CLASSIFICATION

Classification Algorithms and Support Vector Machines

WHAT IS CLASSIFICATION?

Critical Thinking: What is *classification*? What is the difference between *regression* and *classification*?

Regression: the dependent variable is continuous and we want to predict the expected value given the input features.

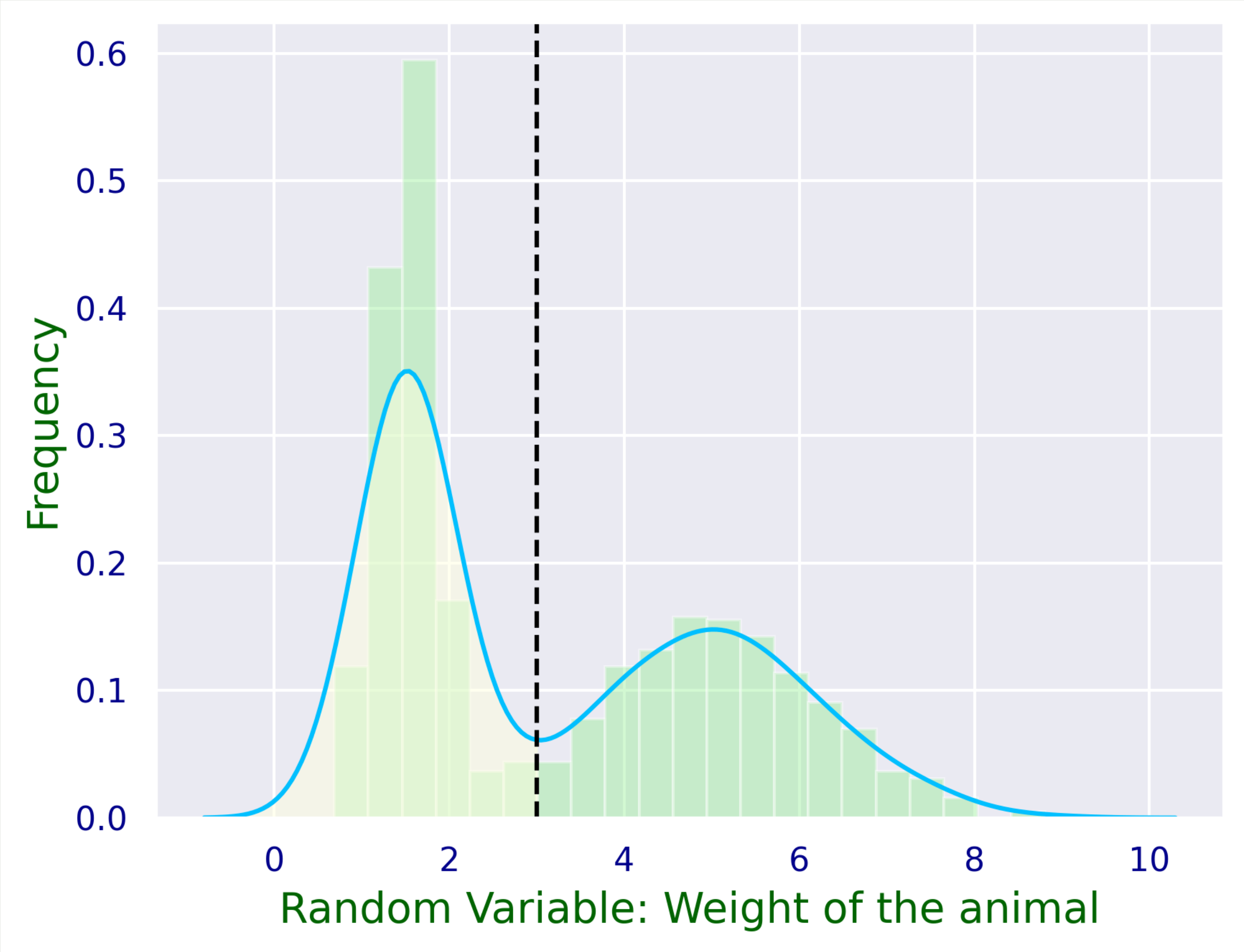
Classification: the dependent variable is binary or nominal and we want to predict the corect class given the input features.

If we had one input feature as a continuous variable we could ***see*** the classification.

Example Let's imagine we have data for the weights of two different animals and we would like to know whether the *weight* alone may be a good predictor for what type of animal there is.

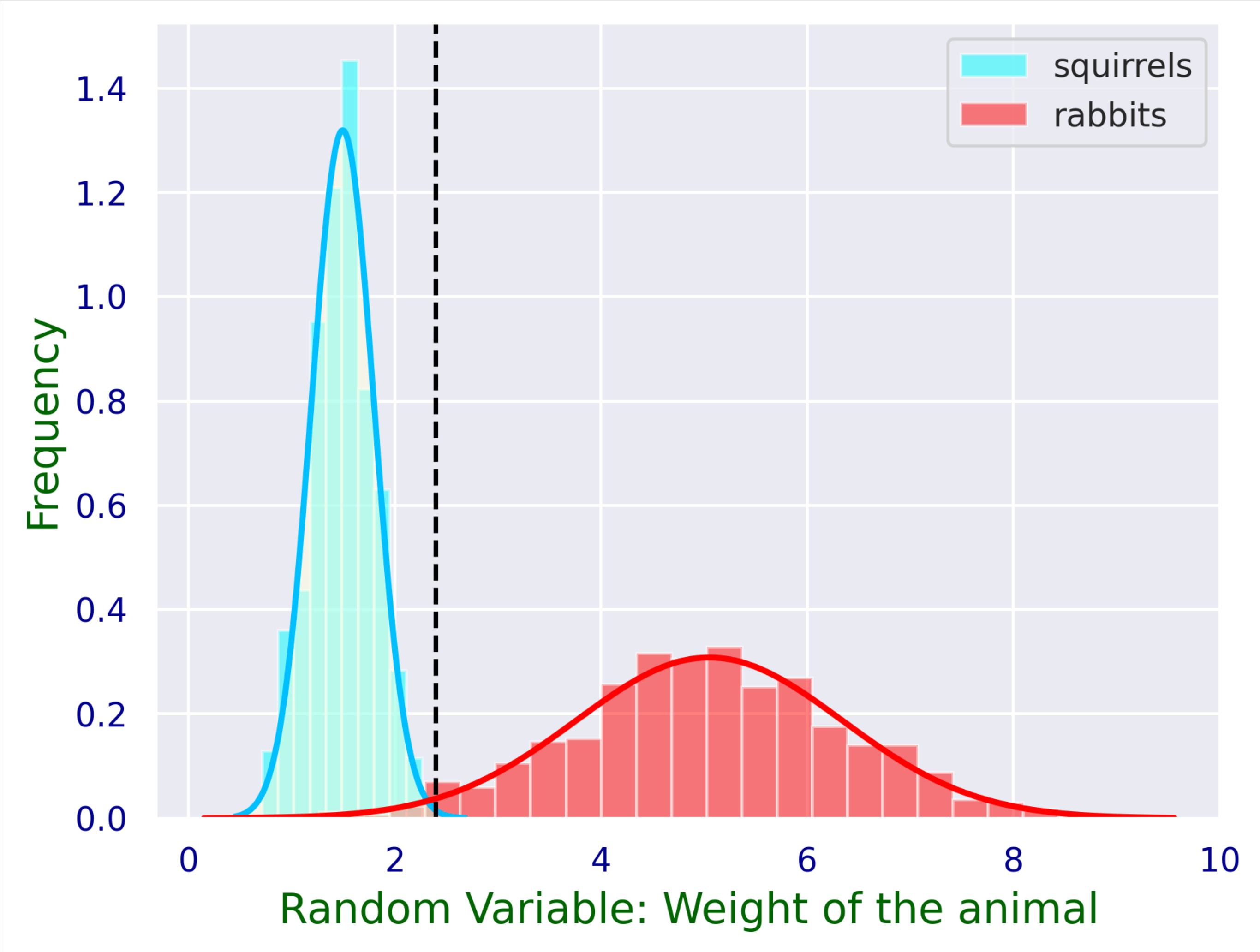
WHAT IS CLASSIFICATION?

Based on the weight alone, we want to know if the animal is more likely to be a squirrel or a rabbit.



