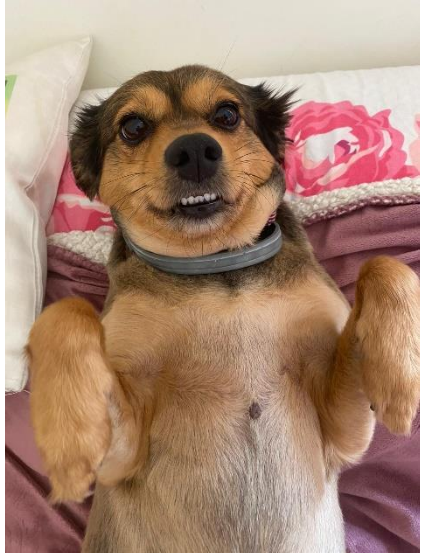
# Support Vector Machine (SVM)

# Image classification

- Image classification is the task of ***assigning a label to an image*** from a ***pre-defined set of categories***.
  - Example:

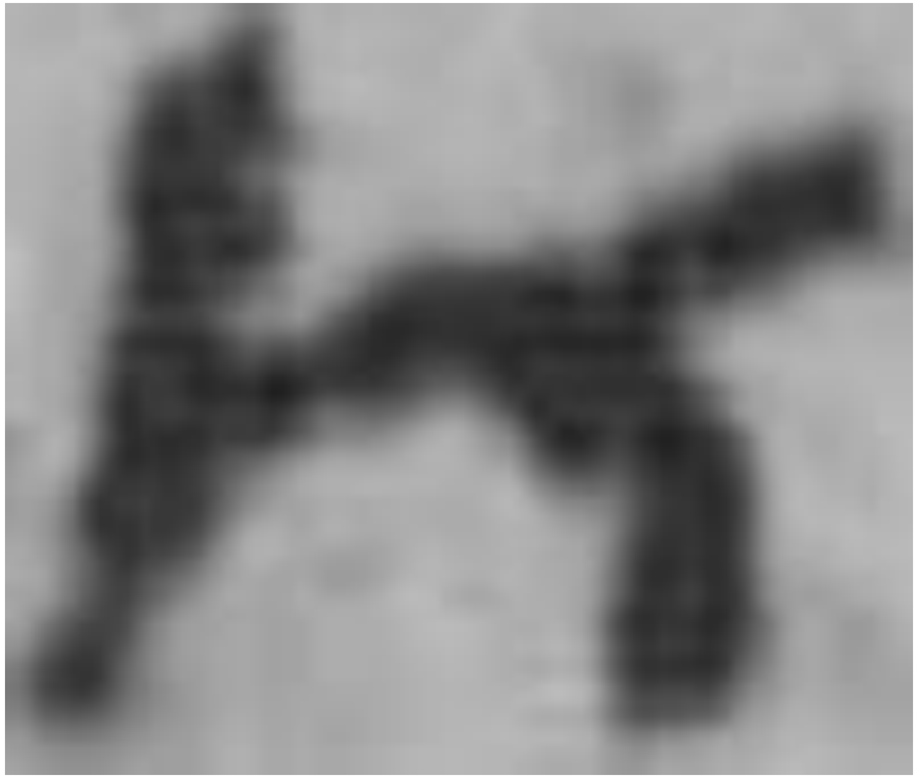
    labels {cat, dog, bear, fish, bird}



Label: dog

# Image classification is a challenging problem

- We see “aleph”



Computer sees a matrix of numbers

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 177 | 180 | 175 | 169 | 179 | 179 | 166 | 111 | 160 | 180 |
| 140 | 148 | 165 | 163 | 179 | 180 | 153 | 63 | 105 | 179 |
| 84 | 57 | 104 | 154 | 170 | 174 | 154 | 64 | 84 | 174 |
| 140 | 96 | 60 | 72 | 87 | 101 | 141 | 77 | 83 | 173 |
| 182 | 172 | 97 | 42 | 49 | 54 | 74 | 71 | 72 | 164 |
| 181 | 156 | 63 | 52 | 111 | 142 | 103 | 58 | 61 | 156 |
| 185 | 140 | 52 | 115 | 171 | 177 | 171 | 113 | 55 | 137 |
| 179 | 129 | 49 | 132 | 180 | 176 | 168 | 154 | 83 | 132 |
| 166 | 132 | 53 | 130 | 182 | 179 | 169 | 166 | 112 | 92 |
| 173 | 170 | 129 | 154 | 183 | 177 | 169 | 168 | 141 | 125 |

