. cross validation)



# Neural Network

- A universal function approximator
  - Loss function: MSE
  - No sigmoid activation function at the output node

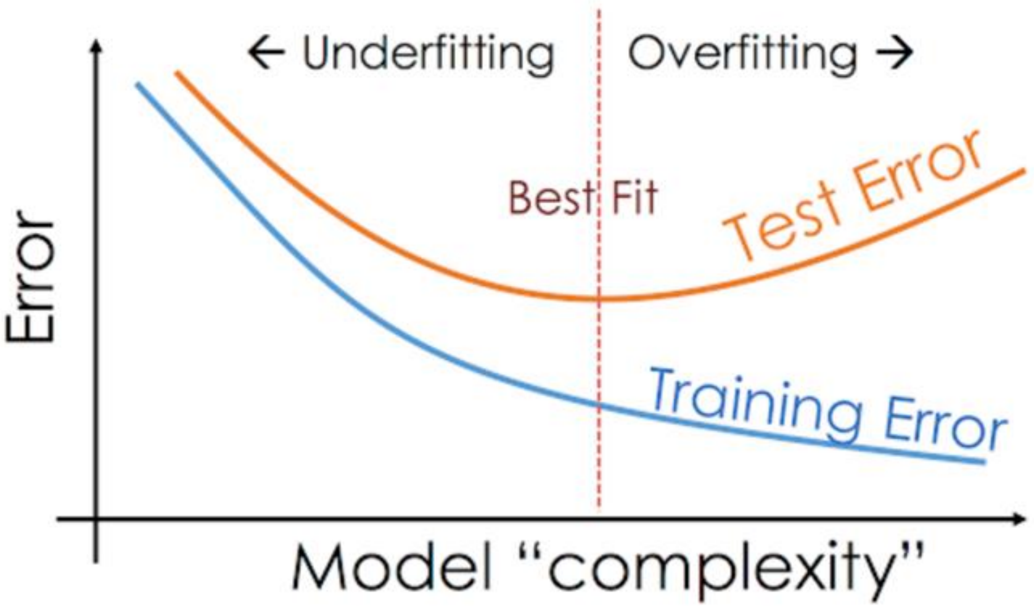





https://joshuagoings.com/2020/05/05/neural-network/

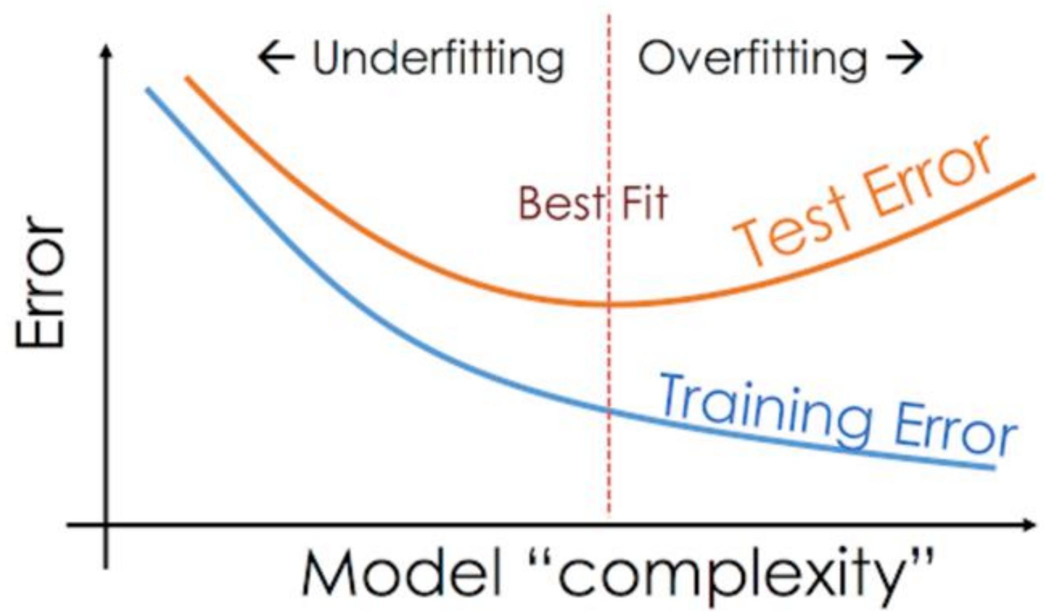
# Over/underfit and bias-variance tradeoff

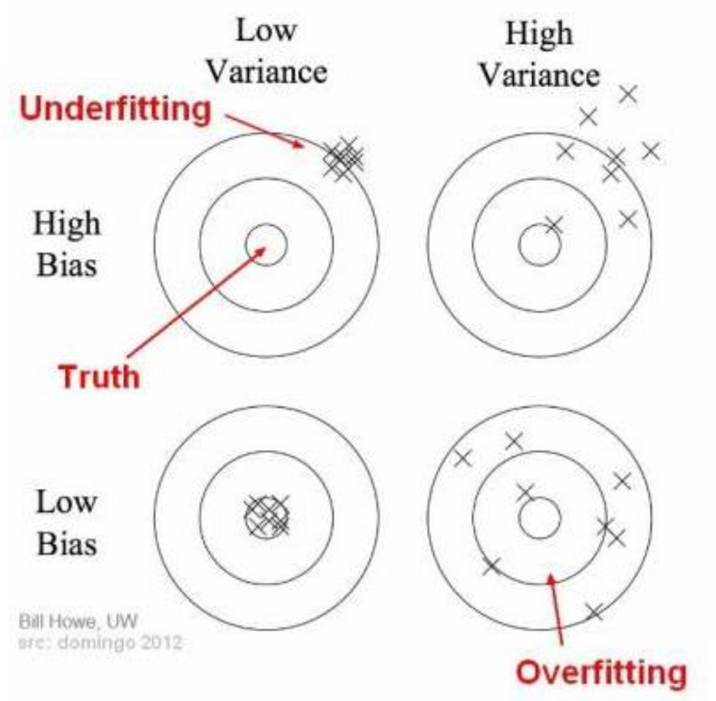





https://medium.com/@minions.k/underfit-and-overfit-explained-8161559b37db

# Over/underfit and bias-variance tradeoff



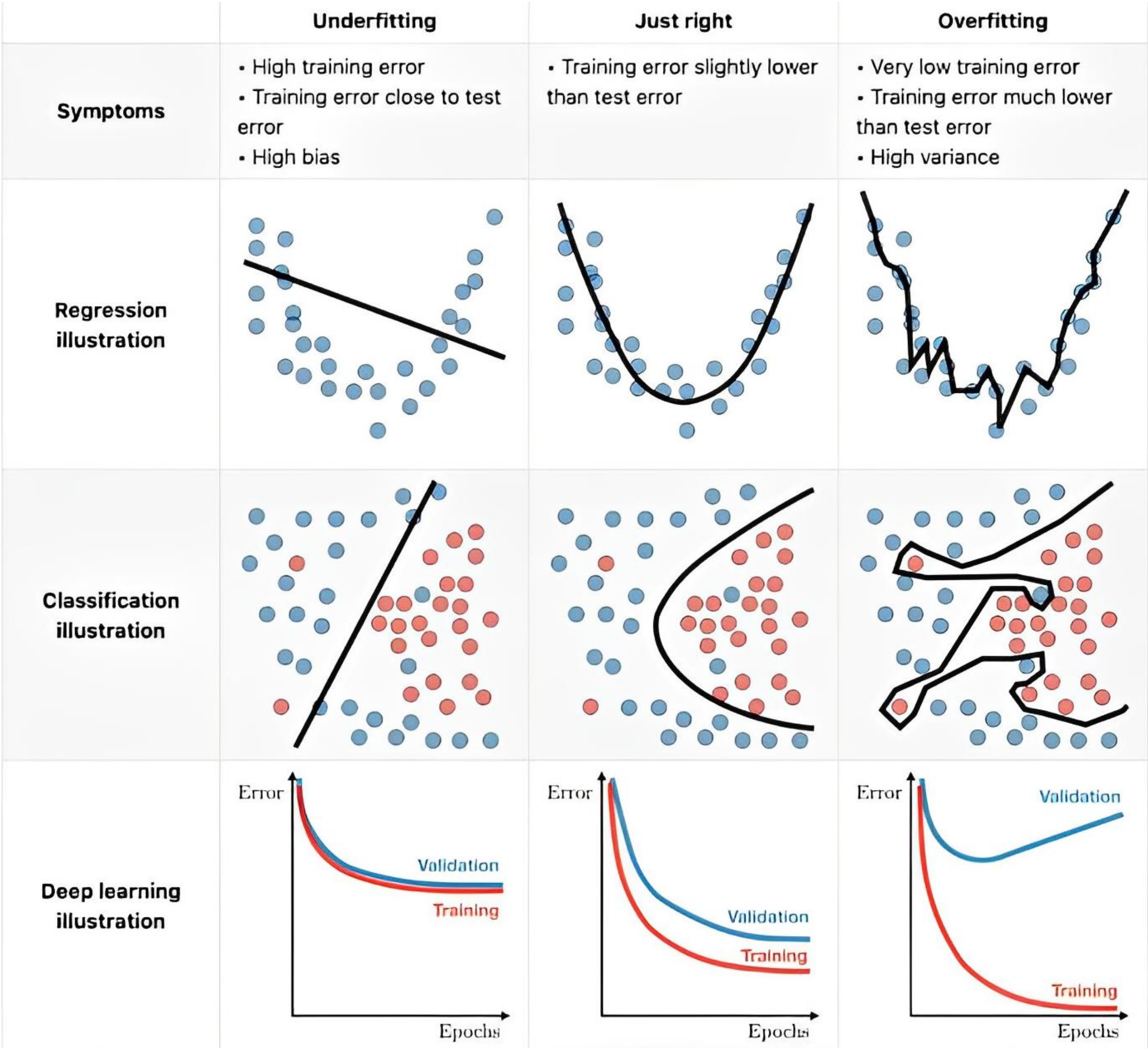




The bias-variance trade off



https://medium.com/@minions.k/underfit-and-overfit-explained-8161559b37db

# Underfit & Overfit




(too simple to explain the variance)