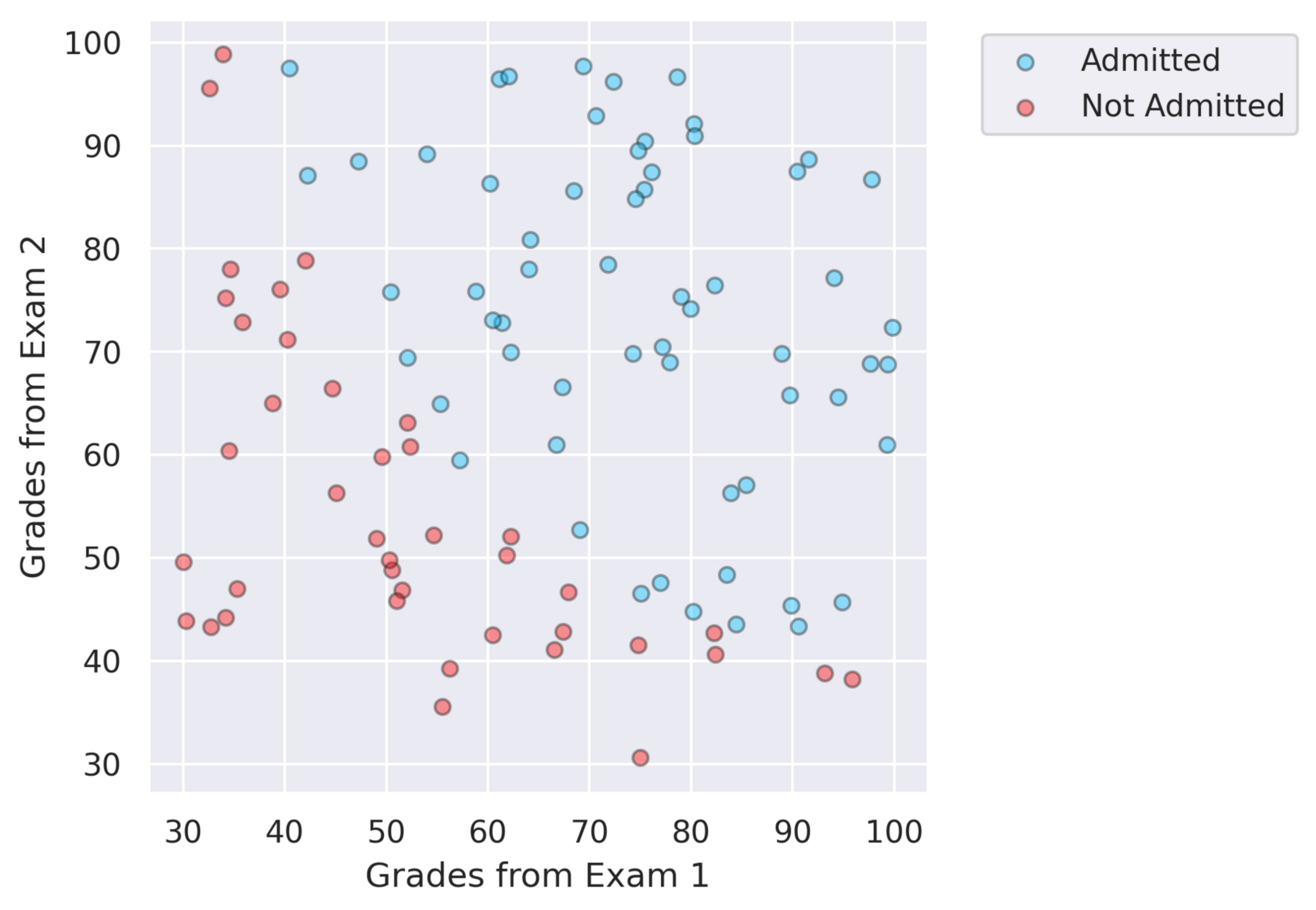
WHAT IS CLASSIFICATION?

Based on the weight alone, we want to know if the animal is more likely to be a squirrel or a rabbit.



WHAT IS CLASSIFICATION?

Example in 2-D: Do the exam grades predict an outcome?



LOGISTIC REGRESSION

What we want: classify by using a probability model (an estimate of the odds-ratio) such as a **straight** line or a **sigmoid** curve.

IMPORTANT: If we divide two probability values we get an output between 0 and ∞ (the infinity is approached when the denominator is very close to 0 and the numerator is very close to 1.

The ***odds-ratio*** is

$$\frac{\mathrm{P}(y_i = 1|\text{feature data})}{\mathrm{P}(y_i = 0|\text{feature data})}$$

Critical Thinking: Can we predict the odds ratio as a regression problem? Why and why not?

LOGISTIC REGRESSION

Classification by a straight line is possible but less desirable (as you can see in the picture.)

The concept of the logistic regression in a multivariate setup is to model the log of the odds ratio as a linear function of the features:

$$\log\left(\frac{\mathrm{P}(y_i = 1|\text{feature data})}{\mathrm{P}(y_i = 0|\text{feature data})}\right) = \beta_0 + \beta_1 \cdot x_i$$

where y_i represents the $i-th$ output (classification) and x_{ij} represent the features of the $i-th$ observation.

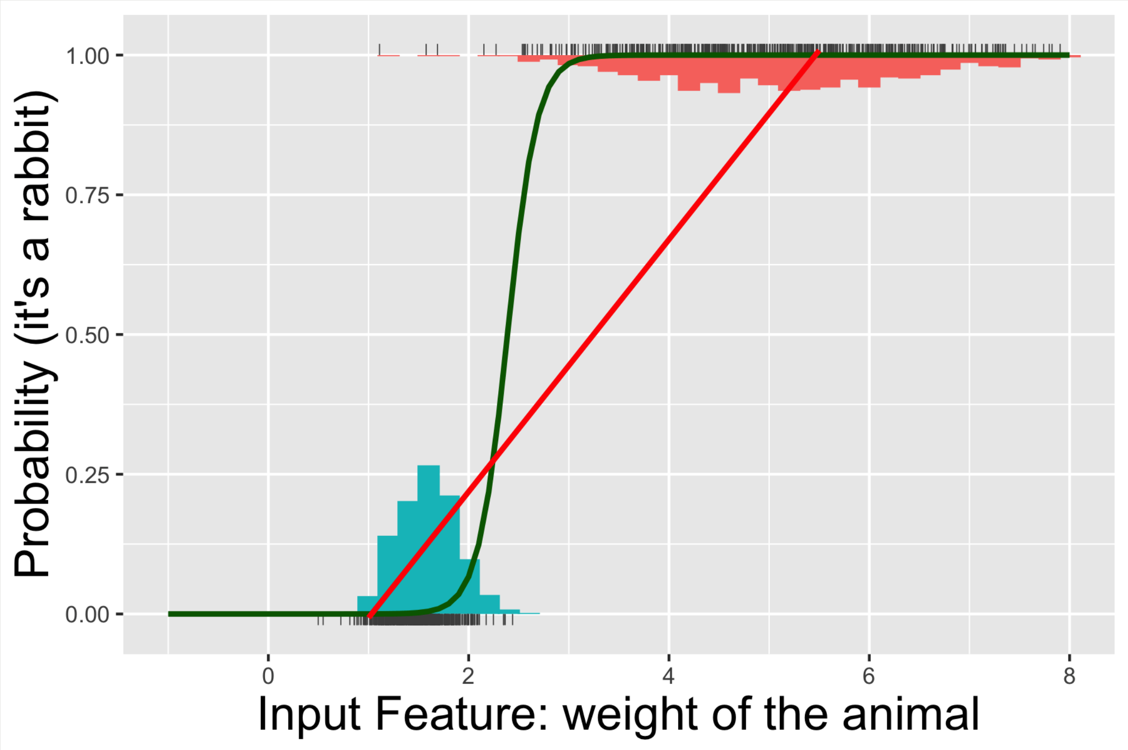
LOGISTIC REGRESSION

$$\mathrm{P}(y_i = \text{rabbit}|\text{weight} = x_i) + \mathrm{P}(y_i = \text{squirrel}|\text{weight} = x_i) = 1$$

We get

$$\mathrm{P}(y_i = \text{rabbit}|\text{weight} = x_i) = \frac{1}{1 + e^{-\beta_0 - \beta_1 x_i}}$$

the above function is called the "Logistic Sigmoid" (ref. Thomas Malthus)



THE MACHINE LEARNING OF LOGISTIC REGRESSION

The main idea is that we approximate the probability of Class 1 by using a logistic sigmoid:

