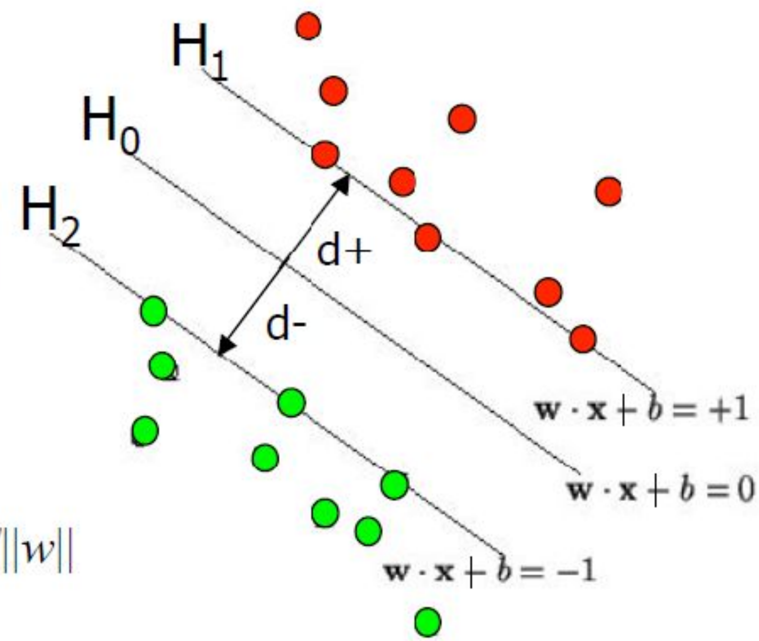
Pattern Recognition

- S. S. Samant

Maximum Margin

We want a classifier (linear separator) with as big a margin as possible.



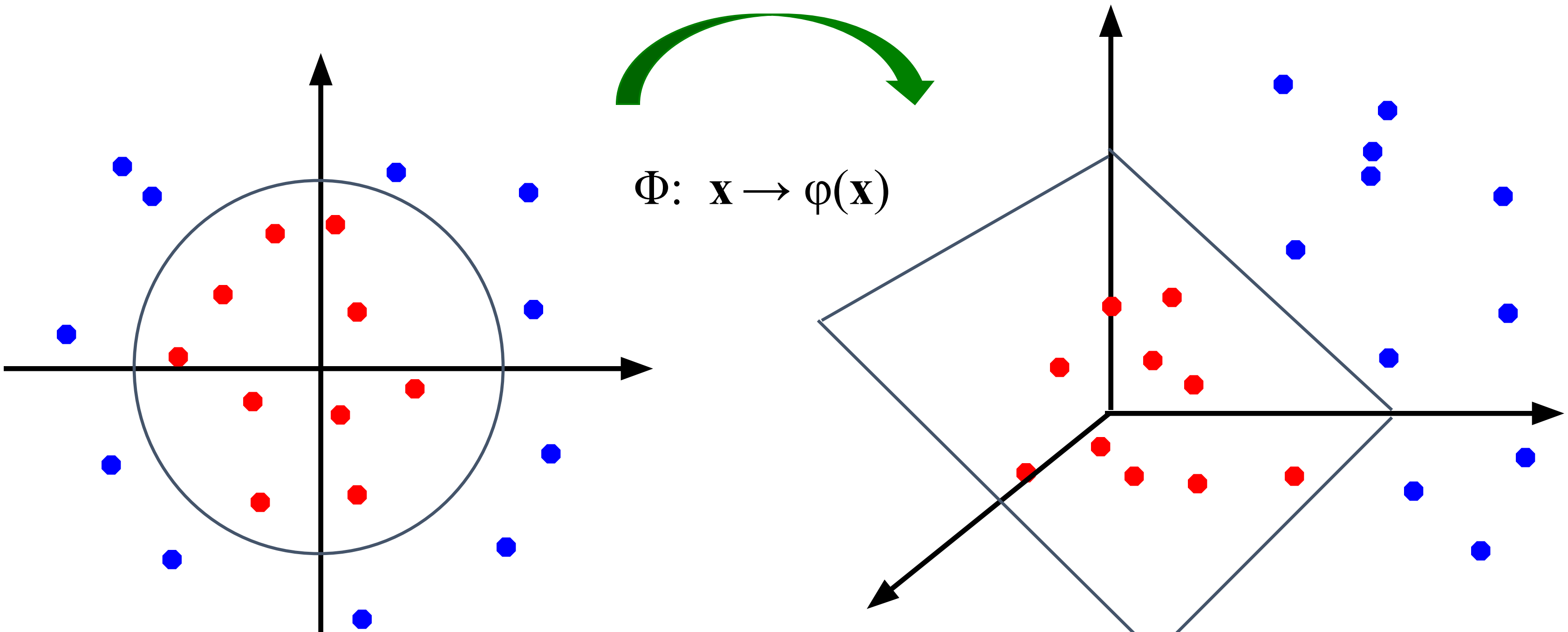
Recall the distance from a point(x_0,y_0) to a line:
$Ax+By+c = 0$ is: $|Ax_0 + By_0 + c|/\text{sqrt}(A^2+B^2)$, so,
The distance between H_0 and H_1 is then:
$|w \bullet x+b|/||w||=1/||w||$, so

