Multi-Class Classification

Combining Binary Classifiers

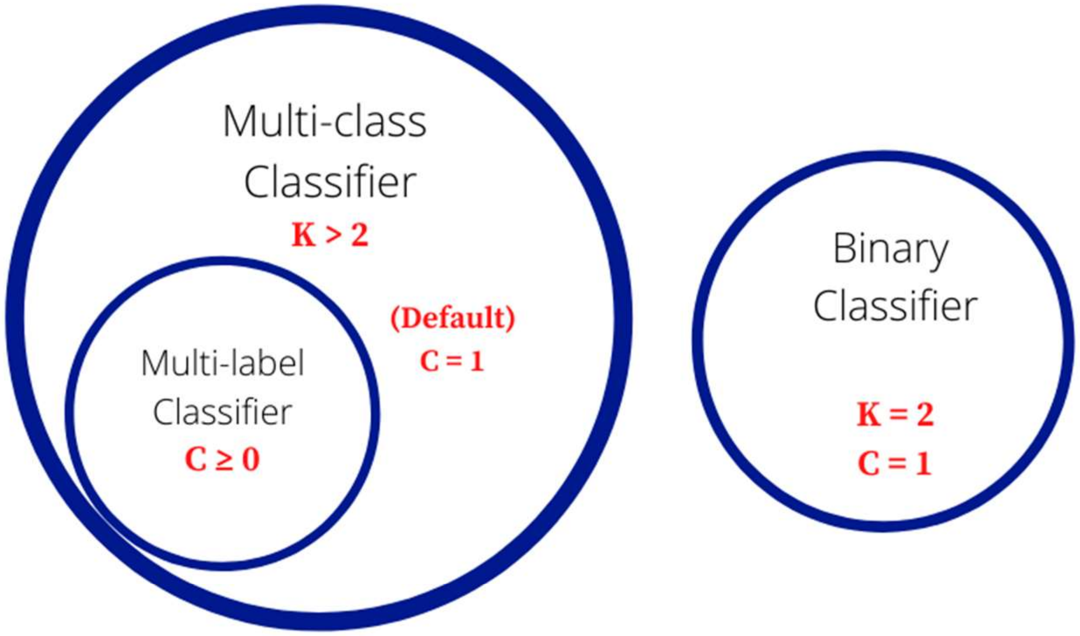
What is multiclass classification?

- An input can belong to one of K classes

- Training data: Input associated with class label (a number from 1 to K)
- Prediction: Given a new input, predict the class label

Each input belongs to exactly one class. Not more, not less.

- Otherwise, the problem is not multiclass classification
- If an input can be assigned multiple labels (think tags for emails rather than folders), it is called *multi-label classification*



K = Total number of classes in the problem statement

C = Number of classes an item maybe assigned to

Multi-Class Classification

- What colour is the dog in this photo?

White

Brown

Black

Multi-Label Classification

- What colour and gender is the dog in this photo?

White
Male

Brown
Female

Black
Female

Approaches for multiclass classification

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
  $x_i \epsilon R^d, y_i \epsilon \{1, 2, \dots, K\}$
- Find $f(x): R^d \rightarrow \{1, 2, \dots, K\}$ that outputs correct labels

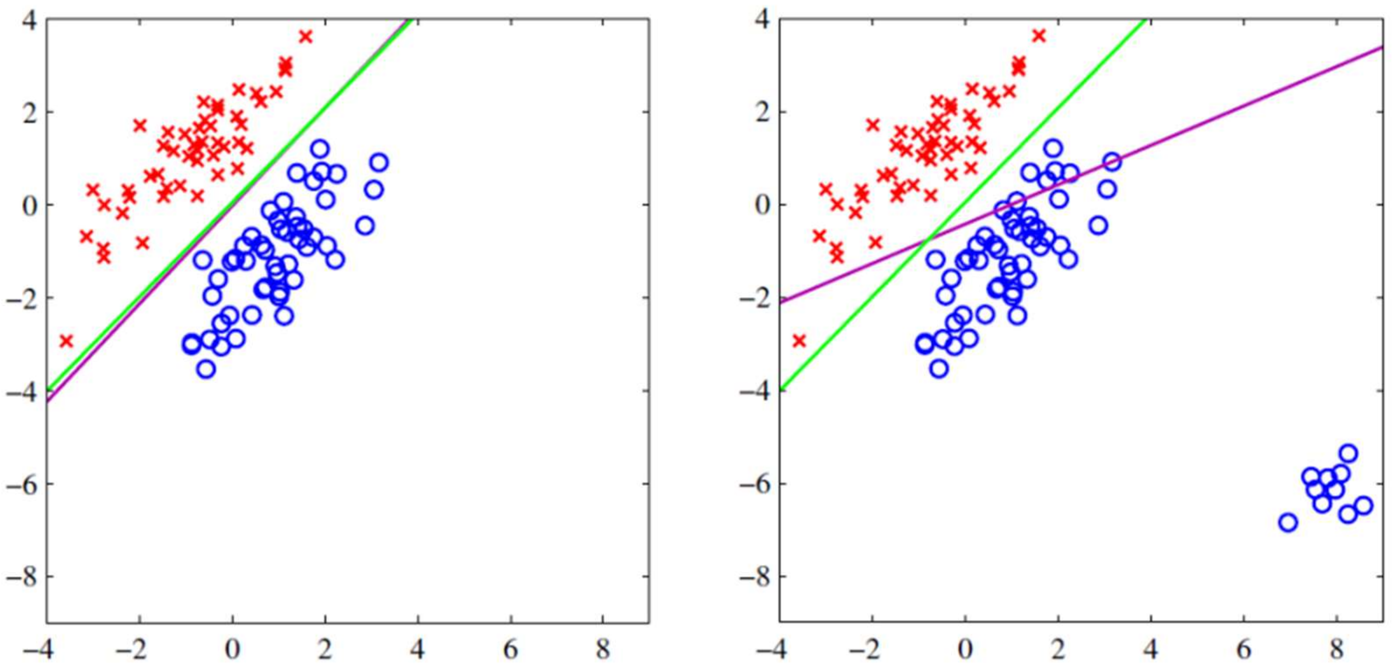
- What kind of f?

Reduce to Regression

➢ Find $f_W(x) = w^T x$ that minimizes $\hat{L}(f_W) = \frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)^2$

➢ Bad idea even for binary classification

Reduce to Regression



Bad idea even for binary classification

Figure from *Pattern Recognition and Machine Learning*, Bishop

Figure 4.4 The left plot shows data from two classes, denoted by red crosses and blue circles, together with the decision boundary found by least squares (magenta curve) and also by the logistic regression model (green curve), which is discussed later in Section 4.3.2. The right-hand plot shows the corresponding results obtained when extra data points are added at the bottom left of the diagram, showing that least squares is highly sensitive to outliers, unlike logistic regression.