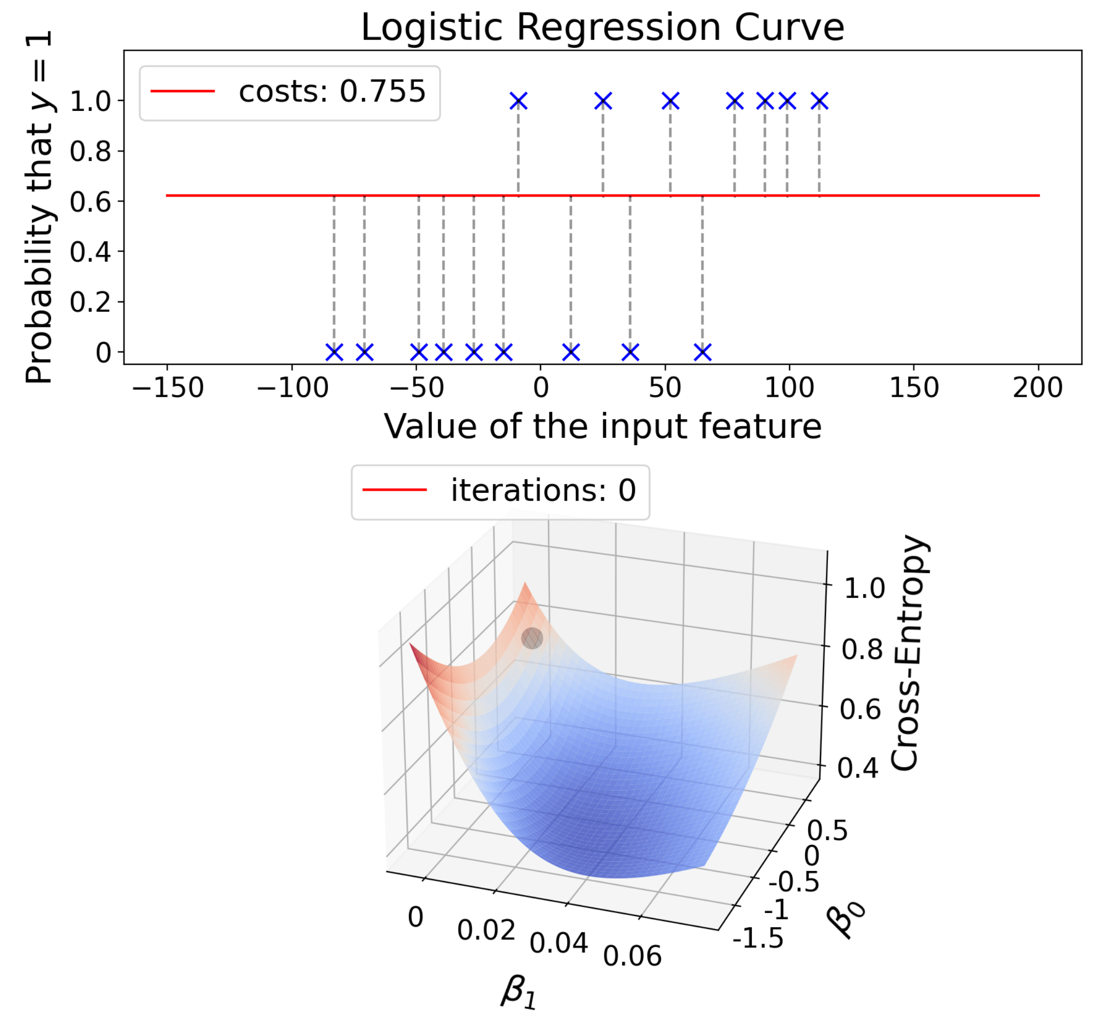
$$p_i \triangleq \mathrm{P}(y_i = 1 | \text{weight} = x_i) = \frac{1}{1 + e^{-\beta_0 - x_i \cdot \beta}}$$

The machine is updating the weights β by using the gradient of the following objective function:

$$\text{Loss}(\beta_0, \beta) \triangleq -\frac{1}{n} \sum_{i=1}^{n} [y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)]$$

Critical Thinking: What else is needed for minimizing the Loss function? How is the algorithm going to work?

THE MACHINE LEARNING OF LOGISTIC REGRESSION



METRICS FOR BINARY CLASSIFICATION

Total population (pop.) = 2030	Test outcome **positive**	Test outcome **negative**	Accuracy (ACC) = (TP + TN) / pop. = (20 + 1820) / 2030 ≈ **90.64%**	F_1 score $= 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$ ≈ **0.174**
Actual condition positive	**True positive** (TP) = 20 (2030 × 1.48% × 67%)	**False negative** (FN) = 10 (2030 × 1.48% × (100% – 67%))	True positive rate (TPR), recall, sensitivity = TP / (TP + FN) = 20 / (20 + 10) ≈ **66.7%**	False negative rate (FNR), miss rate = FN / (TP + FN) = 10 / (20 + 10) ≈ **33.3%**
Actual condition negative	**False positive** (FP) = 180 (2030 × (100% – 1.48%) × (100% – 91%))	**True negative** (TN) = 1820 (2030 × (100% – 1.48%) × 91%)	False positive rate (FPR), fall-out, probability of false alarm = FP / (FP + TN) = 180 / (180 + 1820) = **9.0%**	Specificity, selectivity, true negative rate (TNR) = TN / (FP + TN) = 1820 / (180 + 1820) = **91%**
Prevalence = (TP + FN) / pop. = (20 + 10) / 2030 ≈ **1.48%**	Positive predictive value (PPV), precision = TP / (TP + FP) = 20 / (20 + 180) = **10%**	False omission rate (FOR) = FN / (FN + TN) = 10 / (10 + 1820) ≈ **0.55%**	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$ = (20 / 30) / (180 / 2000) ≈ **7.41**	Negative likelihood ratio (LR–) $= \frac{\text{FNR}}{\text{TNR}}$ = (10 / 30) / (1820 / 2000) ≈ **0.366**
	False discovery rate (FDR) = FP / (TP + FP) = 180 / (20 + 180) = **90.0%**	Negative predictive value (NPV) = TN / (FN + TN) = 1820 / (10 + 1820) ≈ **99.45%**	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR–}}$ ≈ **20.2**	

Source: WiKipedia.

LOGISTIC REGRESSION WITH MULTIPLE CLASSES

When dealing with multiple classes, logistic regression extends to what is known as **multinomial logistic regression** or **softmax regression**.

Multinomial Logistic Regression:

Suppose we have an input $\mathbf{x}$ and weights $\mathbf{W}$, the probability of class k is given by:

$$P(y = k \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \mathbf{x})}$$

- **Goal**: Classify observations into one of K possible classes.
- **Approach**: Use the softmax function to predict the probabilities of each class.
- **Softmax Function**: The softmax function is an extension of the logistic function. It converts raw scores (logits) from the linear model into probabilities that sum to one.

MULTICLASS CROSSENTROPY

To train a multinomial logistic regression model, we use the **cross-entropy loss function**. Cross-entropy is a measure of the difference between two probability distributions for a given random variable or set of events.

1. **Cross-Entropy Loss**:
   - In the context of multi-class classification, the cross-entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1.
   - The loss increases as the predicted probability diverges from the actual label.
2. **Formula**:
   - Suppose we have N samples and K classes. For each sample i, let $\mathbf{y}_i$ be the one-hot encoded true label, and $\hat{\mathbf{y}}_i$ be the predicted probability distribution from the softmax function.
   - The cross-entropy loss for a single sample i is:

$$L_i = -\sum_{k=1}^{K} y_{i,k} \log(\hat{y}_{i,k})$$

MULTICLASS CROSSENTROPY

$$P(y = k \mid \mathbf{x_i}) = \frac{\exp(\mathbf{w}_k^\top \mathbf{x_i})}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \mathbf{x_i})}$$

