Pokémon Recognition

阮柏愷 bkruan.ee11@nycu.edu.tw

黃柏綸 kevin503.ee12@nycu.edu.tw

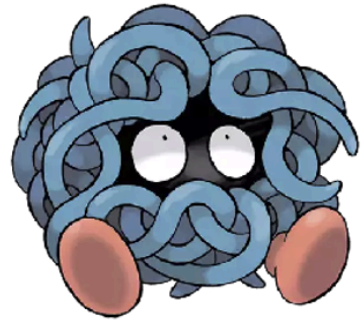
Special Thanks to Yi-Lun Wu 🤗

Outline

- Problem Description & Dataset
- Python Packages
- Tasks
    - Create Virtual Environment
    - Data preprocessing
    - KNN
    - SVM
    - PCA

Problem Description

- There are 4 classes of Pokémon.
- For each class, there are 20 images.



Python Packages

- `scikit-learn` is a free software machine learning library for the Python programming language.
- `numpy` supports large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- `Pillow` supports opening, manipulating, and saving many different image file formats.
- `matplotlib` is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Source Code

💡 We'll use colab in Lab1

Python Environment

- It is recommended that you use Ubuntu or another Linux-like operating system.
- A virtual environment is recommended due to the following advantages:
    - It allows you to distinguish your environments by name.
    - You can specify the Python version for each independent environment.
    - The environment for each project is independent.
    - It helps keep the host environment clean.

Virtual Environment for Python

- There are three recommended tools for managing virtual environments.

Name	Install	Use	Flexibility	Recommend
pyenv stars 42k	Hard	Normal	High	Normal
virtualenv stars 4.9k	Easy	Easy	Low	High
conda stars 6.8k	Normal	Hard	Normal	Low
uv stars 51k	Easy	Easy	Normal	High

Virtual Environment for Python

- There are three recommended tools for managing virtual environments.

Name	Install	Use	Flexibility	Recommend
pyenv stars 42k	Hard	Normal	High	Normal
virtualenv stars 4.9k	Easy	Easy	Low	High
conda stars 6.8k	Normal	Hard	Normal	Low
uv stars 51k	Easy	Easy	Normal	High

Create Virtual Environment

- Install `pip` and `virtualenv` system-widely for all users.

```
$ sudo apt-get install python3-pip python3-venv
```

  Since `apt-get install` require the privilege permission, we had installed these packages for you.

- Create virtual environment for Lab1.

```
$ python3 -m venv {name_of_your_venv}
```

  For example:

```
$ python3 -m venv lab1env
```

How to Use?

- Activate your virutal environment.

```
$ source [name_of_your_venv]/bin/activate
```

  For example:

```
$ source lab1env/bin/activate
(lab1env) $
```

  Once activated, the environment name will be displayed in your terminal prompt.
- Install all python packages.

```
(lab1env) $ pip install -r requirements.txt
```

- Disable/Leave current virtual environment.

```
$ deactivate
```

What is `requirements.txt`

- A list of packages along with the required versions. For example,

```
gdown
matplotlib
scikit-learn==1.0.2
```

- `pip` can interpret this file for installation.

Data Preprocessing

- Download dataset from google drive and unzip the image. link

```
$ gdown 19zDhGAb7hDNKHV80qwwcdQvGcqVeWjyh
$ unzip -q pokemon.zip
```

Data Preprocessing

- The Pokemon images are in different resolutions:

```
$ file pokemon/Charizard/01.png
pokemon/Charizard/01.png: ..., 844 x 587, ...
$ file pokemon/Charizard/02.png
pokemon/Charizard/02.png: ..., 594 x 463, ...
```

- We need to:
  - Resize all images to (200×200).
  - Convert all images to grayscale.

Checkpoint 1

- Update the following snippet in the checkpoint 1 cell.

```
img = Image.open(path)
# TODO: Checkpoint 1.
# - Convert `img` to grayscale.
# - Resize `img` to h x w.
####
img = img.convert("???")
img = img.resize("???")
```

Checkpoint 1

- Hints:
  In following package documents, you can find how to assign correct function arguments.
    - `Image.convert`
      Convert all images to grayscale.
    - `Image.resize`
      Resize all images to (200×200).

