CPSC 340/540 Tutorial 3

Winter 2024 Term 1

T1A: Tuesday 16:00-17:00;
T1C: Thursday 10:00-11:00;
Office Hour: Wednesday 15:00-16:00

Slides can be found at Piazza and my personal page after T1C.

Yi (Joshua) Ren

https://joshua-ren.github.io/
renyi.joshua@gmail.com
PhD with Danica

Machine Learning:
Learning dynamics, LLM, Compositional Generalization

More helpful on theory

Less helpful on coding

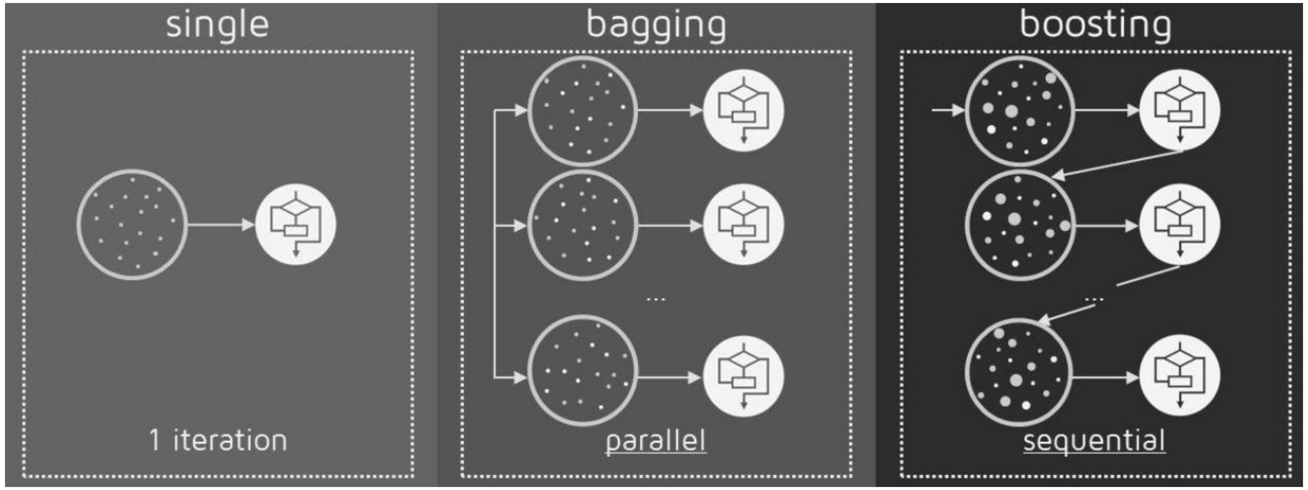
Slides Credit: To various pervious TA's of this course

- Ensemble Methods
- K-means and Expectation-Maximization
- Recap of Part 1 (supervised learning)

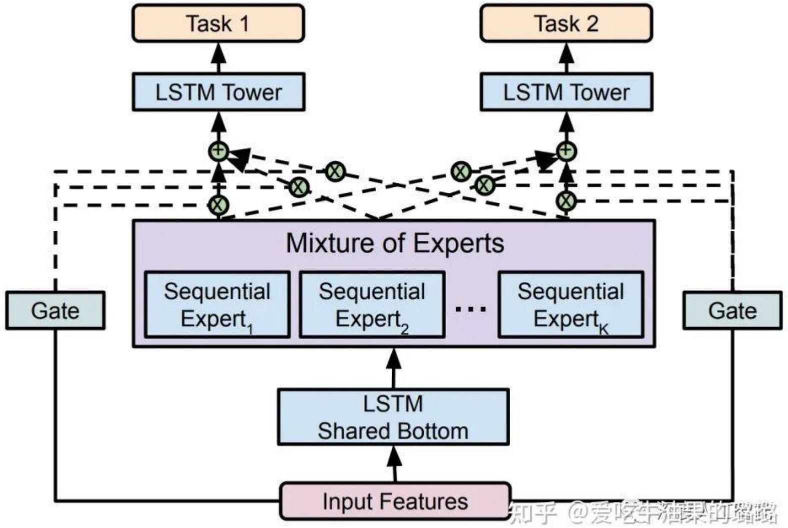
Ensemble Methods (intro)

- They have interesting names:
  - Averaging.
  - Blending.
  - Boosting.
  - Bootstrapping.
  - Bagging.
  - Cascading.
  - Random Forests.
  - Stacking.
  - Voting.

