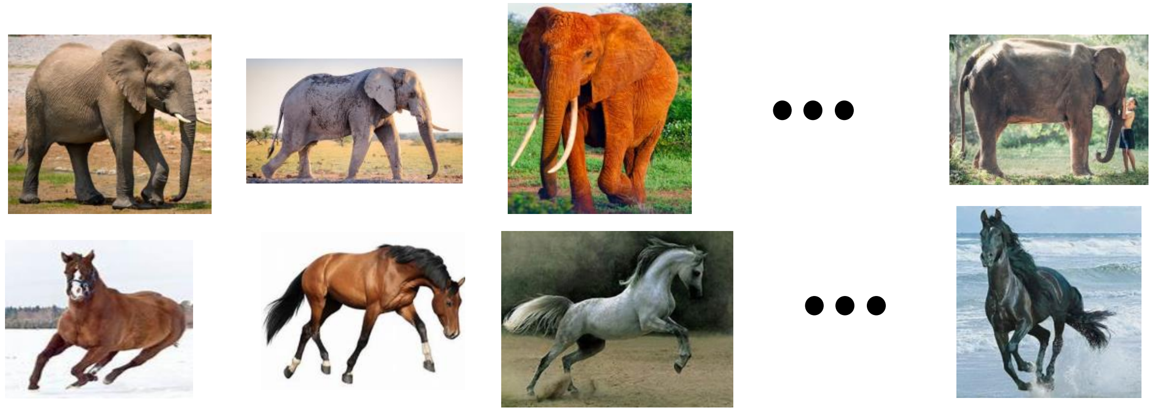
Deep Learning (CS 470, CS 570)

Module 2, Lecture 1: Perceptron Learning Algorithm

Toy Example: Animal Classification

We want to write an machine learning algorithm to classify each image into two animals classes; elephant and horse. This is a toy example for ease of understanding but we will soon see some real-world image classification examples.

Dataset: Contains image samples from each classes we want to detect. A part of these images are used for training and rest for testing.

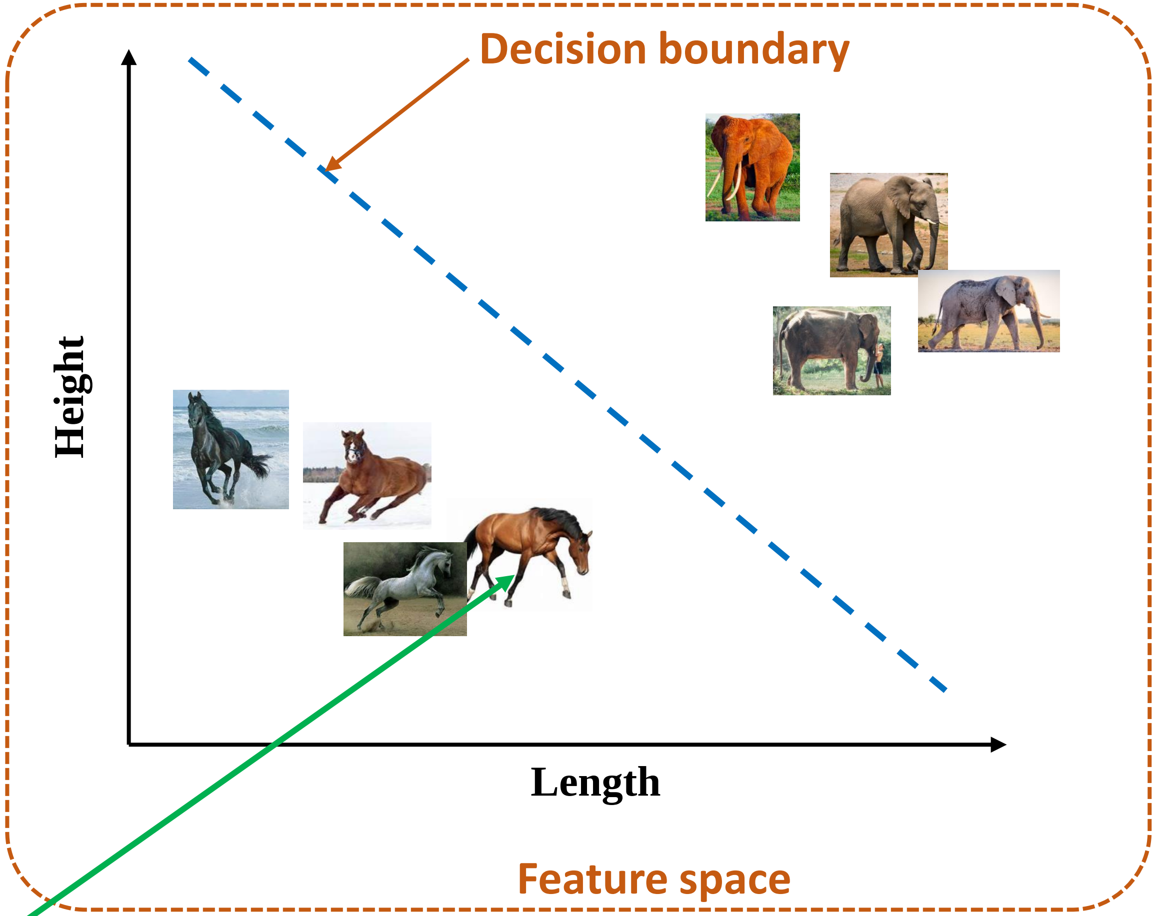


For any image we can use some image processing techniques to extract image features that encode crucial information useful for classification. We will later learn how to extract image features, but for this toy example we will assume that we know how to compute two features such as length and height of the animal present in the image.

Any image

[8ft 5ft]

