# Machine Learning
# Week 7 Lecture: Softmax Regression, Non-Parametric Classification, and Mixture of Experts

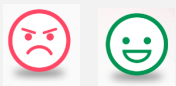
Debasis Ganguly

School of Computing Science
University of Glasgow

November 13, 2025

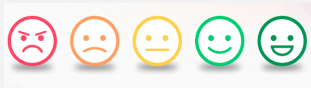
# Multi-class Classification

## Binary Classification



- Target values can only be binary, i.e. $y \in \{0, 1\}$.
- Common applications - sentiment classification, spam detection.
- Model: $h_\theta(\mathbf{x}) = 1/(1 + \exp(-\theta\mathbf{x}))$

## Multi-class Classification



- More general
- Target variables can be categorical, i.e., $y \in \{0, 1, \ldots, k-1\}$.
- Categories can be mapped to integers.
- More practical use-case than binary classification.
- Model: $h_\theta(\mathbf{x}) = 1/(1 + \exp(-\theta\mathbf{x}))$?

# Motivation

- Does $\underbrace{h_\theta(\mathbf{x}) = 1/(1+\exp(-\theta\mathbf{x}))}_{\text{Sigmoid}}$ work for categorical values, i.e., when $y \in \{0, 1, \ldots, k-1\}$?
- We need to first verify (just as we did from linear regression $\mapsto$ logistic regression) if slight modifications to existing solutions work.
- Recall from last week that:

## Linear regression

- Target variables are unbounded.
- Objective function treats the parameter vector (line in the data space) as a **close fit**.

## Logistic regression

- Target values in $\{0, 1\}$.
- Objective function has to treat the parameter vector as a **decision boundary**.

# Logistic Regression for Multiple class prediction

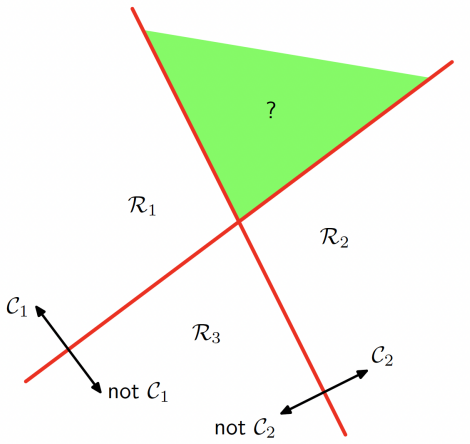



Figure: One vs. Rest

- For $k$ classes, draw $k-1$ decision boundaries - **one for each class**.
- $\mathbf{x} \in C_k$ if $\mathbf{x} \notin C_i$ $(i = 1, \ldots, k-1)$.
  - $h_\theta^{(i)}(\mathbf{x}) = 1/(1+\exp(-\theta^{(i)}\mathbf{x}))$
- Interpretation of $i^{th}$ boundary:
  - Positive part of the hyper-plane $(h_\theta^{(i)}(\mathbf{x}) > 0)$: Member of the $i^{th}$ class $C_i$.
  - Negative part of the hyper-plane: Not a member of the $i^{th}$ class $C_i$.
- Leads to **ambiguous regions**.
  - Green Region: Both $C_1$ and $C_2$!

# Logistic Regression for Multiple class prediction

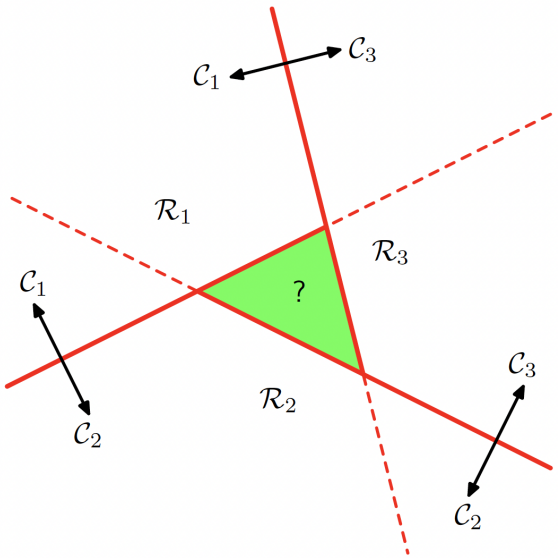



Figure: Pairwise

- Learn a decision boundary for each pair to distinguish between $C_i$ and $C_j$ $(j \neq i)$.
- How many boundaries?

# Logistic Regression for Multiple class prediction

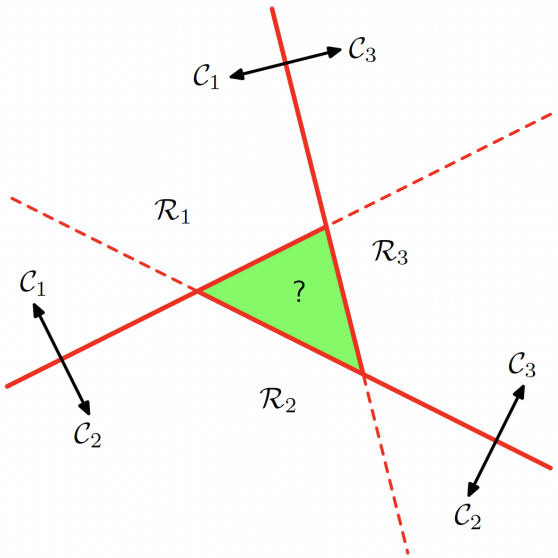
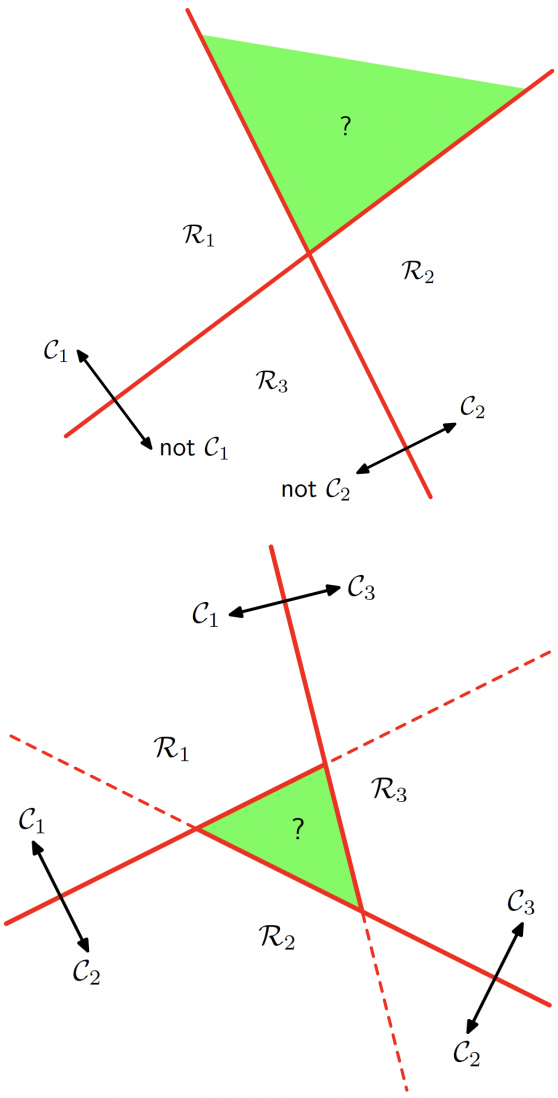



Figure: Pairwise

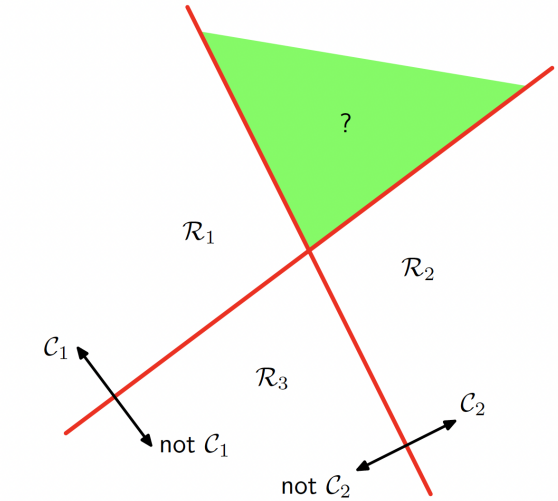
- Learn a decision boundary for each pair to distinguish between $C_i$ and $C_j$ $(j \neq i)$.
- How many boundaries?
  - Number of unordered pairs (combinations): $k(k-1)/2$.
- Leads to **ambiguous regions**.
  - Green Region: Both $C_1$, $C_2$ and $C_3$!
- Quadratic number of decision boundaries $\rightarrow$ Increased training time!

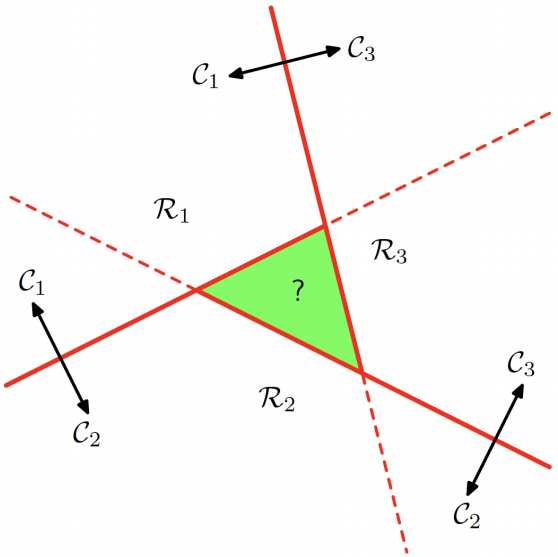
# Can we resolve these ambiguities?




- Yes, we can by using the posteriors
- Recall: $P(y|\mathbf{x};\theta) = h_\theta(\mathbf{x}) = 1/(1+\exp(-\theta\mathbf{x}))$
  - Higher $P(y|\mathbf{x};\theta) \rightarrow$ more confidence!
- Despite ambiguity of the green regions, we can make a **per-instance decision** to resolve ambiguity!
- For both 'one vs. rest' and pairwise:
  - $\hat{y} = i$ if $P(\mathbf{x}|\theta^{(i)}) > P(\mathbf{x}|\theta^{(j)})\ \forall j$ .
- However, this seems like a hack?

# Can we resolve these ambiguities?

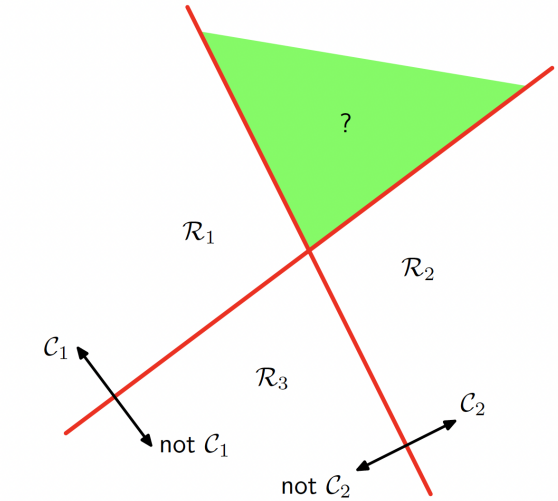


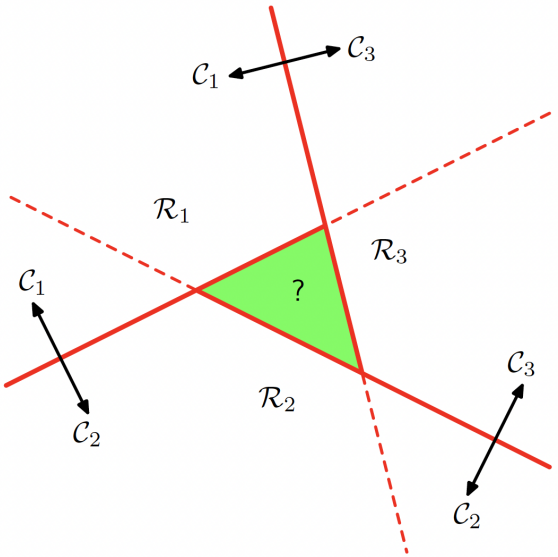




## Menti-Quiz (Code: 1373-7721)

Are the posterior probabilities even comparable?