Merge the predictions of different models



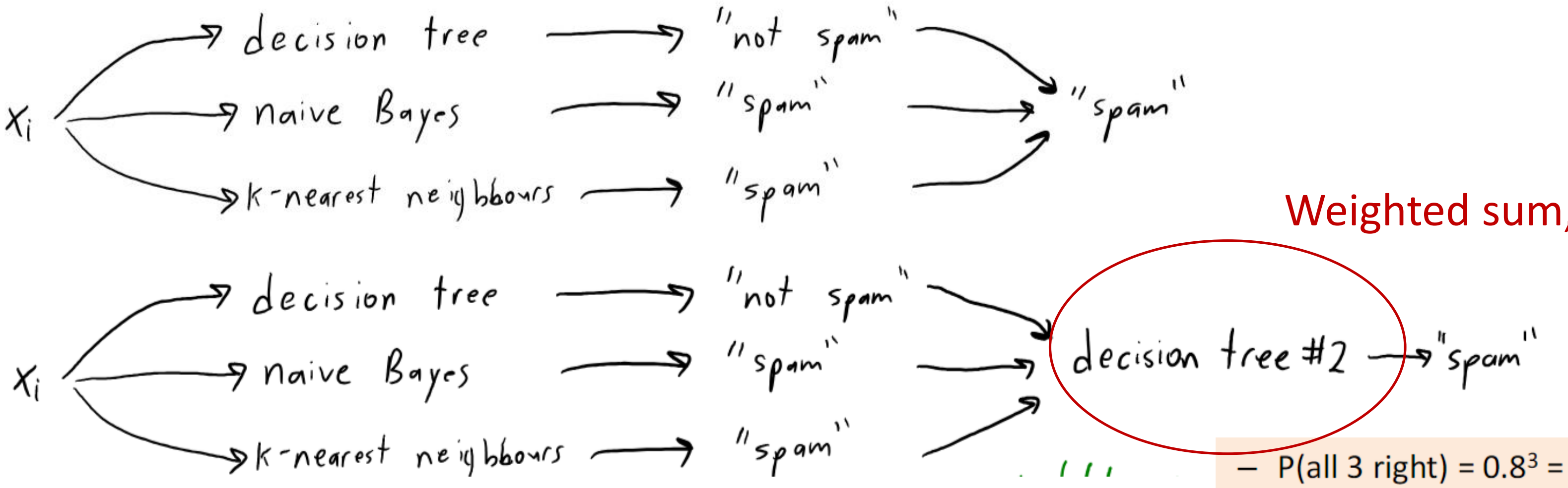
- Not only popular for Kaggle, but also very popular in SOTA deep learning systems, e.g., **Mixture of Experts (MoE)** in ChatGPT



Ensemble Methods (why and when they works)

- Voting and stacking (parallel & sequential)

Weighted sum, similar to DNN



– P(all 3 right) = 0.8^3 = 0.512.
– P(2 rights, 1 wrong) = $3*0.8^2(1-0.8)$ = 0.384.
– P(1 right, 2 wrongs) = $3*(1-0.8)^2 0.8$ = 0.096.
– P(all 3 wrong) = $(1-0.8)^3$ = 0.008.
– So ensemble is right with probability 0.896

➢ It is **less likely** that all models make wrong predictions together.

➢ But, note the following facts:
  - We need **independence** of different models (sub-sample different features, use different models)
  - <u>**Almost impossible to achieve independence**</u> (since the dataset is fixed)
  - The basic idea can be generalize to many applications (**Multi-mode (Interesting example)**, MoE, etc.)

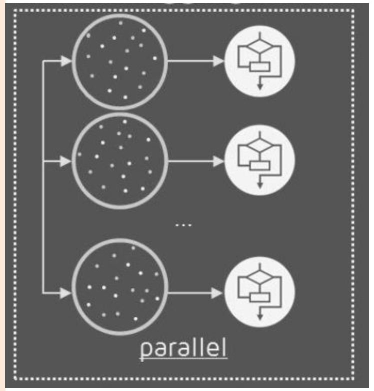
Types and Goals of Ensemble Methods

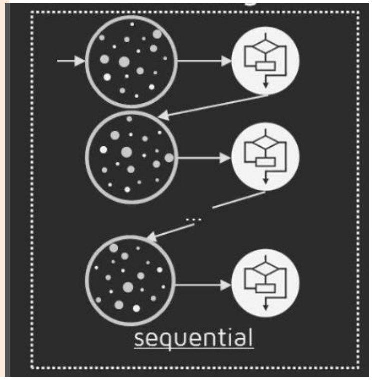
- Remember the fundamental trade-off:
  1. E_{train}: How small you can make the training error. **Capacity**

     vs.

  2. E_{gap}: how close training error is to test error. **Generalization**
- Goal of ensemble methods is that meta-classifier:
  - Does much better on one of these than individual classifiers.
  - Does not do too much worse on the other.
- This suggests two types of ensemble methods:
  1. Averaging: improves generalization gap of classifiers with high E_{gap}.
     - This is the point of "voting".
  2. Boosting: improves training error of classifiers with high E_{train}.
     - Covered later in course.

Although overfit in different ways, averaging them can mitigate that.





Individual model underfit (not capable enough), boosting them can increase the equivalent capacity.

K-means (Unsupervised learning)

- Supervised learning:
  - We have features x_i and class labels y_i.
  - Write a program that produces y_i from x_i.
- Unsupervised learning:
  - We only have x_i values, but no explicit target labels.
  - You want to do "something" with them.
- Some unsupervised learning tasks:
  - Outlier detection: Is this a 'normal' x_i?
  - Similarity search: Which examples look like this x_i?
  - Association rules: Which x^j occur together?
  - Latent-factors: What 'parts' are the x_i made from?
  - Data visualization: What does the high-dimensional X look like?
  - Ranking: Which are the most important x_i?
  - Clustering: What types of x_i are there?