Two key ideas

- Reducing multiclass to binary
  - Decompose the multiclass prediction into multiple binary decisions
  - Make final decision based on multiple binary classifiers

✓ Future-proof: binary classification improved so does muti-class

✓ Easy to implement

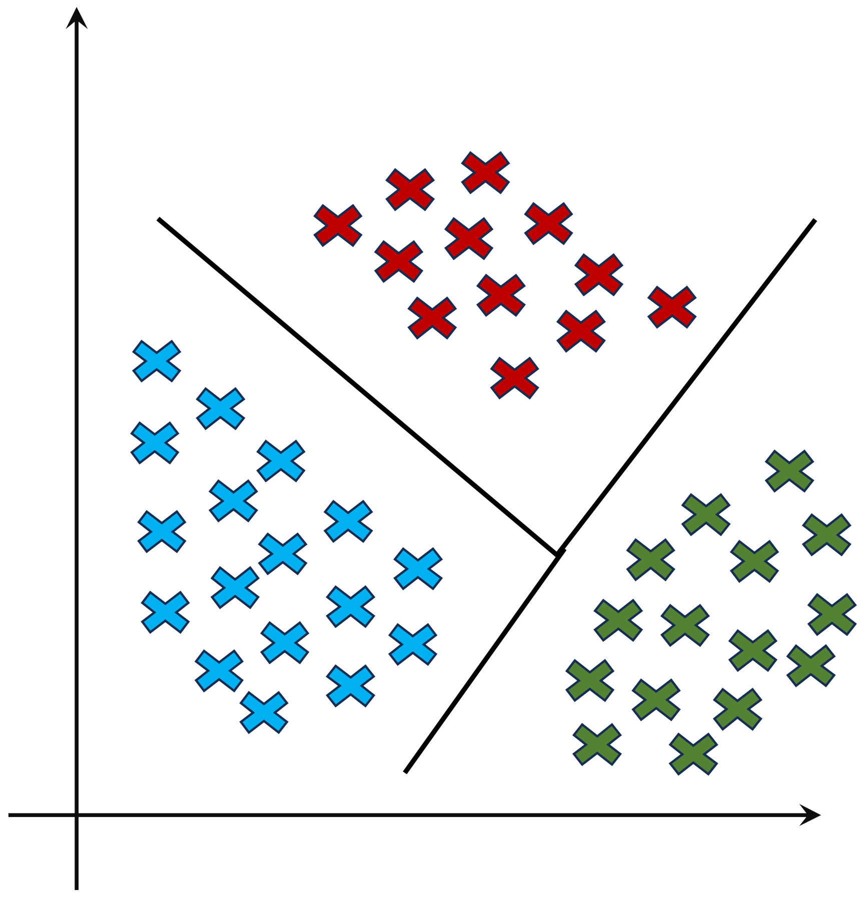
- Training a single classifier
  - Minimize the empirical risk
  - Consider all classes simultaneously

✓ Global optimization: directly minimize the empirical loss; easier for joint prediction

✓ Easy to add constraints and domain knowledge

Linear Classifiers

- Linear Classifier between 2 classes is parametrized by $\mathbf{w}^{\mathrm{T}}\mathbf{x} = 0$:
  - $\mathbf{w}^{\mathrm{T}}\mathbf{x} < 0$ is class 0
  - $\mathbf{w}^{\mathrm{T}}\mathbf{x} \geq 0$ is class 1
- What if we have 3 or more classes?
  - Can we find piece-wise linear boundaries that can partitions the feature space into three?
  - How do we find the lines? How do we combine them?



One Against All Learning

- Learning: Given a dataset $\mathrm{D} = \{(\boldsymbol{x_i}, \boldsymbol{y_i}): \mathbf{1} \leq \boldsymbol{i} \leq \boldsymbol{n}\},\ \boldsymbol{x_i} \,\boldsymbol{\epsilon}\, \boldsymbol{R^n}, \boldsymbol{y_i} \,\boldsymbol{\epsilon}\, \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \ldots \boldsymbol{K}\}$

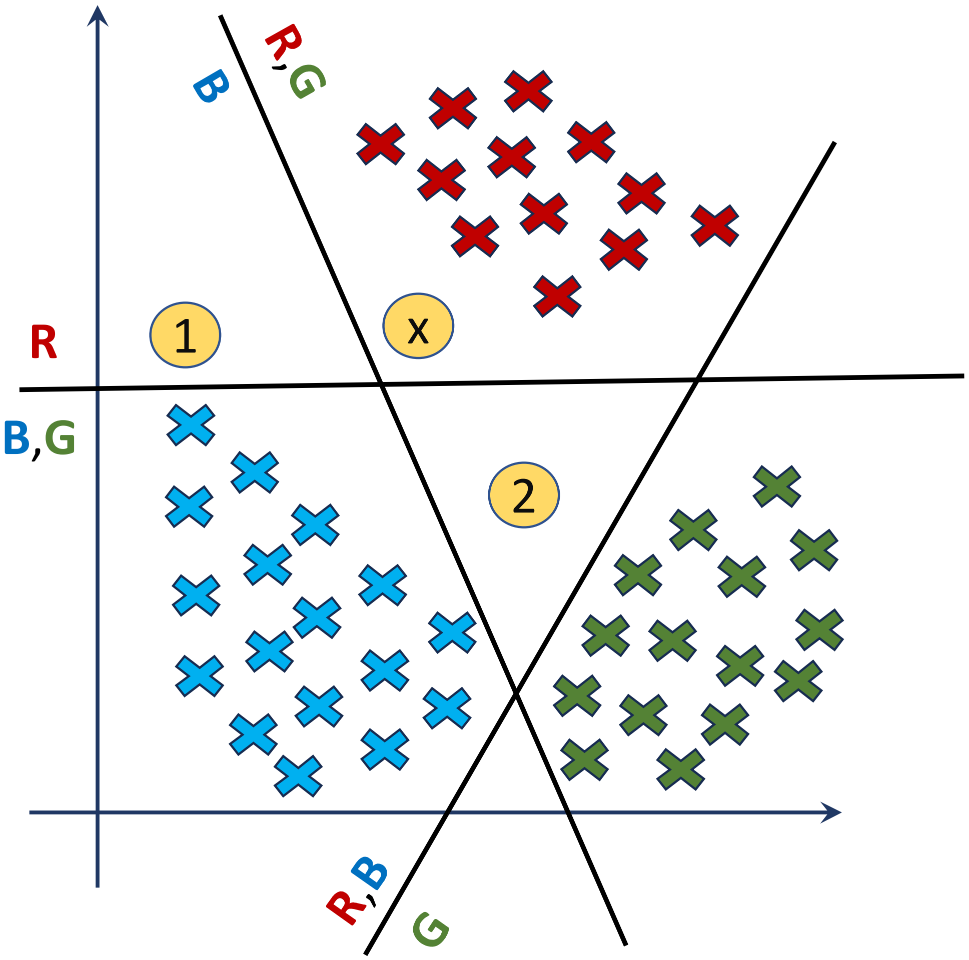
- Decompose into binary problems with $\boldsymbol{K}$ binary classification tasks
  - Learn $\boldsymbol{K}$ models: $w_1, w_2, w_3, \ldots, w_k$