The total distance between H_1 and H_2 is thus: $2/||w||$

In order to <u>maximize</u> the margin, we thus need to <u>minimize</u> $||w||$. <u>With the condition that there are no datapoints between II_1 and II_2:</u>

Non-linear SVMs: Feature spaces

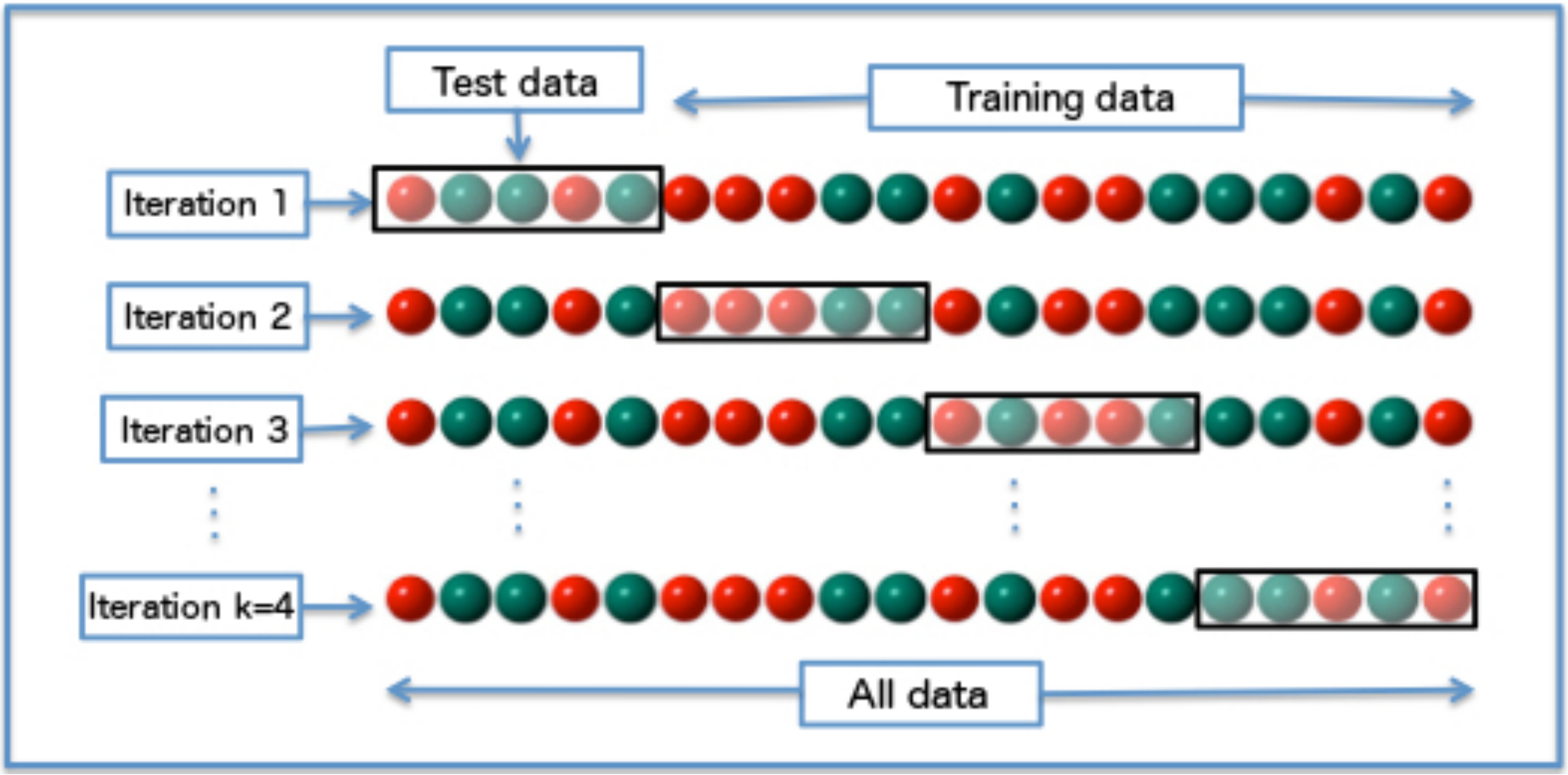
- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