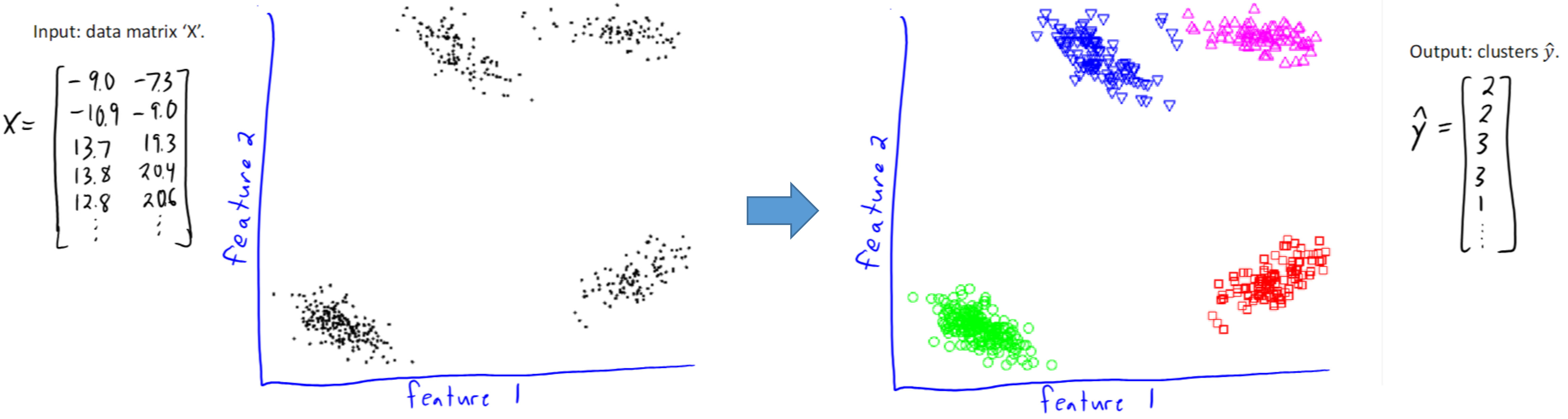
Bayesian: $p(\mathbf{x}_i|\mathbf{y}_i)$, also other generation models
DNN: $p(\mathbf{y}_i|\mathbf{x}_i)$, end2end, use the data more efficient

Self-supervised learning:
Very common way to get good representations

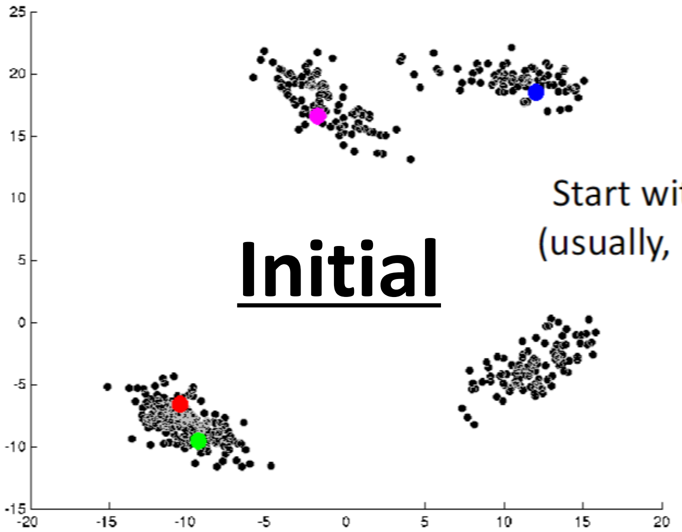
- GPT
- Diffusion model
- Variational autoencoder (VAE)
- Generative adversarial network (GAN)

K-means (Goal)

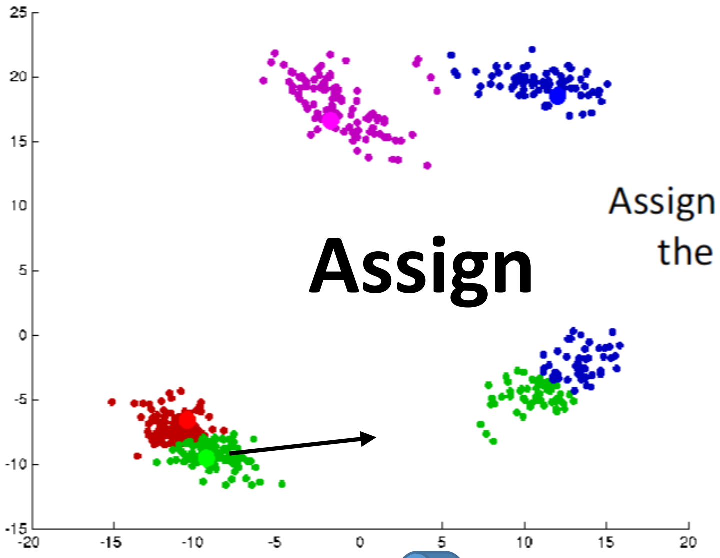
- In clustering we want to assign examples to “groups”:



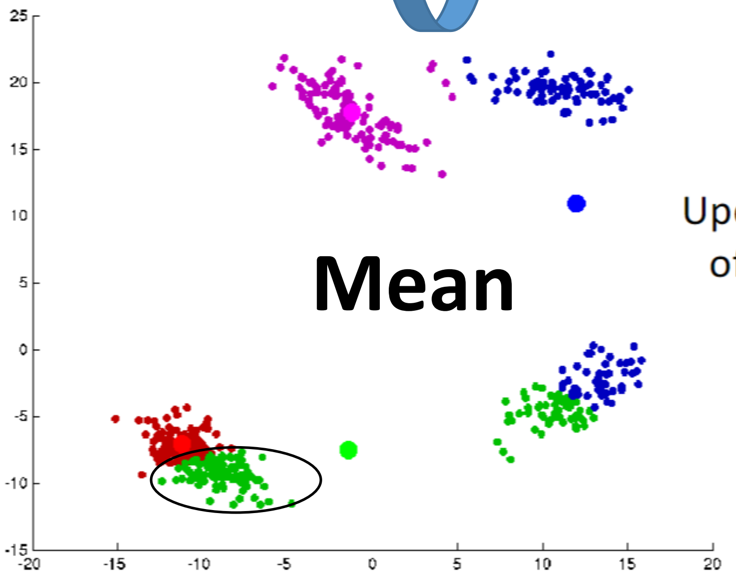
K-means (Algorithm)