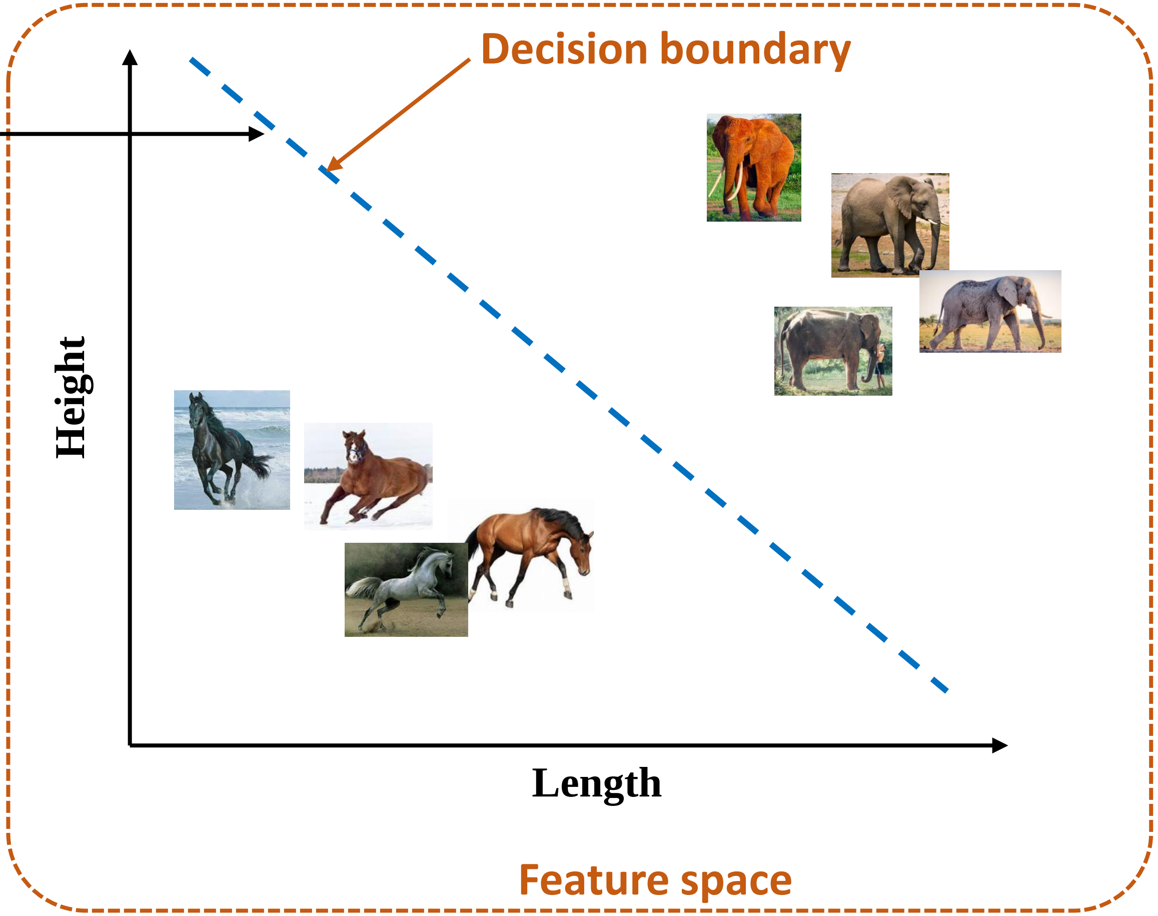
Represented by the length and height of the animal



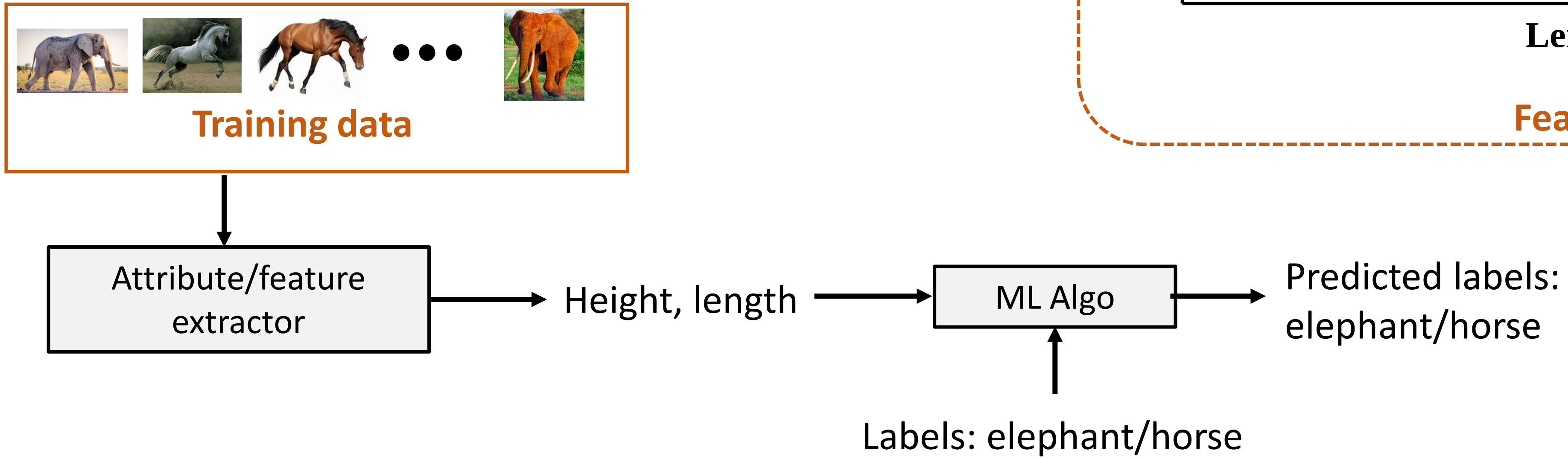
Becomes a point in a coordinate space where each axis of the coordinate space represents a feature. We refer it as data point or sample point. If we extract a ***d*** dimensional feature vector, then the dimension of the coordinate space is also ***d*** .

Toy Example: Animal Classification

Given enough training image samples, a machine learning algorithm learns to draw a boundary in the coordinate space (known as feature space) such that all horse images are in one side of the boundary (decision boundary) and elephant images are on the other side.



Once the decision boundary is learnt, the algorithm can predict the class of a new unknown image depending on which side of the decision boundary it belongs to. A typical ML algorithm framework is give below.



Taxonomy

Supervised learning:

These are the notations we will use throughout the course to explain machine learning algorithms.

Input: $\boldsymbol{x} = (x_1, x_2, \dots, x_d)$ Vector of dimension d

Output: y Scalar

$$\text{Continuous -> regression} \xleftarrow{} y \xrightarrow{} \text{Categorical -> classification}$$

Target function: $f : \boldsymbol{x} \to y$ Unknown

Data: $(\boldsymbol{x}^1, y^1), (\boldsymbol{x}^2, y^2), \dots, (\boldsymbol{x}^N, \boldsymbol{y}^N)$ N is the number of data points

Hypothesis Set: $H = \{h\}$ H is a set that defines what are the possible decision boundaries a machine learning algorithm can generate. h is a member of H.