Ideal case: only the correct label will have a positive score

- Inference: "Winner takes all"
  - $\boldsymbol{\hat{y} = argmax_{y\epsilon\{1,2,\ldots,k\}} f(y; w, x)}$
  - $\boldsymbol{w = \{w_1, w_2, w_3, \ldots, w_k\}, \mathrm{f}(\mathrm{y}; \mathrm{w}, \mathrm{x}) = \ w_y^T x}$

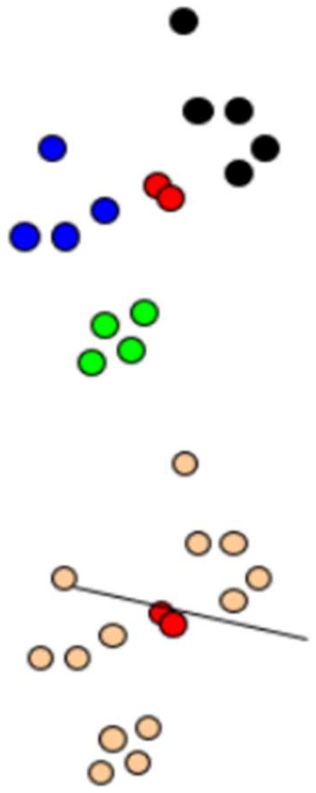
One-vs-Rest

- Compute 1 line per class:
  - R vs. {B,G} or ¬R
  - B vs. {R,G} or ¬B
  - G vs. {R,B} or ¬G
- Test Point: x
  - R, {R,G}, {R,B}
- What happens to points in between?
  1. R, B and {R,B} : Tie
  2. ¬R, ¬B and ¬G : 3-way tie
- How do we break the ties?
  - Weighted Voting?



One-vs-Rest

- Not always possible to learn
  - Assumption: each class individually separable from all the others
- No theoretical justification
  - Need to make sure the range of all classifiers is the same – we are comparing scores produced by K classifiers trained independently.
- Easy to implement; work well in practice
- Several possibilities for indecision/tie

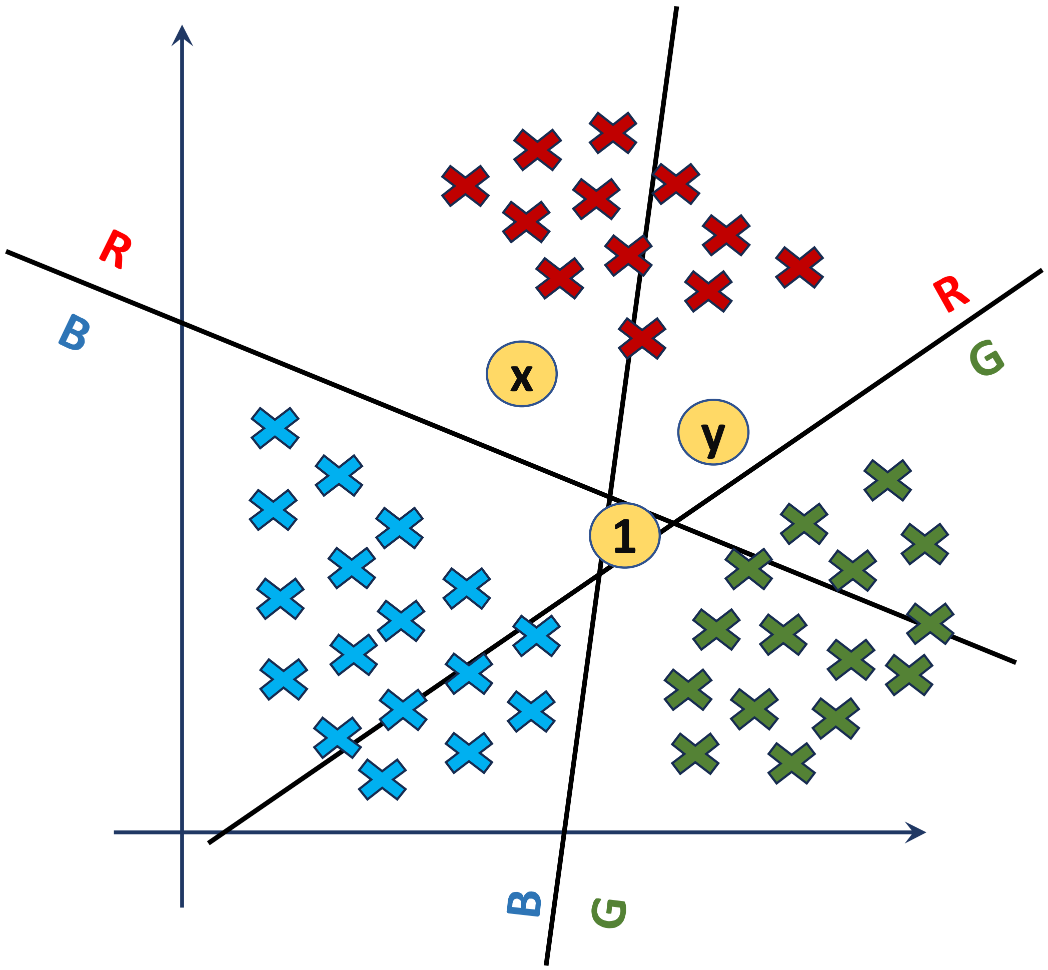


One vs. One

- Learning: Given a dataset $\mathrm{D} = \{(\boldsymbol{x_i}, \boldsymbol{y_i}): \mathbf{1} \leq \boldsymbol{i} \leq \boldsymbol{n}\},\ \boldsymbol{x_i} \,\epsilon\, \boldsymbol{R^n}, \boldsymbol{y_i} \,\epsilon\, \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \dots \boldsymbol{K}\}$
- Decompose into binary problems with ${}^{K}C_{2}$ binary classification tasks
  - Learn ${}^{K}C_{2}$ models: $w_1, w_2, w_3, \dots, w_{k*(k-1)/2}$
  - For each class pair **(i,j)**, construct a binary classification task as:
    - Positive examples: Elements of **D** with label **i**
    - Negative examples Elements of **D** with label **j**
    - The binary classification can be solved by any algorithm we have seen

One-vs-One [Pairwise]

- Compute 1 line per class-pair:
  - R vs. B
  - B vs. G
  - G vs. R
- Test Point:
  - x: R, B, R
  - y: R, G, R
  - 1: B, G, R : 3-way tie
- How do we break the ties?