# Can we resolve these ambiguities?

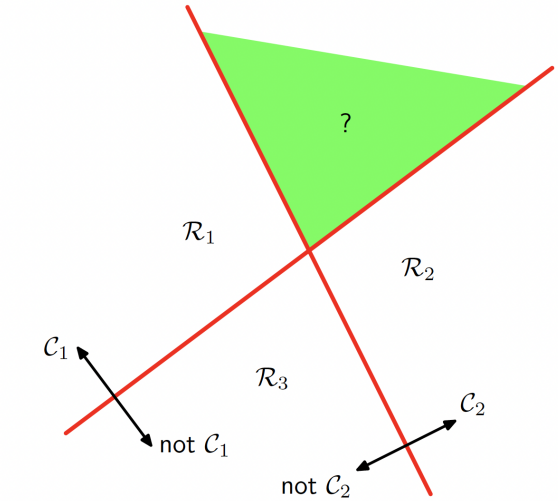


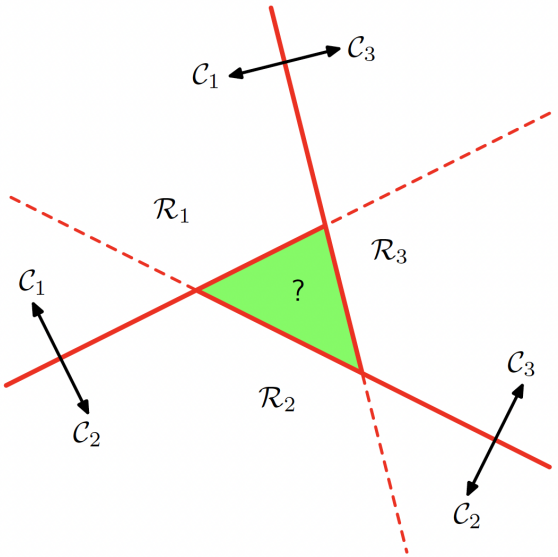




## Menti-Quiz (Code: 1373-7721)

Are the posterior probabilities even comparable?

# Can we resolve these ambiguities?




## Menti-Quiz (Code: 1373-7721)

Are the posterior probabilities even comparable?

- No, they are not because the **parameter vectors are trained independently** of one another.

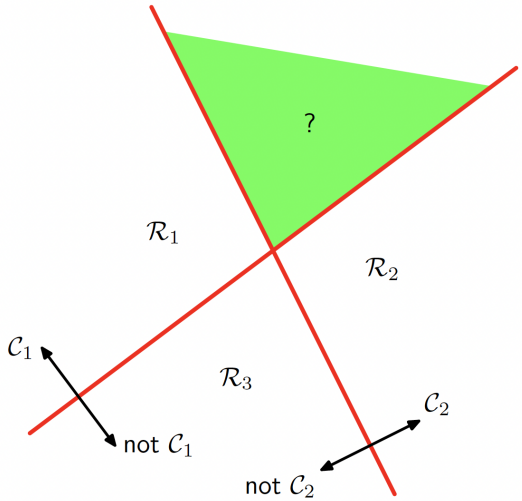
What we need:

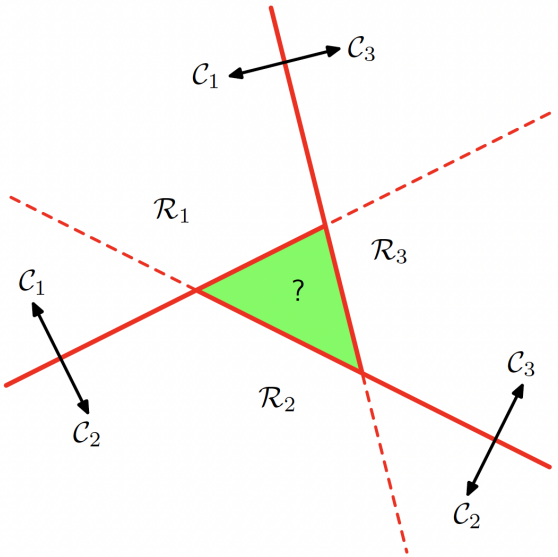
- A single model that is trained **jointly**.
- A decision for one class needs to be done in **awareness of the other possibilities**.

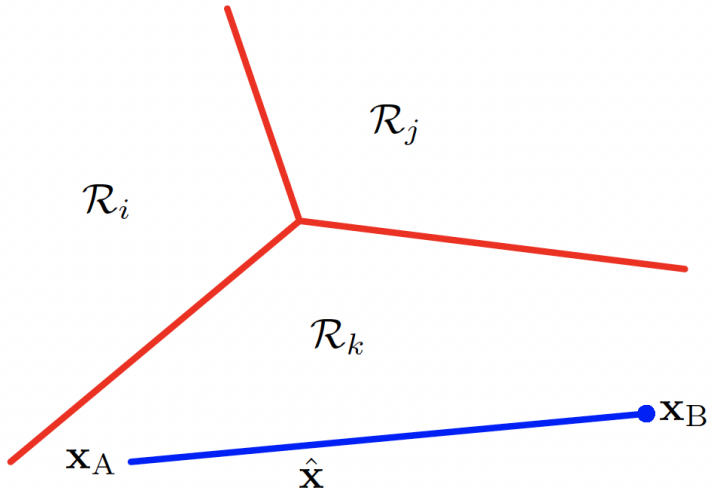
# Softmax Regression

- What then needs to be changed?
  - Short answer: The notion of decision boundary - because it only applies for two classes.
- Long answer:
  - Linear $\mapsto$ Logistic: Objective function was made to represent a decision boundary.
  - Logistic $\mapsto$ Multi-class: Objective function needs to be **jointly modeling the posteriors** of each class.
  - $k$-class discriminant: composed of $k$ linear functions **trained jointly**.
- Joint Modeling means that:
  - **Not independent decision boundaries** anymore.
  - Our hacky idea of taking the maximum over the posteriors now works!
  - Because the **probabilities are now comparable**!

# Softmax Regression




- Joint training ensures:
  - Piecewise decision boundaries.
  - **Convex** decision regions.
  - No ambiguity.

# Softmax Regression



## Logistic Regression

- Distribution: Bernoulli
- $h_\theta(\mathbf{x})^y(1-h_\theta(\mathbf{x}))^{1-y}$
- Model:
- $h_\theta(\mathbf{x}) = g(\theta\mathbf{x}) = 1/(1+exp(-\theta\mathbf{x}))$

## Softmax Regression

- Distribution: Multinomial $h_{\theta^{(1)}}(\mathbf{x})^{y=1} h_{\theta^{(2)}}(\mathbf{x})^{y=2} \ldots h_{\theta^{(k)}}(\mathbf{x})^{y=k}$
- Model:
- $h_{\theta^{(i)}}(\mathbf{x}) = \phi_i/\Phi$
  - $\phi_i = \exp(\theta^{(i)}\mathbf{x})$
  - $\Phi = \sum_{j=1}^{k} \phi_j$
  - Belief in class $i$ **relative to** the other class beliefs.

## Menti-Quiz (Code: 1373-7721)

What is the key insight which tells us that the decision region is convex?

# Softmax Regression



## Logistic Regression

- Distribution: Bernoulli
- $h_\theta(\mathbf{x})^y(1-h_\theta(\mathbf{x}))^{1-y}$
- Model:
- $h_\theta(\mathbf{x}) = g(\theta\mathbf{x}) = 1/(1+exp(-\theta\mathbf{x}))$

## Softmax Regression

- Distribution: Multinomial $h_{\theta^{(1)}}(\mathbf{x})^{y=1} h_{\theta^{(2)}}(\mathbf{x})^{y=2} \ldots h_{\theta^{(k)}}(\mathbf{x})^{y=k}$
- Model:
- $h_{\theta^{(i)}}(\mathbf{x}) = \phi_i/\Phi$
  - $\phi_i = \exp(\theta^{(i)}\mathbf{x})$
  - $\Phi = \sum_{j=1}^{k} \phi_j$
  - Belief in class $i$ **relative to** the other class beliefs.

### Menti-Quiz (Code: 1373-7721)

What is the key insight which tells us that the decision region is convex?

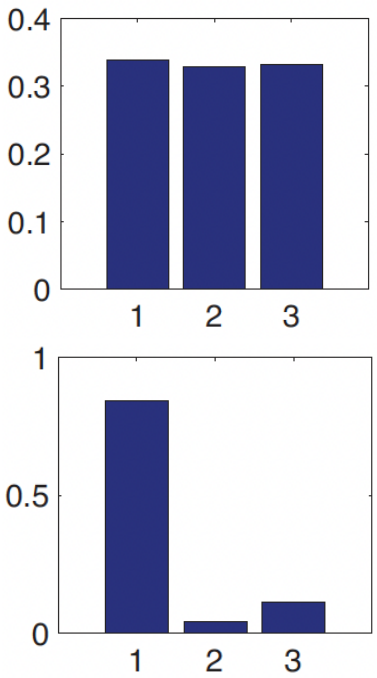
- There is a point with the same posterior probability for each class.

# Ground-truth Labels and Objective Function

- Need to employ the cross-entropy objective, meaning:
  - Predicted probability for a class (close to 1) and a target value of 1 $\rightarrow$ good prediction.
  - **Cross-entropy can't work directly on $y$ values** because probabilities are bounded in $\{0, 1\}$, whereas the target values are bounded in $\{0, \ldots, k-1\}$.
- Intuitively, the target variables need to also be comprised of '1's and '0's as was the case for logistic regression.
- Encode each target variable (a class id) $y \in \mathbb{Z}_k$ as a $k$-dimensional 1-hot vector $\mathbf{y}$.
  - $\mathbf{y}_i = 1$ if $y(\mathbf{x}) = i$ (i.e., $\mathbf{x} \in C_i$)
  - Example: Consider a 3-class classification problem.
    - For $y = 1$, $\mathbf{y} = (1, 0, 0)$
    - For $y = 2$, $\mathbf{y} = (0, 1, 0)$
    - For $y = 3$, $\mathbf{y} = (0, 0, 1)$

# Softmax Regression

- Each ground-truth value (a 1-hot vector) is now the **ideal probability distribution**
  - That is, when you **know that** $\mathbf{x} \in C_i$, the probability distribution **collapses** to $(y_1 = 0, \ldots, y_i = 1, \ldots, y_k = 0)$.

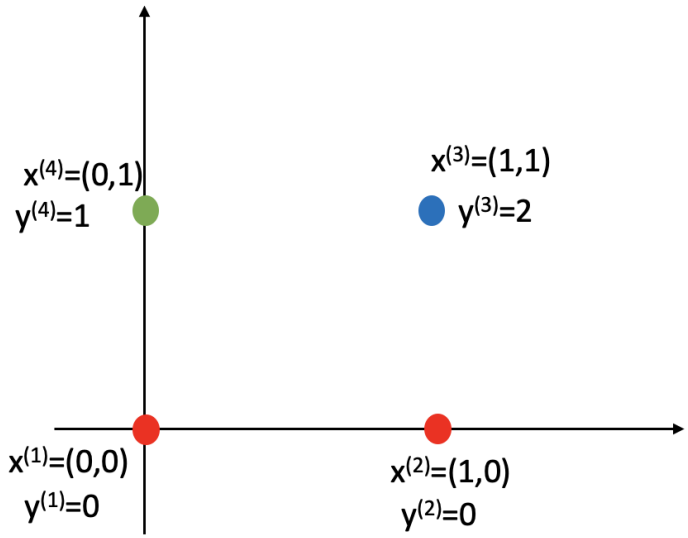



## Generalized Cross-Entropy Loss

- $h_\Theta(\mathbf{x}) = (\frac{\phi_1}{\Phi}, \ldots, \frac{\phi_k}{\Phi})$
- Good prediction: When the correct class gets a high probability and every other probability value is low.
- Bad predictions:
  - When the incorrect class gets a high predicted probability.
  - When the distribution is close to uniform.