Learned Hypothesis: $g : \boldsymbol{x} \to y$, $g \in H$ g is an approximation of f

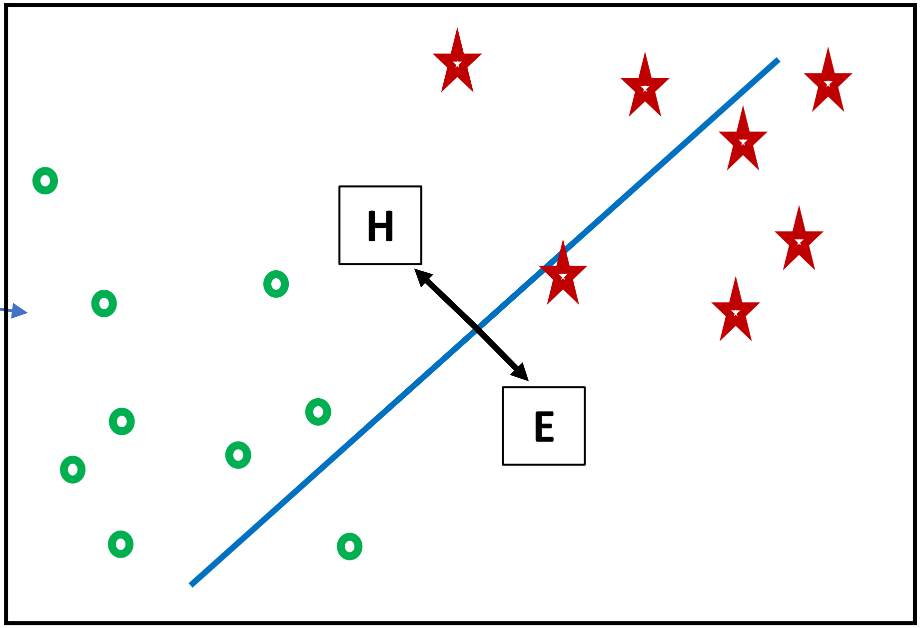
Classification Framework

Training data

Attribute/feature extractor

Data: $(\boldsymbol{x}^1, y^1), (\boldsymbol{x}^2, y^2), \ldots, (\boldsymbol{x}^N, \boldsymbol{y}^N)$

Plot data



Feature space

Horse

Elephant

Hypothesis:

Select a random line

Right side of the line is Elephant

Left side of the line is Horse

Model training/learning/optimization:

Change the line orientation

The above steps describes how a typical ML algorithm learns to classify two different group (here horse and elephant). A hypothesis h is a set of rules that defines the two classes in the feature space. The ML algorithm starts with a random hypothesis that consist of a decision boundary (here a line in arbitrary direction), and a few rules. During training, the algorithm learns from data examples how to draw a correct decision boundary (i.e. orientation and position of the line that divides the two groups of data points).

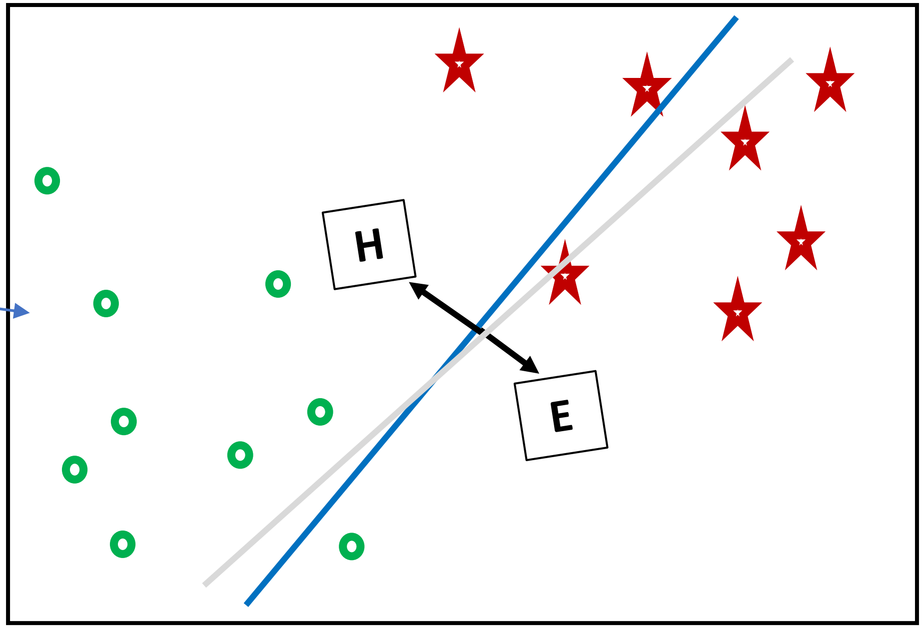
Classification Framework

Training data

Attribute/feature extractor

Data: $(\boldsymbol{x}^1, y^1), (\boldsymbol{x}^2, y^2), \ldots, (\boldsymbol{x}^N, \boldsymbol{y}^N)$

Plot data



Feature space

Horse

Elephant

Hypothesis:

Select a random line

Right side of the line is Elephant

Left side of the line is Horse

Model training/optimization:

Change the line orientation

To learn how to draw the correct decision boundary, the ML algorithm runs a iterative process that slightly changes the line orientation (and position but the example doesn't show that) in every step following certain rules.

Classification Framework

Training data

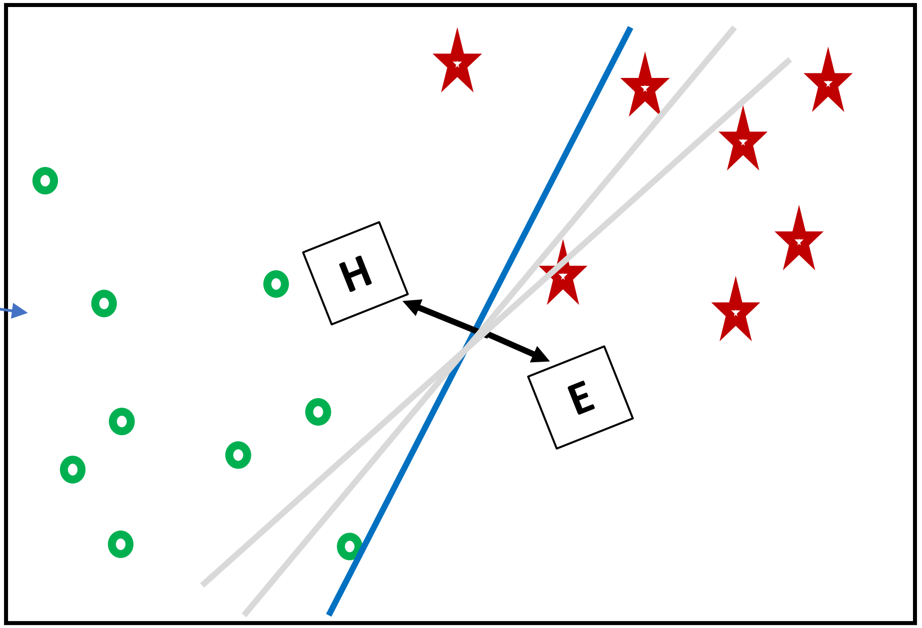
↓

Attribute/feature extractor

↓

Data: $(\boldsymbol{x}^1, y^1), (\boldsymbol{x}^2, y^2), \ldots, (\boldsymbol{x}^N, \boldsymbol{y}^N)$

Plot data



Feature space

Horse

Elephant

Hypothesis:

Select a random line

Right side of the line is Elephant

Left side of the line is Horse

Model training/optimization:

Change the line orientation

To learn how to draw the correct decision boundary, the ML algorithm runs a iterative process that slightly changes the line orientation (and position but the example doesn't show that) in every step following certain rules.