Checkpoint 1

- Execute the cell
- Run the `file` command in the next cell

```
# Check the metadata of the image
$ file pokemon_processed/Charizard/1.jpg
```

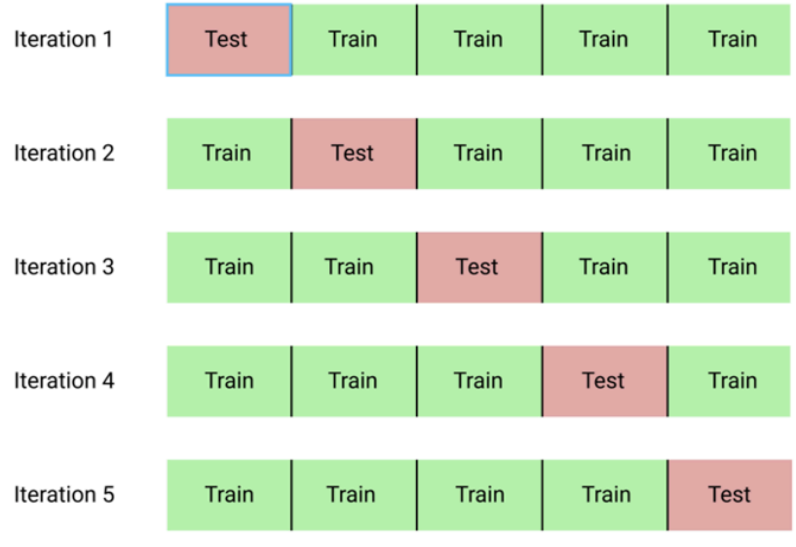
- Show gradscale images in `./pokemon_processed`

Evaluation

- Split your dataset into two sets before doing anything.
    - Train set
    - Independent test set
      "Independent" means that this dataset must not have been directly or indirectly used in training and **parameter tuning**. The independence can simulate the unseen data in the real world.

Parameter Tunning: K-Fold Cross Validation

- Split the dataset into K subsets, taking one subset as testing set and the other $K-1$ subsets as training sets. Repeat this procedure K times, then average the evaluation results.



Evaluation Metrics

- `sklearn.metrics.accuracy_score`
- `sklearn.metrics.precision_score`
- `sklearn.metrics.recall_score`

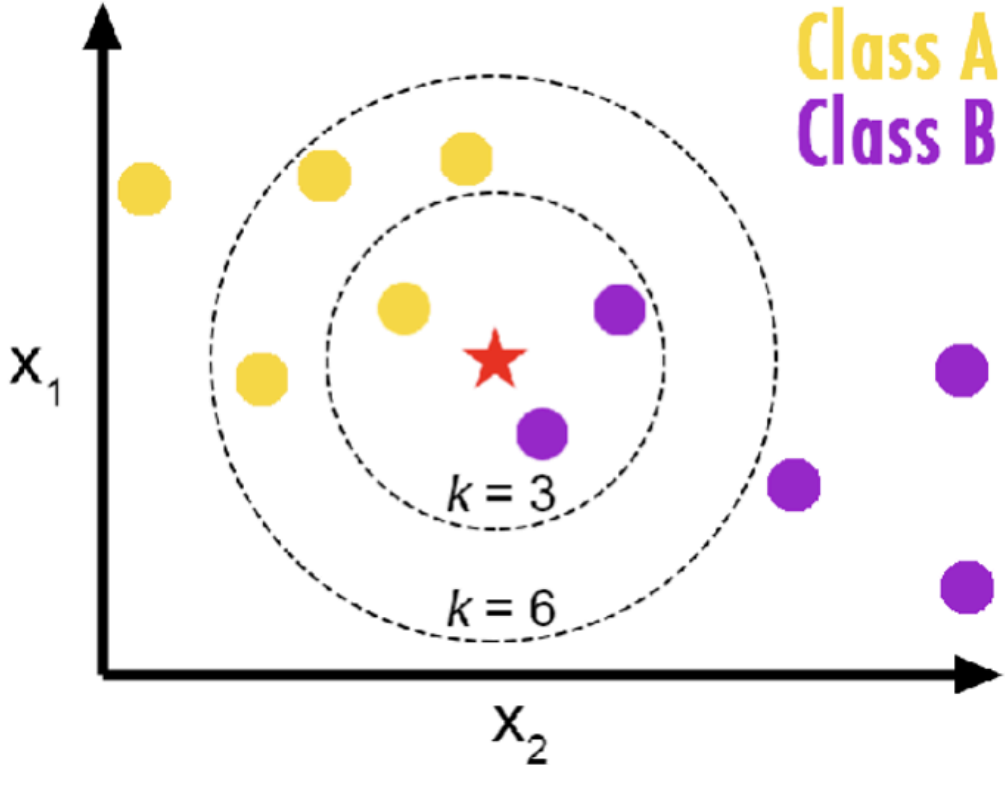
```
Evaluate on test set:
  Accuracy : 0.85
  Precision: 0.87
  Recall   : 0.88
```

- `sklearn.metrics.confusion_matrix`

	Charizard	Gengar	Pikachu	Tangela
Charizard	3	1	1	0
Gengar	5	2	0	0
Pikachu	0	0	2	0
Tangela	4	0	0	1

KNN

A classical classification model.