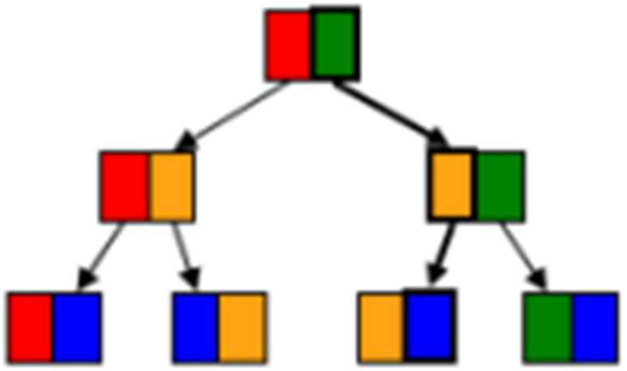


One-vs-One [Pairwise]

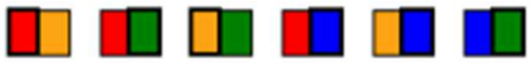
- Build ${}^{n}C_{2}$ classifiers and vote (compute heavy)

- Decision Options:
  - More complex;
  - Output of binary classifier may not cohere.
  - Majority: classify example x to take label $\mathbf{i}$ if $\mathbf{i}$ wins on $\mathbf{x}$ more often than j $(\mathbf{j}=1,\ldots\mathbf{k})$
  - A tournament: start with n/2 pairs; continue with winners

- Weighted voting can break ties



Majority Vote

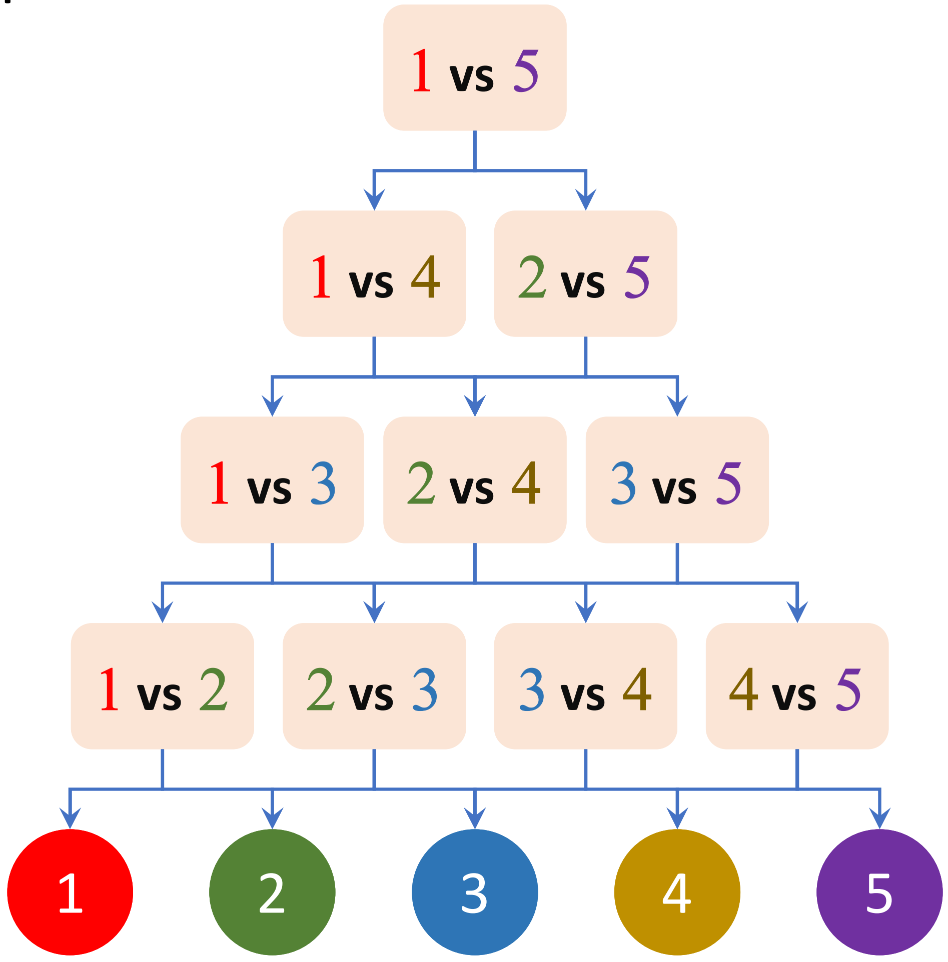


1 red, 2 yellow, 2 green

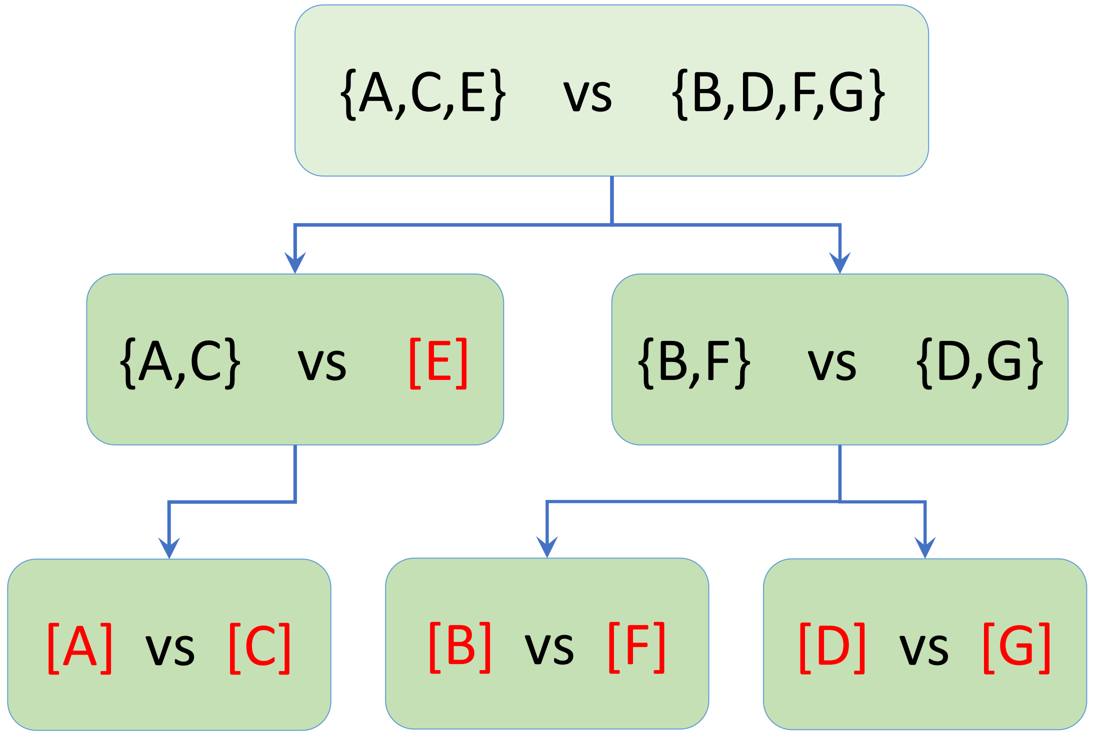
$\rightarrow$?