(forcefitting -- too good to be true)

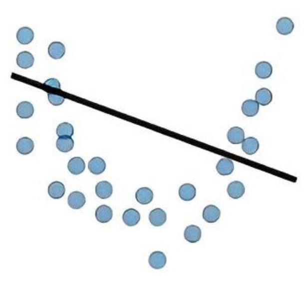
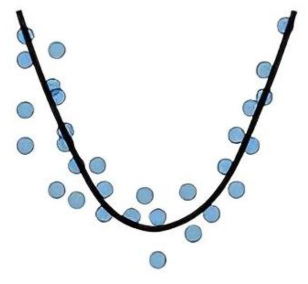
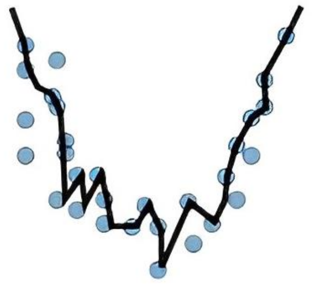
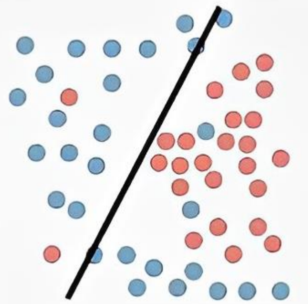
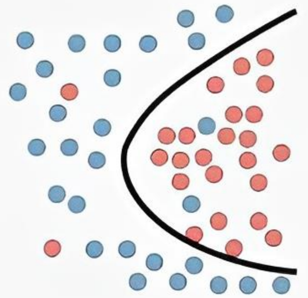
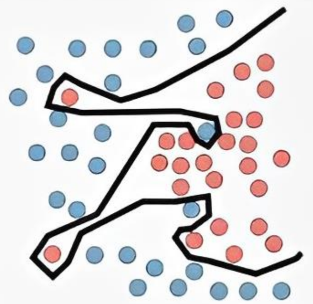
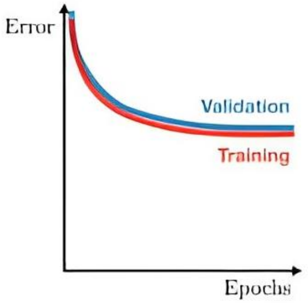
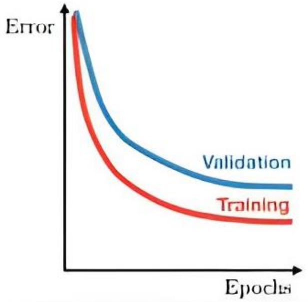
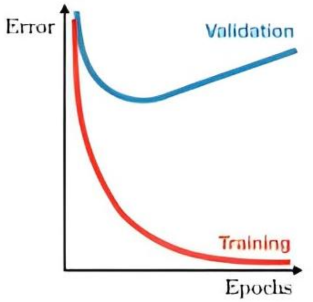
## Symptom and solutions



https://www.kaggle.com/getting-started/180365

# Underfit & Overfit

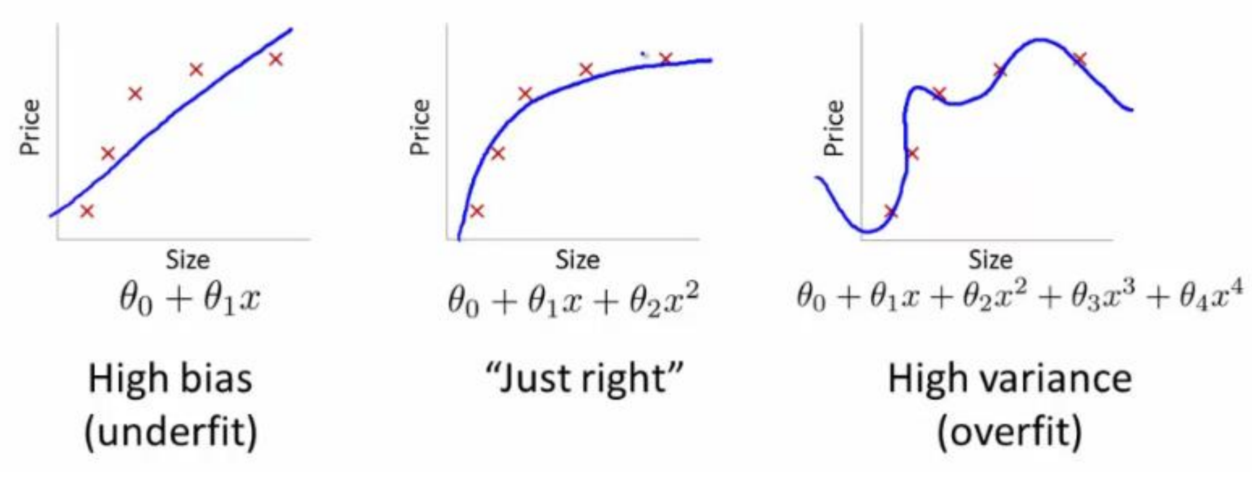
Symptom and solutions

| | Underfitting | Just right | Overfitting |
|---|---|---|---|
| **Symptoms** | • High training error<br>• Training error close to test error<br>• High bias | • Training error slightly lower than test error | • Very low training error<br>• Training error much lower than test error<br>• High variance |
| **Regression illustration** |  |  |  |
| **Classification illustration** |  |  |  |
| **Deep learning illustration** |  |  |  |
| **Possible remedies** | • Complexify model<br>• Add more features<br>• Train longer | | • Perform regularization<br>• Get more data |



https://www.kaggle.com/getting-started/180365

# Regularization techniques in regression




- Regularization: to put constrains on the model
  - The fewer degrees of freedom, the harder it overfits
  - In reality, it's common to have d >> n, regularization is usually needed



https://www.endtoend.ai/blog/bias-variance-tradeoff-in-reinforcement-learning/

# 3 typical constrains in regression

- Lasso (L1 penalty)

$$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + \alpha \sum_{i=1}^{n} |\theta_i|$$

- Ridge (L2 penalty)

$$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + \alpha \frac{1}{2} \sum_{i=1}^{n} {\theta_i}^2$$

This forces the learning algorithm to not only fit the data but also keep the model weights as small as possible!

- Elastic-net

$$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + r\alpha \sum_{i=1}^{n} |\theta_i| + \frac{1-r}{2} \alpha \sum_{i=1}^{n} {\theta_i}^2$$

# 3 typical constrains in regression

- Lasso (L1 penalty)

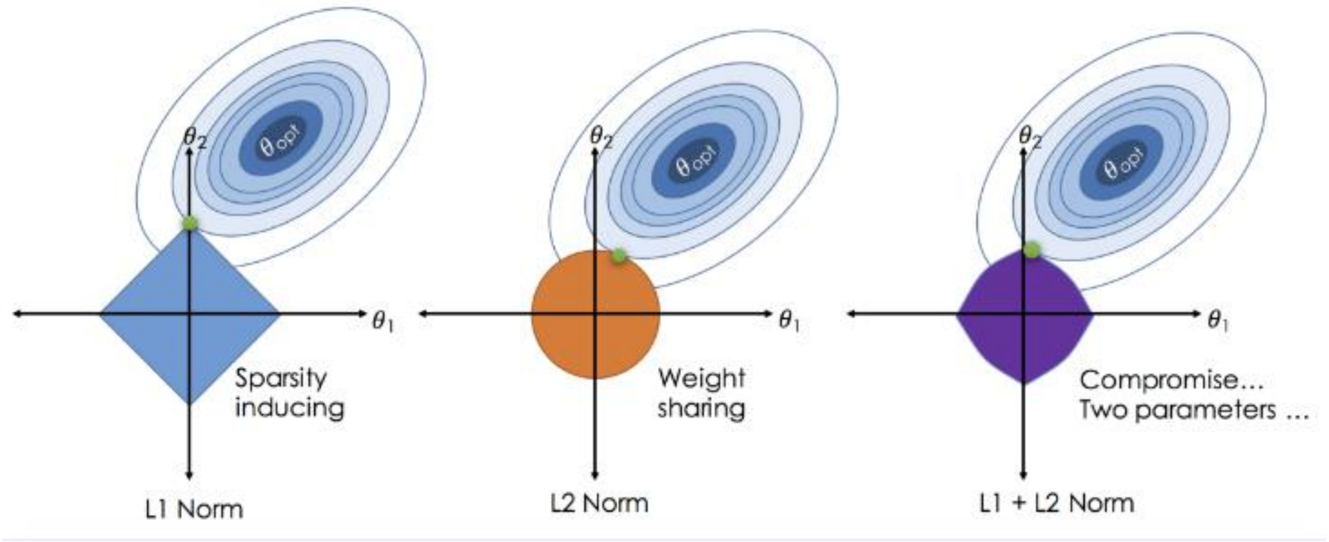
$$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + \alpha \sum_{i=1}^{n} |\theta_i|$$

- Ridge (L2 penalty)

$$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^{\,2}$$

- Elastic-net

$$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + r\alpha \sum_{i=1}^{n} |\theta_i| + \frac{1-r}{2} \alpha \sum_{i=1}^{n} \theta_i^{\,2}$$






https://towardsdatascience.com/from-linear-regression-to-ridge-regression-the-lasso-and-the-elastic-net-4eaecaf5f7e6