Checkpoint 2

- Update the following snippet in the `checkpoint 2` cell

```
# TODO: Checkpoint 2. Train a KNN classification model.
# - Extend 'n_neighbors' to let GridSearchCV find the best value.
####
param_grid = {
    'n_neighbors': [1]
}
clf = GridSearchCV(KNeighborsClassifier(), param_grid, cv=3)
clf = clf.fit(X_train, y_train)
```

Document links: `KNeighborsClassifier`, `GridSearchCV`.

Checkpoint 2

- Execute the cell and show the accuracy, precision and recall:

```
Evaluate on test set:
  Accuracy : 0.75
  Precision: 0.82
  Recall   : 0.77
```

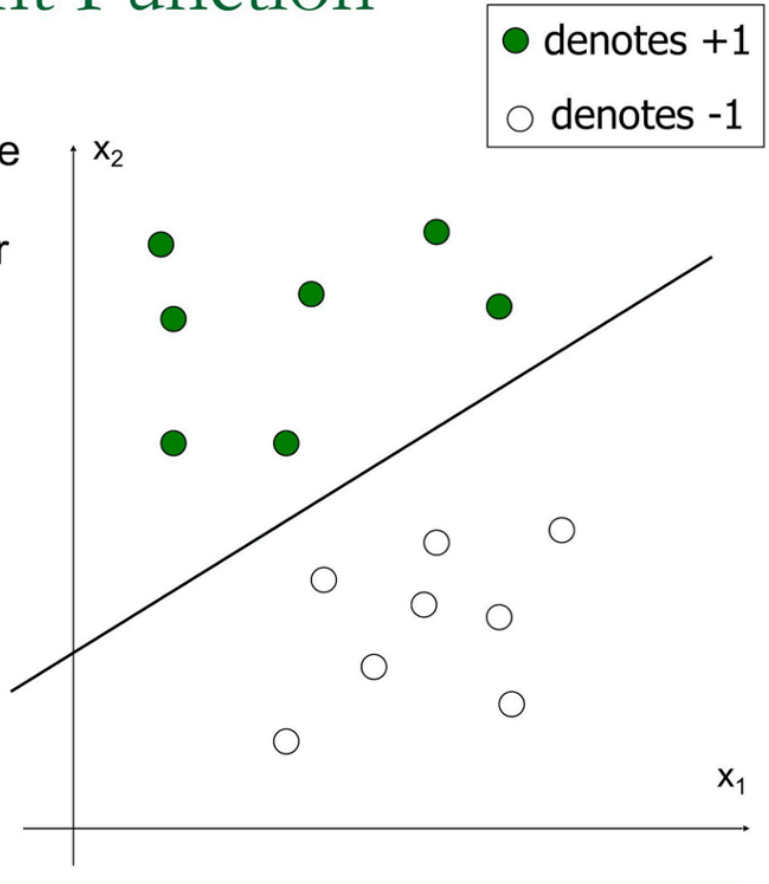
Support Vector Machine (SVM)

source: https://slideplayer.com/slide/10982739/

Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?

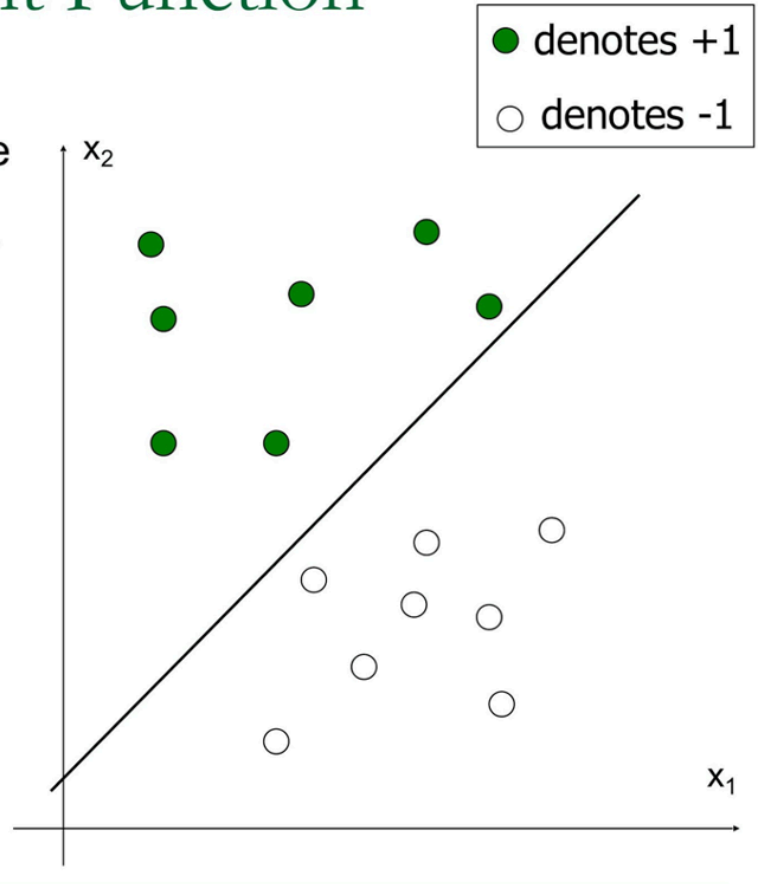
- Infinite number of answers!



Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?

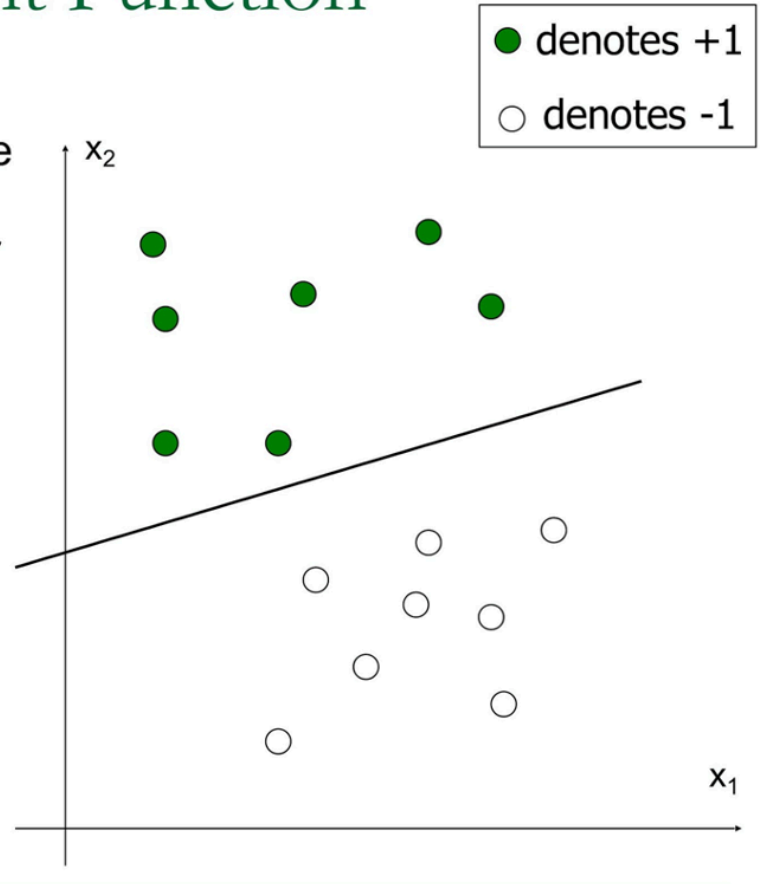
- Infinite number of answers!