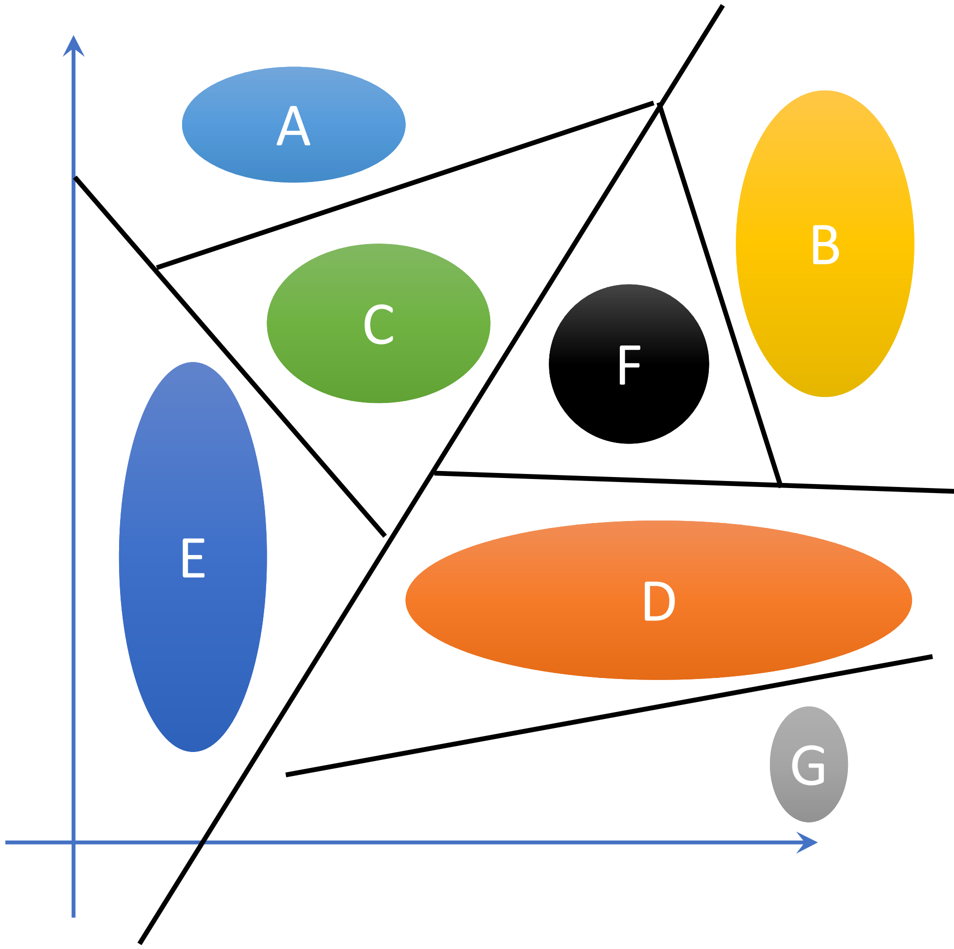
Decision Directed Acyclic Graph

- What can we infer if X_1 vs X_2 classifier gives X_2 in a 5-class problem?
  - We cannot say the test sample belongs to X_2.
  - However, we can infer that the sample most likely is not in X_1.
- Use this idea to eliminate classes as we move along
- This is not a tree, but a graph (acyclic), where edges are based on decisions from a classifier



Hierarchical Classifier Combination





Summary: Efficient Approaches

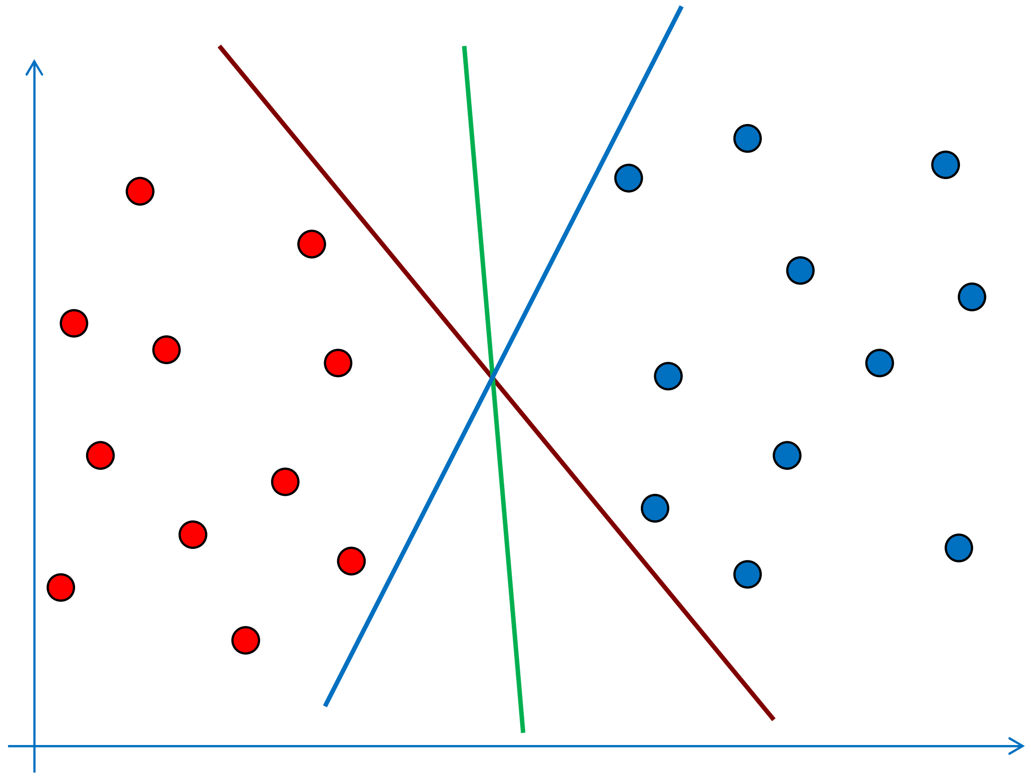
- DDAG:
  - Build $n(n-1)/2$ classifiers and classify using $n-1$ of them
  - Compute heavy to build, but efficient in inference
  - Effective in classification (uses 1-vs-1)
- Hierarchical
  - Build $O(n)$ classifiers by appropriate grouping of classes
    - Infer with $O(\log n)$ classifiers.
  - Far more efficient that 1-vs-1 (or DDAG)
  - Similar in accuracy to 1-vs-1 and DDAG.
  - Difficult to find the right grouping of classes
- No Ties

Training a single classifier

Improving Test Accuracy

Perceptron Learning

- Multiple solutions exist for linearly separable data
- Perceptron learning (any GD) results in a feasible solution



Are all solutions equally good?