Start with 'k' initial 'means'
(usually, random data points)

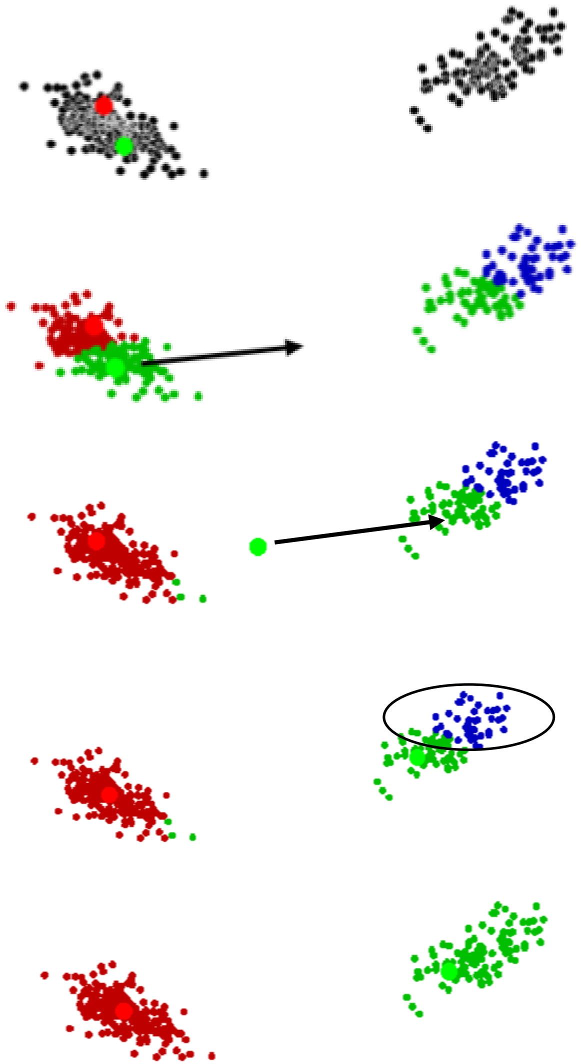


Assign each example to
the closest mean.



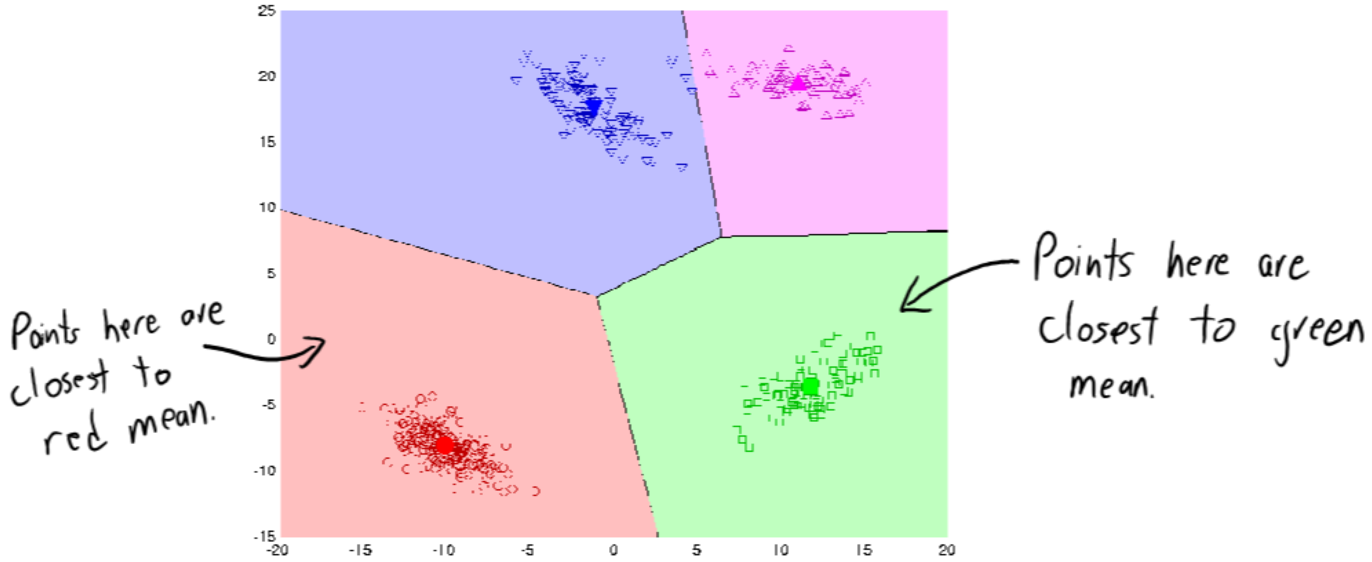
Update the mean
of each group.