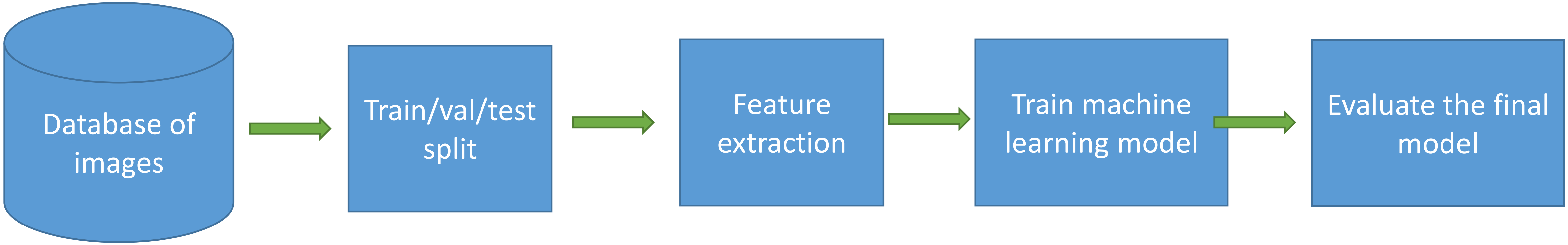
# Machine learning for image classification

- Describe an image(image part) by a descriptor
- Apply machine learning to “teach” the computer how objects from each class look
    - Supervised learning
    - Unsupervised learning
    - Semi-supervised learning

# Machine learning

- How to tell the difference between a dog and other categories?
- Instead of a coding set of rules to recognize each class, in machine learning we use a **data-driven approach**
    - Supply examples of each class and teach the model to recognize between them
        - Training set: each example consist of a pair (image, label)

# The image classification pipeline



Database of images

Train/val/test split

Feature extraction

Train machine learning model

Evaluate the final model

# Compiling the dataset

- Collect the images from each class
- The dataset should be balanced – approximately the same number of images for each class

# Train/Val/Test Sets

- Training set – examples used to train the model ($\approx 80\%$)
- Validation set – a “test set” used in training to tune various hyperparameters ($\approx 10\%$)
- Test set – evaluate the performance of the model ($\approx 10\%$)

# Feature extraction

[4, 0.25, 1.7, …..]

[3, 0.37, 7.1, …..]

# Training

- Machine learning models
    - Support Vector Machine (SVM)
    - Decision Trees
    - Random Forest
    - K-Nearest Neighbor
    - Neural networks
    - …

# Evaluation

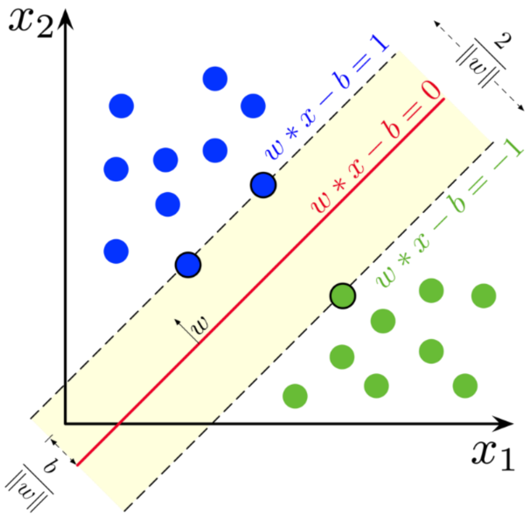
- For each of the images in the test set we predict the label and compare to it ground-truth (actual) label
- Calculate various metrics based on the results, e.g., precision, recall, f-measure

# Support Vector Machine (SVM)

- **Support-vector machines** (**SVMs)**, are supervised learning models with associated learning algorithms that analyze data for classification
- Developed at AT&T Bell Laboratories by Vladimir Vapnik with colleagues (1992)
- Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other

