ml-01

November 29, 2019

1 Python for Data Science

Machine Learning 1 - Introduction and Simple Supervised Learning

2 Overview

- What is Machine Learning (ML)?
- Where does it come from?
- What ML is not
- Application Examples

3 What is Machine Learning?

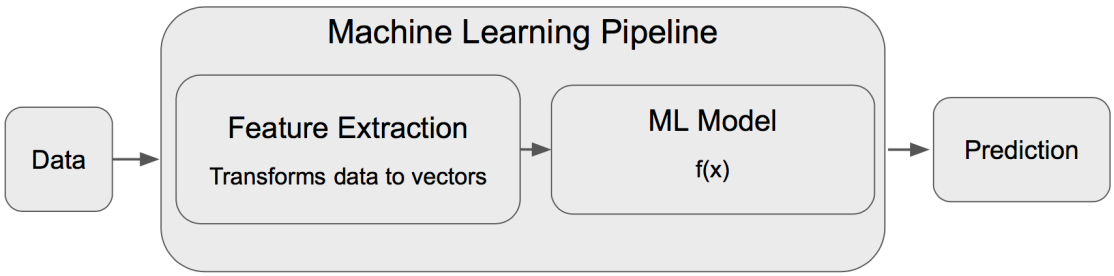
Given

- a model class $f(.)$ (Neural Networks, Linear models, Random Forests, ...)
- some data x

Machine Learning:

Learn function $f(x)$ that takes data as input and makes a prediction

4 Machine Learning Systems in Practice



5 Some ML Applications

Data	Prediction	Popular models
Email Texts	Spam or not?	Linear Models
Clicks/Likes	Personality, credit risk, playlists, …	Linear Models
Text in one language	Text in another language	Neural networks
Images	Object on image	Neural Networks
Text	Spoken Language	Neural Networks
Spoken Language	Text	Neural Networks

6 What is Machine Learning

- ML researchers come from many disciplines:
  - Statistics
  - Mathematics
  - Physics
  - Computer Science
- Some consider ML (or its methods) a subfield of the above genres
- Also, ML has many fields / synonyms:
  - Artificial Intelligence
  - Data Mining
  - Information Retrieval
  - Data Science
- Hence there are many definitions

7 What is Machine Learning

Some definitions that can be helpful:

ML systems

- **learn rules from data** (as opposed to handcrafted rules)
- **perform cognitive tasks** (as in: emulating human cognition)

8 A Short History of Machine Learning

- 1632: Bodies as biological machines, [Descartes, 1632]
- 1940/50: From machines to intelligent machines
  - Cybernetics – mechanics of cognition [Wiener, 1948]
  - Turing Test [Turing, 1950]: Can machines think?

  - Perceptron [Rosenblatt, 1958]: Artificial 'brains'
- Eliza [Weizenbaum, 1966]: Better than psychotherapy!
- Chinese Room argument [Searle, 1980]: Against functionalism
- 1973 AI winter [Lighthill, 1973] Classical AI was mainly built on hand made rules

  Problem: No scalability, no unified theory of cognition
- New AI: Let machines learn rules from data [Russell and Norvig, 2010]

9 What is Artificial Intelligence?

9.1 Alan Turing

- Important theoretical work on computability
- Programmed first chess program
- Helped to crack Enigma encryption
- Most important Computer Science Prize: Turing Award
- Many people at the time were complaining about the hybris behind the research on intelligent machines
- Turing thought that in order to argue about this, one needs a proper definition of intelligence

9.2 The Imitation Game

In 1950 A.M. Turing proposed the Imitation Game [Turing, 1950].

- Interrogator communicates with two participants via teleprompter
- One of the participants is a computer
- The task of the interrogator is to find out who is human.

This is often referred to as the Turing Test:

- Many humans fail the this test
- Many machines pass it nowadays - some passed it already in the 1960

9.3 ELIZA - An Early Chatbot

- **Human** All men are alike
- **Eliza** In what way?
- **Human** They're always bugging us about something or other
- **Eliza** Can you think of a specific example
- **Human** Well, my boyfriend made me come here
- **Eliza** Your boyfriend made you come here?
- **Human** He says I'm depressed much of the time
- **Eliza** I am sorry to hear you're depressed
- **Human** It's true, I am unhappy

- **Eliza** Do you think coming here will help you not to be unhappy?
- **Human** I need some help, that much seems certain.
- **Eliza** What would it mean to you if you got some help?
- **Human** Perhaps I could learn to get along with my mother.
- **Eliza** Tell me more about your family …

10 What Machine Learning is not

- Algorithms that do not learn from data
- Expert systems with hand crafted logic / rules

10.1 Example: Rule Based System For Spam Classification

```
def classify_mail(text):
    spam_score = 0

    if 'Viagra' in text:
            spam_score += 1
    if 'money' not in text:
            Spam_score -= 1

    is_spam = spam_score > 0

    return is_spam
```

11 Why Machine Learning

- Rule based systems (old AI) are great:
  - Testing is easy
  - Always interpretable
  - Debugging is easy
- But rule based systems are problematic
  - Require human experts to set up rules
  - Rules need to be kept up to date
  - Doesn't scale
  - Doesn't generalize