# Summary

- 6 regression algorithms
- MSE/MAE loss function
- Over/Underfit and Bias-variance tradeoff
- Regularization
  - LASSO
  - Ridge
  - Elastic net

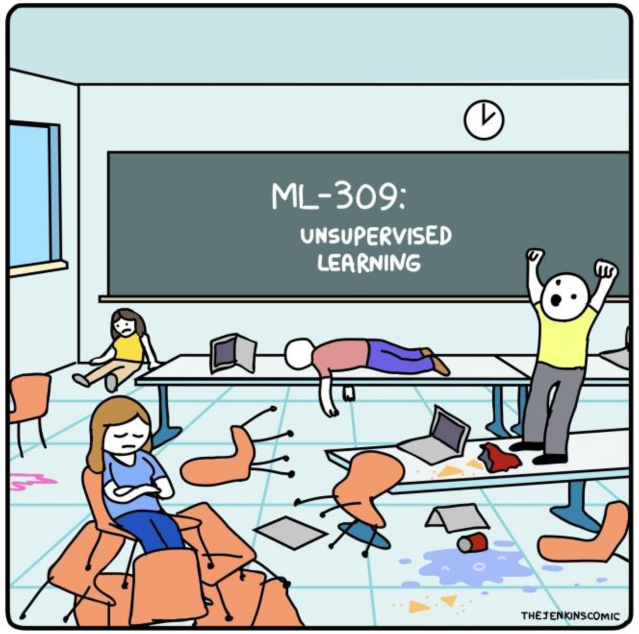
# Unsupervised learning

- Dimension reduction
  - PCA
  - t-SNE
  - Autoencoder

  …

- Clustering
  - K-means
  - Hierarchical
  - DBSCAN

  …



# Dimension reduction

- High dimensional: a blessing and a curse

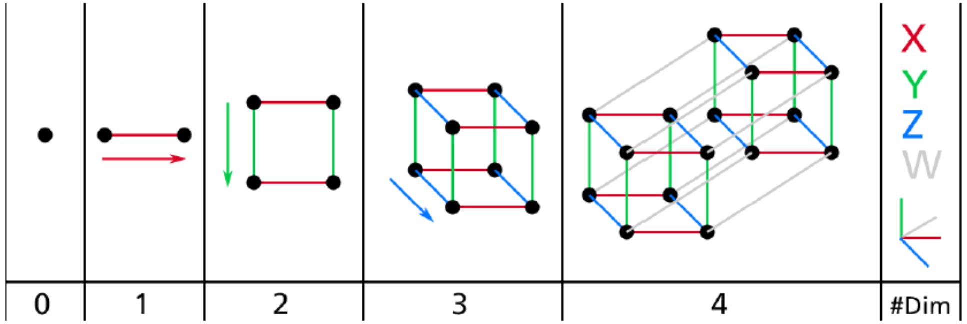


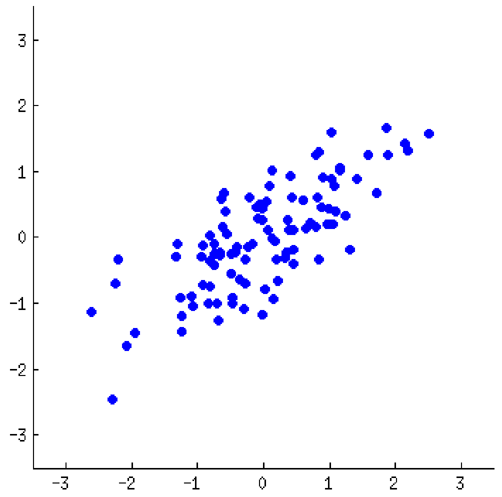

*Figure 8-1. Point, segment, square, cube, and tesseract (0D to 4D hypercubes)*[2]

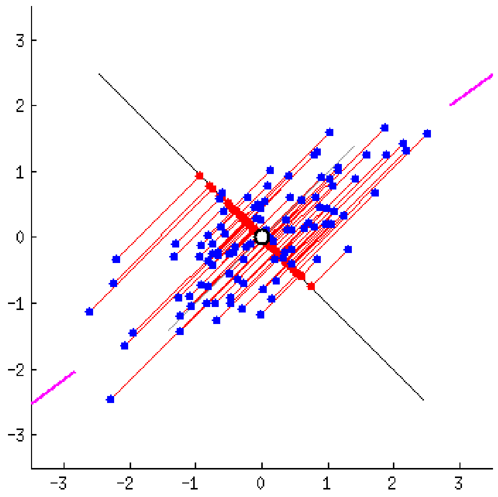
- Curse of dimensionality:
  - Many machine learning algorithm have hard time to find good solutions in high dimensional setting
  - The training can be extremely slow when dimension (number of features) is high

# Principal component analysis (PCA)

Preserving the maximum variance

- Data compression
- Noise removal
- Data visualization






https://sagarsaha455.medium.com/pca-for-visualization-and-dimension-reduction-14492e2acf2b

# Principal component analysis (PCA)

Preserving the maximum variance

- Data compression
- Noise removal
- Data visualization

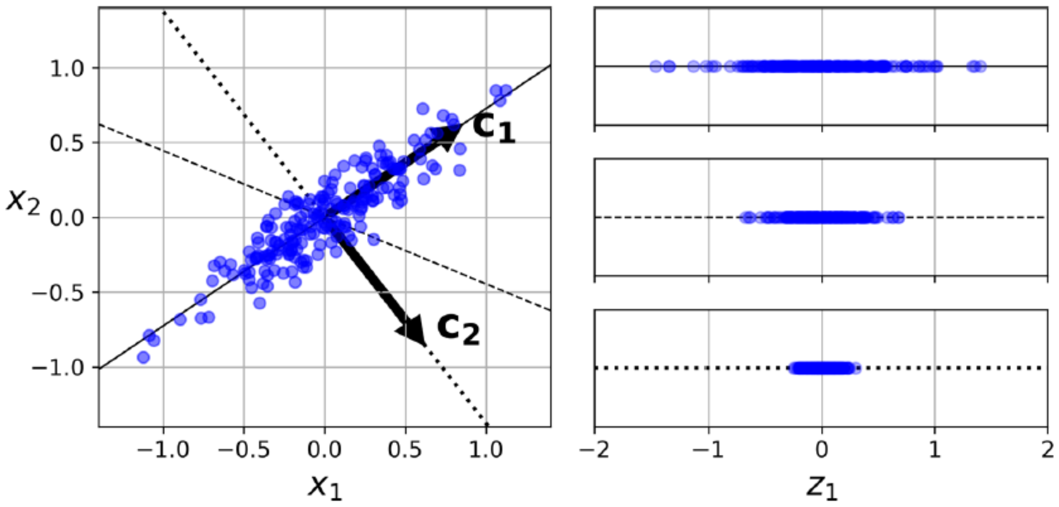



*Figure 8-7. Selecting the subspace to project on*

# Principal component analysis (PCA)

Preserving the maximum variance

- Data compression
- Noise removal
- Data visualization

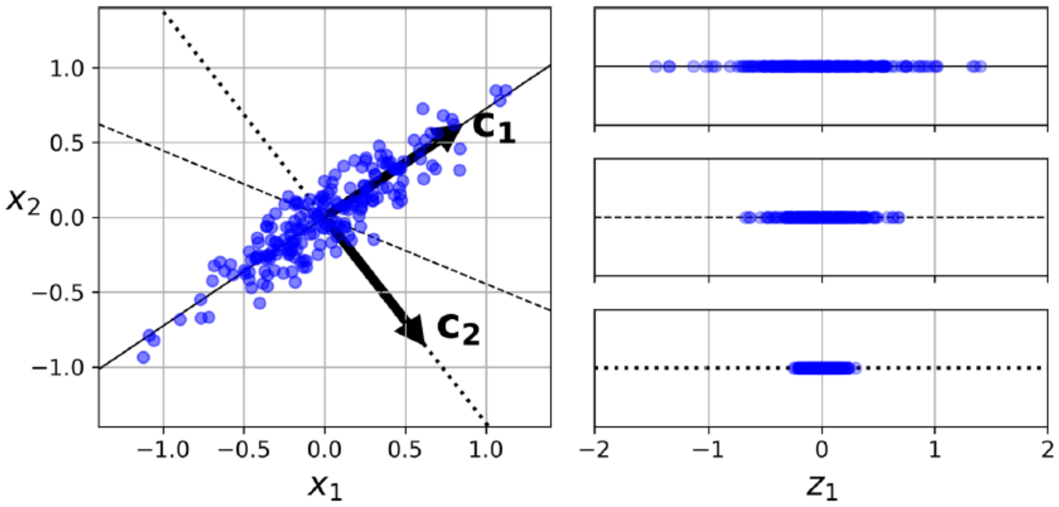


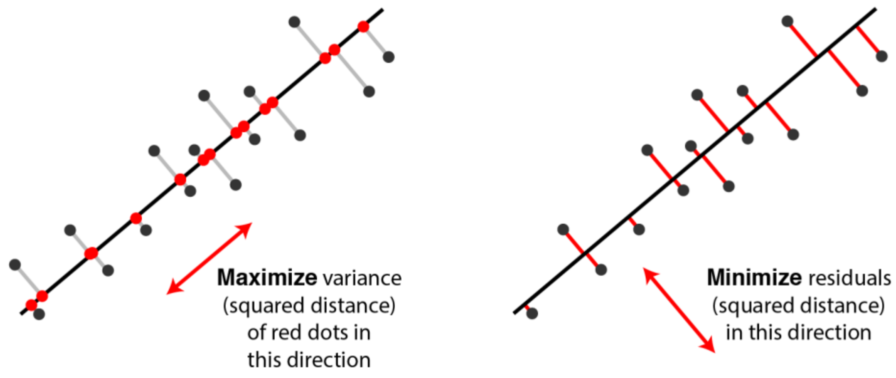

*Figure 8-7. Selecting the subspace to project on*

Goal:

- *find an orthogonal set of r linear basis vectors $w_j \in R^d$ and the corresponding score $z_i \in R^r$, such that we minimize the average* **reconstruction error**

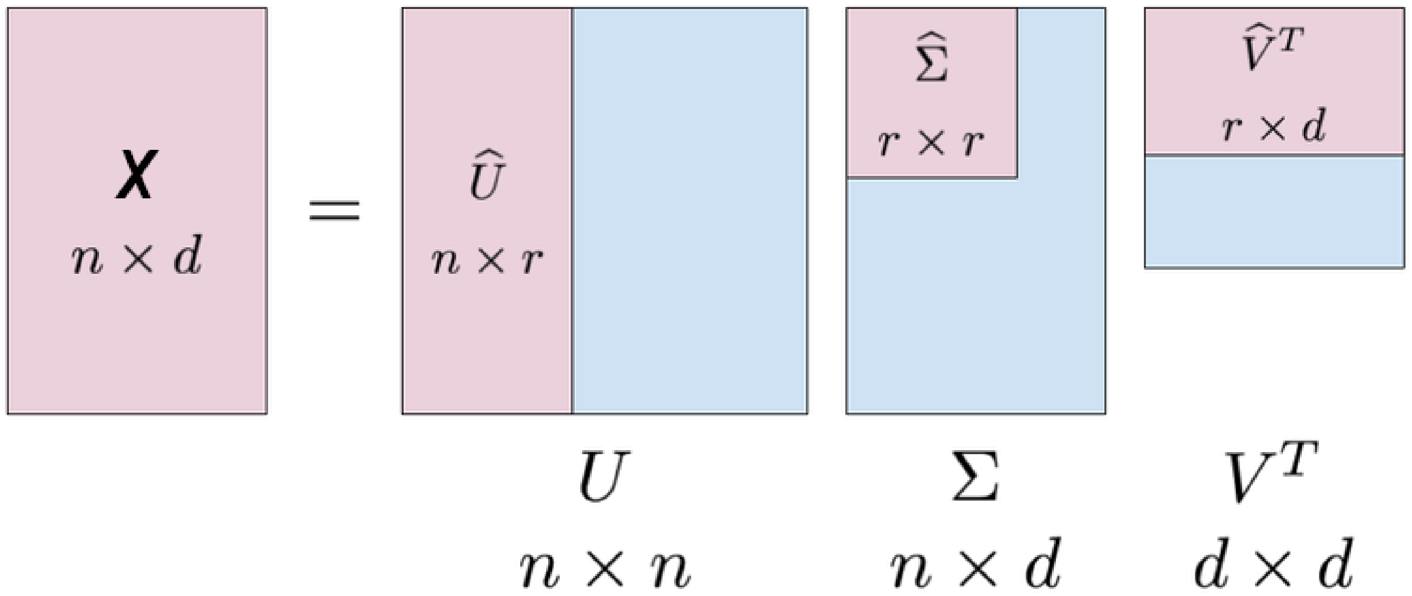
$$J(\mathbf{W}, \mathbf{Z}) = \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2$$

$$where\ \hat{\mathbf{x}}_i = \mathbf{W}\mathbf{z}_i$$



# Principal component analysis (PCA)

- PCA achieved by Singular value decomposition (SVD)




$\hat{U}\,\hat{\Sigma}$: Principal component scores

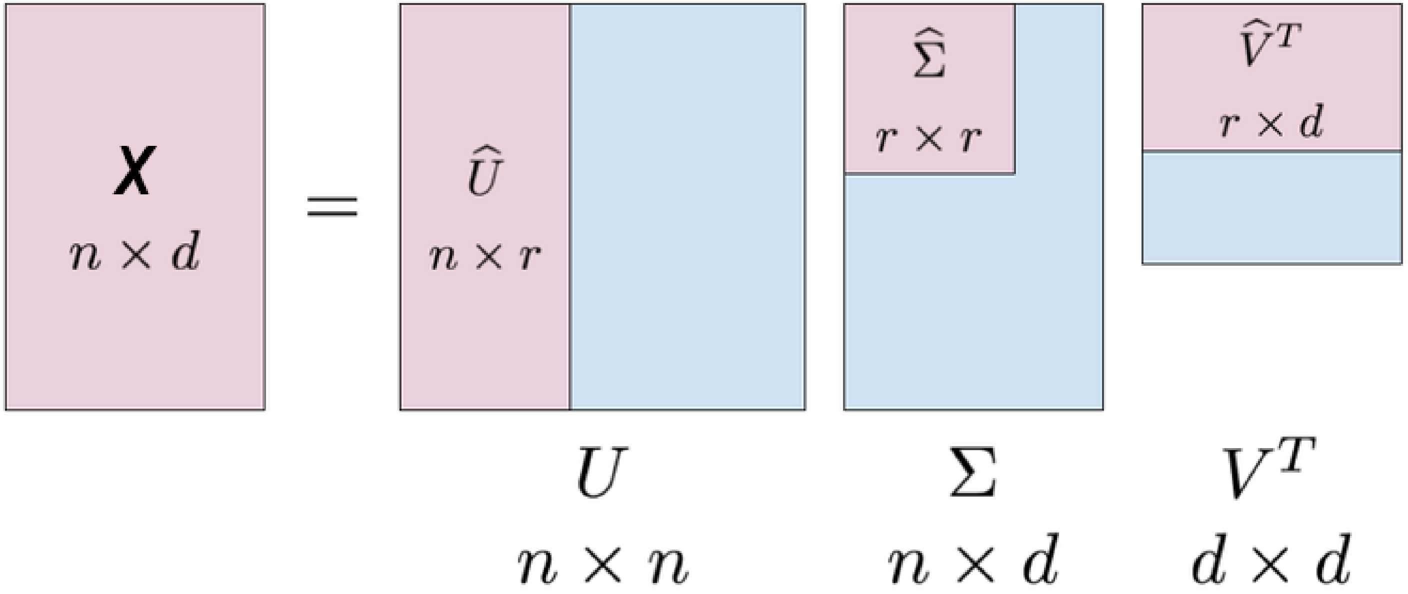
$\hat{V}$: Principal directions



https://intoli.com/blog/pca-and-svd/

# Principal component analysis (PCA)

- PCA achieved by Singular value decomposition (SVD)




- Project data on to the reduced dimension space

$$\boldsymbol{X}_{d-proj} = \boldsymbol{X}\widehat{\boldsymbol{V}}$$



https://intoli.com/blog/pca-and-svd/

# t-Distributed Stochastic Neighbor Embedding (t-SNE)

- Nonlinear dimensionality reduction
  - PCA uses the global covariance matrix
  - t-SNE focus more on the local structure

- Core algorithm
  - In high dimensional space

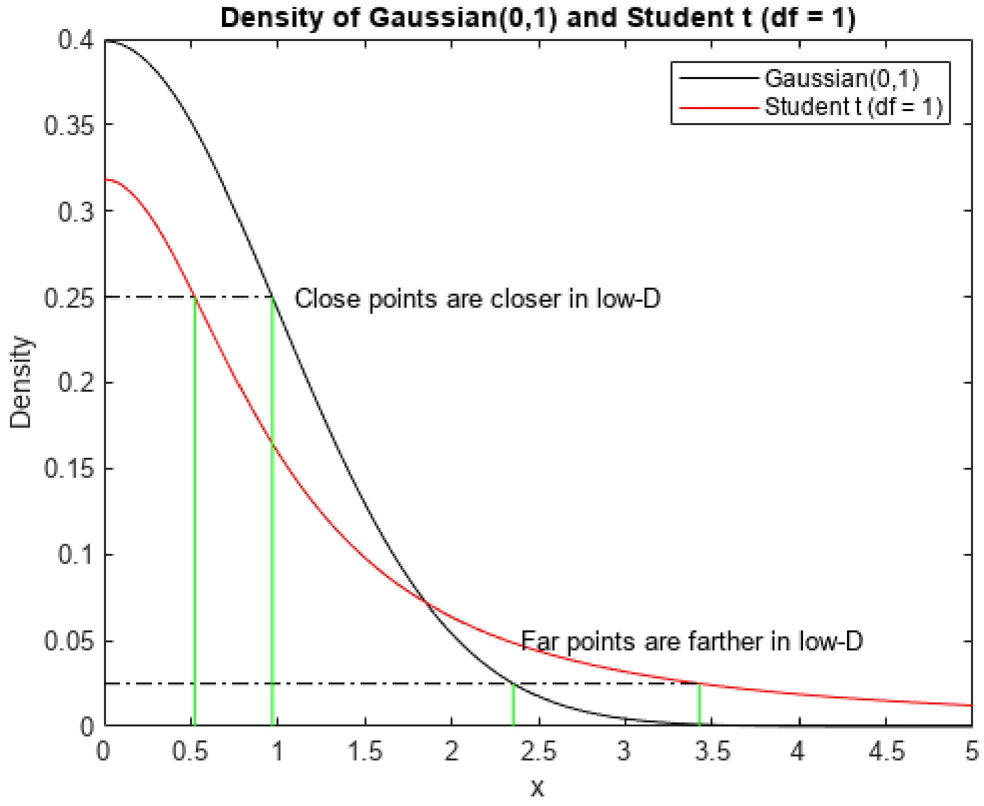
$$p_{ij} = \frac{\exp(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2)}{\sum_{k \neq l} \exp(-\left\|x_k - x_l\right\|^2 / 2\sigma_i^2)}$$