Linear Discriminant Function

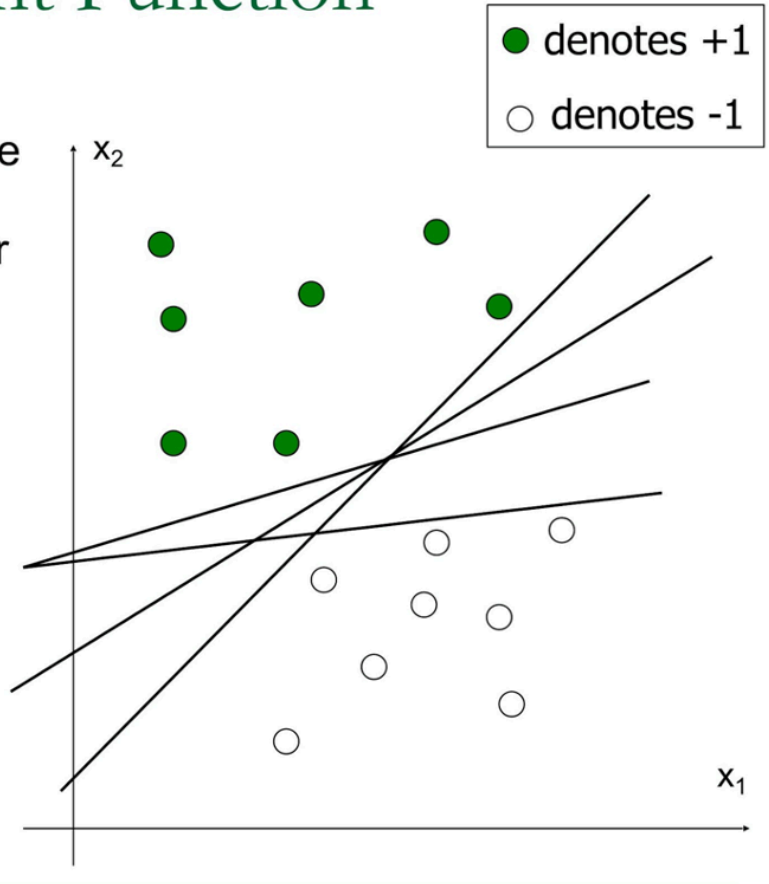
- How would you classify these points using a linear discriminant function in order to minimize the error rate?

- Infinite number of answers!



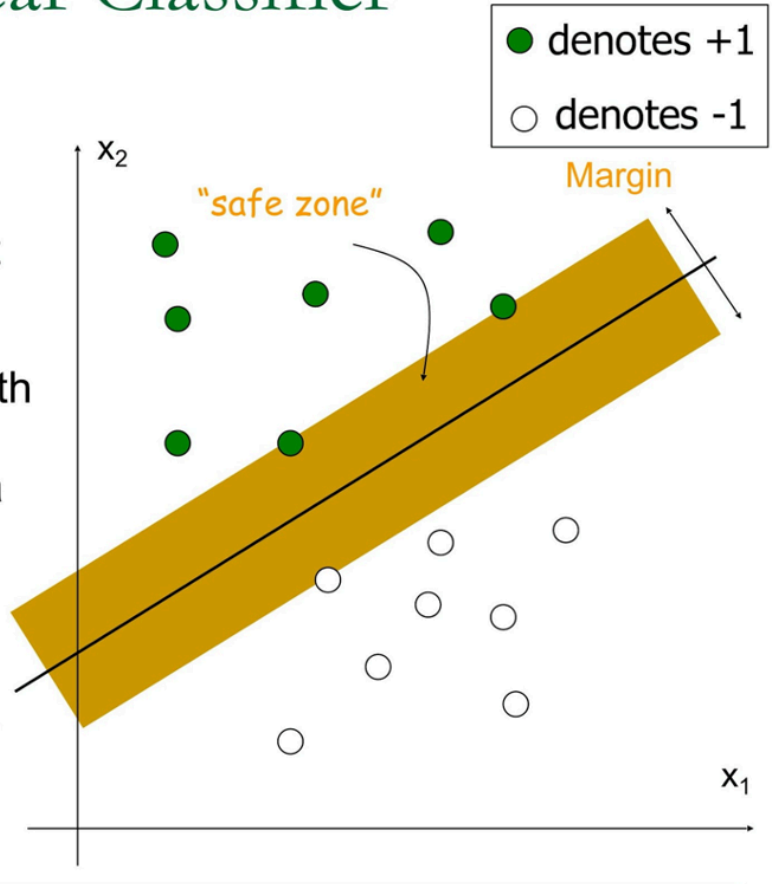
Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?
- Infinite number of answers!
- Which one is the best?



Large Margin Linear Classifier

- The linear discriminant function (classifier) with the maximum margin is the best
- Margin is defined as the width that the boundary could be increased by before hitting a data point
- Why it is the best?
  - Robust to outliners and thus strong generalization ability