Evolution of the mean



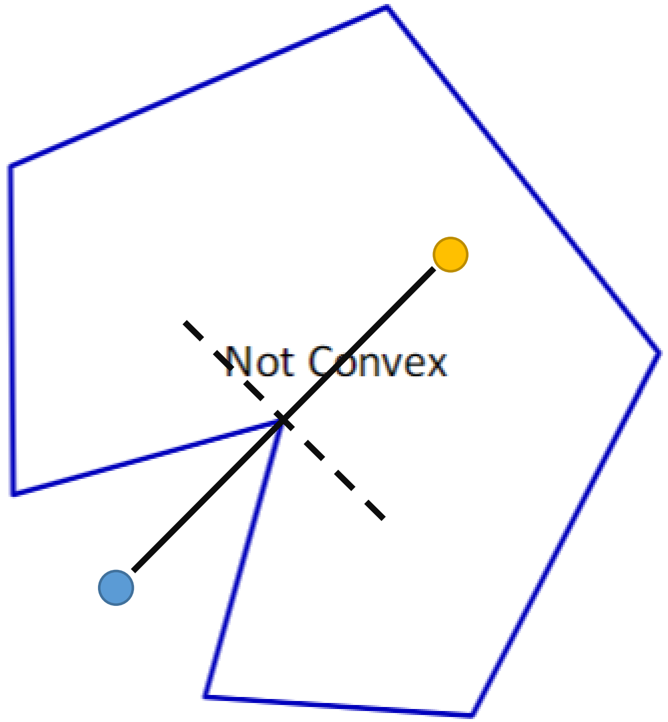
K-means (Shape of Clusters)

- Recall that k-means assigns cluster based on nearest mean.
- This leads to partitions the space :



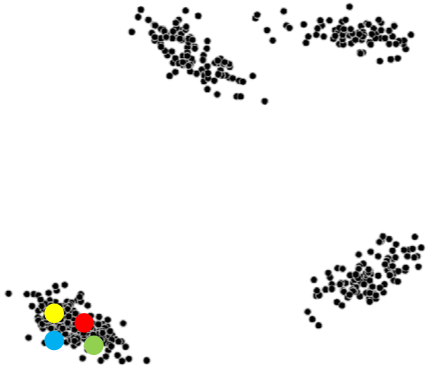
- Observe that the clusters are convex regions (proof in bonus).

- Why must be convex? An intuitive proof.

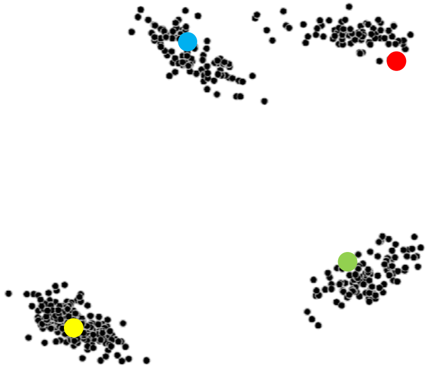


- There are many other clustering methods who can provide non-convex shapes (Bottom-up based, density based, etc.)

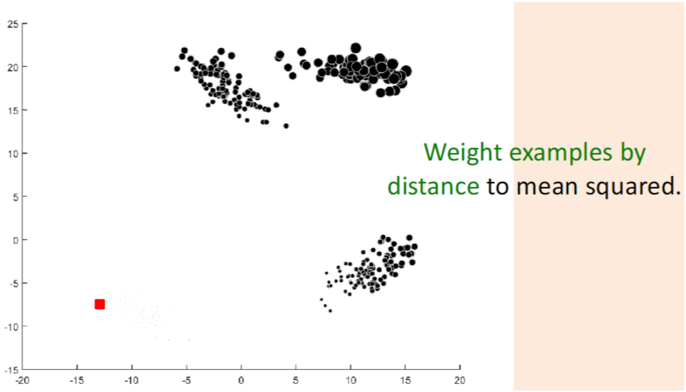
K-means (Influence of initialization)

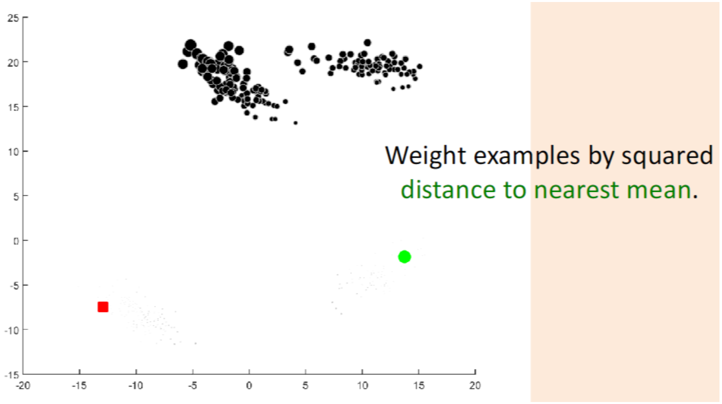


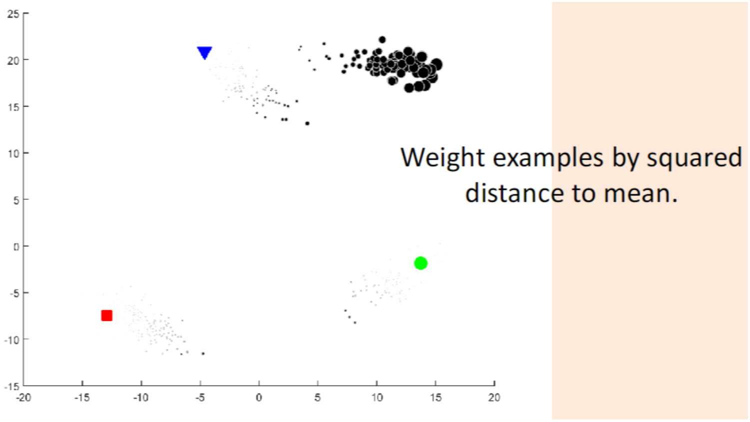
V.S.