  - In lower dimensional space

$$q_{ij} = \frac{(1 + \left\|y_i - y_j\right\|^2)^{-1}}{\sum_{k \neq l}(1 + \left\|y_k - y_l\right\|^2)^{-1}}$$

  - Try to minimize the difference of 2 distributions

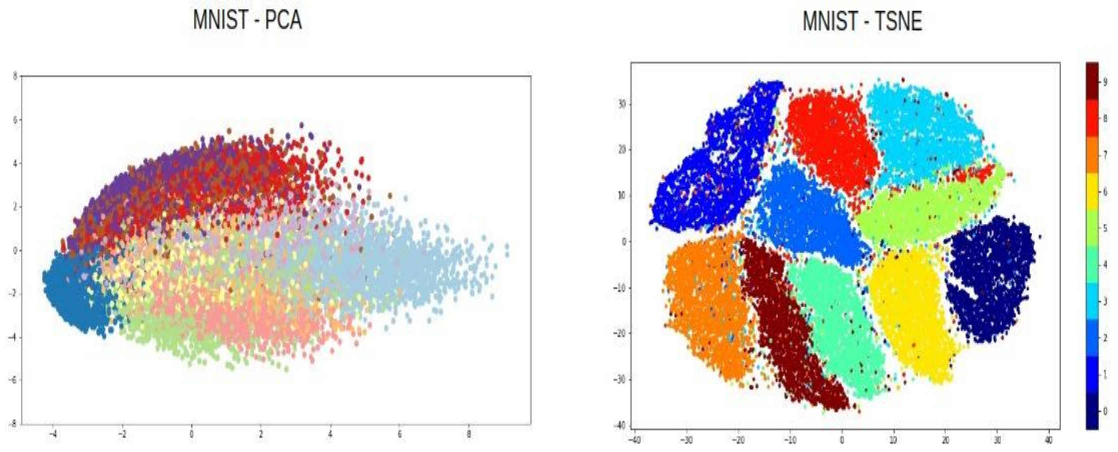
$$C = D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$




https://www.mathworks.com/help/stats/t-sne.html

# t-Distributed Stochastic Neighbor Embedding (t-SNE)

- Nonlinear dimensionality reduction
  - PCA uses the global covariance matrix
  - t-SNE focus more on the local structure




- Core algorithm
  - In high dimensional space

$$p_{ij} = \frac{\exp(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2)}{\sum_{k \neq l} \exp(-\left\|x_k - x_l\right\|^2 / 2\sigma_i^2)}$$

  - In lower dimensional space

$$q_{ij} = \frac{(1 + \left\|y_i - y_j\right\|^2)^{-1}}{\sum_{k \neq l} (1 + \left\|y_k - y_l\right\|^2)^{-1}}$$

  - Try to minimize the difference of 2 distributions

$$C = D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

# Autoencoder

- Nonlinear dimension reduction with NN

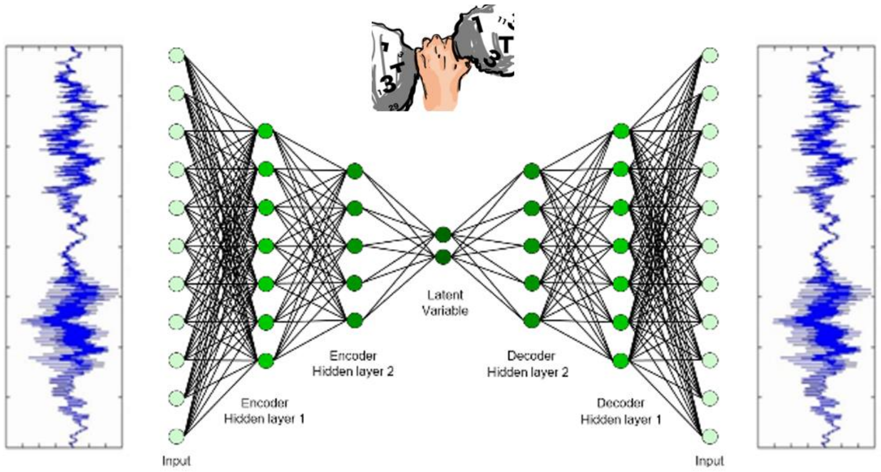


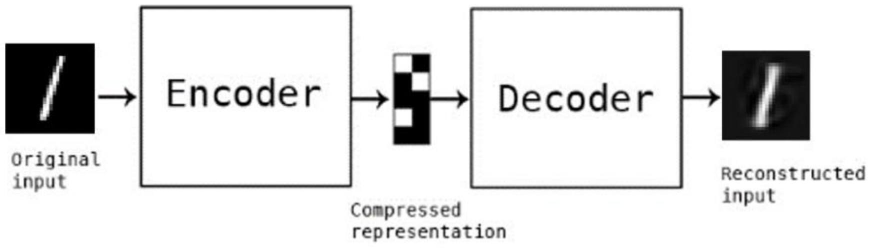

https://i-systems.github.io/teaching/ML/iNotes/15_Autoencoder.html

# Autoencoder

- Nonlinear dimension reduction with NN

- Key idea:
  - An encoder function $z = f(x)$
  - A Decoder function $x = g(z)$
  - Learn to set $g\big(f(x)\big) \cong x$





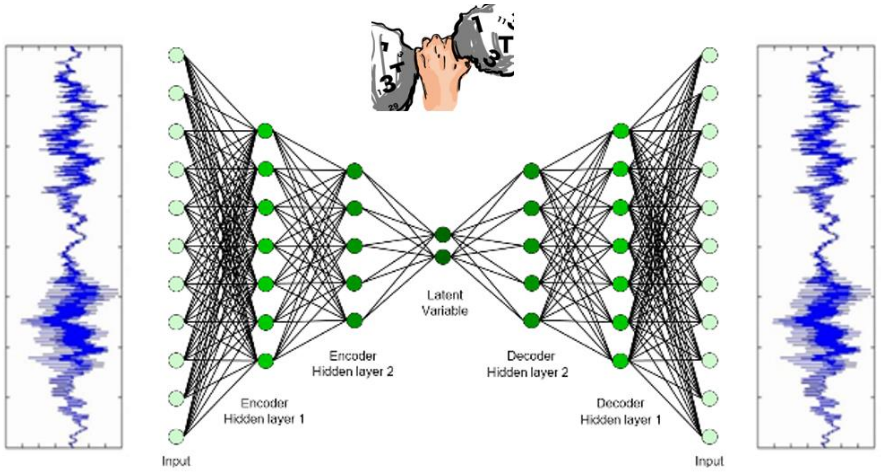

https://i-systems.github.io/teaching/ML/iNotes/15_Autoencoder.html

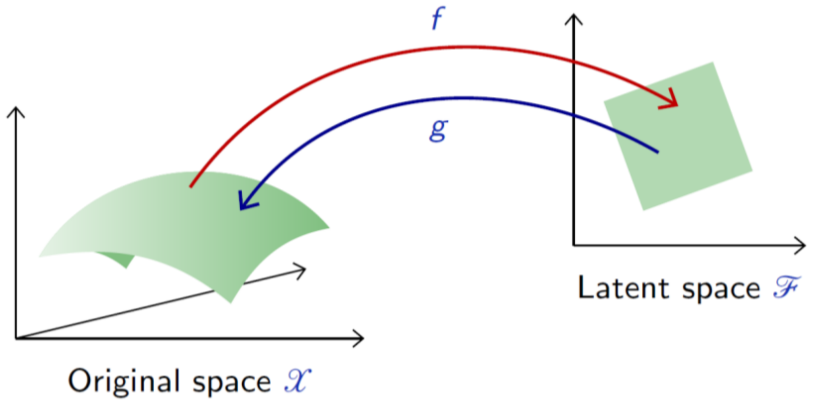
# Autoencoder

- Nonlinear dimension reduction with NN

- Key idea:
  - An encoder function $z = f(x)$
  - A Decoder function $x = g(z)$
  - Learn to set $g\big(f(x)\big) \cong x$

- Loss function: reconstruction error
  - Minimize $||X - g \circ f(X)||^2$






https://i-systems.github.io/teaching/ML/iNotes/15_Autoencoder.html

# Clustering

The vast majority of data is unlabeled. The clustering algorithm tries to identify similar instances and assigning them to *clusters*