Linear Discriminant Function

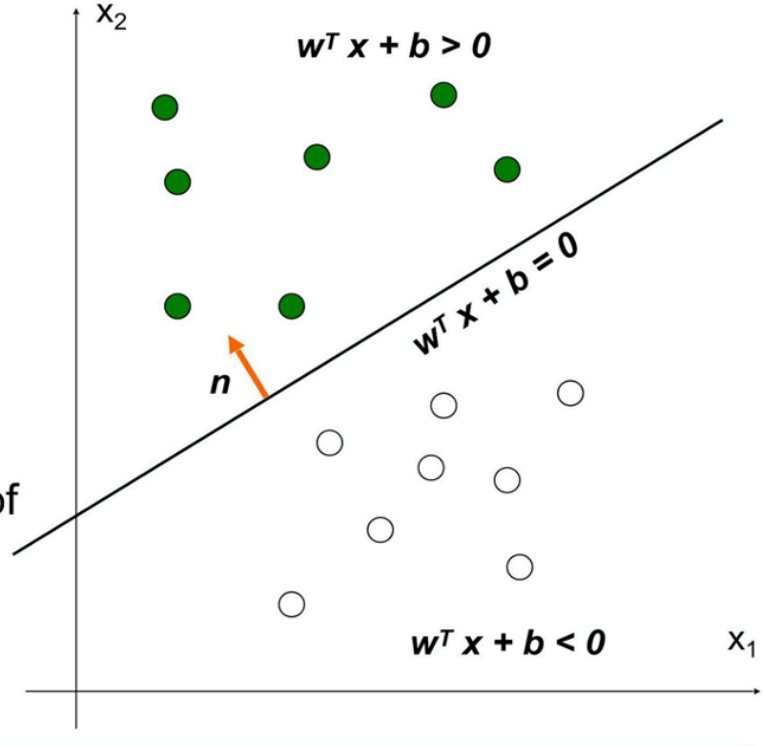
- g(x) is a linear function:

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

- A hyper-plane in the feature space

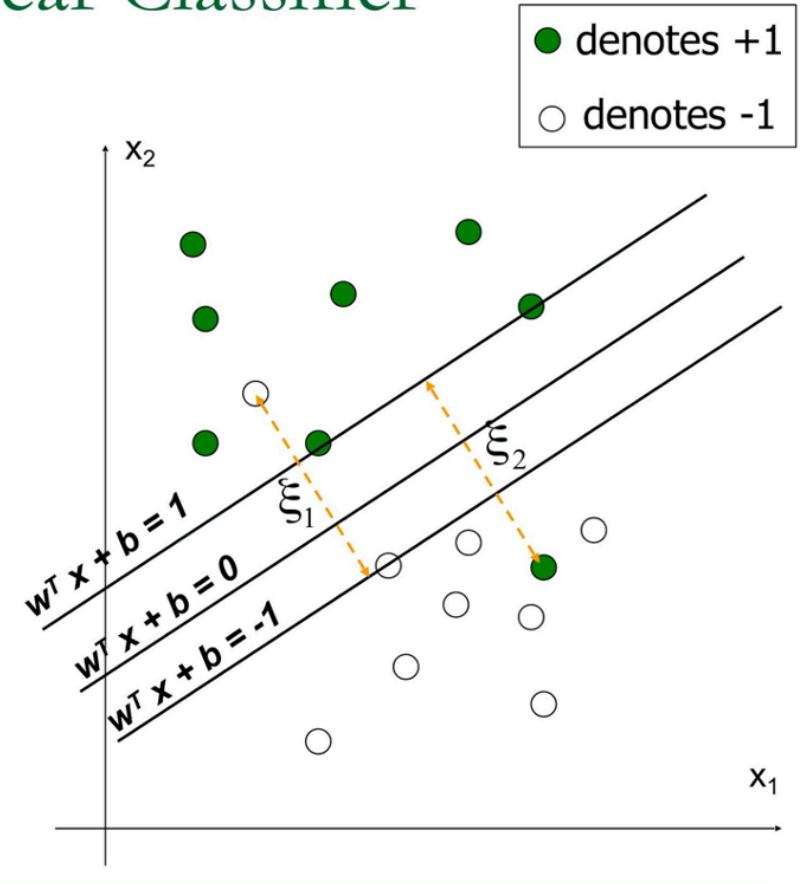
- (Unit-length) normal vector of the hyper-plane:

$$\mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$



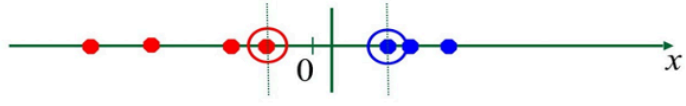
Large Margin Linear Classifier

- What if data is not linear separable? (noisy data, outliers, etc.)
- Slack variables ξ_i can be added to allow mis-classification of difficult or noisy data points



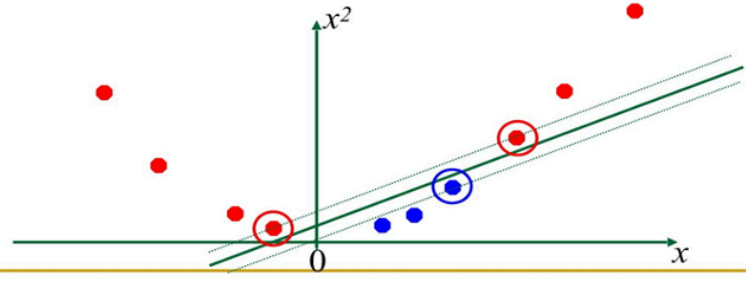
Non-linear SVMs

- Datasets that are linearly separable with noise work out great:



- But what are we going to do if the dataset is just too hard?