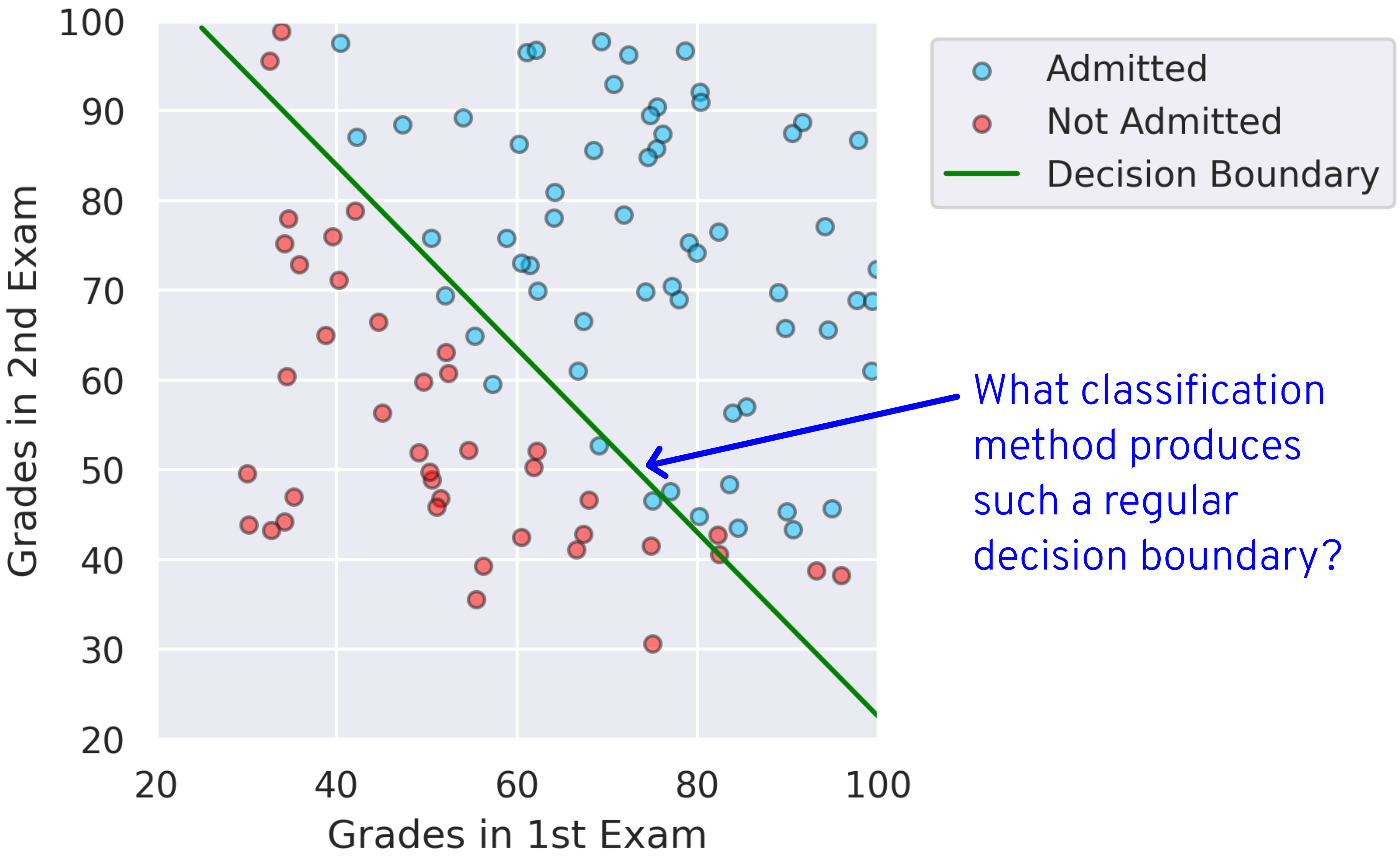
1. **Loss function:**
   - $$L_i = -\sum_{k=1}^{K} y_{i,k} \log(\hat{y}_{i,k})$$

     where $y_{i,k}$ is 1 if sample i belongs to class k, and 0 otherwise.
   - The total cross-entropy loss over all samples is the average of the individual losses:

     $$L = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_{i,k} \log(\hat{y}_{i,k})$$

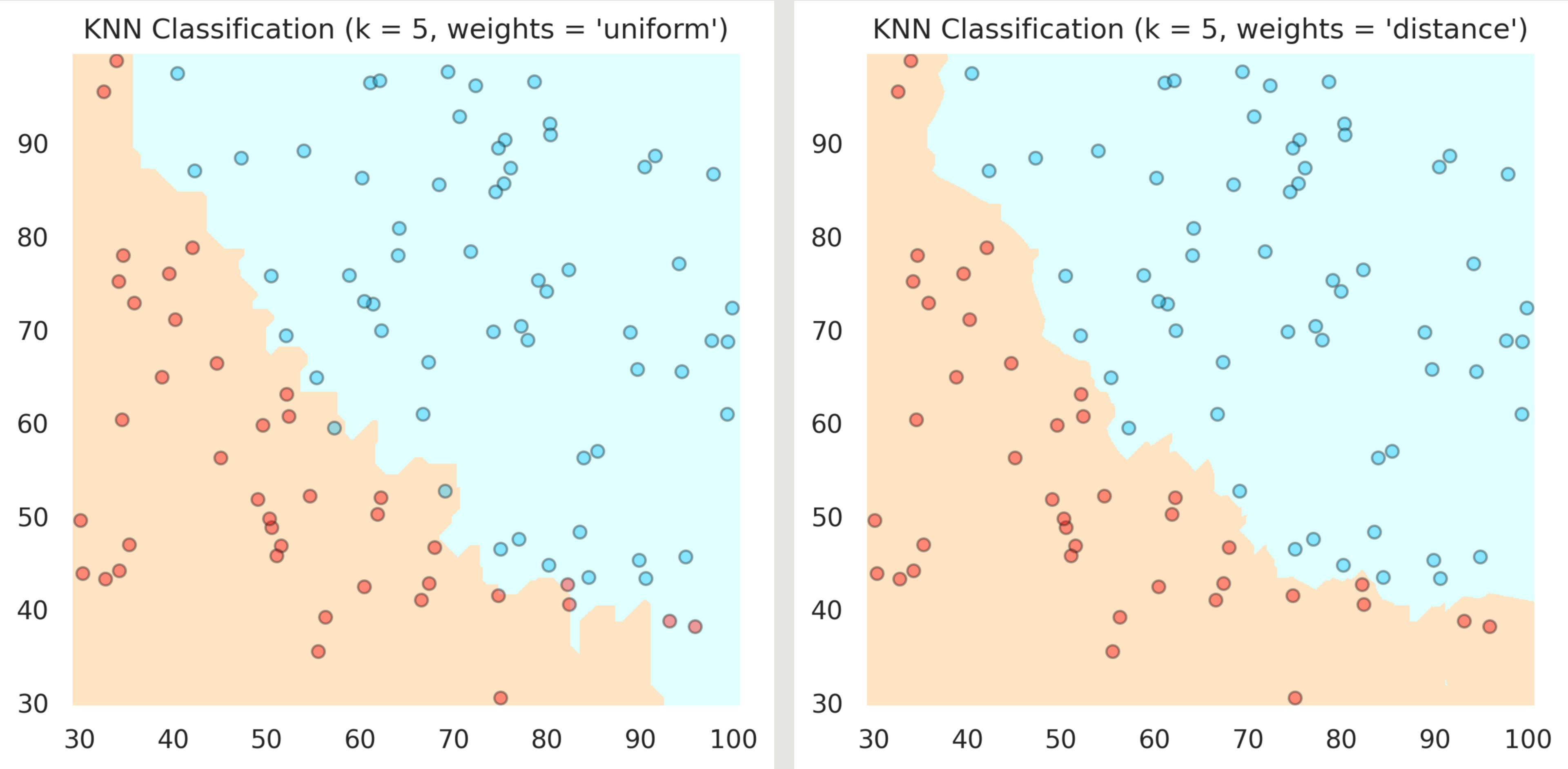
2. **Interpretation**:
   - The cross-entropy loss function effectively penalizes the model more when the predicted probability of the true class is low.
   - Minimizing this loss encourages the model to produce high probabilities for the correct classes.