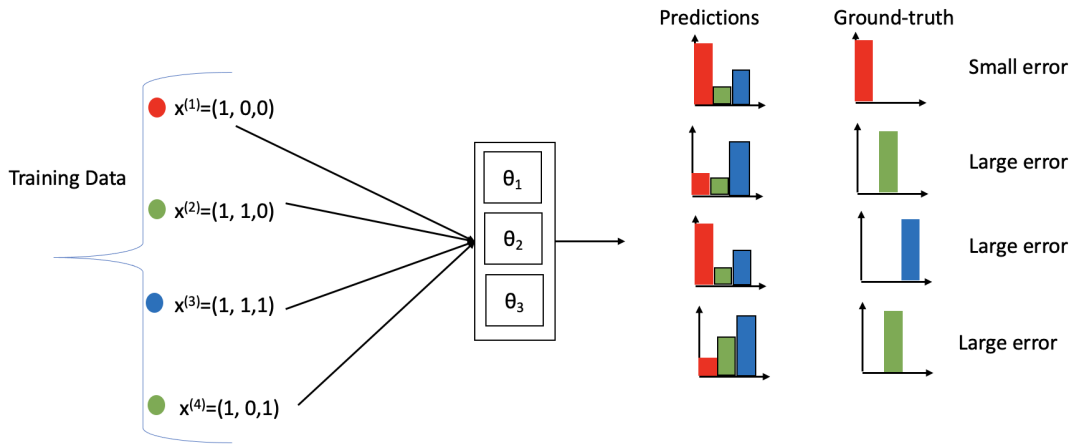
# Multi-class Numerical Working Example

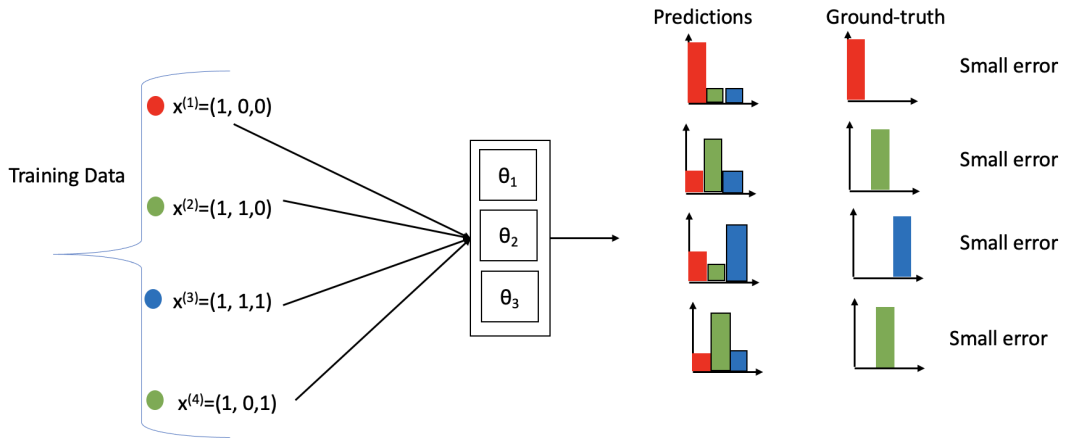
- Recall: In Week 6, we worked out the gradients of a sample problem with logistic regression.
- We modify the problem slightly to introduce one more class (everything else remaining the same).




- We now need 3 $\Theta$ vectors, call them $\Theta_1$, $\Theta_2$ and $\Theta_3$.

# Gradient updates in Softmax Regression



# Gradient Descent for Softmax Regression

- For each training set instance $\mathbf{x} \in \mathbb{R}^{d+1}$ (1 additional for bias)
- Compute: $\Phi = \sum_{k=1}^{K} \exp(\theta_k \mathbf{x})$ $(\theta_k \in \mathbb{R}^{d+1})$
- For each class $k = 1, \ldots, K$
- Compute the probability $P(y = k) = \hat{\mathbf{y}}_k = \exp(\theta_k \cdot \mathbf{x})/\Phi$.
- For each parameter vector component $\theta_{k,j}$ $(j = 1, \ldots, d + 1)$
- $\theta_{k,j} \leftarrow \theta_{k,j} + (y_k - \hat{y}_k).x_j$.

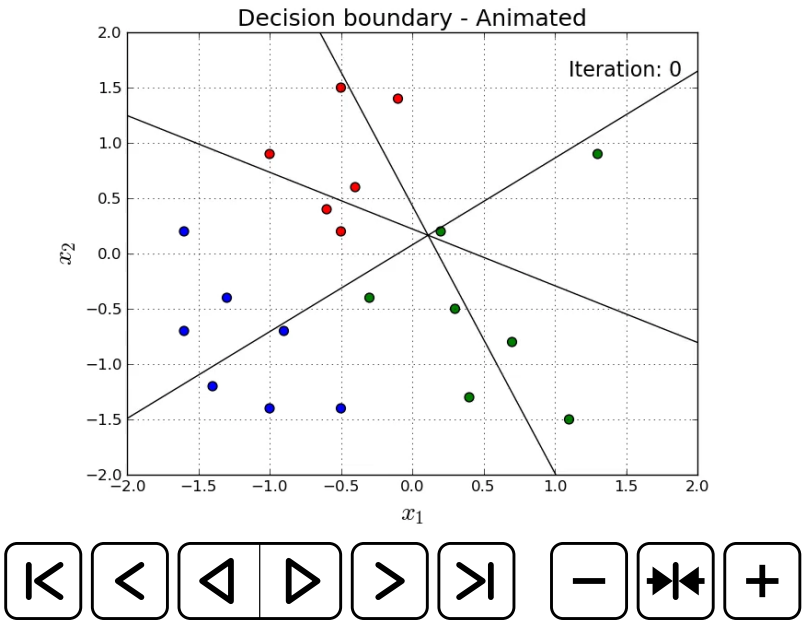
**Compact vector form of the SGD update**

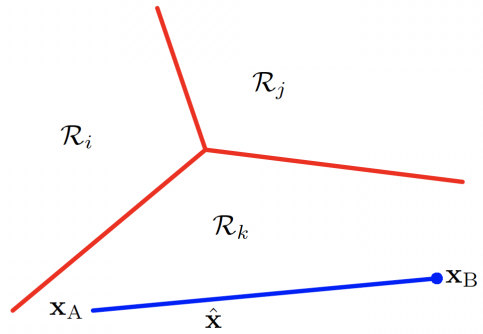
$\theta_k \leftarrow \theta_k + (\mathbf{y} - \hat{\mathbf{y}})x_j$

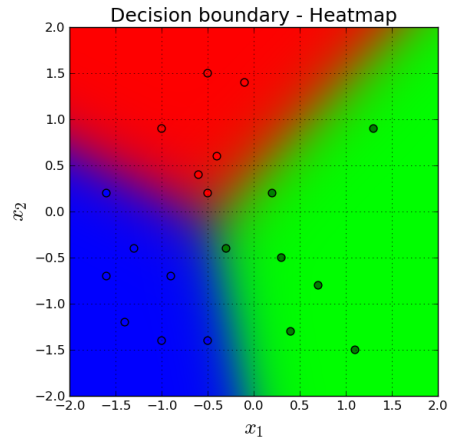
# Evolution of the Decision Boundary

## Softmax regression

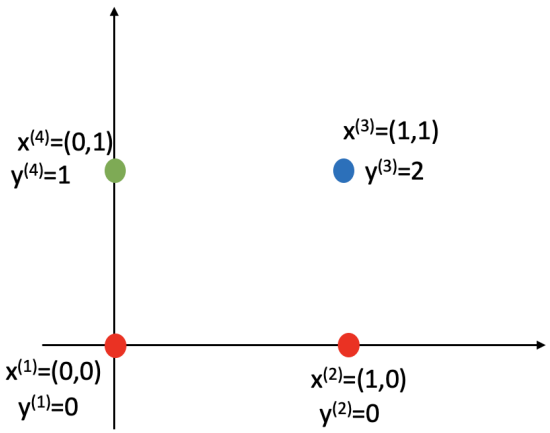
Produces a convex decision boundary comprised of piece-wise linear segments.



# Softmax Regression Numerical Working Example



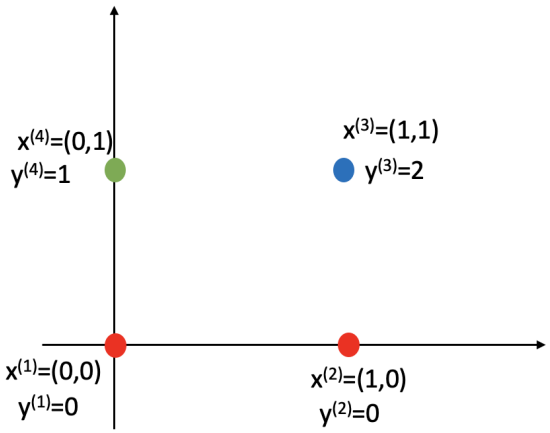

- A random parameter initialization:
  - $\theta^{(1)} =$ $(0.0, 0.0, 0.0)$
  - $\theta^{(2)} =$ $(0.0, 0.0, 0.0)$
  - $\theta^{(3)} =$ $(0.1, 0.1, 0.2)$

Gradient update for $\mathbf{x}^{(3)} = (1, 1, 1)$.

- One-hot encoding of $y^{(3)}$: $\mathbf{y}^{(3)} = (0, 0, 1)$ (Why?)
- $\Phi =$ $\underbrace{\exp(0 \times 1 + 0 \times 1 + 0 \times 1)}_{\theta^{(1)}\mathbf{x}} +$ $\underbrace{\exp(0)}_{\theta^{(2)}\mathbf{x}} + \underbrace{\exp(0.1 + 0.1 + 0.2)}_{\theta^{(3)}\mathbf{x}} =$ $1 + 1 + 1.49 = 3.49$
- Posterior probability $\hat{\mathbf{y}} =$ $(1/3.49, 1/3.49, 1.49/3.49) =$ $(0.28, 0.28, 0.44)$
- $\mathbf{x}^{(3)}$ is thus predicted to be blue

# Softmax Regression Numerical Working Example



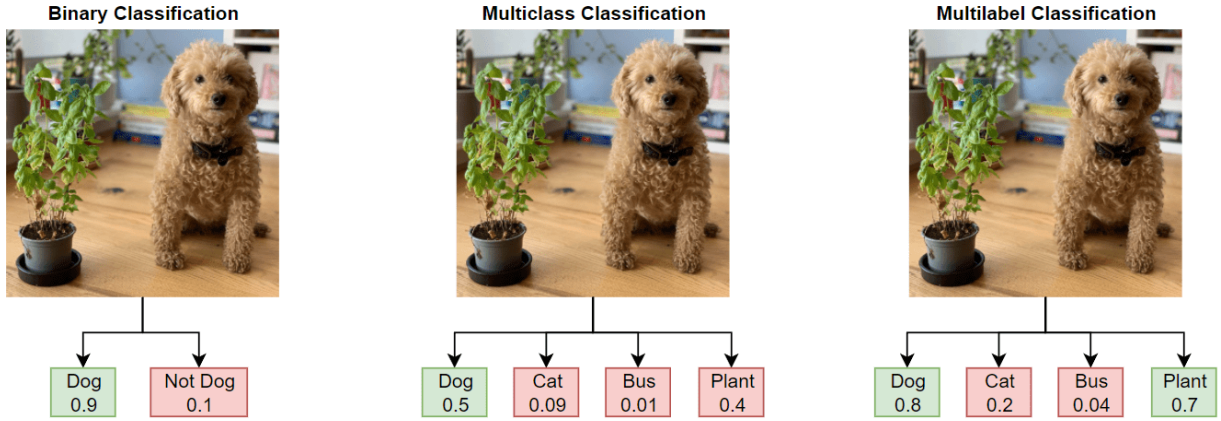



Updates:

- $\theta^{(1)} \leftarrow (0,0,0) + ((0-0.28) \times 1, (0-0.28) \times 1, (1-0.44) \times 1) = (-0.28, -0.28, 0.66) = \theta^{(2)}$
- $\theta^{(3)} \leftarrow \underbrace{(0.1, 0.1, 0.2)}_{\text{initial parameter}} + \underbrace{(-0.28, -0.28, 0.66)}_{\text{gradient-based error}} = (-0.18, -0.18, 0.86)$
- The modified posteriors are better because it assigns a higher probability to the third component — the true class of $\mathbf{x}^{(3)}$.