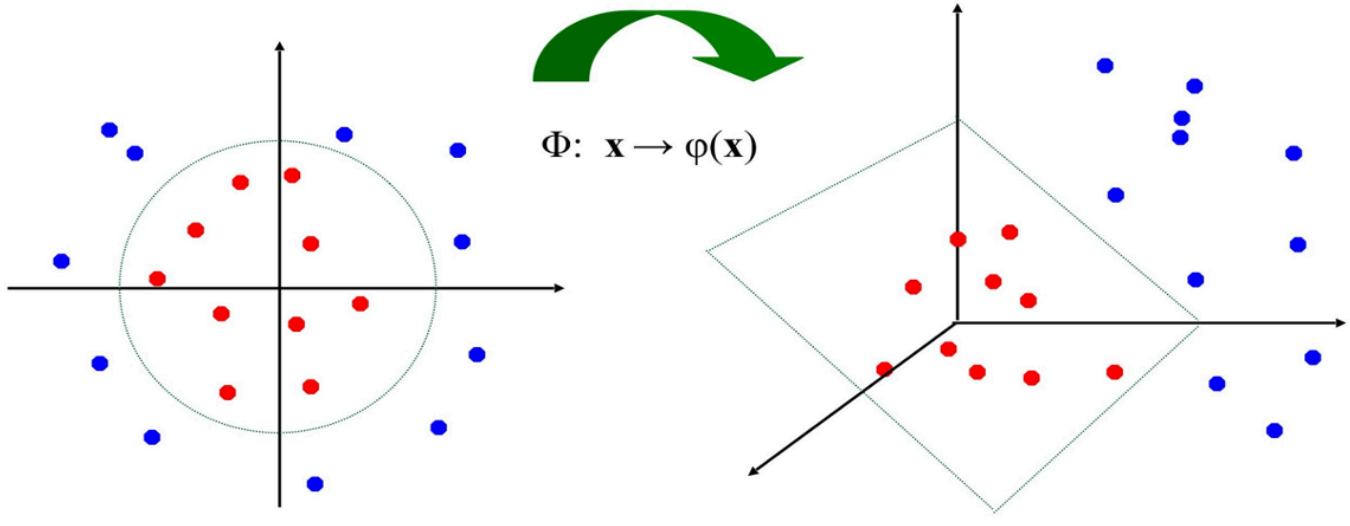
- How about… mapping data to a higher-dimensional space:



This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Non-linear SVMs: Feature Space

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



This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Nonlinear SVMs: The Kernel Trick

- Examples of commonly-used kernel functions:
  - Linear kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
  - Polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$
  - Gaussian (Radial-Basis Function (RBF)) kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}\right)$$

  - Sigmoid:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i^T \mathbf{x}_j + \beta_1)$$

- In general, functions that satisfy *Mercer's condition* can be kernel functions.

Checkpoint 3

- Update the following snippet in the `checkpoint 3` cell.

```
# TODO: Checkpoint 3. Train a SVM classification model.
# - Update `param_grid` to allow GridSearchCV to find the best parameters.
####
param_grid = {
    'kernel': ['rbf', 'linear'],
    'C': [???],
    'gamma': [???],
}
clf = GridSearchCV(
    SVC(class_weight='balanced'), param_grid, cv=3)
clf = clf.fit(X_train, y_train)
```

Documentation: SVC.