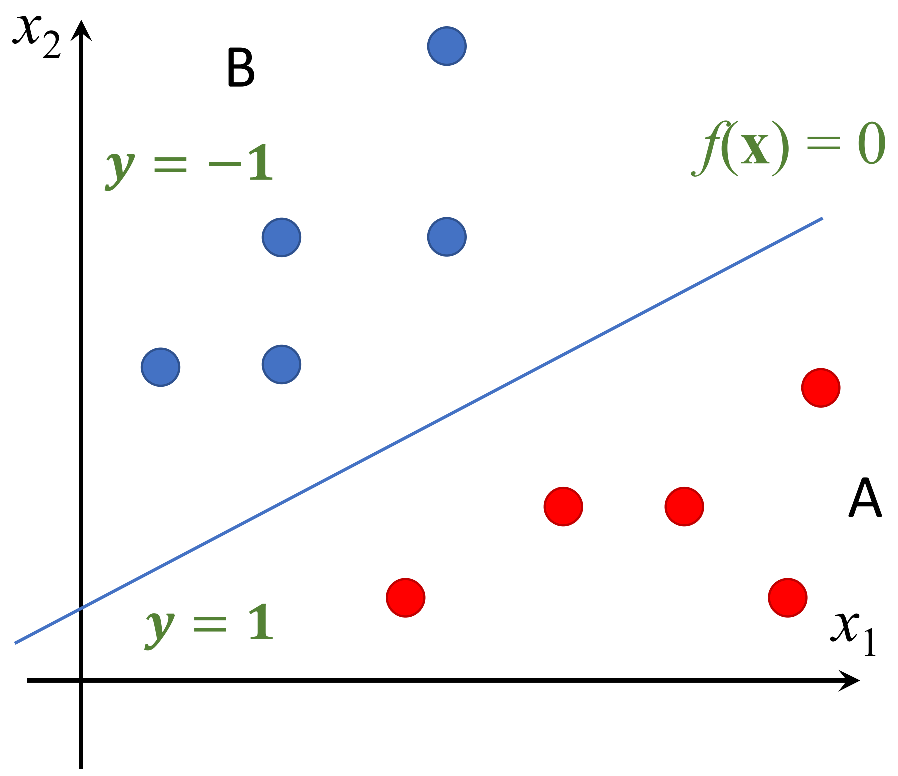
Loss Function

$$f(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + w_0$$
$$= w^T x = 0$$

We have

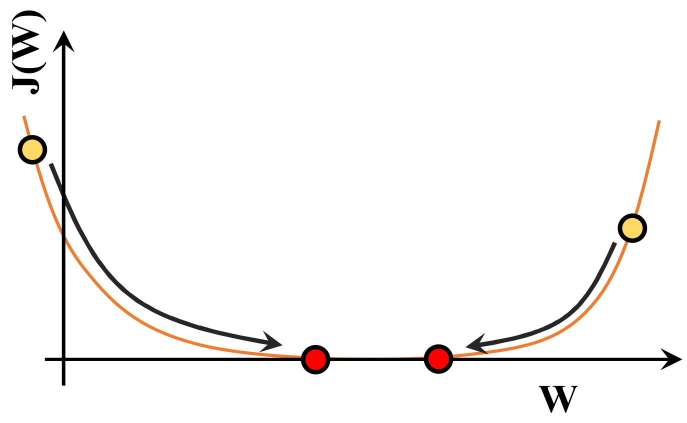
- $\mathbf{w}^T\mathbf{x}_a$ = +ve $|\forall\, x_a \in A$, if $y = 1$
- $\mathbf{w}^T\mathbf{x}_b$ = -ve $|\forall\, x_b \in B$, if $y = -1$

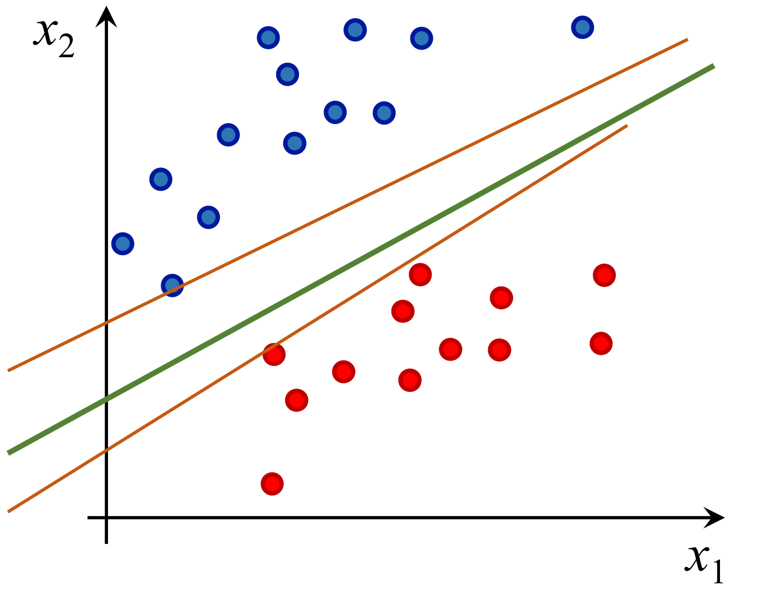
- $y_i(w^T x_i)$ > 0 for all samples
- $J(\mathbf{w}) = -y_i . w^T x_i$