- **Homework**: Do one more step of gradient updates with $\mathbf{x}^{(2)}$.

# Multi-Label Classification (MLC)

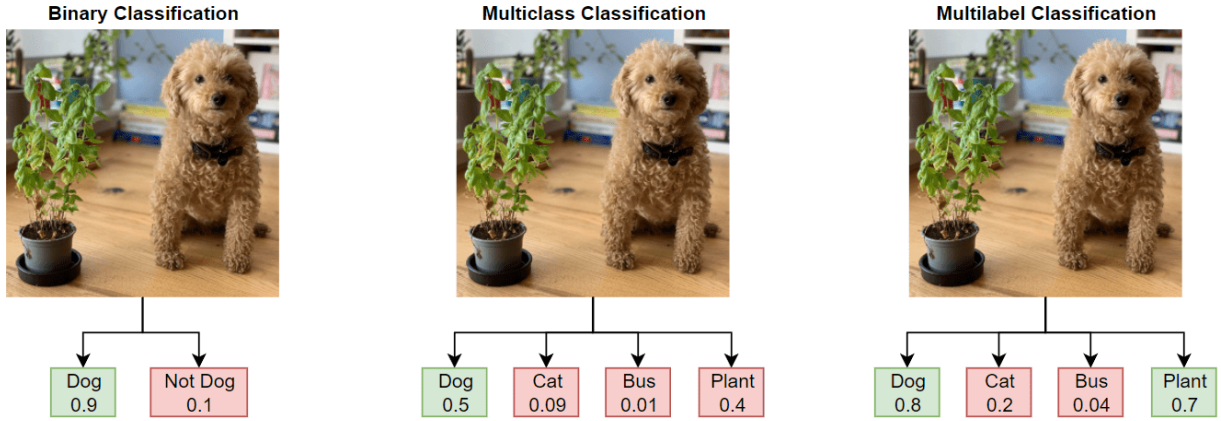



- When an instance is of **composite** type.
- Examples:
  - Multi-topical documents, e.g., long Wikipedia articles.
  - Images containing various objects.
- We saw the journey from linear $\mapsto$ logistic $\mapsto$ softmax

## Menti-Quiz (Code: 1373-7721)

Is MLC more constrained than MCC?

# Multi-Label Classification (MLC)




- When an instance is of **composite** type.
- Examples:
  - Multi-topical documents, e.g., long Wikipedia articles.
  - Images containing various objects.
- We saw the journey from linear $\mapsto$ logistic $\mapsto$ softmax

## Menti-Quiz (Code: 1373-7721)

Is MLC more constrained than MCC?

No, we can apply independent logistic regression classifiers, i.e., presence or absence of each individual class.

# MLC is algorithmically simpler than MCC!

## MCC

- Output: 1-hot vector of dimension $k$ (#classes)
- Parameter: $k$ parameter vectors each of dimension $d+1$.
- Objective function:

$$\sum_{i=1}^{k} \log y_i \frac{\exp(\theta^{(i)}\mathbf{x})}{\Phi}$$

  - Cross Entropy

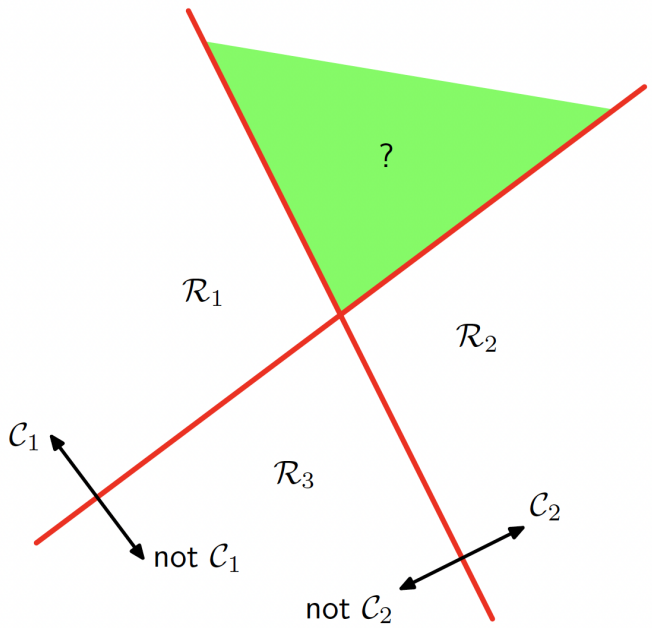
## MLC

- Output: $p$-hot vector of dimension $k$ (#classes), where $p \leq k$.
- Parameter: $k$ parameter vectors each of dimension $d+1$ (**identical to MCC**).
- Objective function:

$$\sum_{i=1}^{k} \log y_i \frac{\exp(\theta^{(i)}\mathbf{x})}{1+\exp(\theta^{(i)}\mathbf{x})}$$

  - **Sum over Binary Cross Entropy** values

# We had seen MLC before!




## MLC

- Objective function:

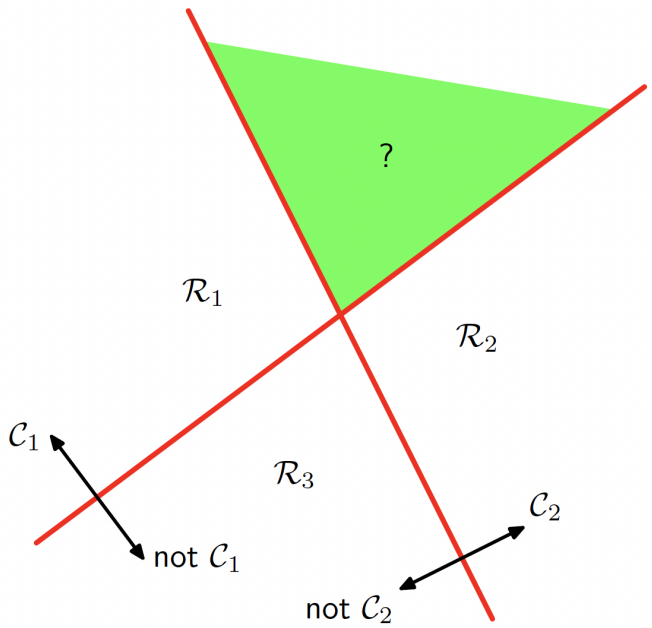
$$\sum_{i=1}^{k} \log y_i \frac{\exp(\theta^{(i)}\mathbf{x})}{1+\exp(\theta^{(i)}\mathbf{x})}$$

  - **Sum over Binary Cross Entropy** values
- Similar to one vs. rest for MCC.

## Menti-Quiz (Code: 1373-7721)

No problems with the green region here! (Why?)

# We had seen MLC before!




## MLC

- Objective function:

$$\sum_{i=1}^{k} \log y_i \frac{\exp(\theta^{(i)}\mathbf{x})}{1+\exp(\theta^{(i)}\mathbf{x})}$$

  - **Sum over Binary Cross Entropy** values
- Similar to one vs. rest for MCC.

## Menti-Quiz (Code: 1373-7721)

No problems with the green region here! (Why?)

An instance could be classified to multiple classes, e.g., object detection.

# K-Nearest Neighbors (KNN)

- Linear/Logistic/Softmax regression belong to the family of **parametric models**.
  - Learn a set of weights/parameters $\mathbf{w}/\theta$ from a given set of observations (training set of labeled data).
  - During inference, **we don't need the training set**.
  - All we need is the learned parameter vector.
- In general, **non-parametric models** refer to a class of methods which **doesn't use a fixed set of parameters**, i.e., the **number of parameters can dynamically grow**.
- What's the closest we have seen so far? –

# K-Nearest Neighbors (KNN)

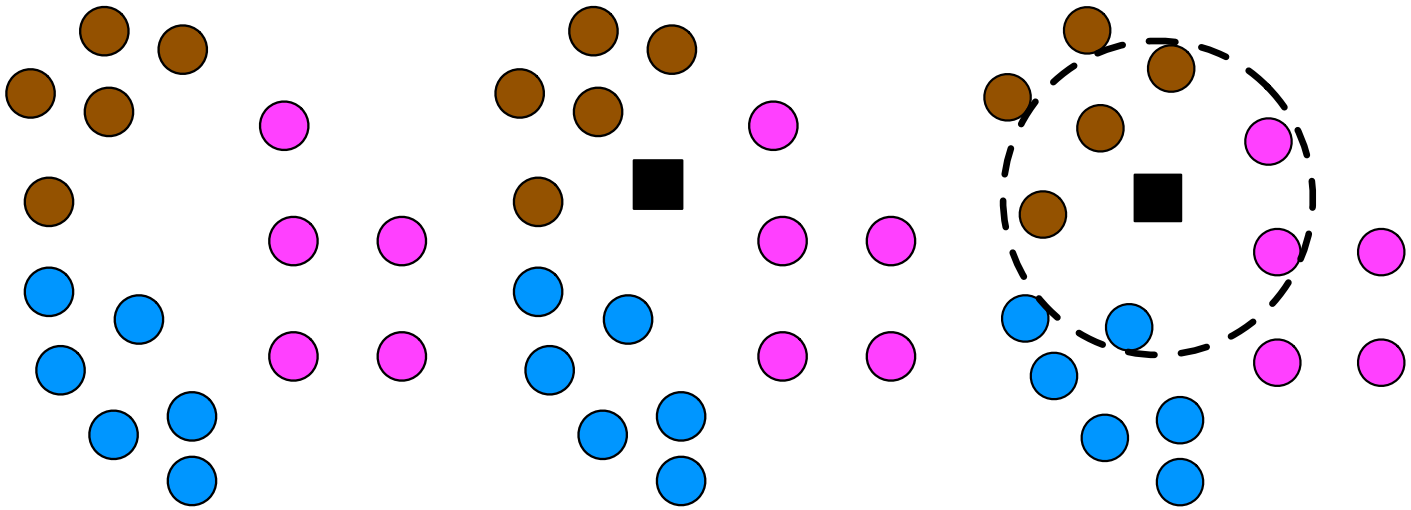
- Linear/Logistic/Softmax regression belong to the family of **parametric models**.
  - Learn a set of weights/parameters $\mathbf{w}/\theta$ from a given set of observations (training set of labeled data).
  - During inference, **we don't need the training set**.
  - All we need is the learned parameter vector.

- In general, **non-parametric models** refer to a class of methods which **doesn't use a fixed set of parameters**, i.e., the **number of parameters can dynamically grow**.
- What's the closest we have seen so far? –
  - Higher-order features via feature maps – but it is not non-parametric. Why?
- K-NN is one of the simplest non-parametric models.
- It **doesn't learn any parameters**.

# So how does KNN work?

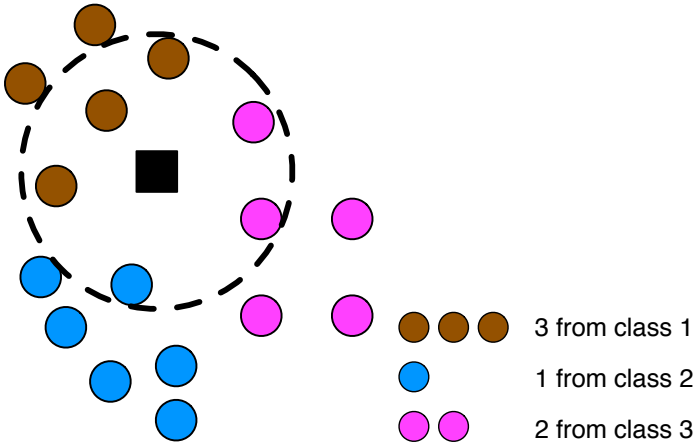
- Uses the **locality of reference** hypothesis.
- Similar inputs will have similar labels.
  - Example: If two houses are quite similar (in terms of the features, e.g., #bedrooms, front_yard_area, back_yard_area, furnished_kitchen, ...), then they will be in the same price range.
- Advantages:
  - Same algorithm works for binary and multi-class (all the trouble of devising different objective functions saved).