DECISION BOUNDARIES



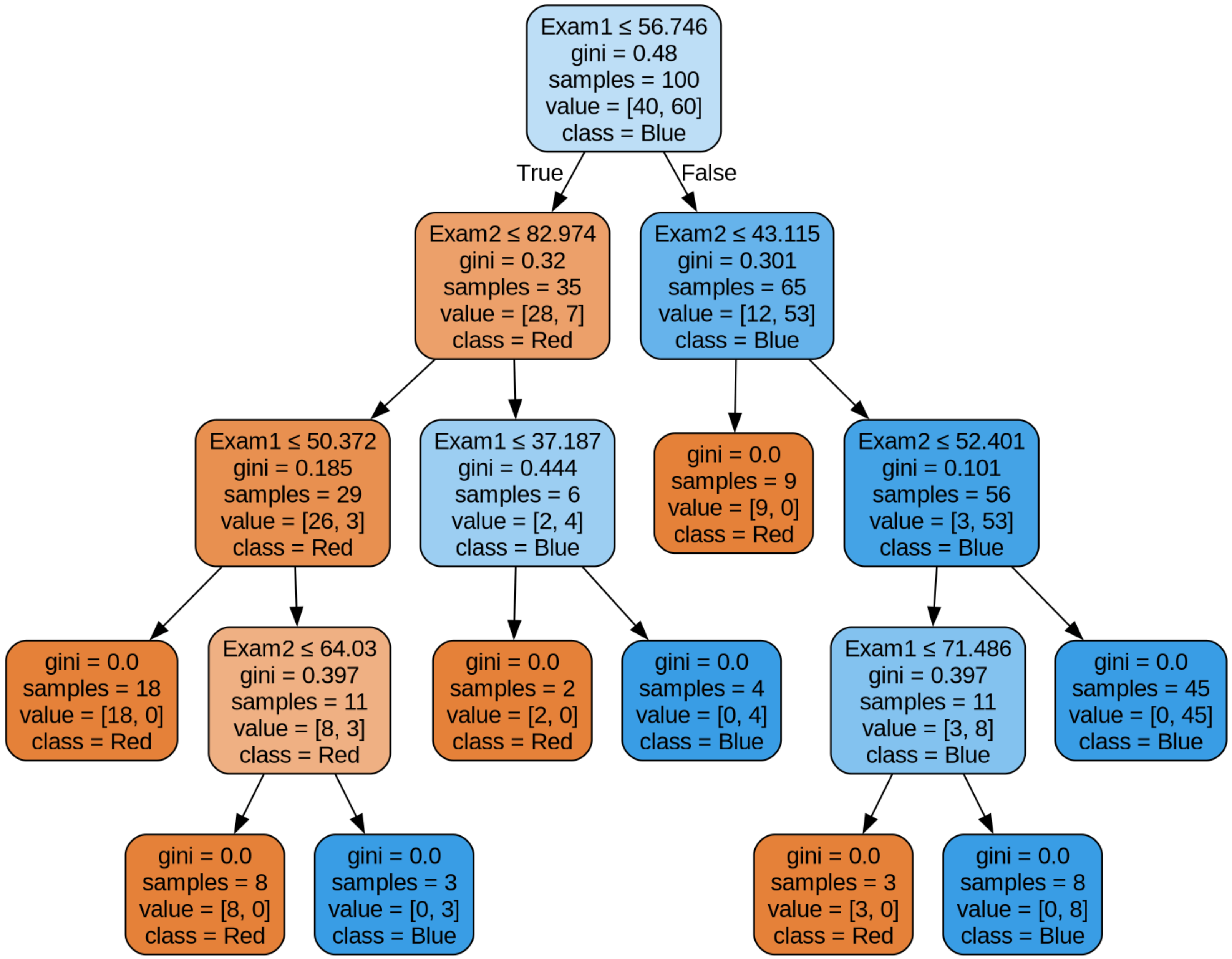
What classification method produces such a regular decision boundary?

DECISION BOUNDARIES -KNN

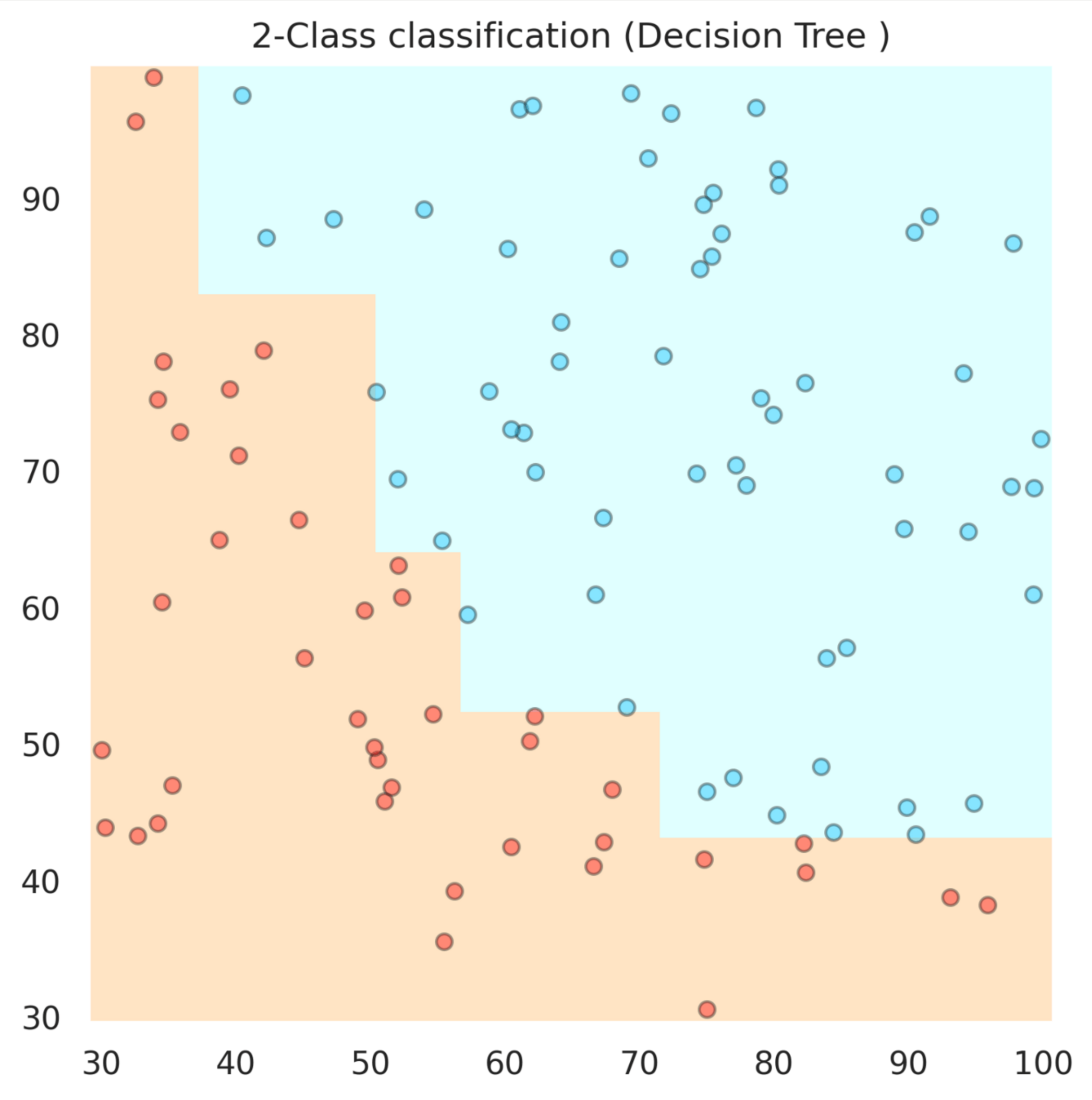


For this example $k = 5$.