Classification Framework

Training data

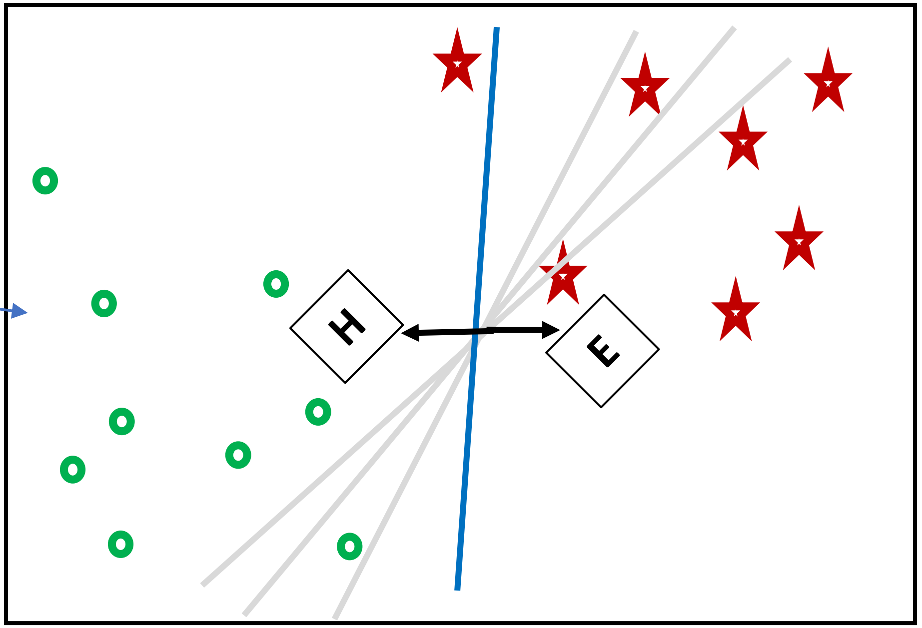
↓

Attribute/feature extractor

↓

Data: $(\boldsymbol{x}^1, y^1), (\boldsymbol{x}^2, y^2), \ldots, (\boldsymbol{x}^N, \boldsymbol{y}^N)$

Plot data



Feature space

Horse

Elephant

Hypothesis:

Select a random line

Right side of the line is Elephant

Left side of the line is Horse

Model training/optimization:

Change the line orientation

To learn how to draw the correct decision boundary, the ML algorithm runs a iterative process that slightly changes the line orientation (and position but the example doesn't show that) in every step following certain rules.

Classification Framework

Training data

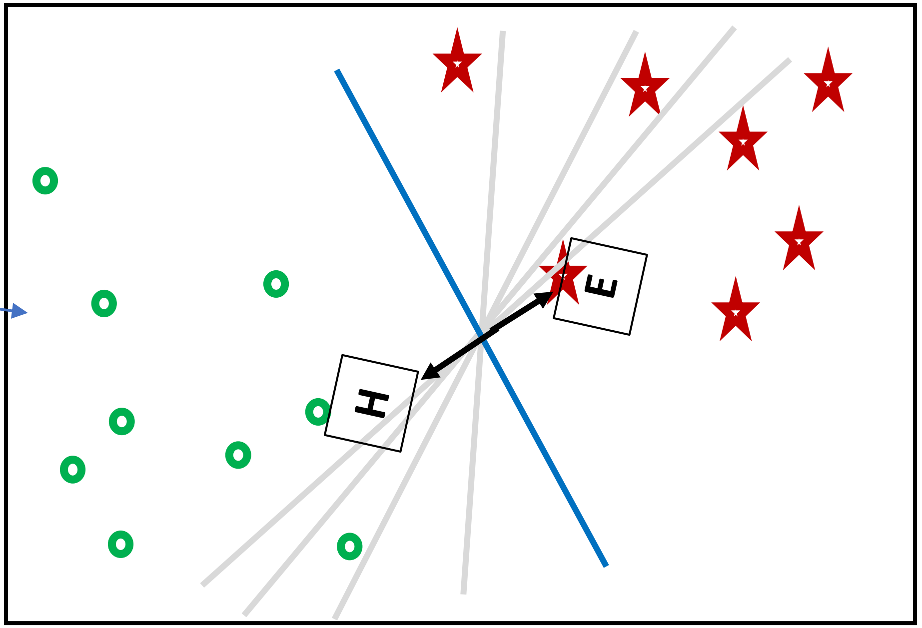
↓

Attribute/feature extractor

↓

Data: $(\boldsymbol{x}^1, y^1), (\boldsymbol{x}^2, y^2), \dots, (\boldsymbol{x}^N, \boldsymbol{y}^N)$

Plot data



Feature space

Horse

Elephant

Hypothesis:

Select a random line

Right side of the line is Elephant

Left side of the line is Horse

Model training/optimization:

Change the line orientation

Select the line orientation that best classifies the training data

Hypothesis set:

Hypotheses associated with all line orientations

Finally, the algorithm finds the orientation (and position) that best classifies the two groups of training examples. This is when training stops.

Perceptron Learning Algorithm: Hypothesis Set

Input: $\mathbf{x} = (x_1, x_2, \dots, x_d)$
$x_i, i \in \{1, \dots, d\}$ is a feature such as length and height

Consider the example of elephant vs. horse classification.

Hypothesis:

Image of an elephant if $\sum_{i=1}^{d} w_i x_i >$ threshold   Right side of line

Image of a horse if $\sum_{i=1}^{d} w_i x_i <$ threshold   Left side of line

OR,

$$h(x) = sign\left(\sum_{i=1}^{d} w_i x_i - threshold\right)$$

'+'ve : right side, '-'ve left side

Hypothesis Set:

$H = \{\, h(x)$: for all possible values of $\boldsymbol{w}$ and threshold $\}$

Here, we will learn the PLA algorithm that follows the learning technique that we just seen in the previous example. We will use a little math to understand the algorithm in details.