# KNN

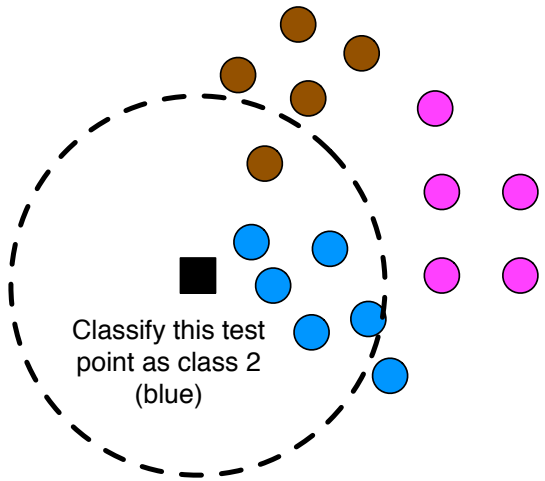


- Choose a $K$ (a small integer) to define the **neighborhood size**.
- For a test object $\mathbf{x}_{\text{new}}$:
  - Find the $K$ closest points from the training set.
  - Find majority class of these $K$ neighbors.
  - Resolve ties arbitrarily.

# KNN

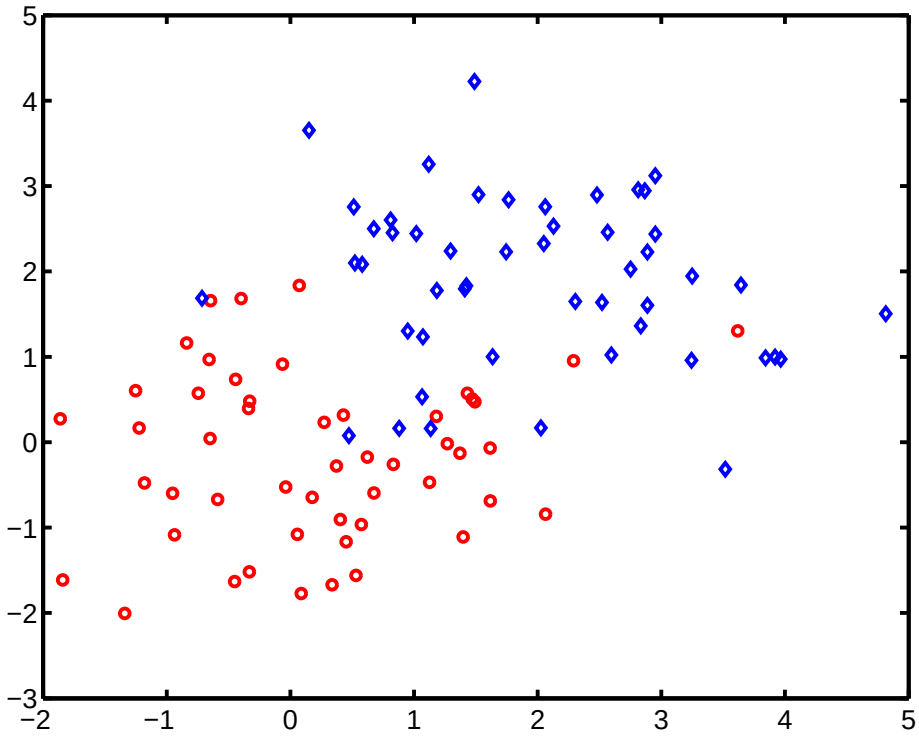



- Class one (brown) has most votes – classify $\mathbf{x}_{\text{new}}$ as belonging to class 1.



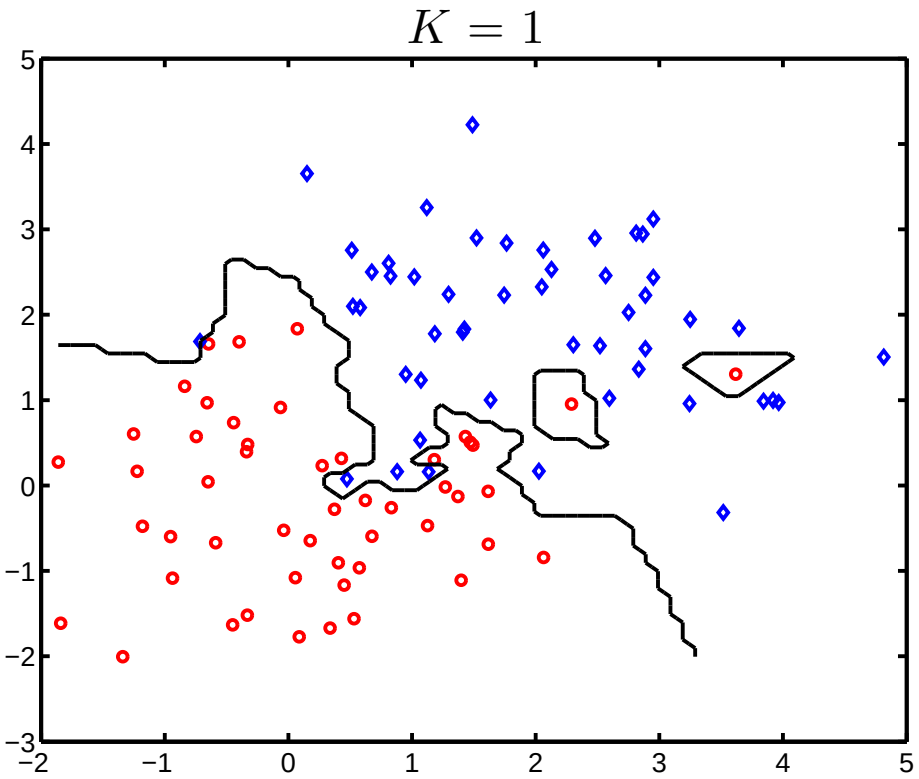

- Class 2 has most votes.

# KNN – real example



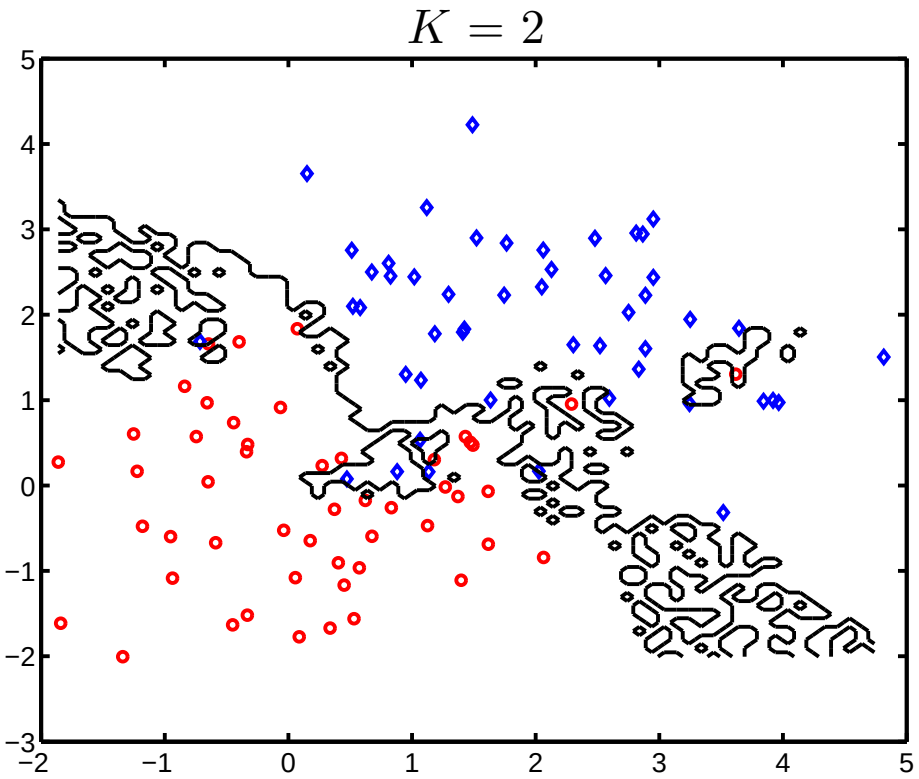

- Binary data.

# KNN – real example



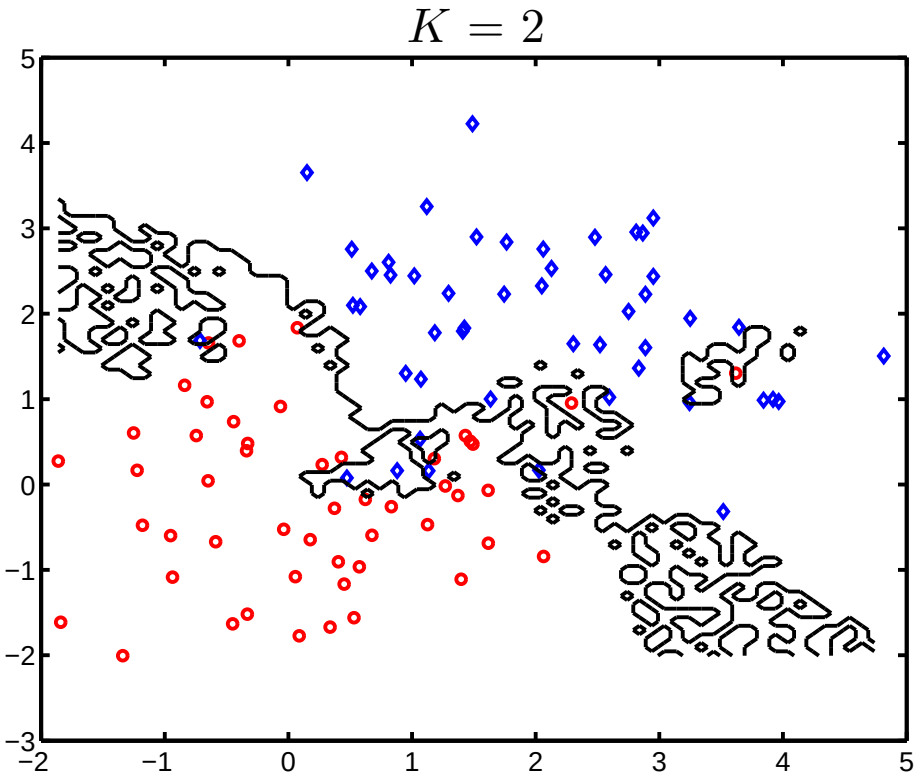

- 1-Nearest Neighbour.
- Line shows decision boundary.
- Too complex – should the islands exist?

# KNN – real example



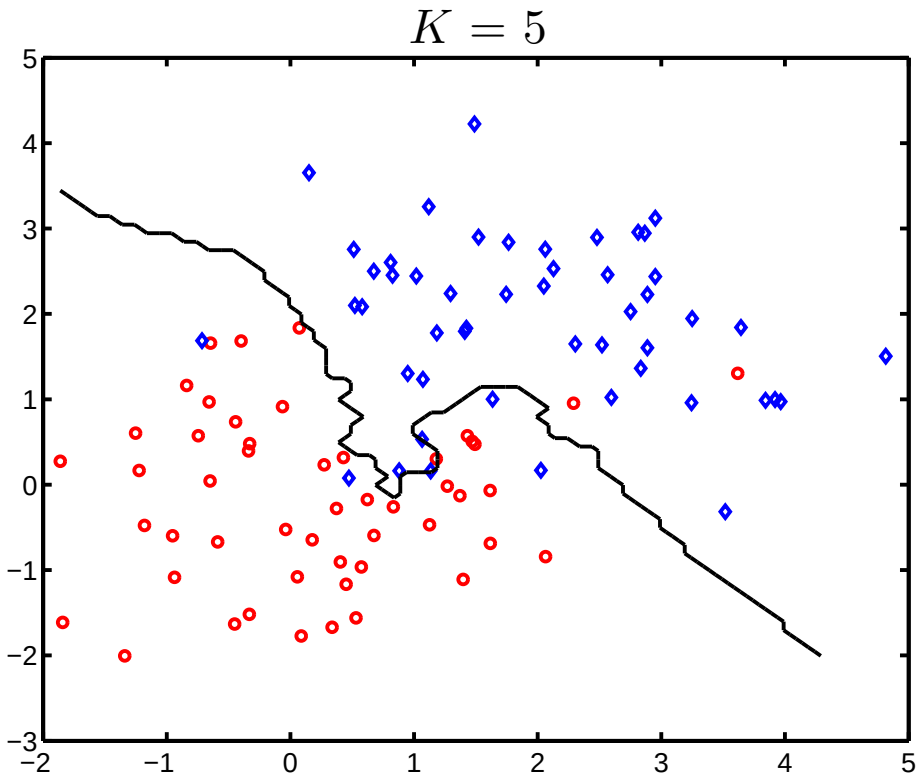

- 2-Nearest Neighbour.
- What's going on?