# Support Vector Machine (SVM)

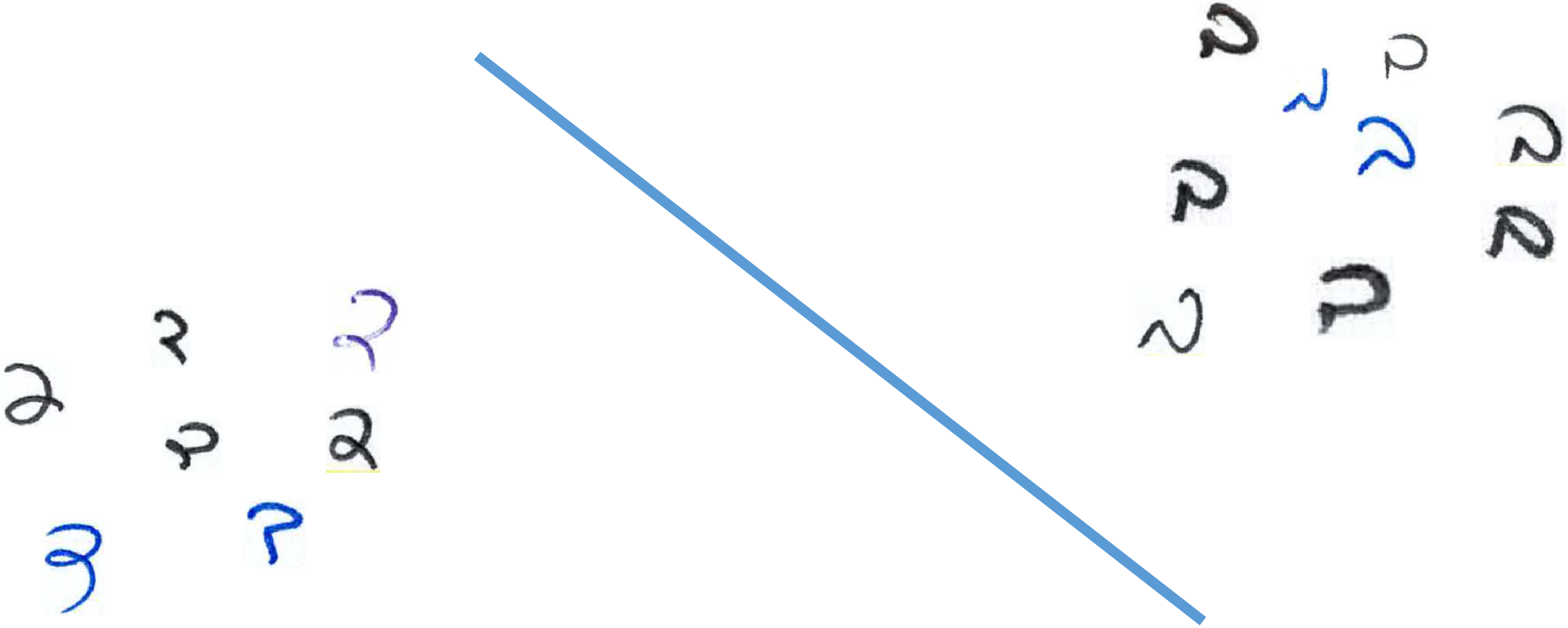
- SVM maps training examples to points in space so as to maximize the width of the gap between the two categories
- New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall



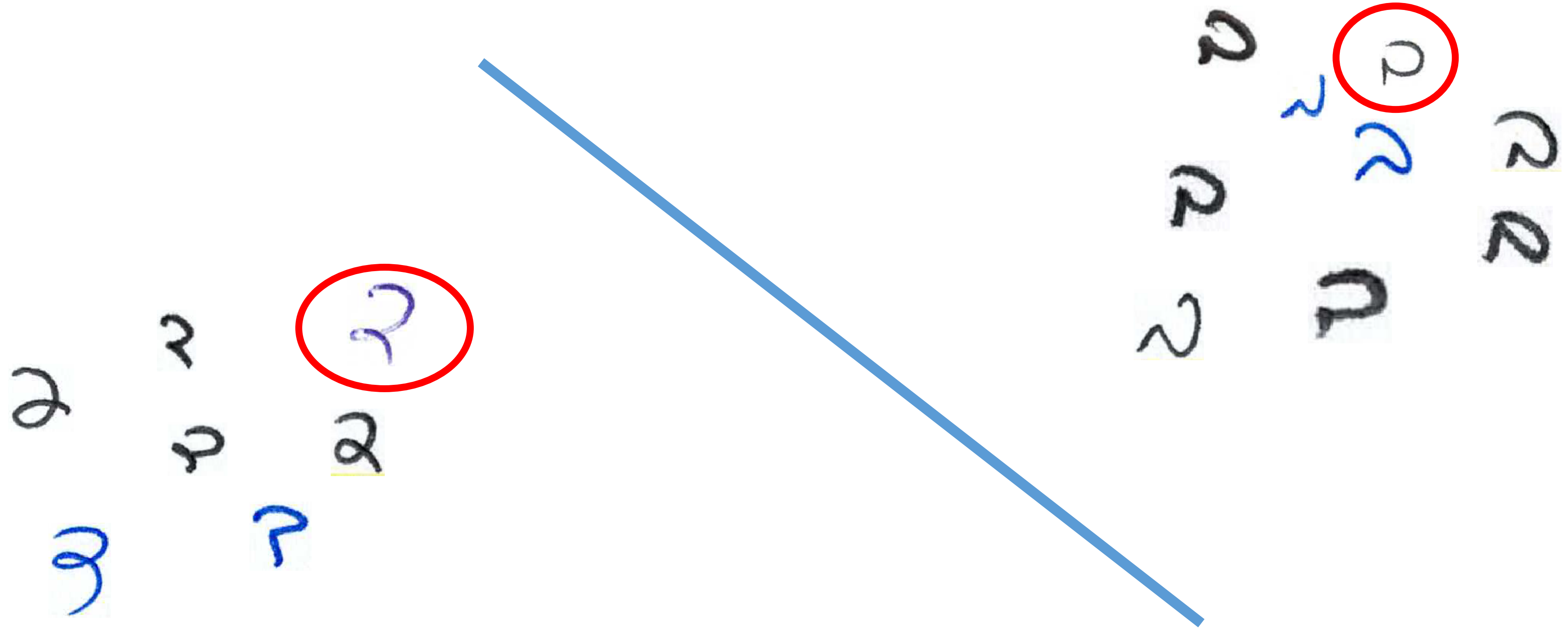
Refence  wikipedia

# Separating Hyperplane

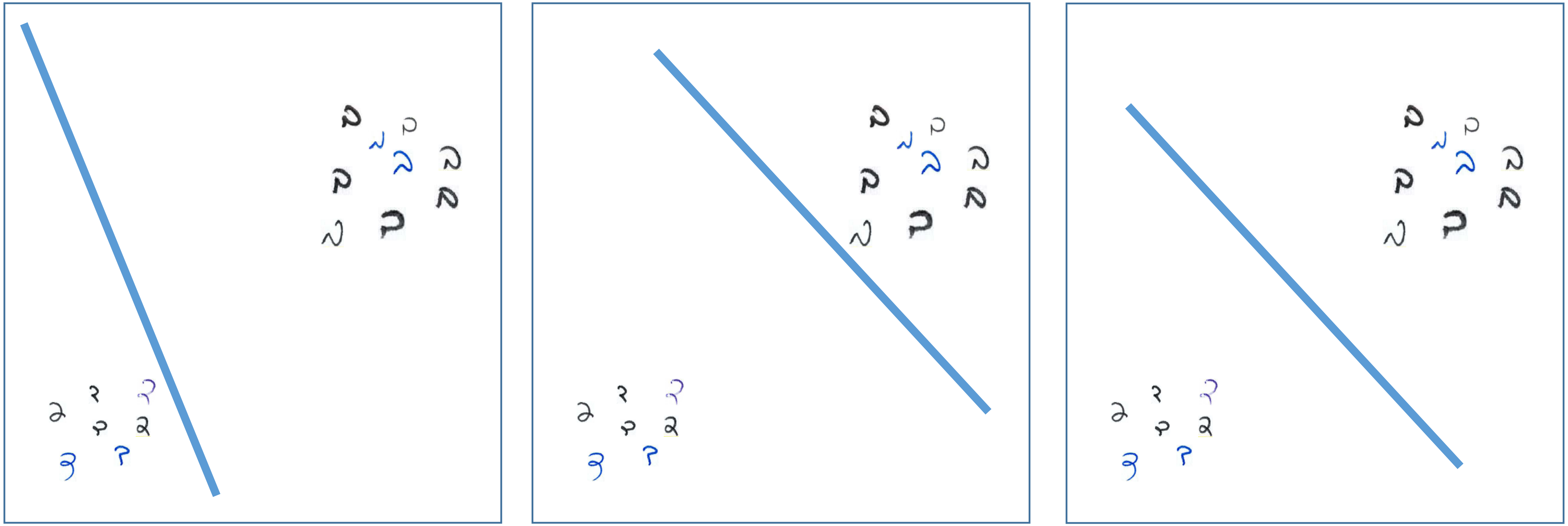
- **Primary assumption:** feature vectors of similar images lie *close together* in an *n*-dimensional space
- The line used to separate the data is called the separating hyperplane.