Perceptron Learning Algorithm: Hypothesis Set

$$\sum_{i=1}^{d} w_i x_i \; - threshold$$

Is this equation familiar?

$$ax + by + c = 0$$

Equation of a line

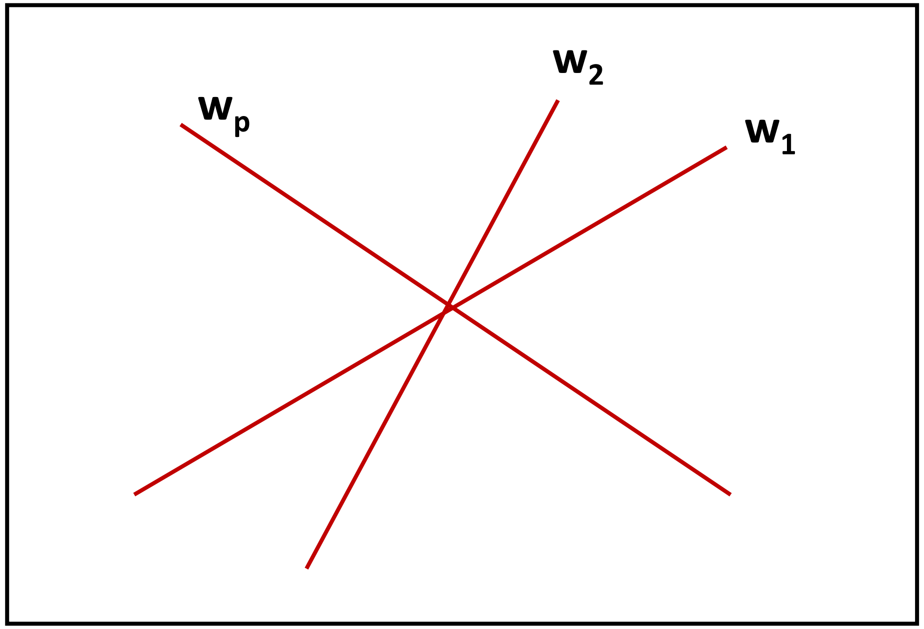
$$ax + by + cz + d = 0$$

Equation of a plane

$$\mathrm{w}_1 x_1 + \mathrm{w}_2 x_2 + \cdots + \mathrm{w}_\mathrm{d} x_d + threshold = 0$$

Equation of a hyperplane in a d dimensional space

Hypothesis set: Set of all possible hyper-planes



Hypothesis:

$$h(x) = sign(\textstyle\sum_{i=0}^{d} w_i x_i)$$

, x_0 is always 1

$$h(x) = sign(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x})$$

Vector form

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_d \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_d \end{bmatrix}$$

Perceptron Learning Algorithm: Learning

Random initialization of $\boldsymbol{w}$

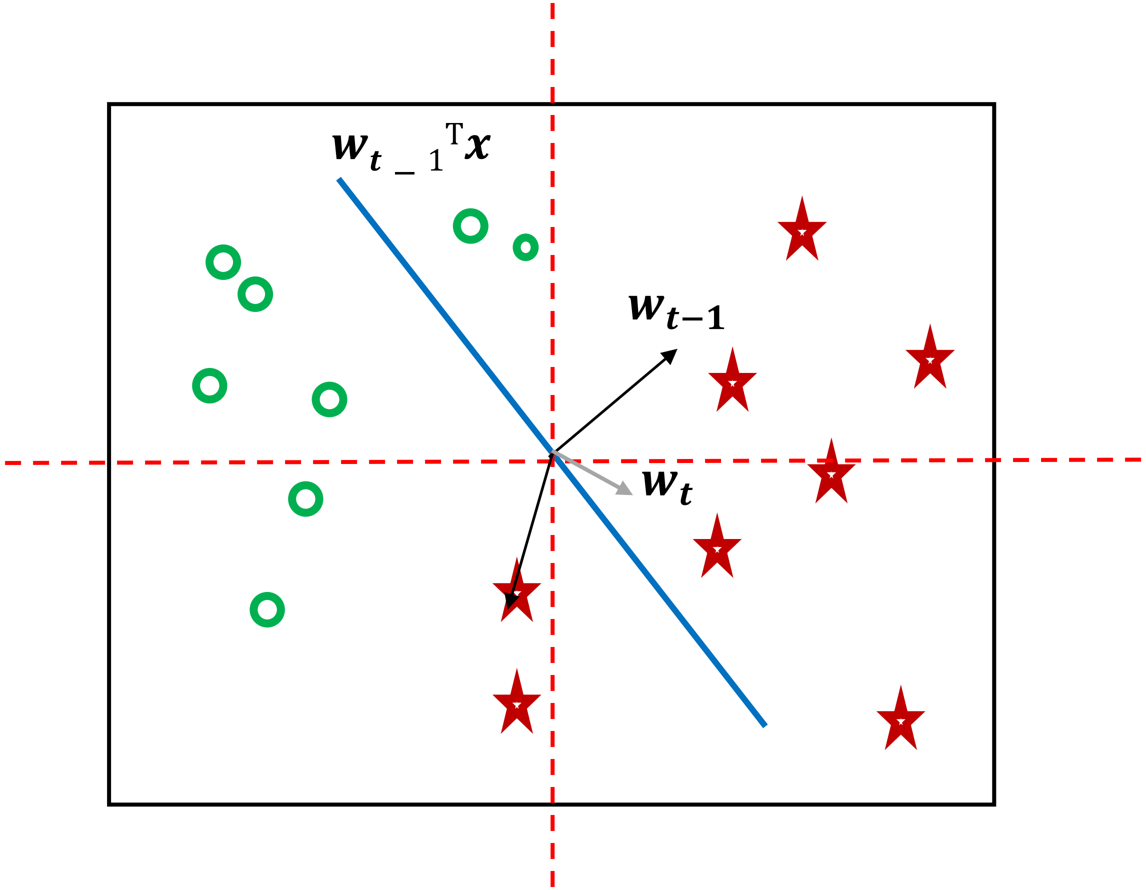
$\boldsymbol{w}$ is perpendicular to the hyperplane $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$ WHY?

At any step, t-1, of the algorithm, pick a misclassified point

$$h(x) = sign(\boldsymbol{w}_{t-1}{}^{\mathrm{T}}\boldsymbol{x}_n) \neq sign(y_n)$$

Update the weight vector as:

$\boldsymbol{w}_{\mathrm{t}} \leftarrow \boldsymbol{w}_{\mathrm{t}-1} + \mathrm{y_n}\boldsymbol{x}_{\mathrm{n}}$ Update rule