# KNN – real example



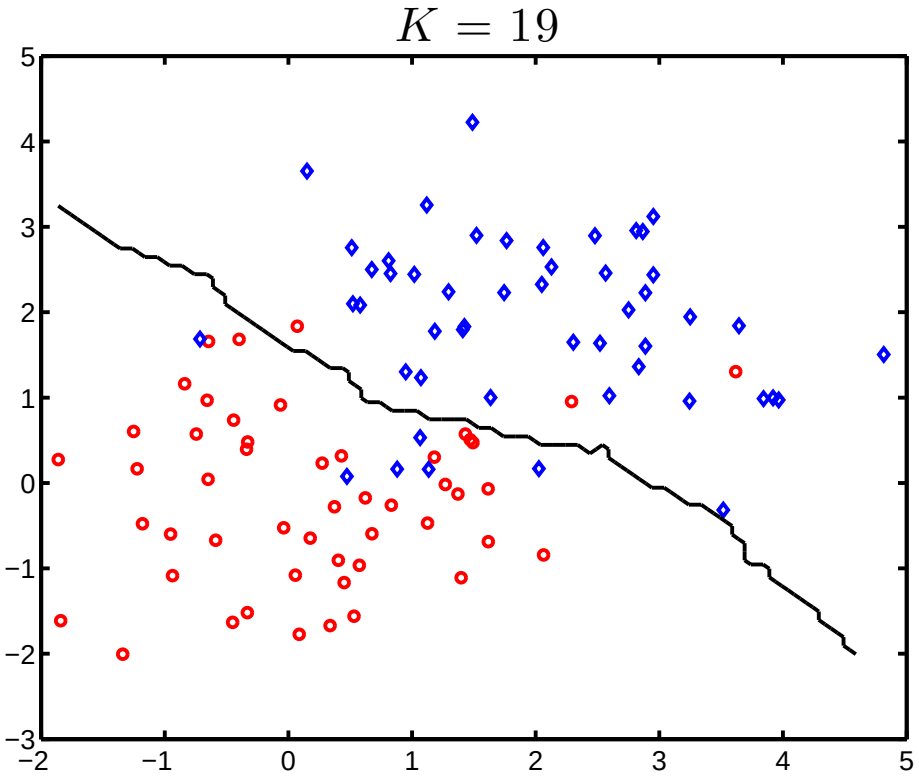

- 2-Nearest Neighbour.
- What's going on?
- Lots of ties – random guessing.

# KNN – real example



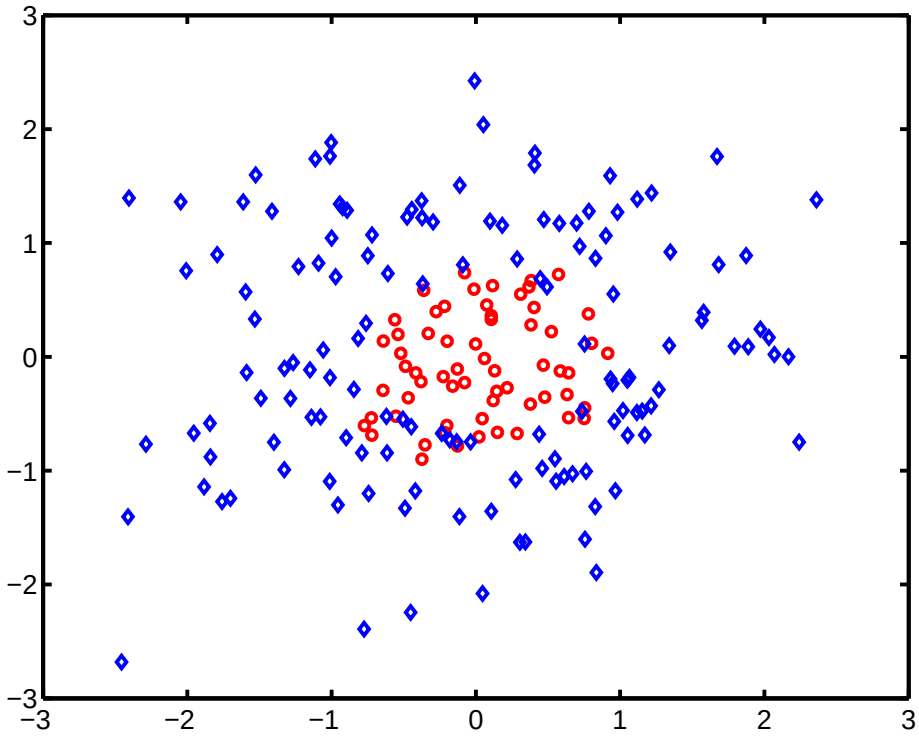

- 5-Nearest Neighbour.
- Much smoother.

# KNN – real example



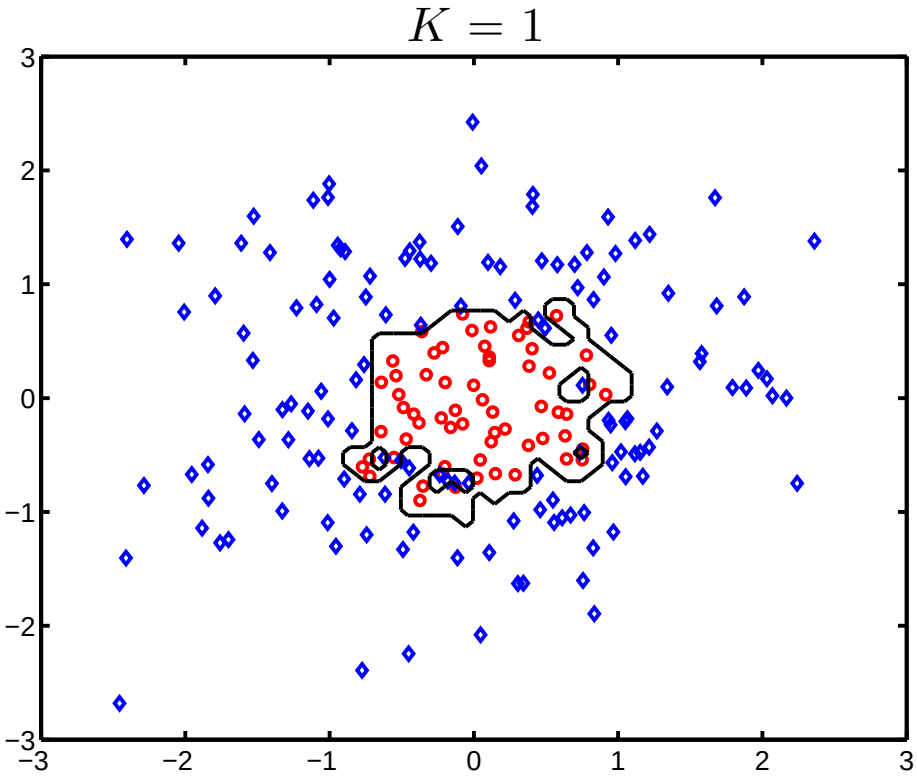

- 19-Nearest Neighbour.
- Very smooth.

# KNN – real example 2



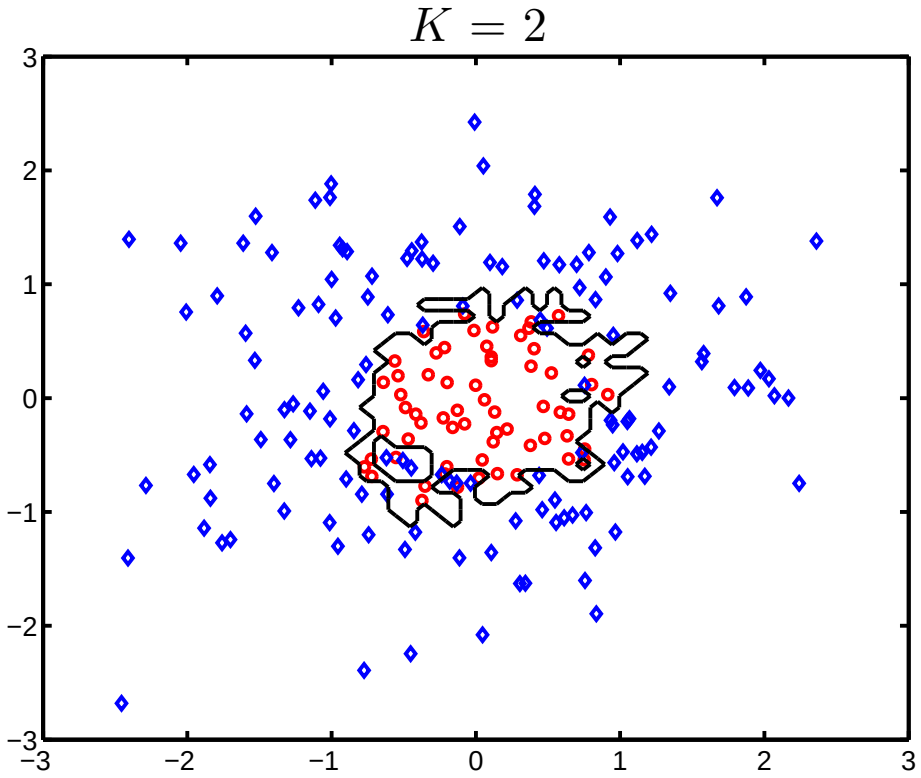

- Binary data.

# KNN – real example 2



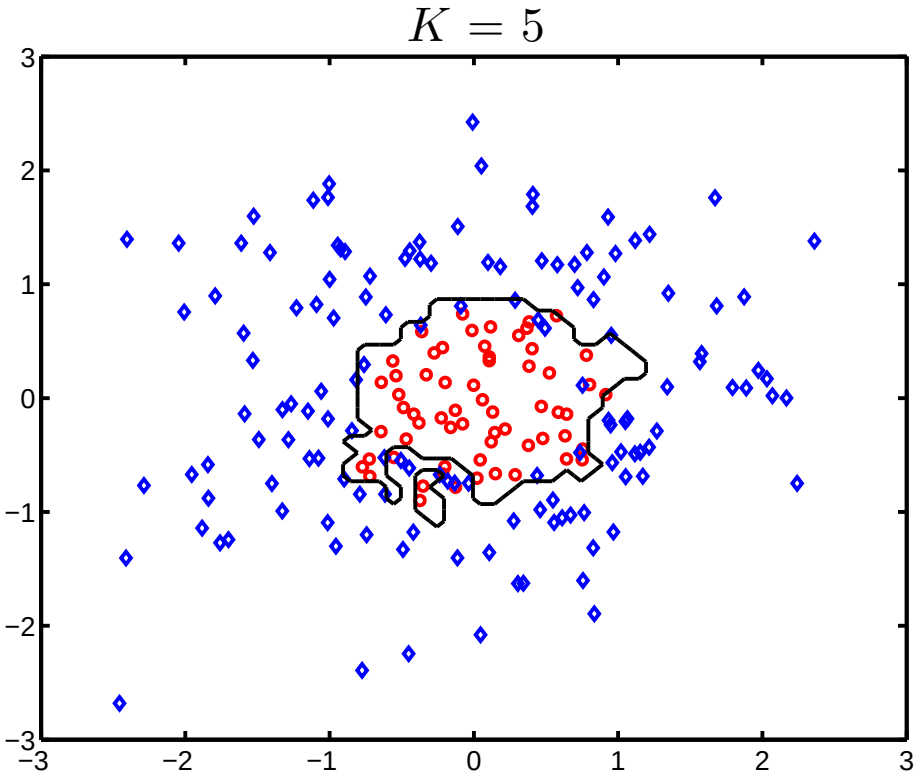

- Non-smooth – too complex again?