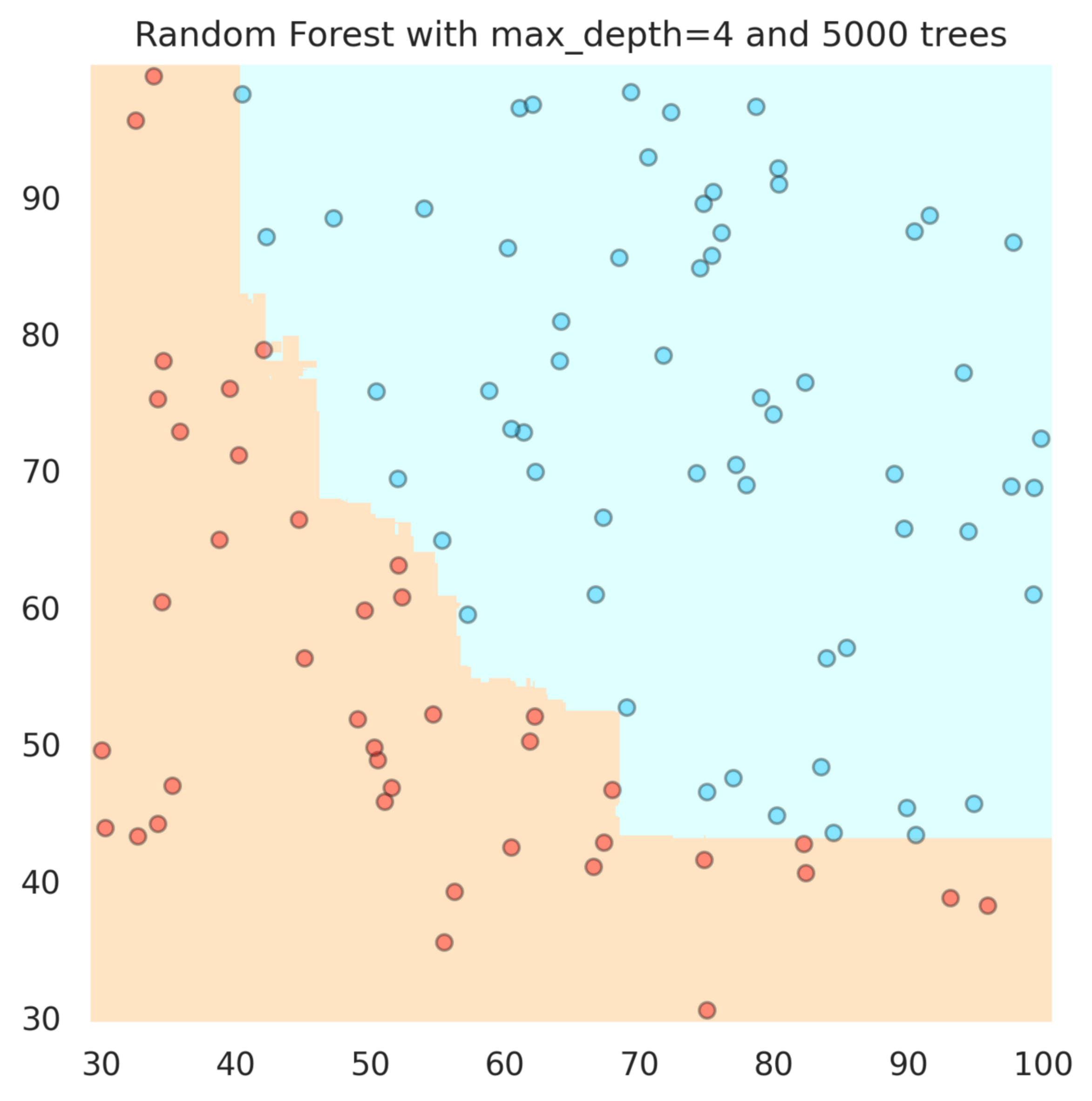
HOW ABOUT DECISION TREES?



HOW ABOUT DECISION TREES?



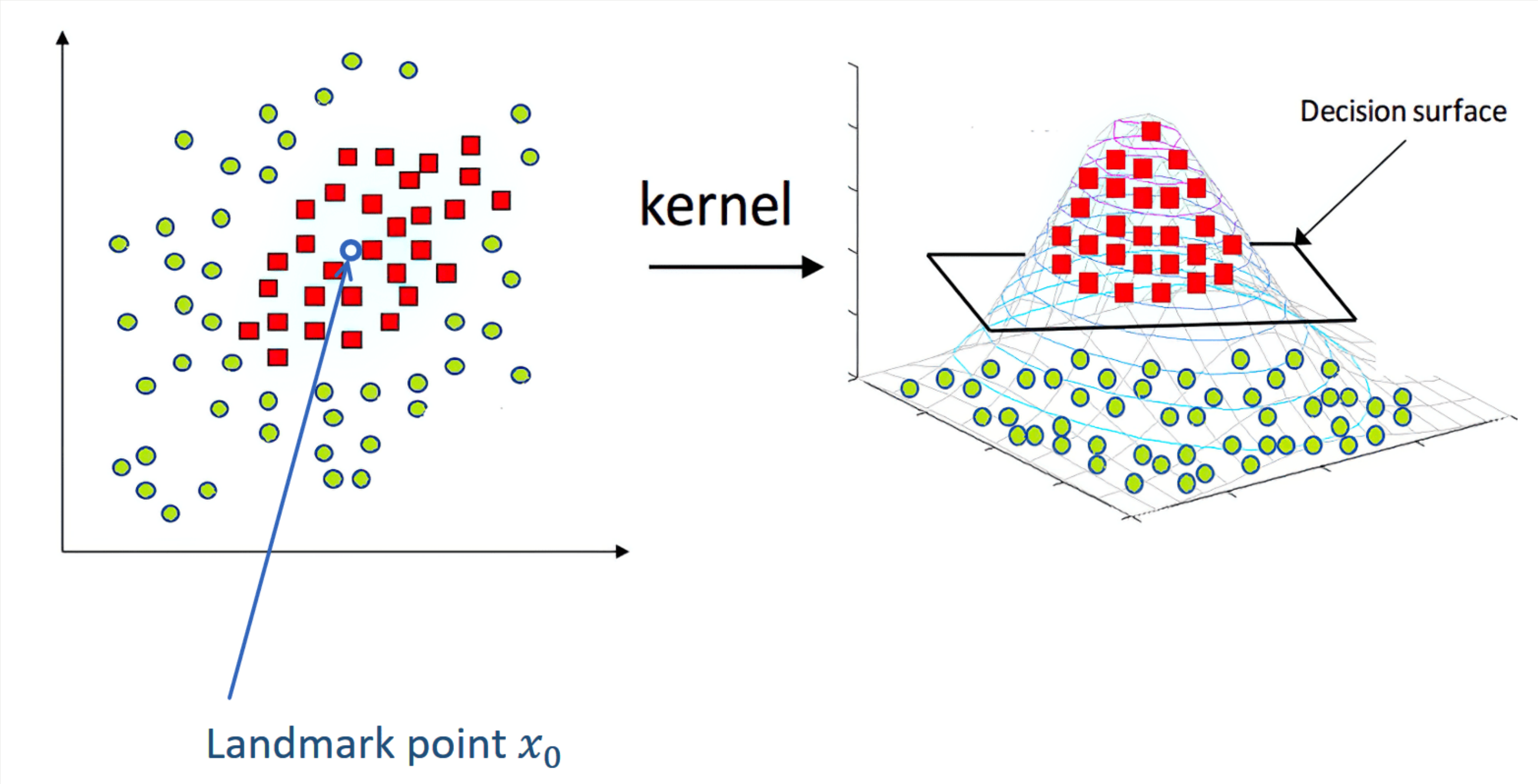
AND RANDOM FORESTS?



HOW ABOUT ENGINEERING A DECISION BOUNDARY?

Why do we care about such an idea?

Example:



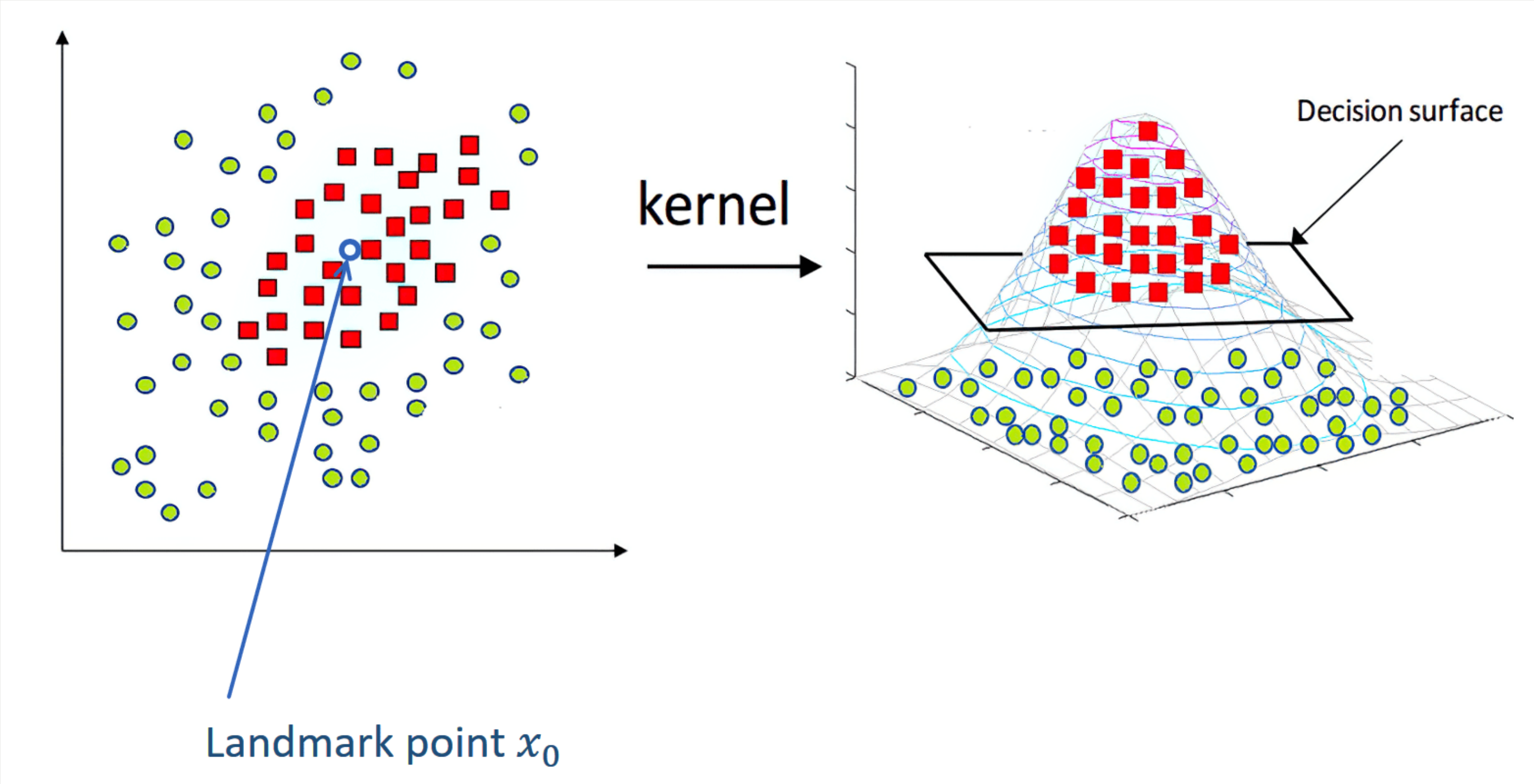
SUPPORT VECTOR MACHINES

Main Idea: For classification, focus on designing the shape of the boundary between classes. This could sometimes be informed by visualizations of the data in lower dimensions.

Imagine a simple 2D case where we have two classes of points that are linearly separable:

- SVM looks for a **line (hyperplane in higher dimensions)** that best separates these two classes.
- The **best** hyperplane is the one that **maximizes the margin**—the distance between the hyperplane and the **nearest data points** from each class.
- These nearest points are called **support vectors**, and they are critical in defining the position of the hyperplane.

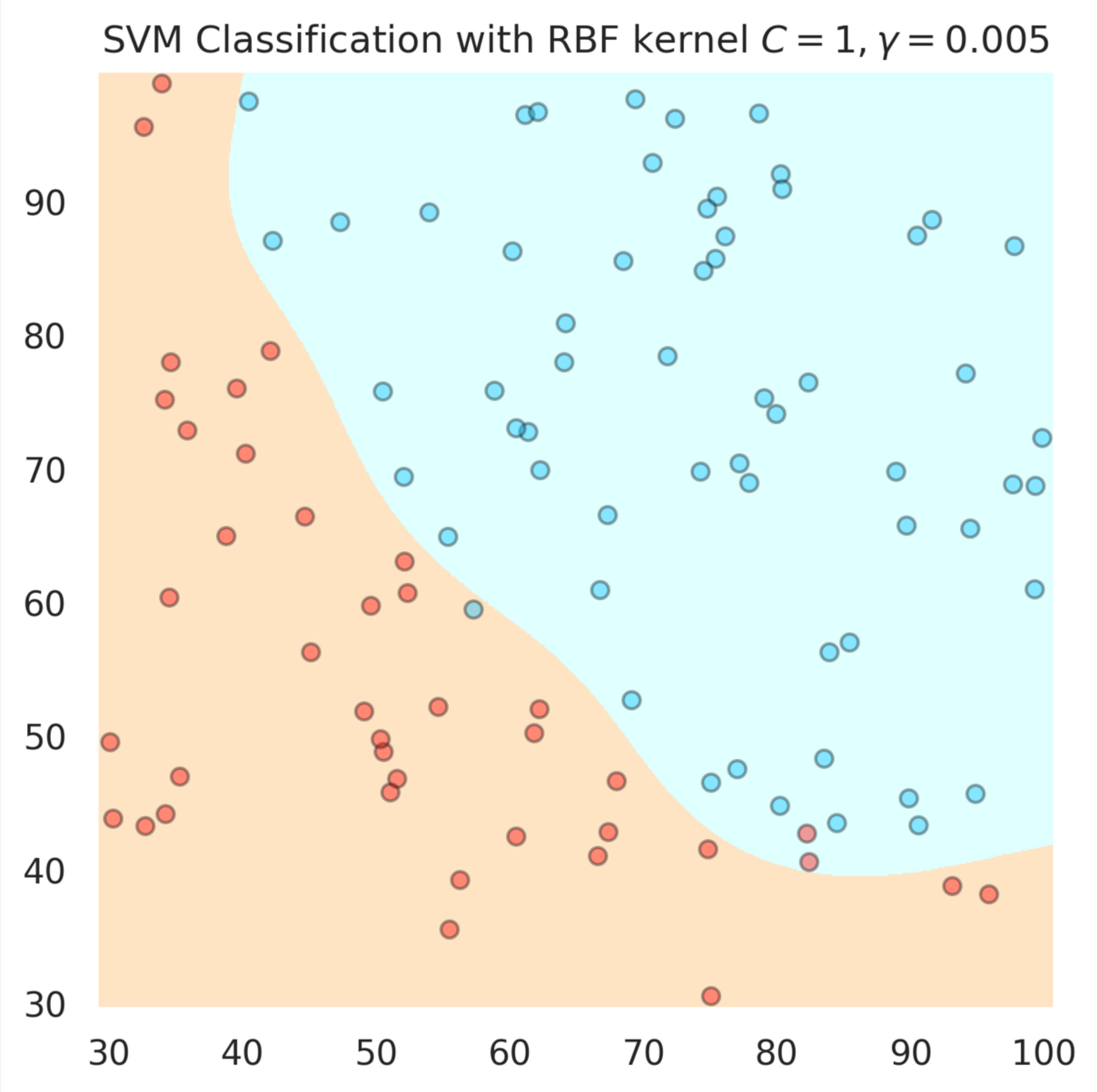
SUPPORT VECTOR MACHINES



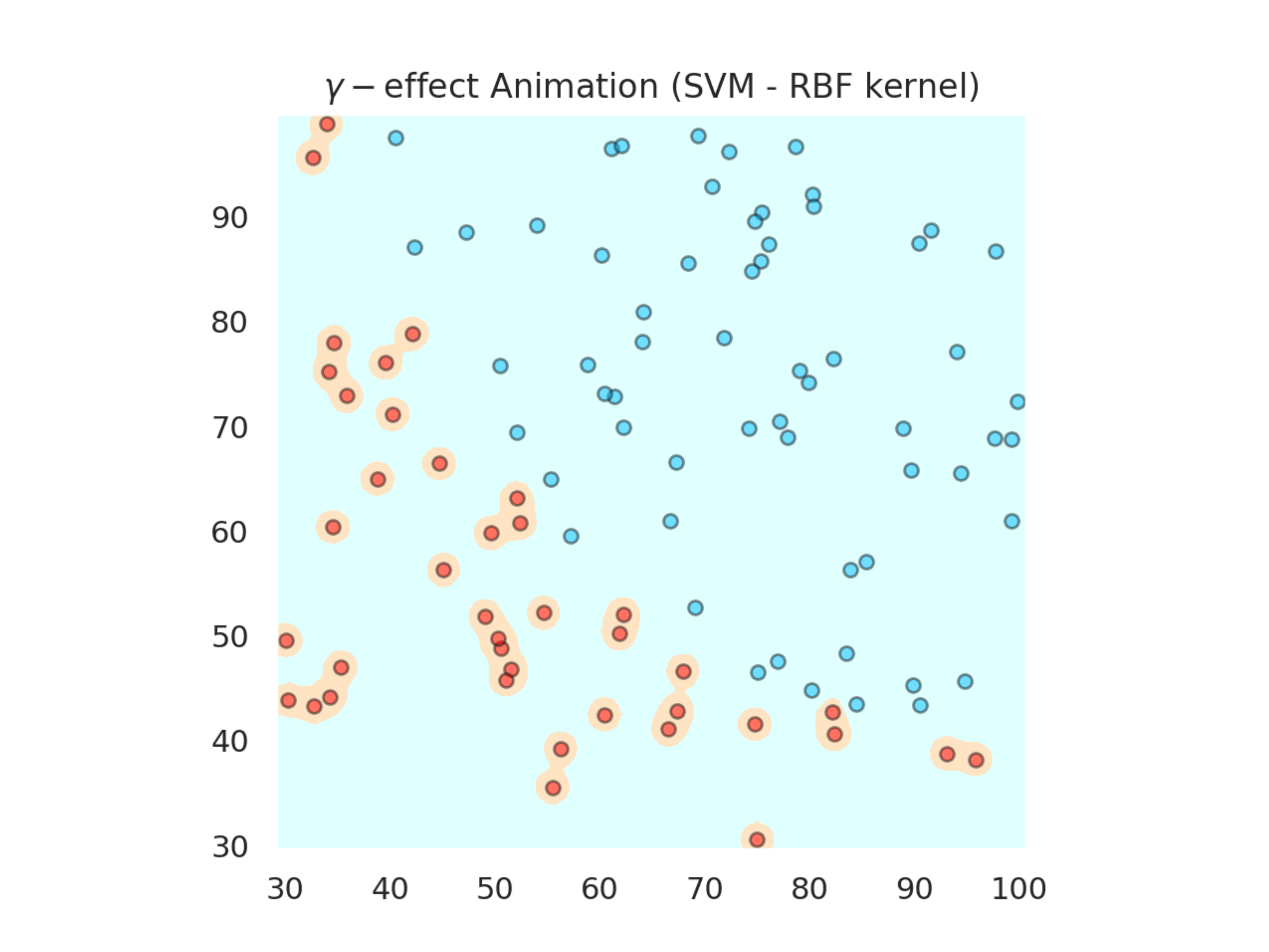
For this we would need at least one landmark point x_0. The following is also called a "Gaussian" kernel