- K-means
- Hierarchical clustering
- DBSCAN

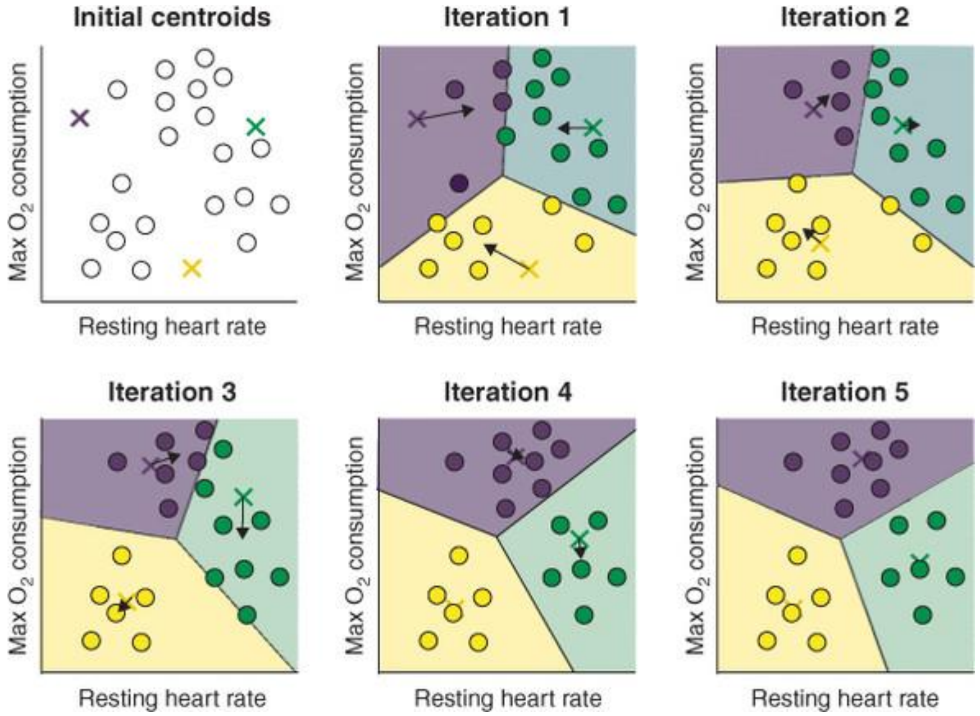
# K-means

- Look for instances centered around a particular point (centroid)

- Core algorithm

**Algorithm 1** $k$-means algorithm

1: Specify the number $k$ of clusters to assign.
2: Randomly initialize $k$ centroids.
3: **repeat**
4: **expectation:** Assign each point to its closest centroid.
5: **maximization:** Compute the new centroid (mean) of each cluster.
6: **until** The centroid positions do not change.






https://pub.towardsai.net/k-means-clustering-from-scratch-4e357ac4716f

# K-means

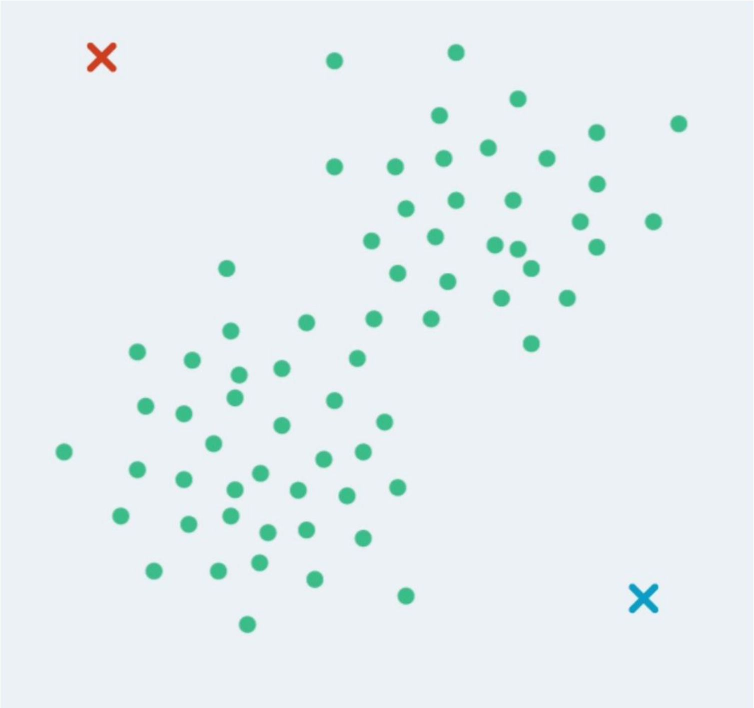
- Look for instances centered around a particular point (centroid)

- Core algorithm

**Algorithm 1** $k$-means algorithm

Specify the number $k$ of clusters to assign.
Randomly initialize $k$ centroids.
**repeat**
    **expectation:** Assign each point to its closest centroid.
    **maximization:** Compute the new centroid (mean) of each cluster.
**until** The centroid positions do not change.

The algorithm is guaranteed to converge in a finite number of steps (usually quite small).



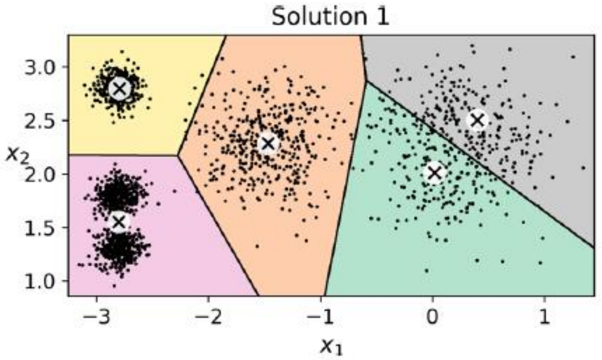
# K-means

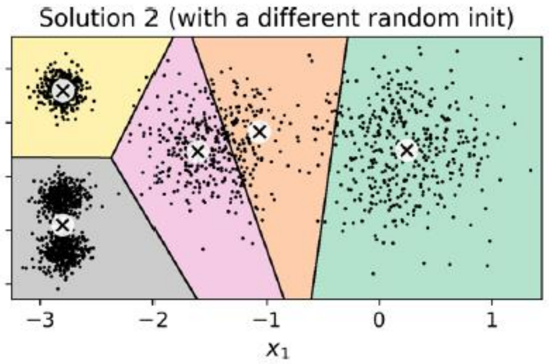
- Look for instances centered around a particular point (centroid)

- Core algorithm

**Algorithm 1** $k$-means algorithm

```
Specify the number k of clusters to assign.
Randomly initialize k centroids.
repeat
    expectation: Assign each point to its closest centroid.
    maximization: Compute the new centroid (mean) of each cluster.
until The centroid positions do not change.
```




The algorithm is guaranteed to converge in a finite number of steps (usually quite small). But it might not converge to a right solution (local optimum)

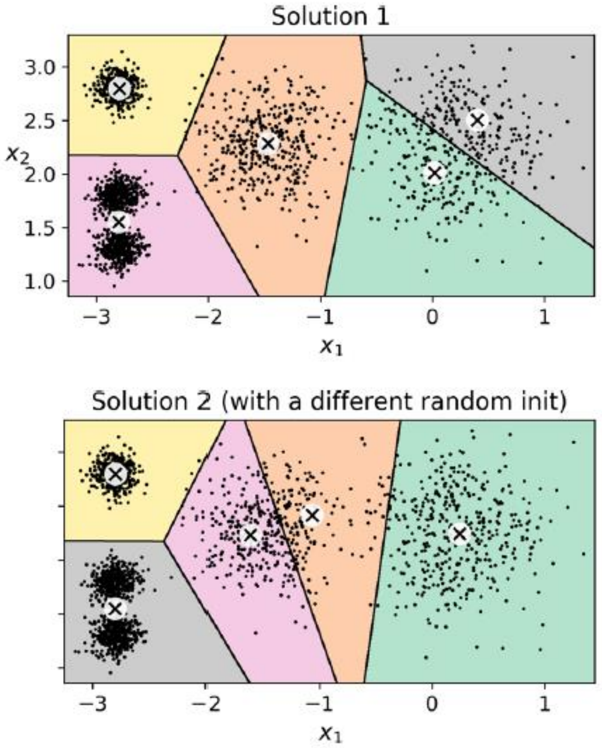
# K-means

- Look for instances centered around a particular point (centroid)

- Core algorithm

**Algorithm 1** $k$-means algorithm

1: Specify the number $k$ of clusters to assign.
2: Randomly initialize $k$ centroids.
3: **repeat**
4: **expectation:** Assign each point to its closest centroid.
5: **maximization:** Compute the new centroid (mean) of each cluster.
6: **until** The centroid positions do not change.




The algorithm is guaranteed to converge in a finite number of steps (usually quite small). But it might not converge to a right solution (local optimum)

Solution:

- If you know the approximate position of centroids, manually set them during initialization
- Run the algorithm multiple times with different random initialization, keep the best solution

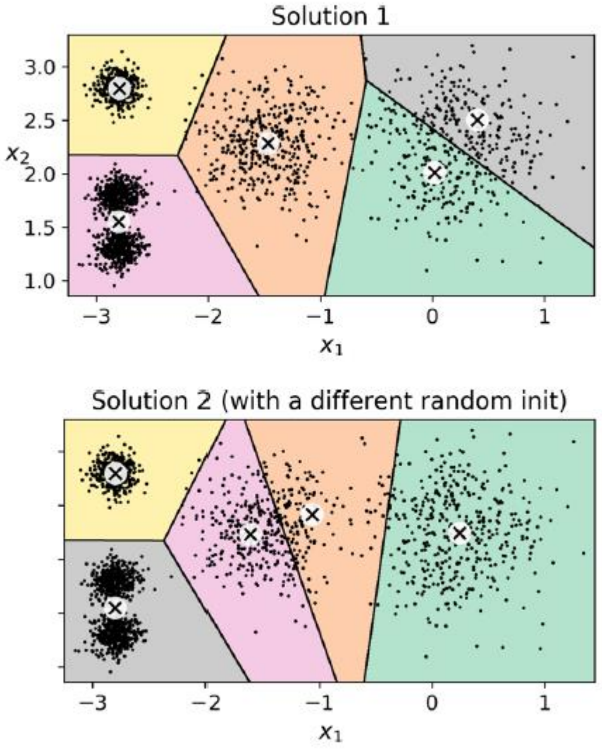
# K-means

- Look for instances centered around a particular point (centroid)

- Core algorithm

**Algorithm 1** $k$-means algorithm

Specify the number $k$ of clusters to assign.
Randomly initialize $k$ centroids.
**repeat**
    **expectation:** Assign each point to its closest centroid.
    **maximization:** Compute the new centroid (mean) of each cluster.
**until** The centroid positions do not change.