Perceptron Learning Algorithm: Learning

Random initialization of $\boldsymbol{w}$

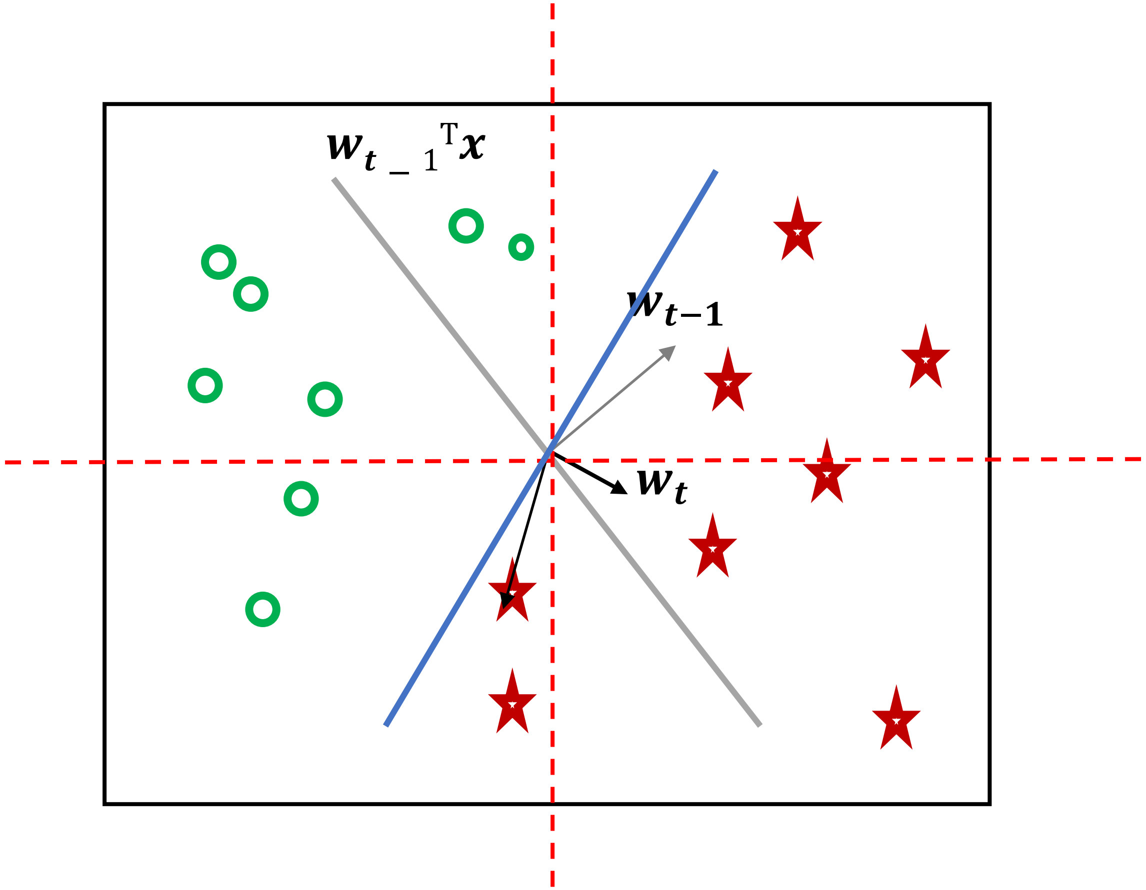
$\boldsymbol{w}$ is perpendicular to the hyperplane $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$ WHY?

At any step, t-1, of the algorithm, pick a misclassified point

$$h(x) = sign(\boldsymbol{w}_{t-1}{}^{\mathrm{T}}\boldsymbol{x}_n) \neq sign(y_n)$$

Update the weight vector as:

$$\boldsymbol{w}_{\mathrm{t}} \leftarrow \boldsymbol{w}_{\mathrm{t}-1} + \mathrm{y}_{\mathrm{n}}\boldsymbol{x}_{\mathrm{n}}$$ Update rule

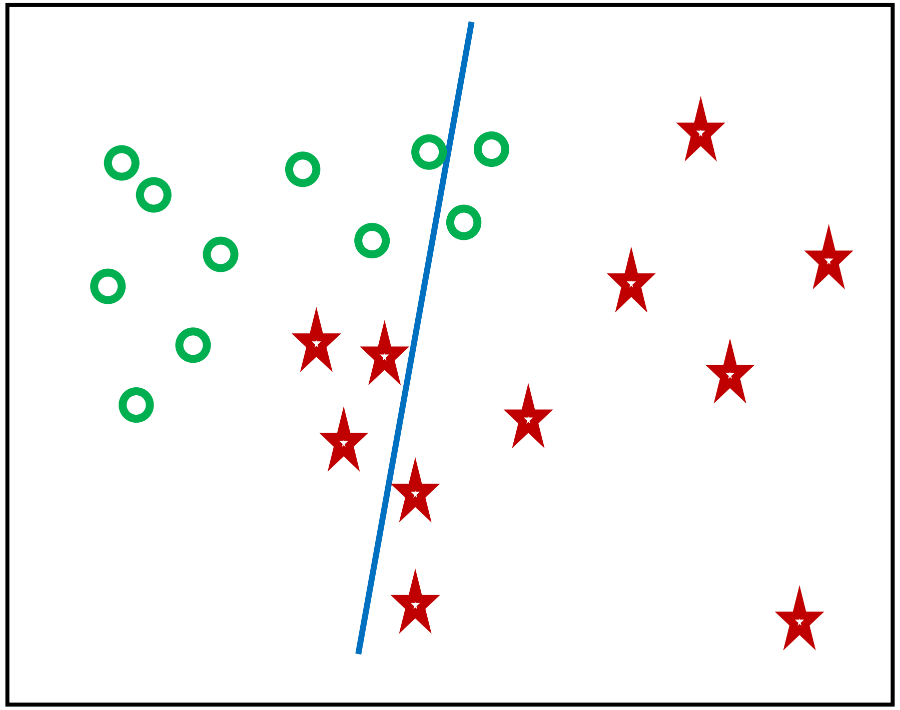


Continue picking points and updating $\boldsymbol{w}$ as long there are any misclassified points.

When data is linearly separable, algorithm will find a $\boldsymbol{w}_{\mathrm{final}}$ that will classify all training points correctly.

$$\mathrm{g}(x) = sign(\boldsymbol{w}_{\mathrm{final}}{}^{\mathrm{T}}\boldsymbol{x})$$

When Data is not Linearly Separable



Random initialization of $\boldsymbol{w}$

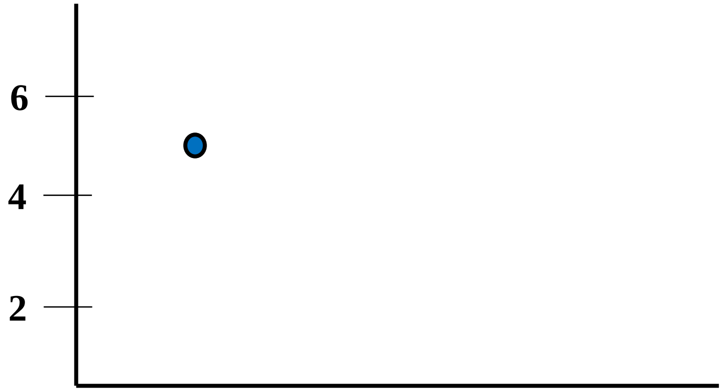
Estimate classification error. Save error and $\boldsymbol{w}$