$$(x, y) \to \left(x, y, z := e^{-\gamma[(x-x_0)^2+(y-y_0)^2]}\right)$$

SUPPORT VECTOR MACHINES



SUPPORT VECTOR MACHINES



LINEAR SVM

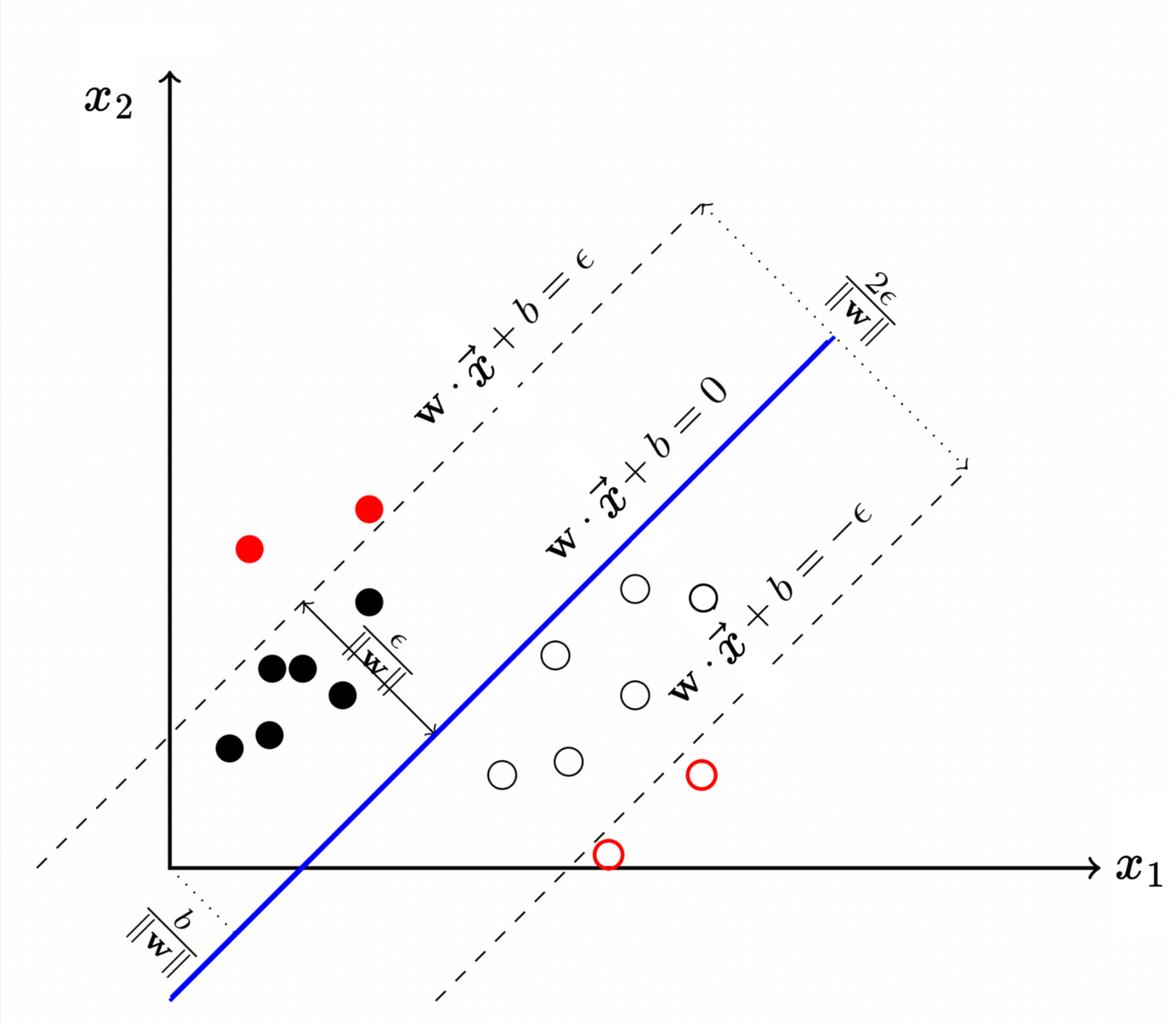
For linearly separable data, SVM tries to solve the optimization problem:

$$\min_{w,b} \frac{1}{2}||w||^2 \quad \text{subject to} \quad y_i(w \cdot x_i + b) \geq 1$$

where:

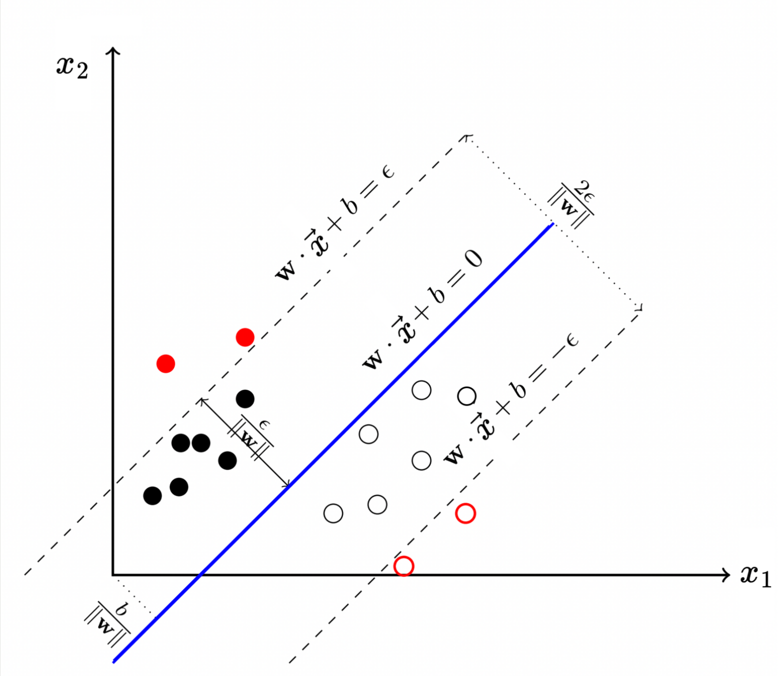
- w is the weight vector,
- b is the bias,
- x_i are the data points,
- $y_i \in \{-1, +1\}$ are the class labels.

LINEAR SVM



x_2
x_1
$\mathbf{w} \cdot \vec{x} + b = \epsilon$
$\mathbf{w} \cdot \vec{x} + b = 0$
$\mathbf{w} \cdot \vec{x} + b = -\epsilon$
$\frac{2\epsilon}{\|\mathbf{w}\|}$
$\frac{\epsilon}{\|\mathbf{w}\|}$
$\frac{b}{\|\mathbf{w}\|}$

LINEAR SVM



Support Vector Machines Seminal Paper (1992): http://www.svms.org/training/BOGV92.pdf

Support Vectors is a method used in Machine Learning for both regression and classification problmes. The main idea is to map the input features into a higher dimensional space and then, in that higher dimensional space, address the problem to solve.

For regression, SVM consists of an algorithm that solves a quadratic optimization problem with constraints:

$$\text{minimize}\frac{1}{2}|w|^2$$

subject to

$$y_i - wx_i - b \leq \epsilon,\ \ wx_i + b - y_i \leq \epsilon$$