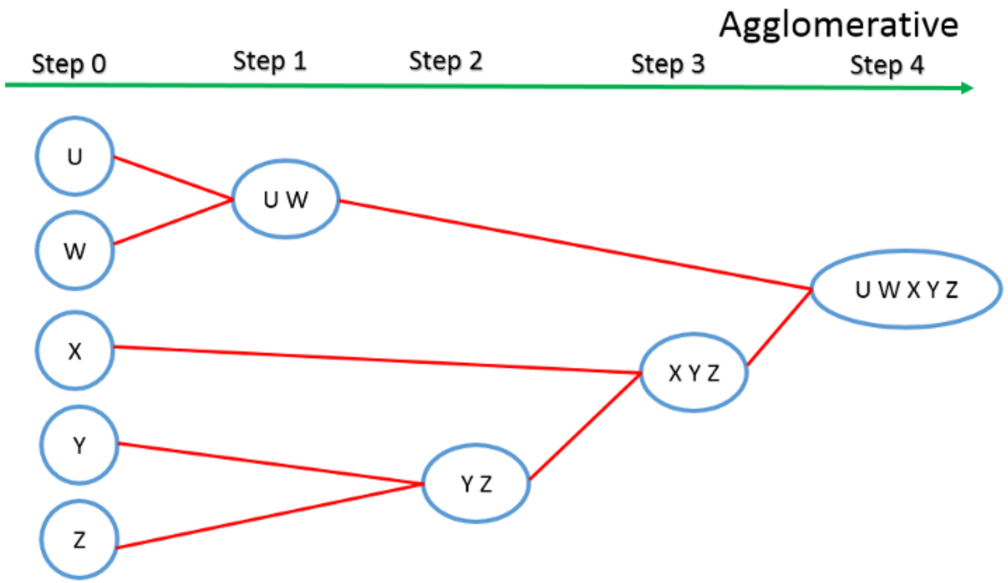

The algorithm is guaranteed to converge in a finite number of steps (usually quite small). But it might not converge to a right solution (local optimum)

Note:
- K-means favor cluster with similar size, it doesn't perform well with varying sizes
- Need to specify the number of clusters

# Hierarchical (Agglomerative) clustering

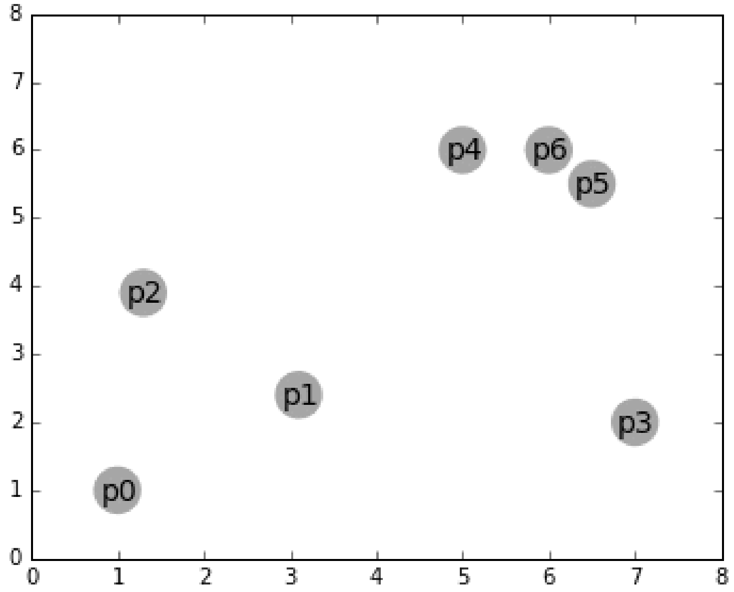
- A bottom-Up approach to connect the nearest pair of clusters

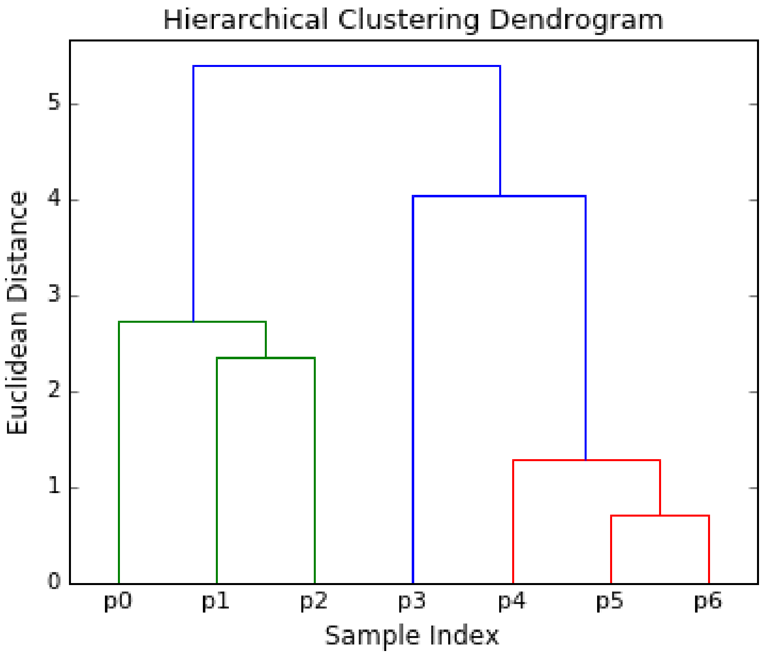
$t = 0$
Choose $R_0 = [C_i = x_i, i = 1, \ldots, N]$ as initial clustering
Repeat
    $t = t + 1$
    Find the closest clusters $C_i, C_j$ in the existing clustering $R_{t-1}$ such that
        $g(C_i, C_j) = \max_{r,s}(C_r, C_s)$ if g is similarity function
        $g(C_i, C_j) = \min_{r,s}(C_r, C_s)$ if g is dissimilarity function
    Define $C_q = C_i \cup C_j$ and produce the new clustering $R_t = [R_{t-1} - C_i - C_j] \cup C_q$
Until only one cluster is left



# Hierarchical (Agglomerative) clustering

- A bottom-Up approach to connect the nearest pair of clusters




Scale up well to large number of instance or clusters

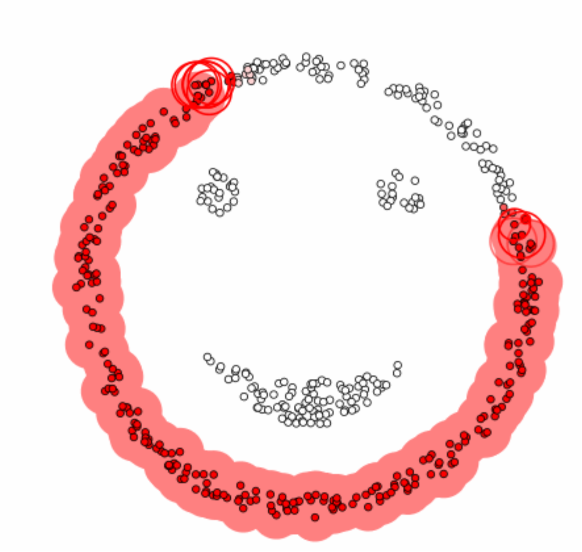
# DBSCAN

- Defines clusters as continuous regions of high density

- Core algorithm

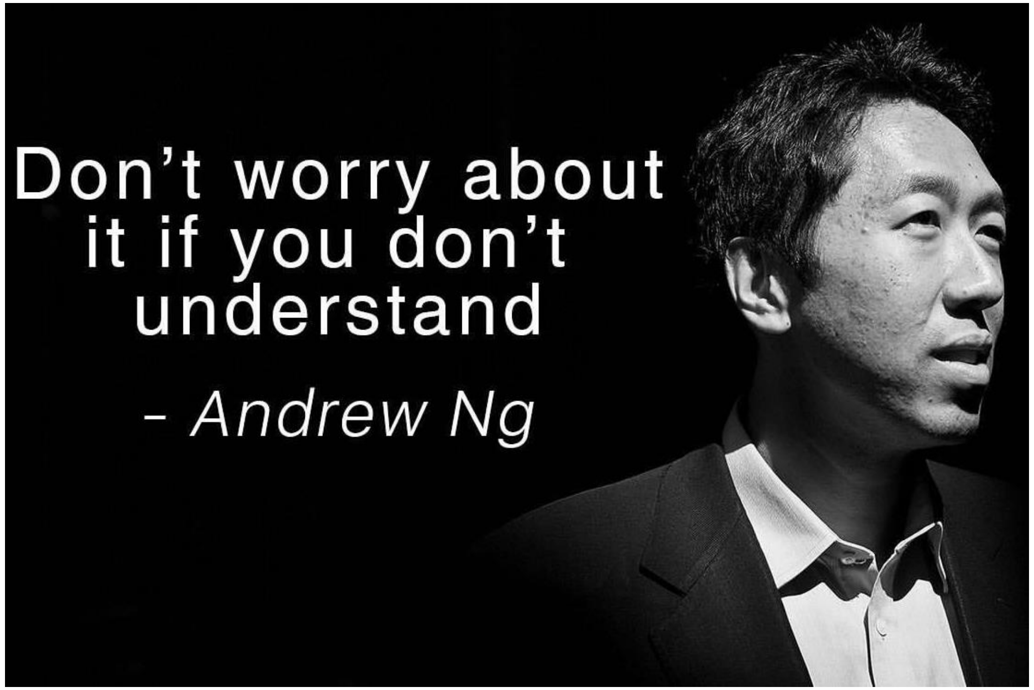
**Algorithm 3 : DBSCAN Clustering**

**Input:** $2D\_Data$ obtained by *Algorithm1* as the input data, |Data| objects to be clustered, the neighborhood radius (ε) and minimum points (μ)
1: Randomly select a point $P$
2: Retrieve all points density-reachable from $P$ based on ε and μ and Similarity Metric (*Algorithm2*)
3: If $P$ is a core point, a cluster is formed.
4: If $P$ is a border point, no points are density-reachable from P and DBSCAN selects the next no-visited point randomly.
5: Continue the procedure until all points have been processed.