# KNN – real example 2



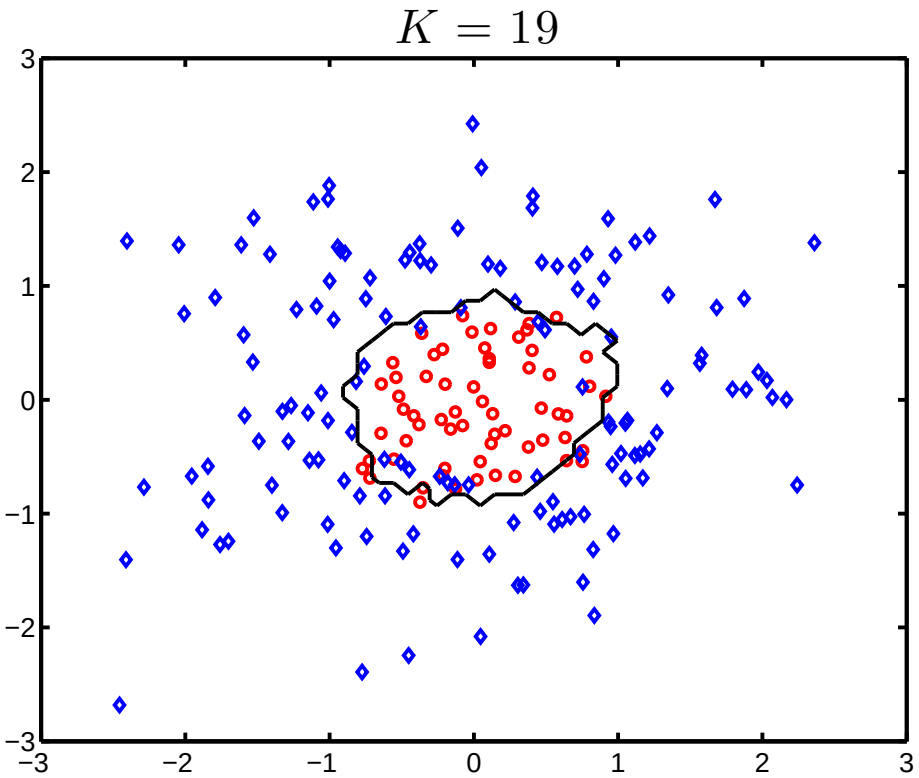

- Random effects again...

# KNN – real example 2




- Getting smoother.

# KNN – real example 2




- Smoother still.

# Problems with KNN

- Class imbalance
  - As $K$ increases, small classes will disappear!
  - Imagine we had only 5 training objects for class 1 and 100 for class 2.
  - For $K \geq 11$, class 2 will **always** win!

**Homework**: Neighborhood size for which class with least prior disappears from predictions?

Work out a general formula for the minimum values of $K(i)$ that make the classes ($C_i$s) disappear with respective priors $\alpha_1, \ldots, \alpha_M$ (assume wlog $\alpha_1 < \alpha_2 \ldots < \alpha_M$).

# Problems with KNN

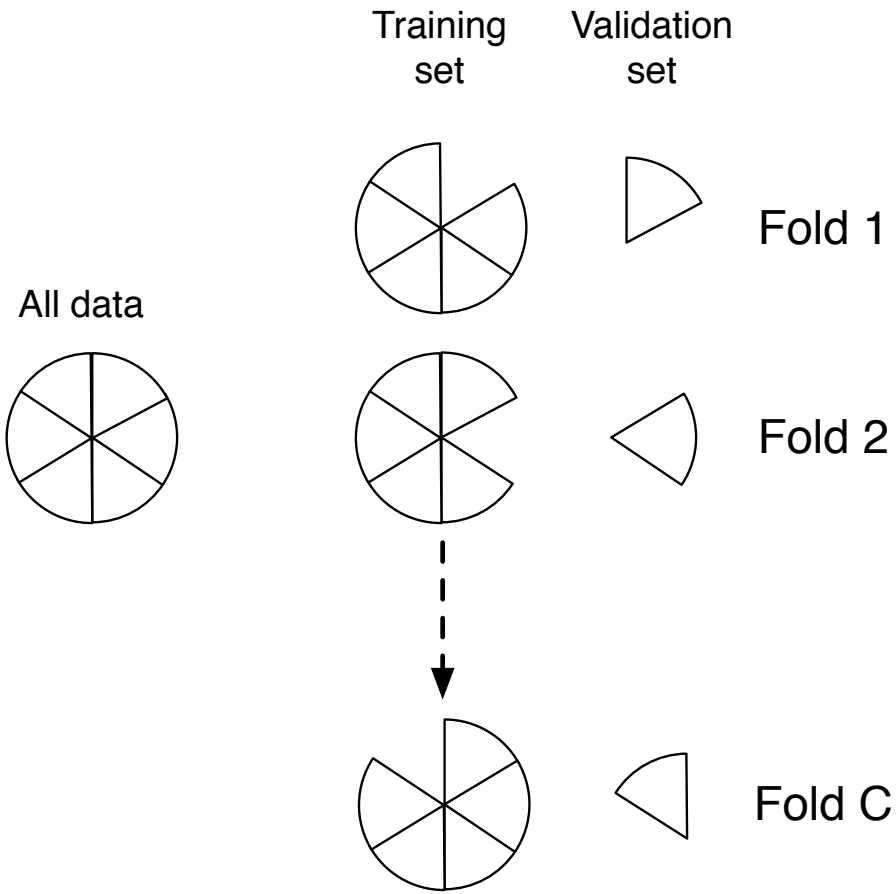
- Class imbalance
  - As $K$ increases, small classes will disappear!
  - Imagine we had only 5 training objects for class 1 and 100 for class 2.
  - For $K \geq 11$, class 2 will **always** win!

**Homework**: Neighborhood size for which class with least prior disappears from predictions?

Work out a general formula for the minimum values of $K(i)$ that make the classes ($C_i$s) disappear with respective priors $\alpha_1, \ldots, \alpha_M$ (assume wlog $\alpha_1 < \alpha_2 \ldots < \alpha_M$).

- What's a pragmatic choice for K?
  - Right value of K will depend on data.
  - Cross-validation!

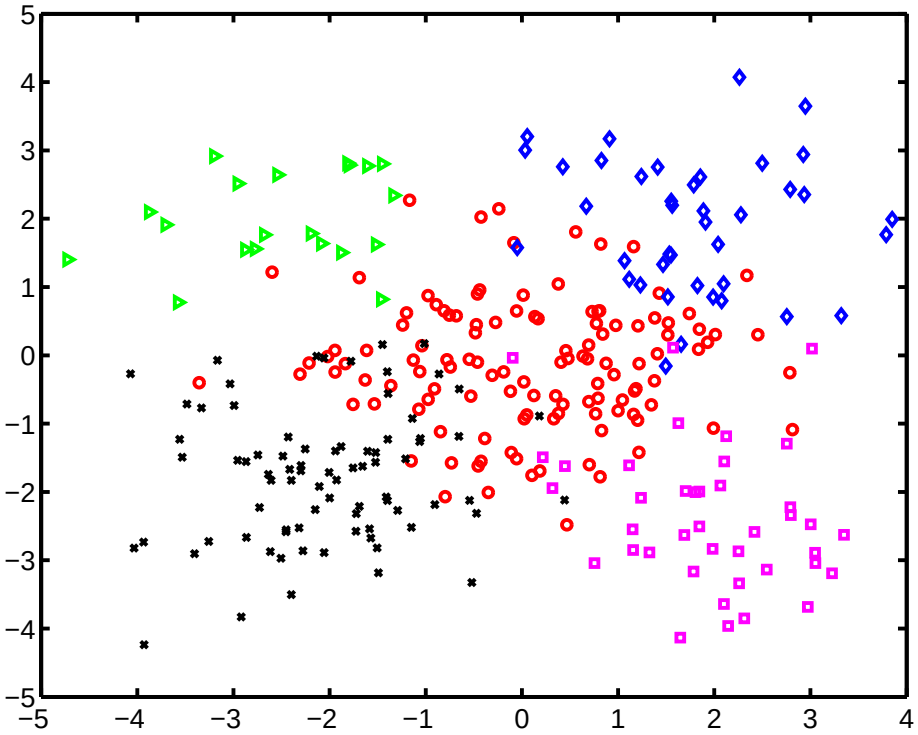
# Cross-validation to select optimal K




- Vary K from 1 to some max size.
- Select the one that leads to highest average performance measure, e.g.,
  - accuracy
  - precision, or
  - recall depending on the task.

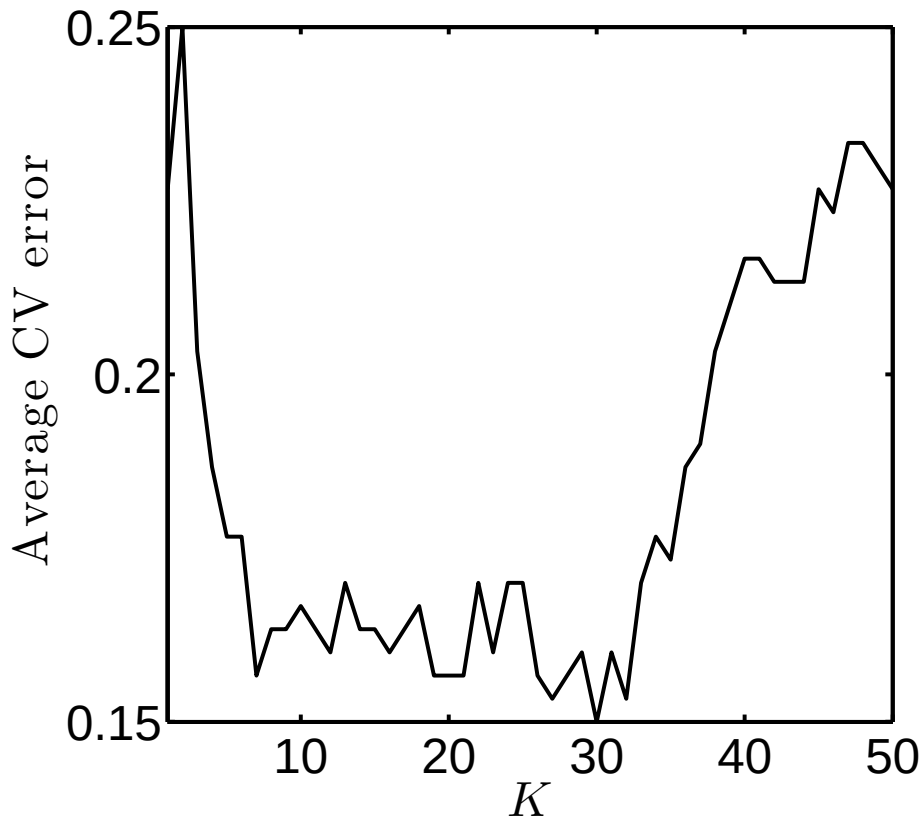
- Maximize the average performance measure across folds.