For a fixed number of iterations

Update $\boldsymbol{w}$

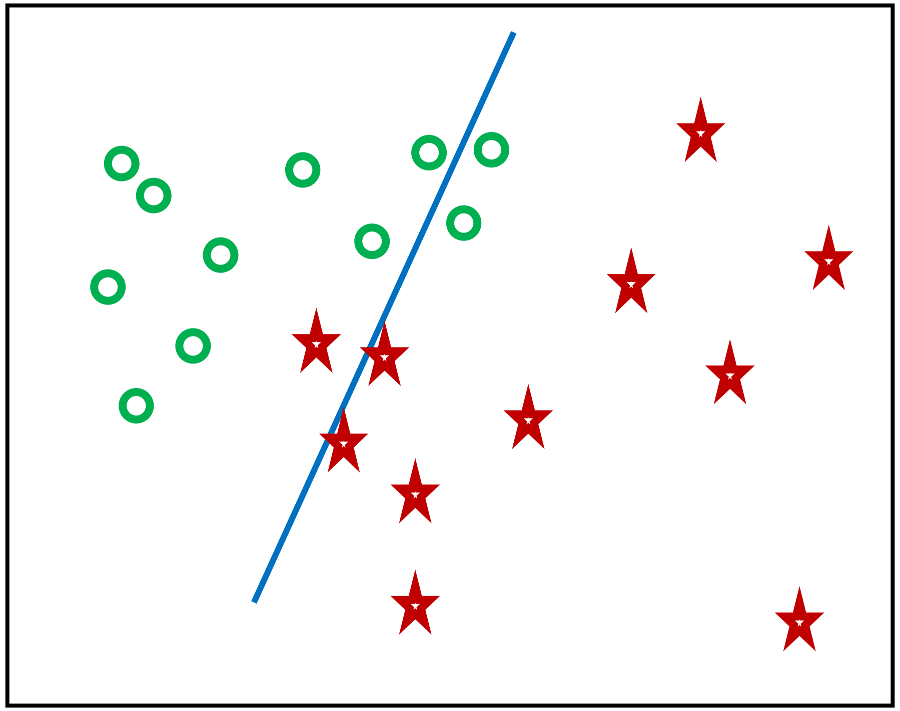
Re-estimate error

Save the error rate and $\boldsymbol{w}$



At any stage of the algorithm, the classification error is the percentage of misclassified sample points.

When Data is not Linearly Separable



Random initialization of $\boldsymbol{w}$

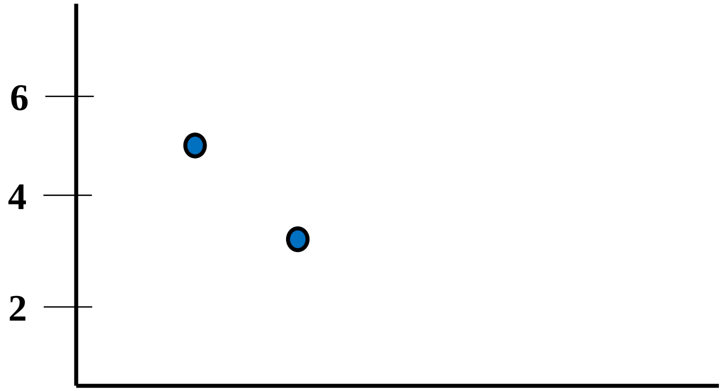
Estimate classification error. Save error and $\boldsymbol{w}$