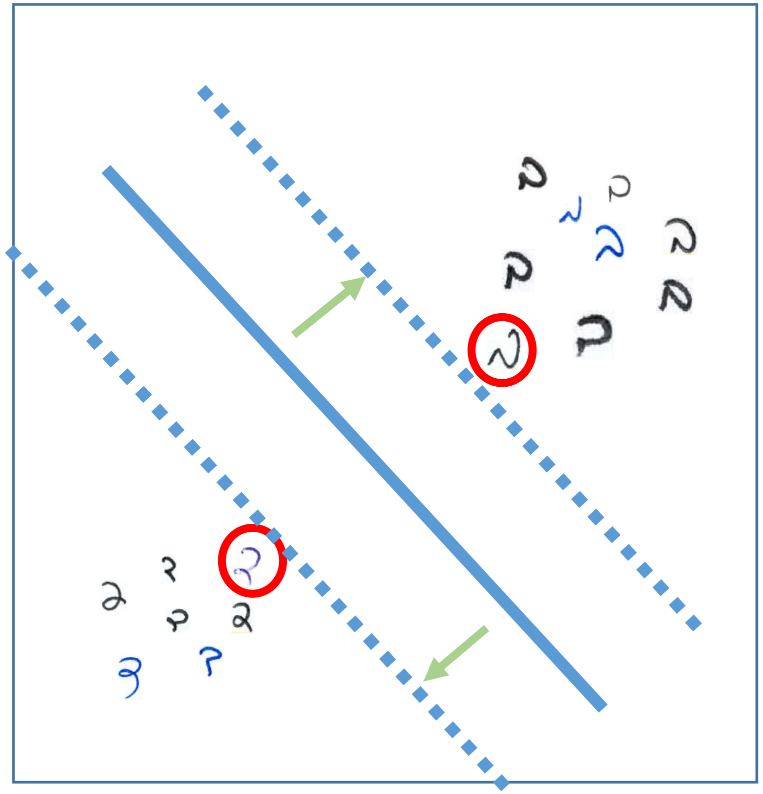
# Separating Hyperplane



# Separating Hyperplane

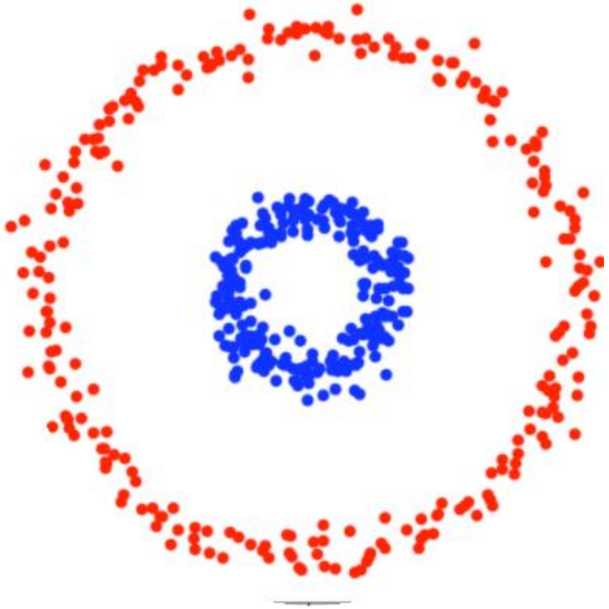


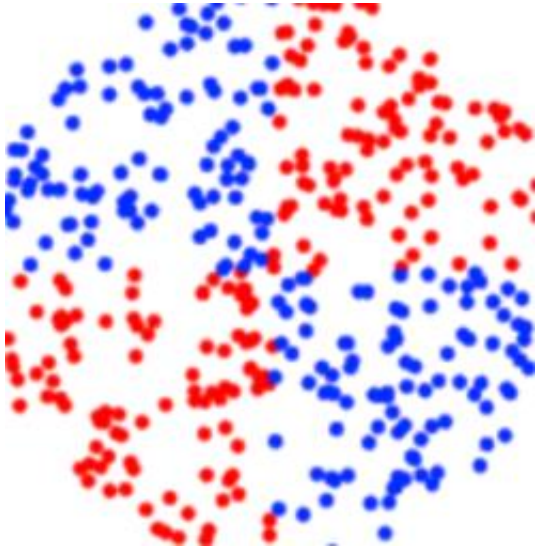
# Maximum-Margin Separating Hyperplane



Maximum-Margin Separating Hyperplane - largest distance to the nearest training-data point of any class

# Non-linearly separable datasets





We cannot separate these two sets by a straight line

# The kernel trick

- Project all the points into a higher dimension using so-called a **kernel matrix**

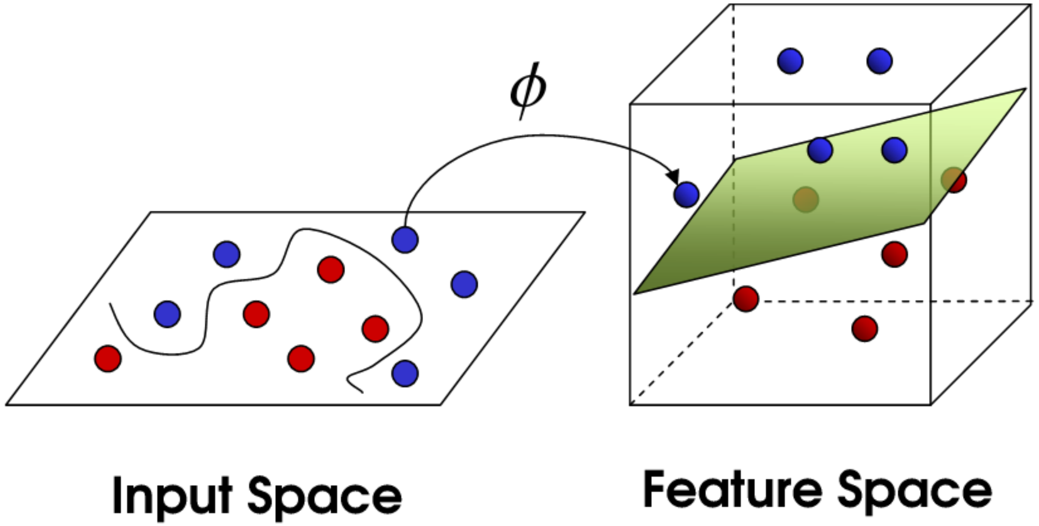


Image taken from https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f

# Types of kernel

- Linear: $K(x, y) = x^T y$
- Polynomial: $K(x, y) = (x^T y + \mathrm{c})^{\ d}$
- Radial basis function kernel: $K(x, y) = \exp(-\gamma \left|\left|x - y\right|\right|^2)$
- Sigmoid: $K(x, y) = \tanh(\gamma(x^T y) + \mathrm{r})$

*x* and *y* are vectors in the *input space*, i.e. vectors of features computed from training or test sets

# SVM implementation

- Scikit-learn library

```
from sklearn import svm
from sklearn.metrics import classification_report


 #Create a linear SVM classifier
clf = svm.SVC(kernel=raenil")

# Train classifier
clf.fit(train_data, train_labels)

# Make predictions on unseen test data
clf_predictions = clf.predict(test_data)
print("Accuracy: {}%".format(clf.score(test_data, test_labels) * 100))

# printing report and different statistics

print(classification_report(test_labels, clf.predict(test_data)))
```

# SVM implementation

- Scikit-learn library

```
from sklearn import svm
from sklearn.metrics import classification_report


 #Create a linear SVM classifier
clf = svm.SVC(kernel=raenil")

# Train classifier
clf.fit(train_data, train_labels)

# Make predictions on unseen test data
clf_predictions = clf.predict(test_data)
print("Accuracy: {}%".format(clf.score(test_data, test_labels) * 100))

# printing report and different statistics

print(classification_report(test_labels, clf.predict(test_data)))
```