# Example – 5 classes



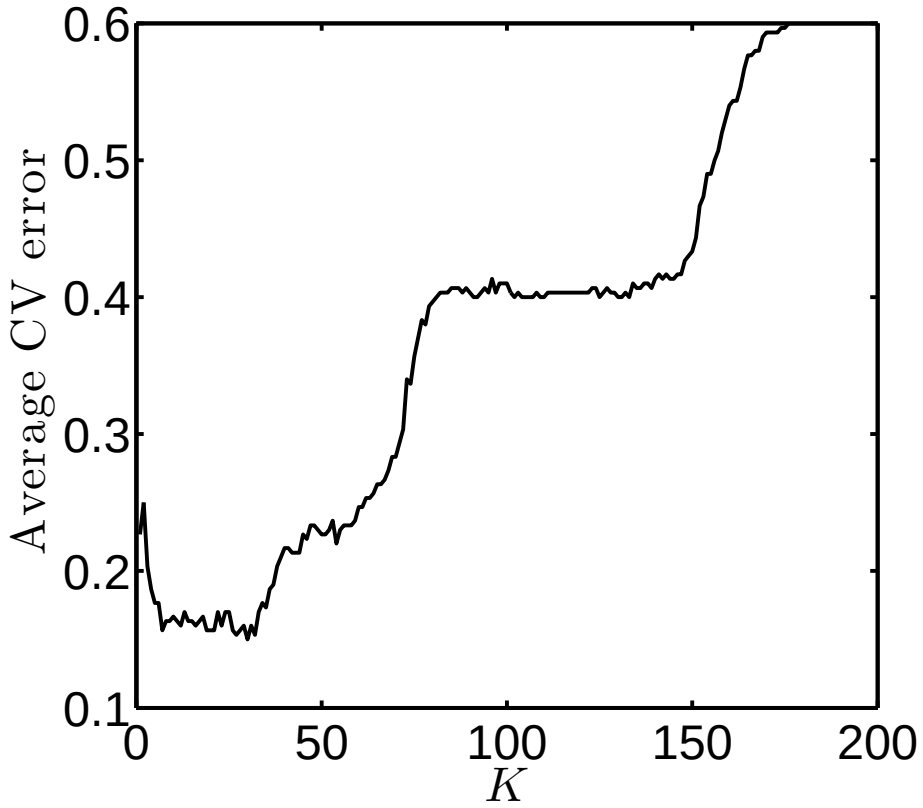

- 5 classes.
- Smallest has 20 instances, biggest 120.

# Example – 5 classes




- Curve shows average misclassification error for 10-fold CV.
- Minimum at approximately $K = 30$.

# Example – 5 classes




- As $K$ increases, classes 'disappear'
- Causes the 'steps' in error.

# Any other problems of KNN?

## Menti-Quiz (Code: 1373-7721)

Can you think of any other problems of K-NN that I didn't enlist in the last slide?

# Any other problems of KNN?

## Menti-Quiz (Code: 1373-7721)

Can you think of any other problems of K-NN that I didn't enlist in the last slide?

- Not efficient if $N$ (the number of data points) is very large.
- Computing the $k$-NN of a test point $\mathbf{x}_{new}$ takes $O(Nd)$ ($d$ the dimensionality of data).
- Solution is to use **approximate nearest neighbor search** (ANN).
- But then the problem is that the neighbourhoods are no longer exact.

# A Probabilistic version of KNN?

- KNN is non-probabilistic (assigns equal weights to each training instance in the neighbourhood).

**Menti-Quiz (Code: 1373-7721)**

Can there be a probabilistic version. How might it work?

# A Probabilistic version of KNN?

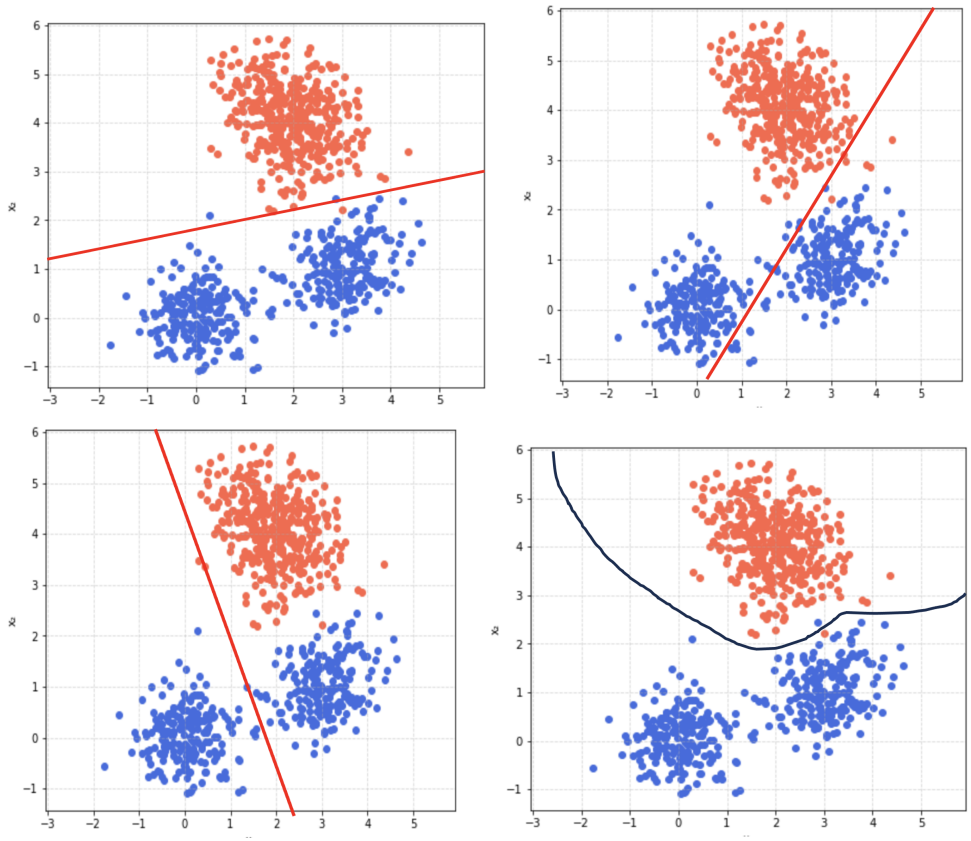
- KNN is non-probabilistic (assigns equal weights to each training instance in the neighbourhood).

## Menti-Quiz (Code: 1373-7721)

Can there be a probabilistic version. How might it work?

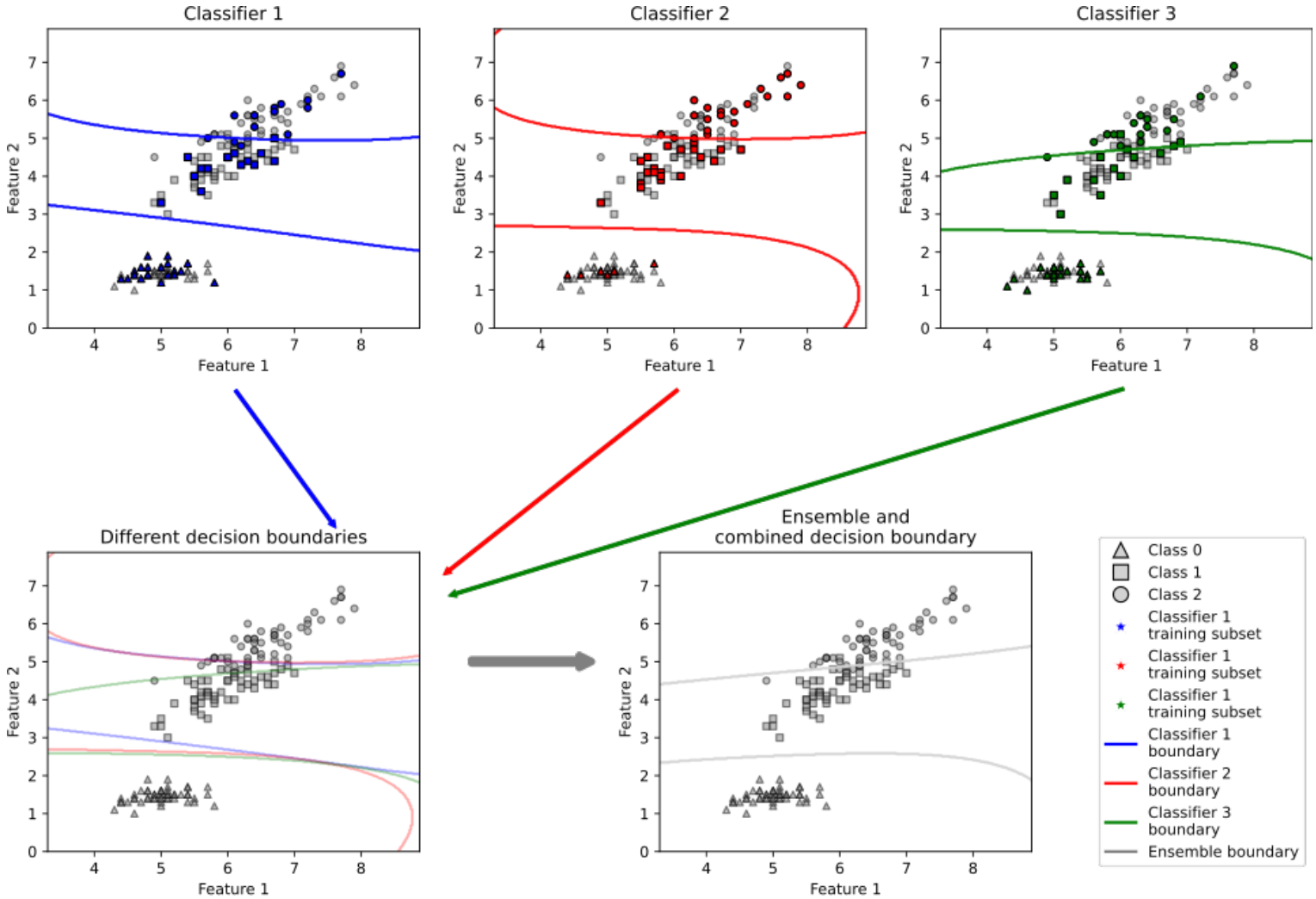
- Can use the similarity of a point $\mathbf{z}$ in the neighbourhood of $\mathbf{x}$.
- Higher the similarity, higher is the confidence that the class of a neighbor matches that of the input.
  - Standard K-NN: $P(y(\mathbf{x}) = k) = \frac{1}{K} \sum_{\mathbf{z}:y(\mathbf{z}=k)} 1$
  - Modified K-NN: $P(y(\mathbf{x}) = k) = \sum_{\mathbf{z}:y(\mathbf{z}=k)} \mathbf{x} \cdot \mathbf{z}$

# Combination of Models



- Different regions of the input space often requires a different local model.
- Learn local models on each cluster of the data.
- Combine the models to get a better decision boundary.

# Combination of Models




- Is particularly useful when data is available in streams.
- Is also able to handle outliers in data similar to Bayesian (next week).

# Mixture of Experts

- By this time you may have realised that this appears in the practical coursework.

## Training individual models on parts of data

- Cluster the data $X$ into $X_1, \ldots, X_n$.
- For each subset $X_i$, train a model $\theta_i$.
  - $\theta_i$'s don't need to be the same model, e.g., one could be logistic regression (week 6), and another could be K-NN (this week).

## A weighted combination of the models

- $P(y|\mathbf{x}; \theta; \phi) = \sum_{i=1}^{n} \phi_i P(y|\mathbf{x}; \theta_i)$
- $\phi$'s are the additional set of $n$ parameters — the weights (or prior beliefs) of each classifier.
- We need to learn the $\phi$'s.

# Mixture of Experts

## Menti-Quiz (Code: 1373-7721)

How will you learn the weights of the experts?

# Mixture of Experts

## Menti-Quiz (Code: 1373-7721)

How will you learn the weights of the experts?

- We can use Softmax!
- We need another set to learn the parameters.
- For each **x** set its target label to the index of the winner model.
- Underlying intuition: We are trying to make a **data-specific choice** of the available classifiers.
- Hoping that: such choices will **generalise** for the test set.

# Inference with MoE

## Menti-Quiz (Code: 1373-7721)

How will you infer the class of a test instance?

# Inference with MoE

## Menti-Quiz (Code: 1373-7721)

How will you infer the class of a test instance?

- $P(y|\mathbf{x} = k; \theta; \phi) = \sum_{i=1}^{n} \phi_i P(y = k|\mathbf{x}; \theta_i)$
- Aggregate (weighted average) the posteriors across each classifier.
- A combination of parametric and non-parametric approaches provides the best of both worlds.
  - Parametric: Captures the global characteristics of data distribution effectively.
  - Non-parametric: Captures the local effects.

# Summary

- Parametric Classification
  - Generalized Logistic Regression to multiple classes.
  - Also discussed multi-label classification
- Discussed a non-parametric approach - KNN
- MoE which can combine different classifiers.
  - MoE is the fundamental concept behind decision tree based classifiers, which we will not cover.