For a fixed number of iterations

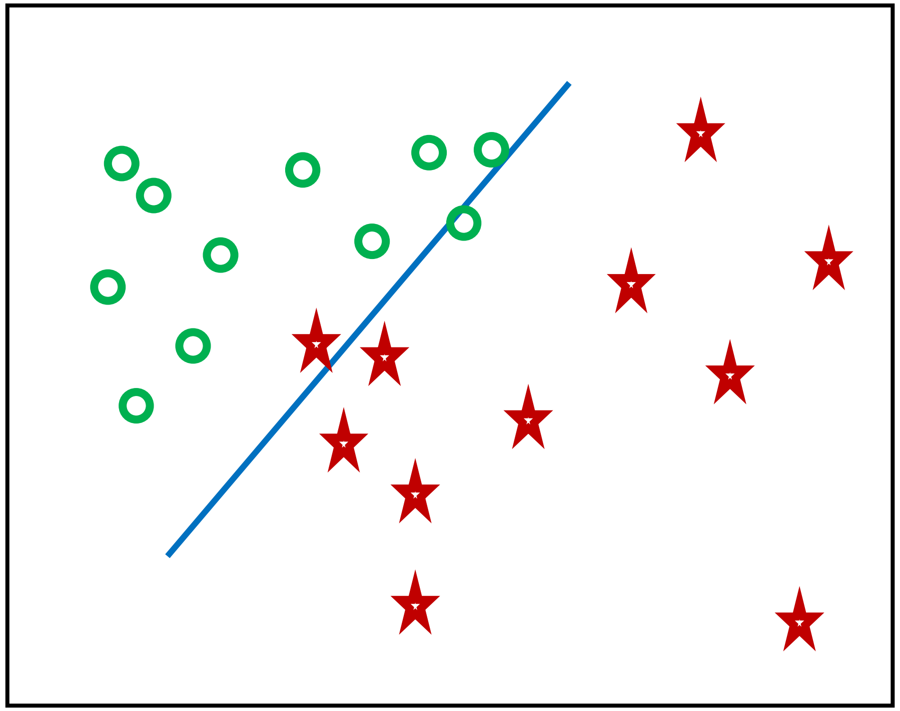
Update $\boldsymbol{w}$

Re-estimate error

Save the error rate and $\boldsymbol{w}$



When Data is not Linearly Separable

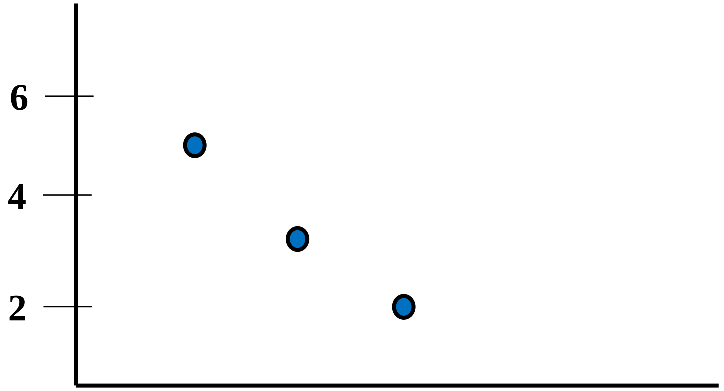


Random initialization of $\boldsymbol{w}$

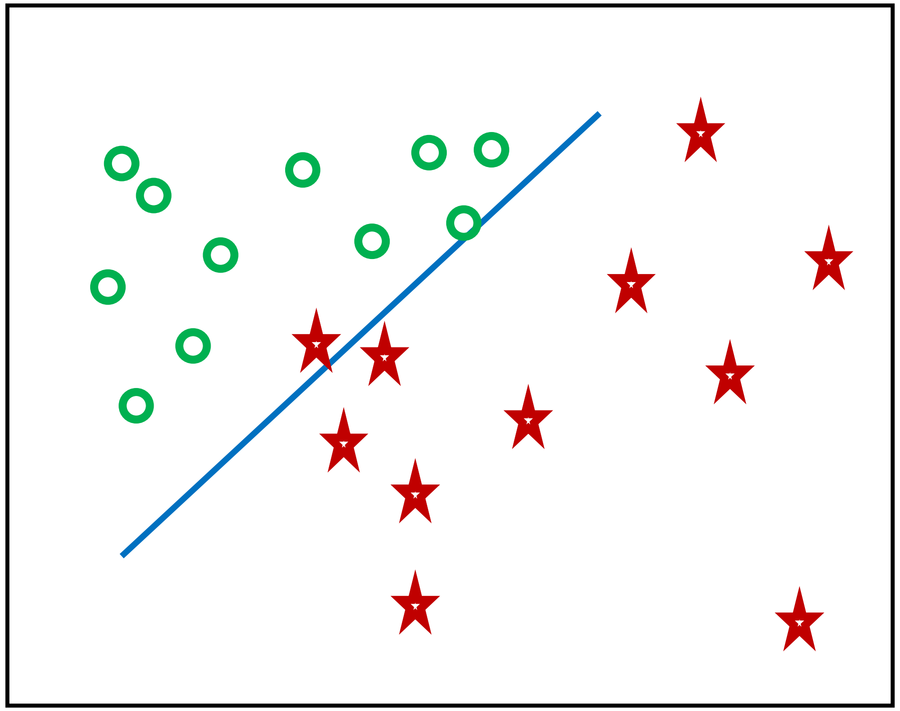
Estimate classification error. Save error and $\boldsymbol{w}$

For a fixed number of iterations

- Update $\boldsymbol{w}$
- Re-estimate error
- Save the error rate and $\boldsymbol{w}$



When Data is not Linearly Separable

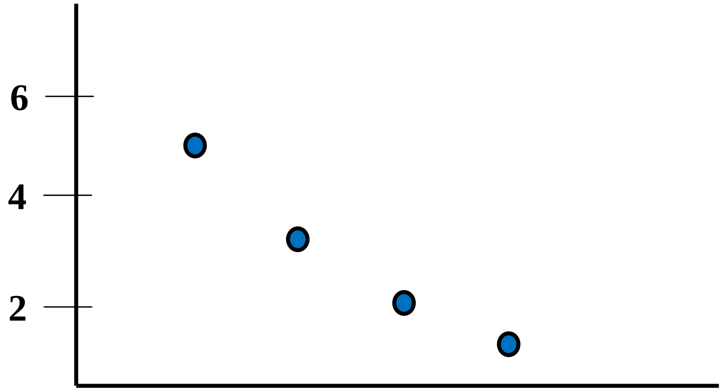


Random initialization of $\boldsymbol{w}$

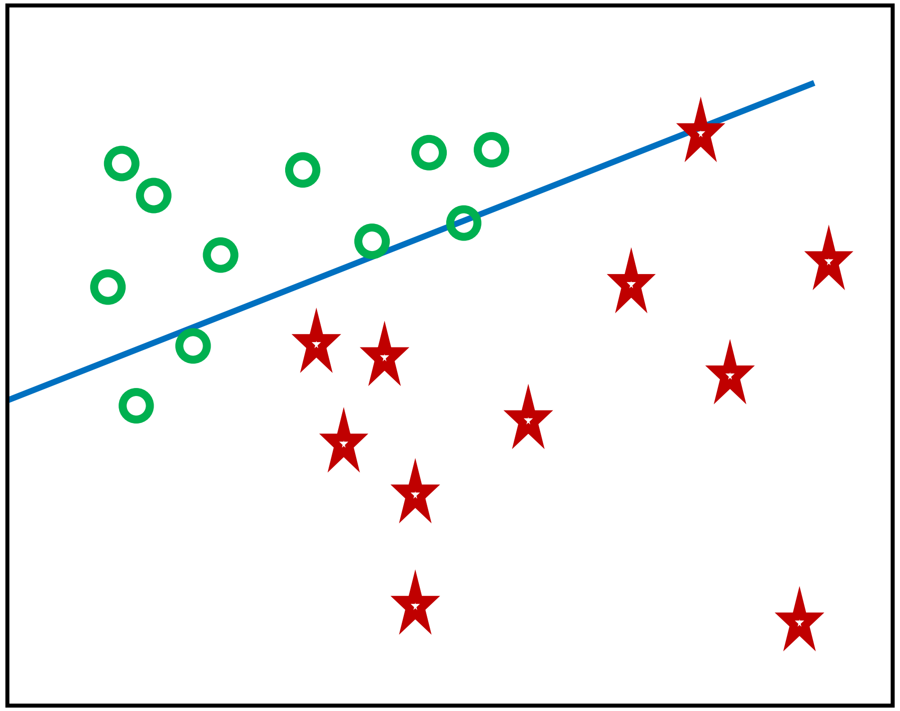
Estimate classification error. Save error and $\boldsymbol{w}$

For a fixed number of iterations

- Update $\boldsymbol{w}$
- Re-estimate error
- Save the error rate and $\boldsymbol{w}$



When Data is not Linearly Separable



Random initialization of $\boldsymbol{w}$

Estimate classification error. Save error and $\boldsymbol{w}$

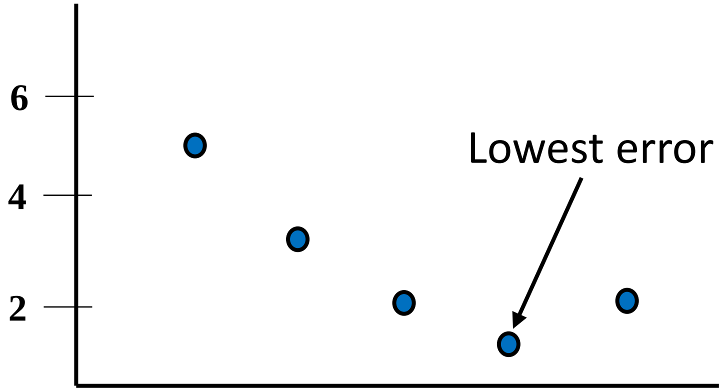
For a fixed number of iterations

    Update $\boldsymbol{w}$

    Re-estimate error

    Save the error rate and $\boldsymbol{w}$

At the end, report lowest error and corresponding $\boldsymbol{w}$



Additional Reading

PLA description and code