- K-means++, select starting point as **sparse** as possible (further sample with higher prob.)

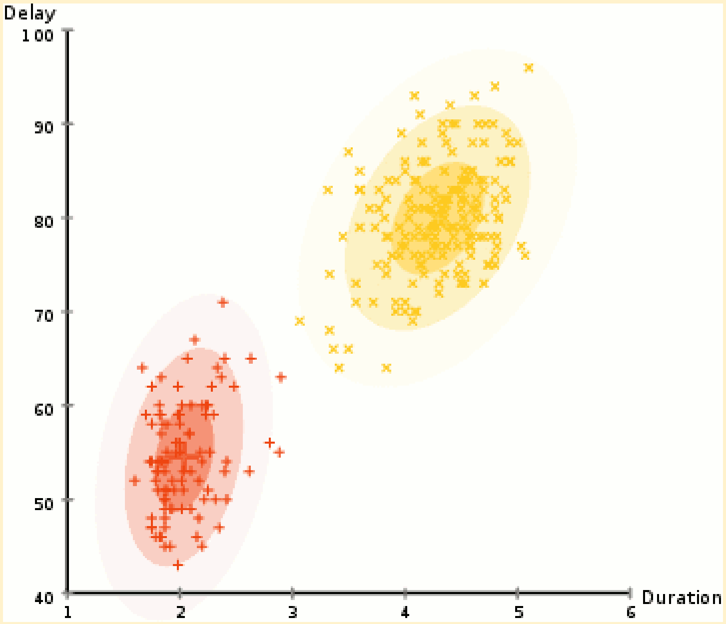




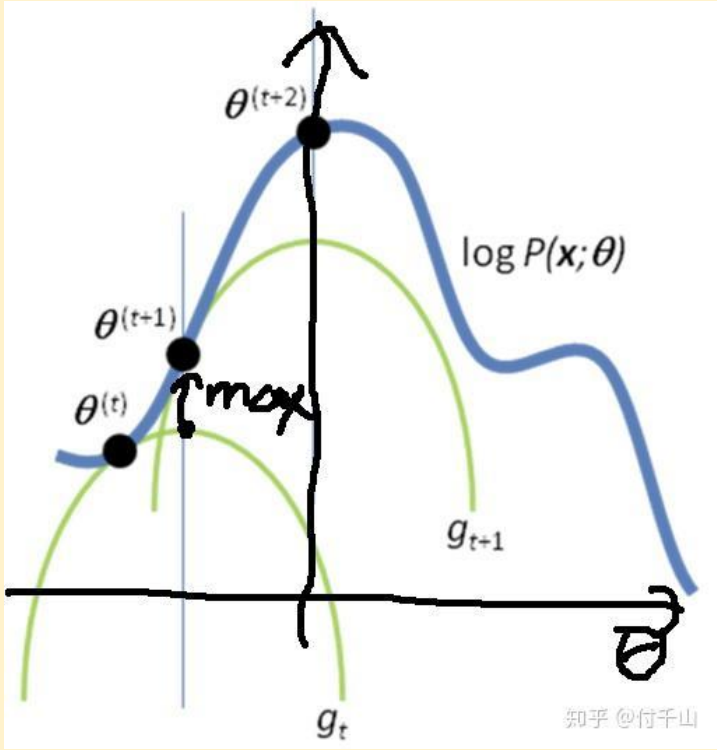


K-means (Theoretical Understanding)

- A special case of **Gaussian Mixture Model (GMM)**
- Guarantee to converge when problem is convex
- Algorithm is called **Expectation-maxmization (EM) algorithm**



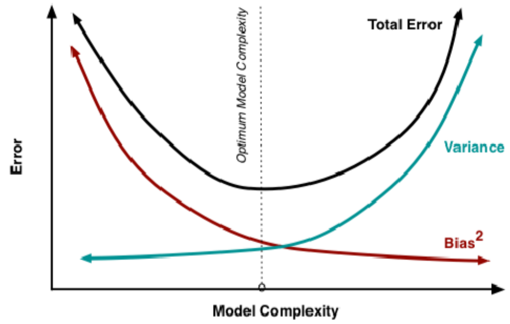
- Task: estimate $\theta = [\mu_1, \ldots, \mu_K, \sigma_1, \ldots, \sigma_K]$ that maximize the likelihood for all given examples
  $$\log P(\mathbf{x}; \theta)$$
- E-step: choose **assignment** to maximize likelihood
  In K-means, assign each sample a closest mean
- M-step: **re-calculate θ** based on assignments
  In K-means, calculate the new mean
- Repeat to converge
- Jensen provides the guarantee for loss decreasing.



Recap of Part 1:

- Fundamental ideas:
  - Training vs. test error (memorization vs. learning).
  - IID assumption (examples come independently from same distribution).
  - Key principle: test set should not influence training
  - Fundamental trade-off (between training error vs. generalization gap).
  - Validation sets and cross-validation (can approximate test error)
  - Optimization bias (we can overfit the training set and the validation set).
  - Decision theory (we should consider costs of predictions).
  - Parametric vs. non-parametric (whether model size depends on 'n').
  - No free lunch theorem (there is no universally "best" model).