$$\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$

Important Topics for Implementation

K-fold cross validation



Averaging multiple folds/categories

Micro-averaging : average using total TP/FP etc.

Macro-averaging: average of all fold's/category's Precision/Recall/F1-score

Average Precision

$$PRE = \frac{TP}{TP + FP}$$

Micro-averaging : average using total TP/FP etc.

$$PRE_{micro} = \frac{TP_1 + \cdots + TP_k}{TP_1 + \cdots + TP_k + FP_1 + \cdots + FP_k}$$

Macro-averaging: average of all fold’s/category’s Precision/Recall/F1-score

$$PRE_{macro} = \frac{PRE_1 + \cdots + PRE_k}{k}$$

Example : Microaverage vs. Macroaverage

Classification on first category

True positive (TP1) = 20
False positive (FP1) = 10
False negative (FN1) = 10

Classification on second category

True positive (TP2) = 40
False positive (FP2) = 20
False negative (FN2) = 10

$$PRE = \frac{TP}{TP + FP}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

$$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$

Example : Microaverage vs. Macroaverage

Classification on first category

True positive (TP1) = 20
False positive (FP1) = 10
False negative (FN1) = 10

Find precision and recall for each category

Classification on second category

True positive (TP2) = 40
False positive (FP2) = 20
False negative (FN2) = 10

$$PRE = \frac{TP}{TP + FP}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

$$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$

Example : Microaverage vs. Macroaverage

Classification on first category

True positive (TP1) = 20
False positive (FP1) = 10
False negative (FN1) = 10

Find F1-score for each category

Classification on second category

True positive (TP2) = 40
False positive (FP2) = 20
False negative (FN2) = 10

$$PRE = \frac{TP}{TP + FP}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

$$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$

Example : Microaverage vs. Macroaverage

Classification on first category

True positive (TP1) = 20
False positive (FP1) = 10
False negative (FN1) = 10

Classification on second category

True positive (TP2) = 40
False positive (FP2) = 20
False negative (FN2) = 10

Find micro/macro averaged F1-score of the two categories

Normalization and Standardization

Standardization (Z-score normalization)

- Variables that are measured at different scales do not contribute equally to the analysis and might end up creating a bias. We use formula:

$$Z = \frac{\overset{\text{Score}}{x} - \overset{\text{Mean}}{\mu}}{\underset{\text{SD}}{\sigma}}$$

Normalization and Standardization

Standardization (Z-score normalization)

- Variables that are measured at different scales do not contribute equally to the analysis and might end up creating a bias. We use formula:
- Ex. a variable that ranges between 0 and 100 will outweigh a variable that ranges between 0 and 1. Without standardization, the variable with the larger range gets higher weight of 100.
- Assumes that the data has a Gaussian (bell curve) distribution.
- In sklearn, we can use:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_data = scaler.fit_transform(unscaled_data)
```

Normalization and Standardization

Normalization (Min-Max Scalar)

- In this data is scaled to a fixed range, usually [0,1].
- Scaling is typically done via the following equation:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- In sklearn, we can use:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(unscaled_data)
```

Thank You!