Improving the Loss Function

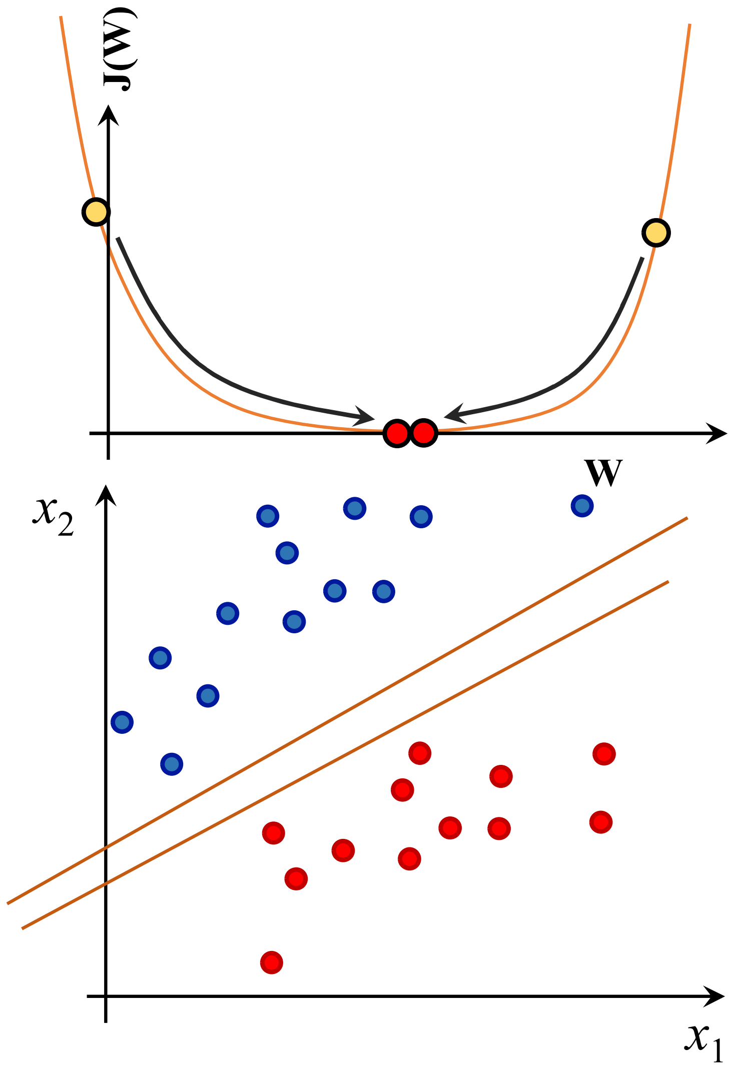
- Perceptron Loss: $J_p(\mathbf{w}) = \sum_{\mathbf{x}_i \in X} -\mathbf{w}^T \mathbf{x}_i$
  - X is the set of samples for which $\mathbf{w}^{\mathrm{T}}\mathbf{x}_i < 0$.
  - $\boldsymbol{x_i}$s are augmented and multiplied by sign
  - Can we improve the smoothness?
- $J_q(\mathbf{w}) = \sum_{\mathbf{x}_i \in X} \left(\mathbf{w}^T \mathbf{x}_i\right)^2$
  - Smoother loss function
  - Challenges:
    - So smooth that it often converges to the boundary
    - Dominated by the longest vectors
  - We would like a boundary that is at some distance (margin) away from the samples





Improving the Loss Function

- $\boldsymbol{J_q(\mathbf{w}) = \sum_{\mathbf{x}_i \in X} (\mathbf{w}^T \mathbf{x}_i)^2}$
- Improvements: Normalize and add a margin
  - We need $\mathbf{w}^{\mathrm{T}}\mathbf{x}_i > k$, where k is the margin
  - Normalize the loss by $\|\mathbf{x}_i\|^2$.
- $\boldsymbol{J_r(\mathbf{w}) = \frac{1}{2} \sum_{\mathbf{x}_i \in X} \frac{(\mathbf{w}^T \mathbf{x}_i - k)^2}{\|\mathbf{x}_i\|^2}}$
  - X is the set of samples for which $\mathbf{w}^{\mathrm{T}}\mathbf{x}_i < k$.
- Note: $\boldsymbol{\nabla J_r = \sum_{\mathbf{x}_i \in X} \frac{\mathbf{w}^T \mathbf{x}_i - k}{\|\mathbf{x}_i\|^2} \mathbf{x}_i}$



Notes on Margin

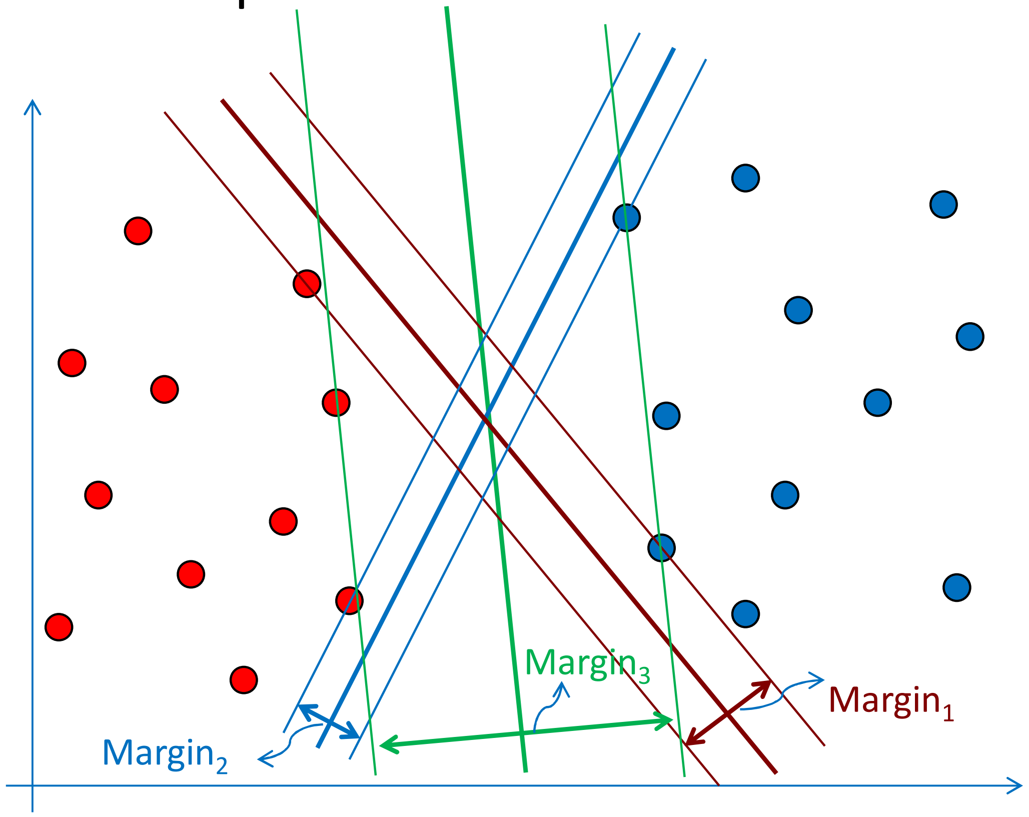
- A larger margin means the boundary is further away from the training samples
  - Convergence during training is faster
  - Small perturbations to test samples do not change their label
    - Better Generalization
- k does not give actual margin
  - Remember that $\mathbf{w}^{\mathrm{T}}\mathbf{x}_i$ is the scaled distance from the hyperplane
- How do we deal with the scale?
- Should we maximize the margin?