✓ Under/over-fit
✓ Variance bias trade-off



✓ otherwise same as one sample
✓ otherwise no reason to generalize

✓ If so, need another clean test set

✓ Use it to select hyper-parameters

✓ Less #validation samples OR more trials → more bias

✓ KNN v.s. Naive Bayes (what is parameter, what is hyper)

✓ Need uniform data assumption, which is usually not the case

Recap of Part 1: Key concepts

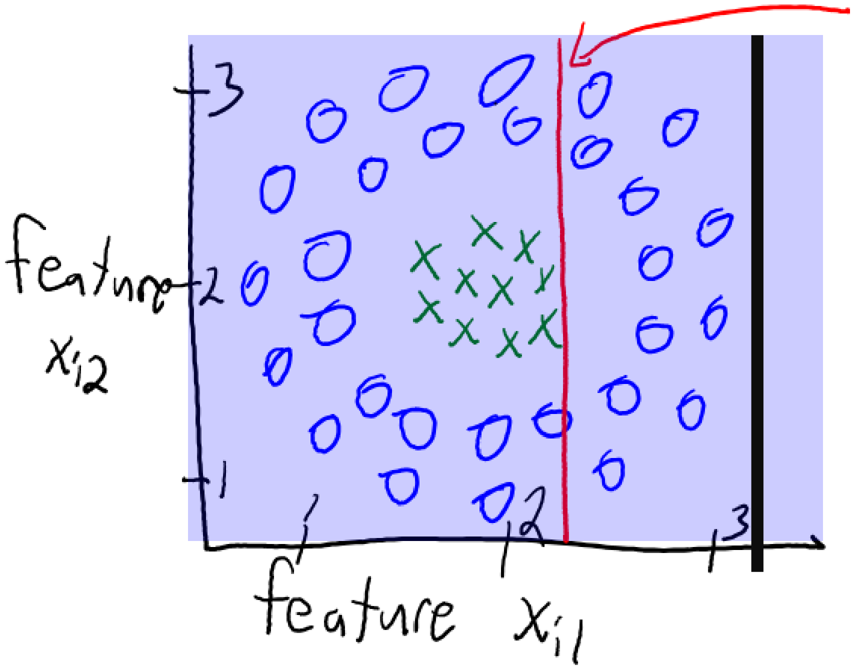
- We saw 3 ways of "learning":
  - Searching for rules.
    - Decision trees (greedy recursive splitting using decision stumps).
  - Counting frequencies.
    - Naïve Bayes (probabilistic classifier based on conditional independence).
  - Measuring distances.
    - K-nearest neighbours (non-parametric classifier based on distances).
- We saw 2 generic ways of improving performance:
  - Encouraging invariances with data augmentation.
  - Ensemble methods (combine predictions of several models).
    - Random forests

Recap of Part 1: Decision trees – why we use “information gain” instead of accuracy

number of examples assigned to leaf 1

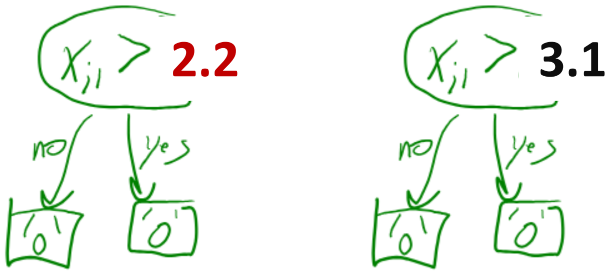
$$\text{information gain} = \underbrace{\text{entropy}(y)}_{\text{entropy of labels before split}} - \frac{n_{leaf1}}{n}\text{entropy}(y_{leaf1}) - \frac{n_{leaf2}}{n}\text{entropy}(y_{leaf2})$$

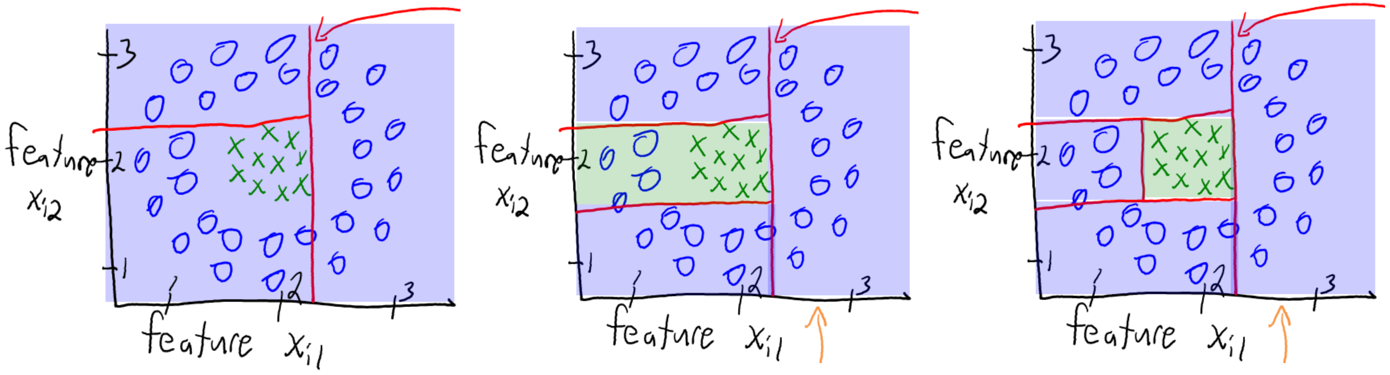
entropy of labels of examples assigned to leaf 1