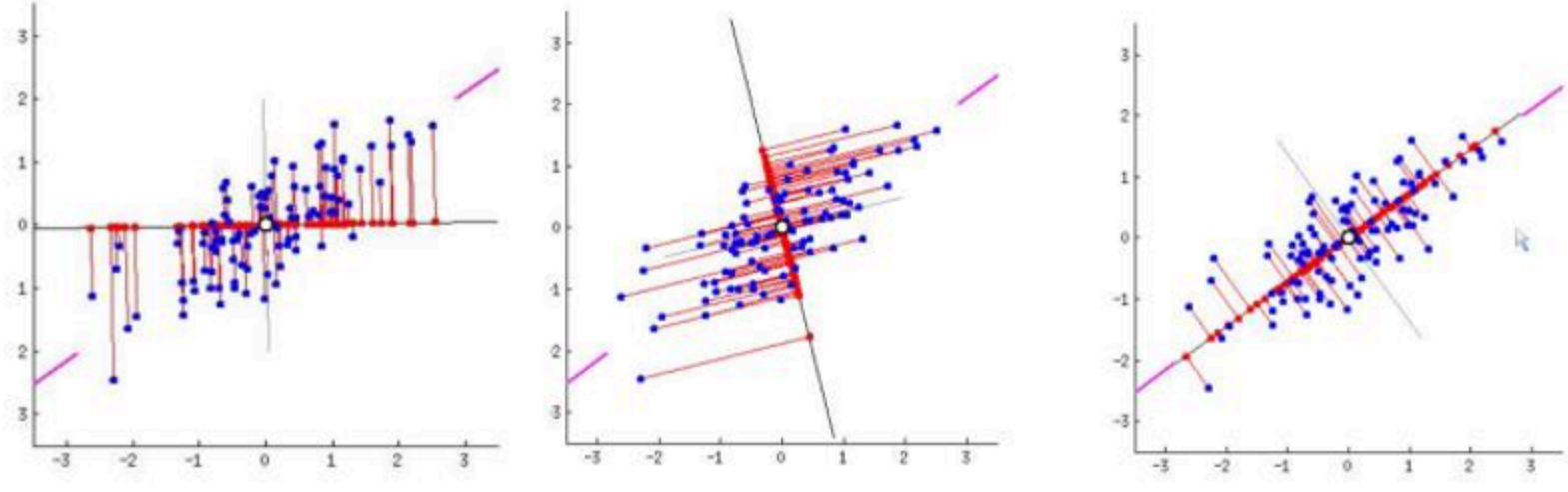
Checkpoint 3

- Execute the cell and show the accuracy, precision and recall:

```
Evaluate on test set:
  Accuracy : 1.00
  Precision: 1.00
  Recall   : 1.00
```

Principal Components Analysis (PCA)

Multi-dimensional data can easily contain noisy features, so it's better to select only those that capture the patterns in our data.



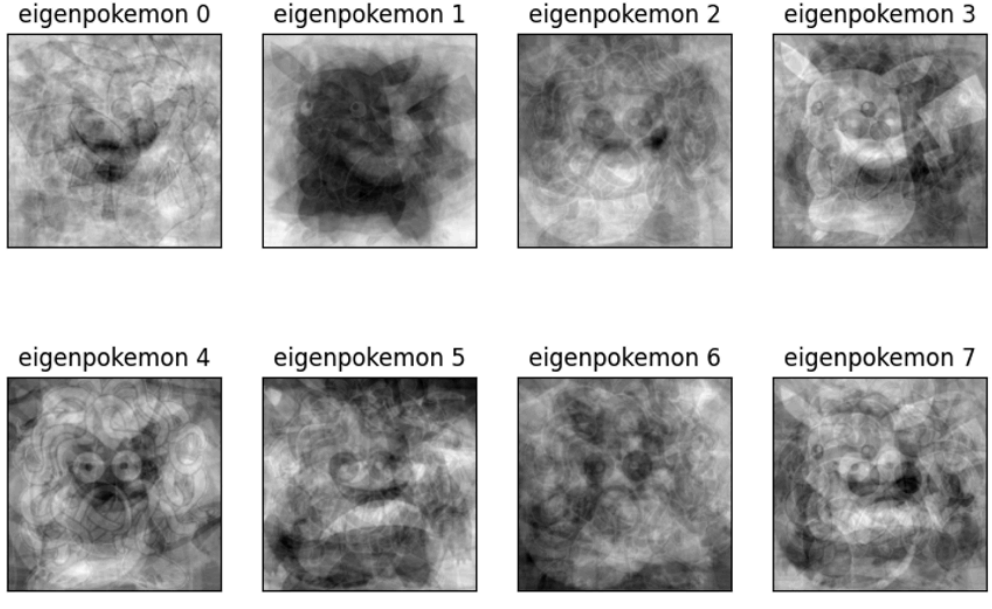
Checkpoint 4

- Update the following snippet in the `checkpoint 4` cell.

```
# TODO: Checkpoint 4. Apply PCA to the dataset before
# training the SVM.
# - Update `n_components` to set the number of eigenvectors.
####
n_components = ???
pca = PCA(n_components=n_components, whiten=True).fit(X_train)
X_train = pca.transform(X_train)
X_test = pca.transform(X_test)
```

Checkpoint 4

- Execute the cell and show the accuracy, precision and recall of KNN and SVC.
- Also, show `PCA.png`.



The End