Maximum Margin Classification

Best Generalizers

Visualizing Margin: The No-mans Band

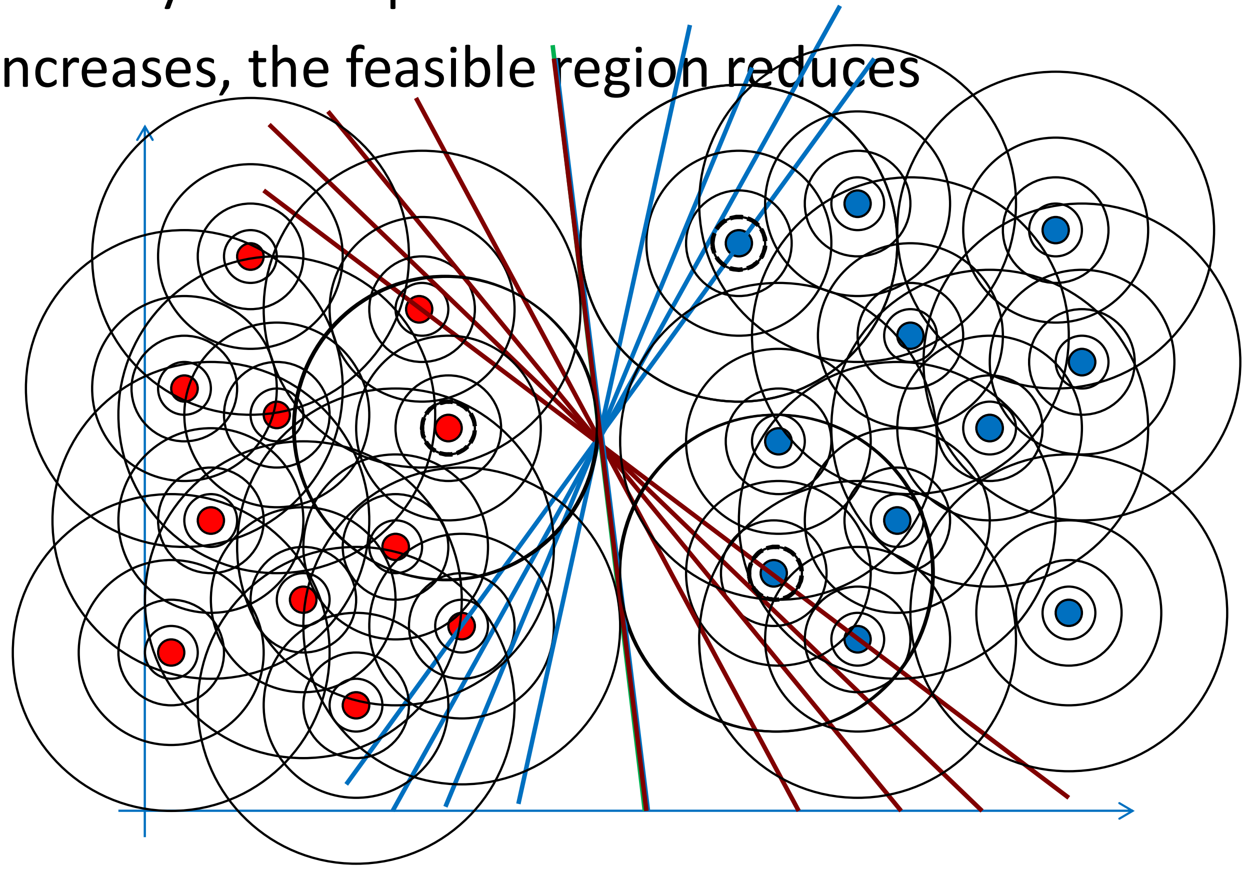
- Margin: Width of a band around decision boundary without any training samples
- Margin varies with the position and orientation of the separating hyperplane



Is a Larger Margin better? Why?

Visualizing Margin: Bubbles around Samples

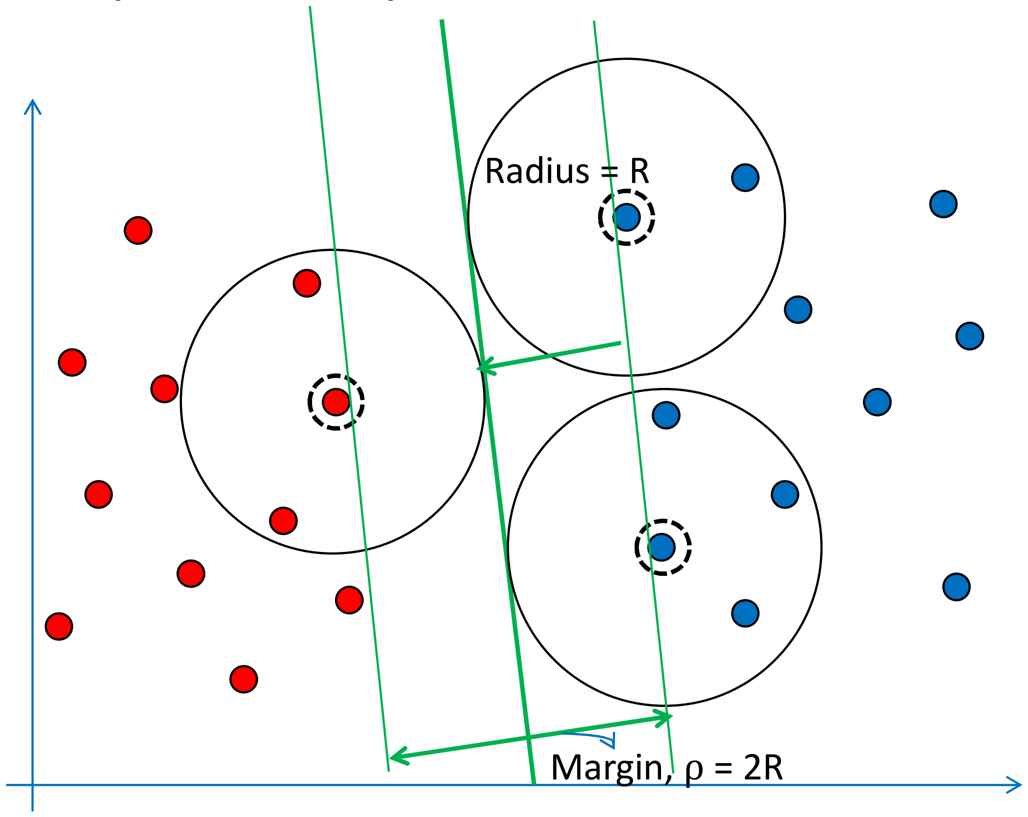
- Margin: Radius of a region around each training sample, through which the decision boundary cannot pass
- As margin increases, the feasible region reduces



A few samples control the Dec. boundary

Margin: Band vs. Bubbles

- Margin: Both interpretations yield the same decision boundary



Samples that support the Decision boundary are called ***Support Vectors***