- Obviously, x>2.2 is better than x>3.1
- But both of the following 2 stumps provide the same accuracy
- However, their info gain is different:
  - For 2.2: IG = entropy(y) – xxx, which is **greater than 0**
  - For 3.1: IG = 0

- Build stump by using the **mode** for each split





Recap of Part 1: Key concepts

- We saw 3 ways of "learning":
  - Searching for rules.
    - Decision trees (greedy recursive splitting using decision stumps).
  - Counting frequencies.
    - Naïve Bayes (probabilistic classifier based on conditional independence).
  - Measuring distances.
    - K-nearest neighbours (non-parametric classifier based on distances).
- We saw 2 generic ways of improving performance:
  - Encouraging invariances with data augmentation.
  - Ensemble methods (combine predictions of several models).
    - Random forests

Step1: get data using BofW

Step2: calculate $p(\mathbf{y}_i = 1|\mathbf{x}_i) > p(\mathbf{y}_i = 0|\mathbf{x}_i)$ using

a. Bayesian and get $\frac{p(\mathbf{x}_i|\mathbf{y}_i)p(\mathbf{y}_i)}{p(\mathbf{x}_i)}$

b. Eliminate $p(\mathbf{x}_i)$ for both sides

c. Calculate $p(\mathbf{y}_i)$ by counting

d. Approximate

$$p(\mathbf{x}_{i1}, \mathbf{x}_{i2}, \dots, \mathbf{x}_{iB}|\mathbf{y}_i) \approx \prod_{b=1}^{B} p(\mathbf{x}_{ib}|\mathbf{y}_i)$$

e. Calculate each $p(\mathbf{x}_{ib}|\mathbf{y}_i)$ by counting

f. Use label smoothing if necessary

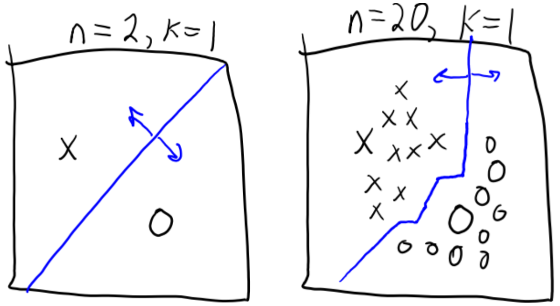
g. Use n-gram BofW if necessary

h. Use log-prob if necessary

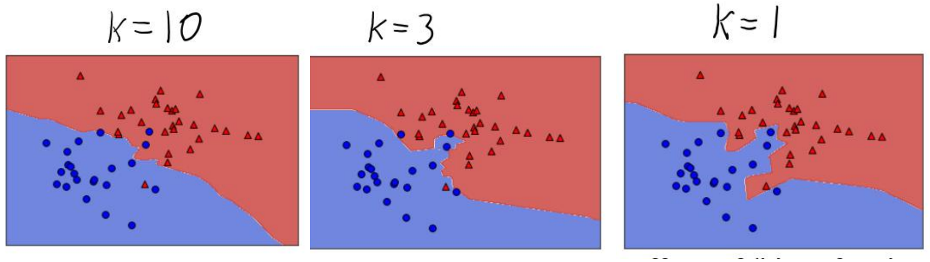
Recap of Part 1: Key concepts

- We saw 3 ways of "learning":
  - Searching for rules.
    - Decision trees (greedy recursive splitting using decision stumps).
  - Counting frequencies.
    - Naïve Bayes (probabilistic classifier based on conditional independence
  - Measuring distances.
    - K-nearest neighbours (non-parametric classifier based on distances).
- We saw 2 generic ways of improving performance:
  - Encouraging invariances with data augmentation.
  - Ensemble methods (combine predictions of several models).
    - Random forests

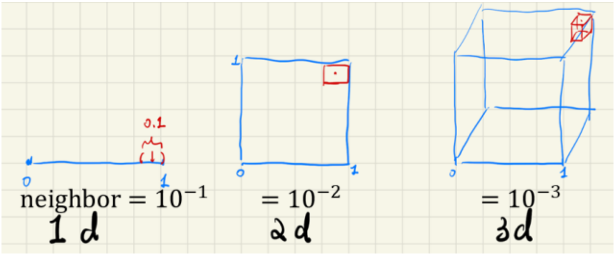
- Hyper-parameter and bias-variance tradeoff

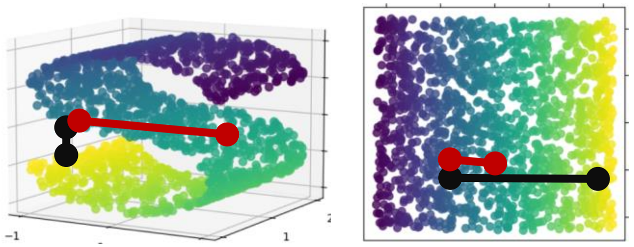


$$f = WX + k|W|_2^2$$
$$= W\begin{bmatrix} x \\ \dots \\ x^n \end{bmatrix} + k|W|_2^2$$



- Curse of dimensionality and low-dim manifold





Thanks for your time!
Questions?