- Works pretty well if clusters are dense enough and separated well by low-density regions
- Robust to outliers
- Computation complexity is $O(m \log m)$

# Let's do some practice!





- ➢ `git clone https://github.com/wbvguo/qcbio-ML_w_Python.git`

# Summary

We have knowledge about

- ❑ Machine learning’s definition and categories
- ❑ The workflow for train a machine learning model
- ❑ Rationale of several major machine learning algorithms!
- ❑ Performance measure for evaluating models
- ❑ Challenges in machine learning and potential solutions

And we have experience in

- ❑ Jupyter notebook
- ❑ NumPy, matplotlib, scikit-learn, keras
- ❑ Build, train, evaluate a classifier or regressor
- ❑ Tune hyperparameters
- ❑ Unsupervised learning (PCA, K-Means)

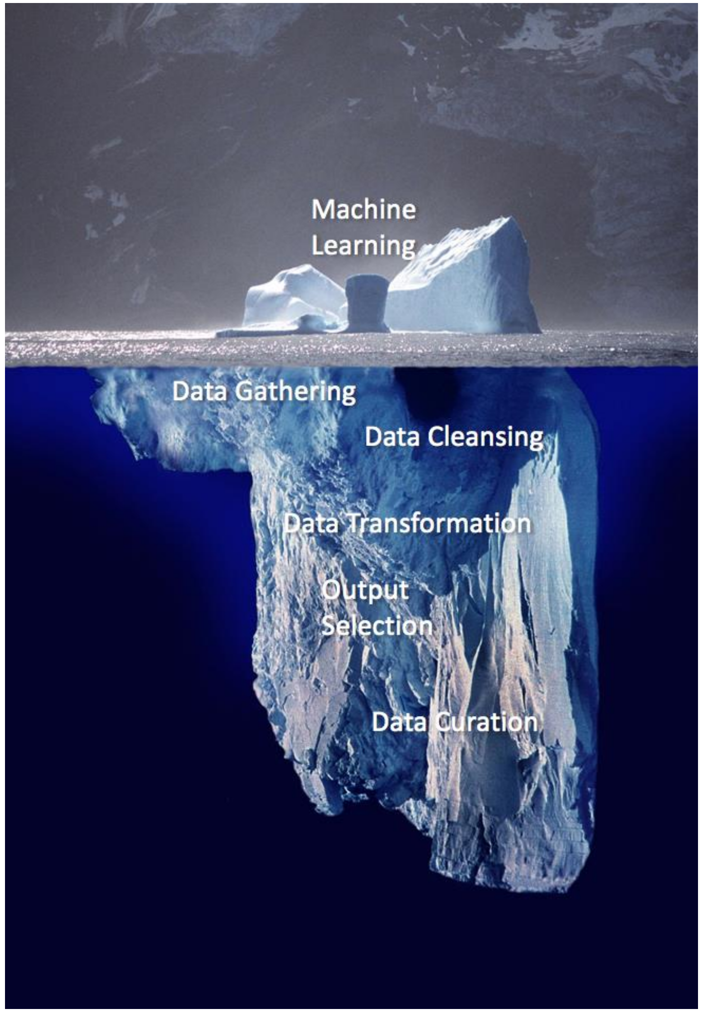
# Summary of algorithms

| Supervised learning | | Unsupervised learning | |
|---|---|---|---|
| Classification | Regression | Dimension reduction | Clustering |
| Logistic regression | Linear regression | PCA | K-means |
| KNN | Polynomial regression | t-SNE | Hierarchical |
| Naïve bayes | SVR | Autoencoder | DBSCAN |
| SVM | Tree-based | | |
| Decision tree | GBM | | |
| Random forest | Neural Network | | |
| Adaboost | | | |
| Gradient boosting | | | |
| Neural network | | | |

# Beyond machine learning algorithms

- Problem formulation
- Data cleansing
- Feature engineering
  - Feature encoding
  - Imputation: Missing data handling
  - Transformation/Normalization/Standardization

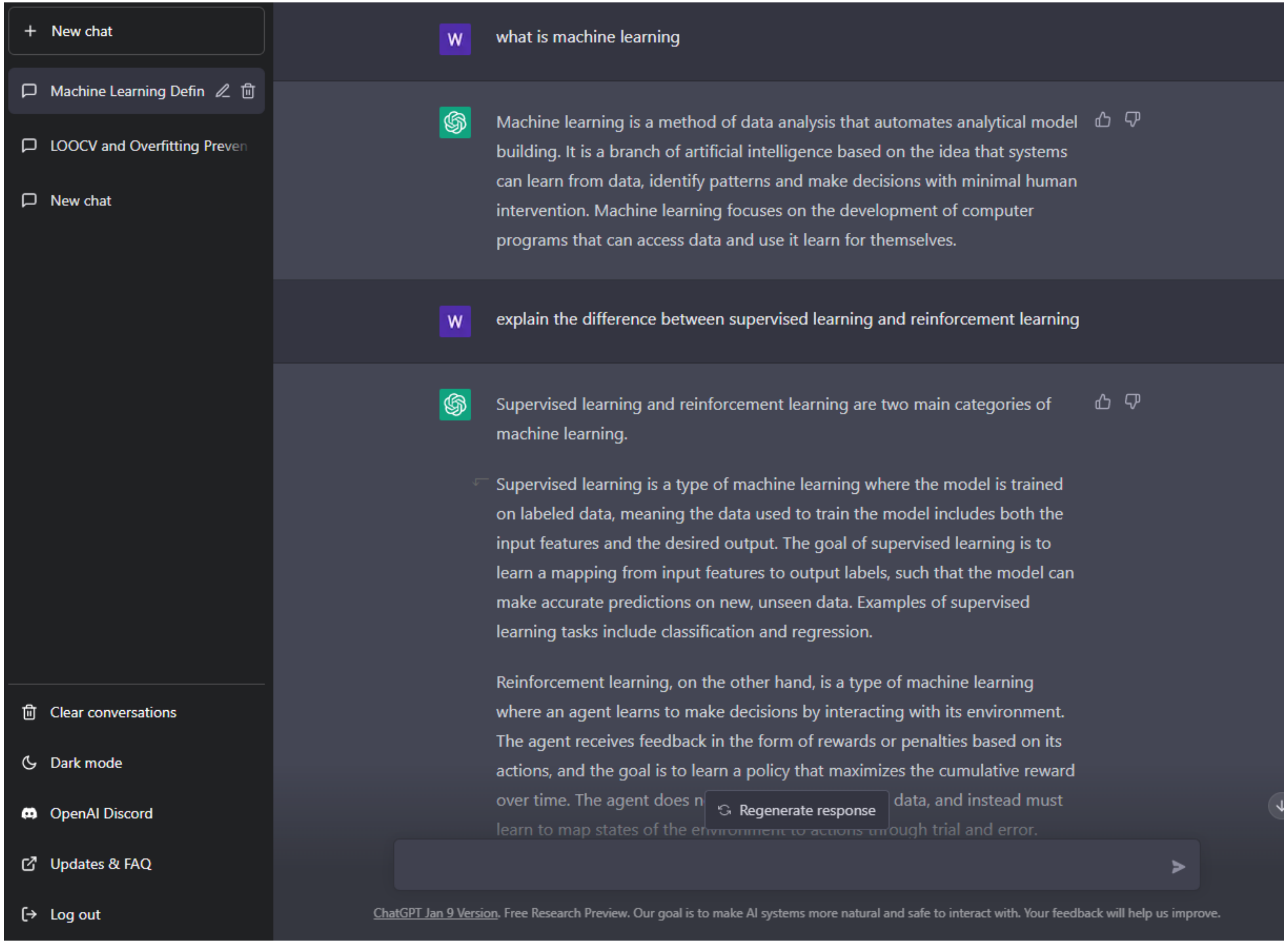
    …



# Where to get help?

- https://www.google.com
- https://stackoverflow.com
- https://stats.stackexchange.com/
- https://towardsdatascience.com/
- https://www.3blue1brown.com
- https://statquest.org/
- https://openai.com/blog/chatgpt/

# ChatGPT



New chat
Machine Learning Defin
LOOCV and Overfitting Preven
New chat
Clear conversations
Dark mode
OpenAI Discord
Updates & FAQ
Log out
W
what is machine learning
Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.
W
explain the difference between supervised learning and reinforcement learning
Supervised learning and reinforcement learning are two main categories of machine learning.
Supervised learning is a type of machine learning where the model is trained on labeled data, meaning the data used to train the model includes both the input features and the desired output. The goal of supervised learning is to learn a mapping from input features to output labels, such that the model can make accurate predictions on new, unseen data. Examples of supervised learning tasks include classification and regression.
Reinforcement learning, on the other hand, is a type of machine learning where an agent learns to make decisions by interacting with its environment. The agent receives feedback in the form of rewards or penalties based on its actions, and the goal is to learn a policy that maximizes the cumulative reward over time. The agent does n
data, and instead must
learn to map states of the
ough trial and error.
Regenerate response
ChatGPT Jan 9 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

# Lastly, GLHF!


"good luck, have fun with machine learning"

W Wenbin × DALL·E
Human & AI



Figure is generated by [DALLE](#) powered by [OpenAI](#)

# Q&A

[Google docs]