12 A ML Spam Classifier

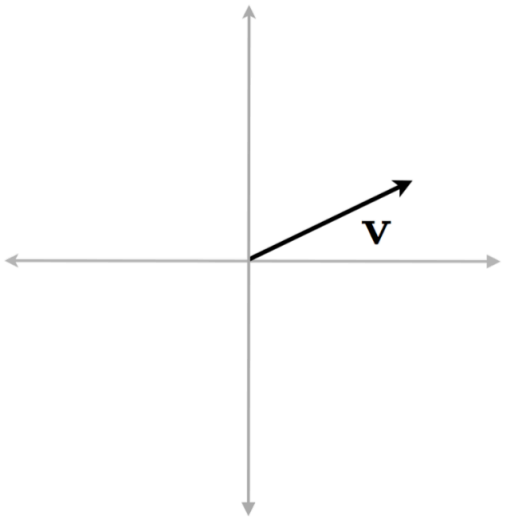
```
def predict_spam(word_histogram = {'Viagra': 1, ...}):
```

```
# Learned rules
rules = {'Viagra': 10.0, 'Homework': -1.0, ...}

spam_score = 0.0

# Word histograms are weighted and summed up
for word, weight in rules.items():
    spam_score += word_histogram[word] * weight

return spam_score > 0
```

13 Rules and Data are Vectors

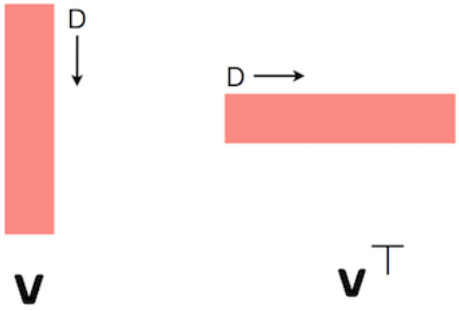
- Rules learned by Machine Learning can be expressed as matrix/vector operations
- In this form, rules can be treated as mathematical functions $f(x)$
- Vector representation helps to understand some ML models
- Mathematical functions can be optimized

14 Vectors

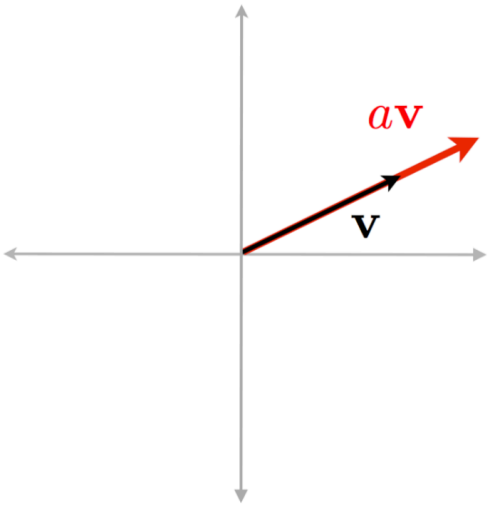


$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_D \end{bmatrix}$$

15 Vector Transpose

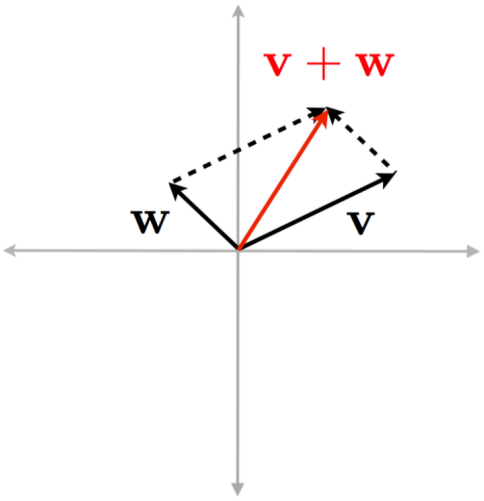


16 Vector Scaling



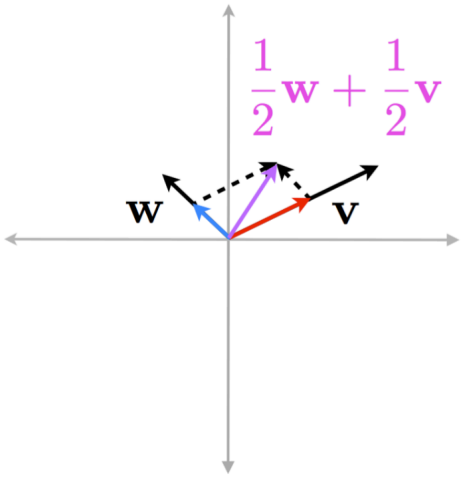
$$\mathbf{a}\mathbf{v} = \begin{bmatrix} av_1 \\ av_2 \\ \vdots \\ av_D \end{bmatrix}$$

17 Vector Addition



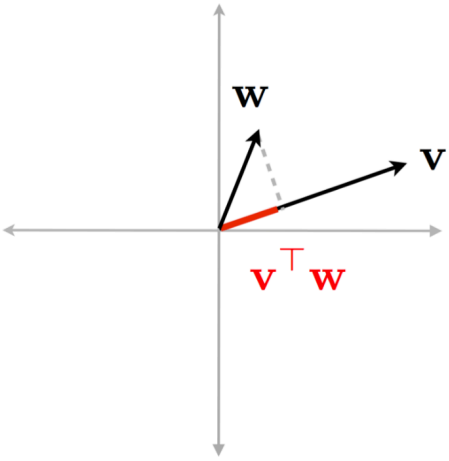
$$\mathbf{v} + \mathbf{w} = \begin{bmatrix} v_1 + w_1 \\ v_2 + w_2 \\ \vdots \\ v_D + w_D \end{bmatrix}$$

18 Vector Scaling and Addition



$$a\mathbf{v} + b\mathbf{w} = \begin{bmatrix} av_1 + bw_1 \\ av_2 + bw_2 \\ \vdots \\ av_D + bw_D \end{bmatrix}$$

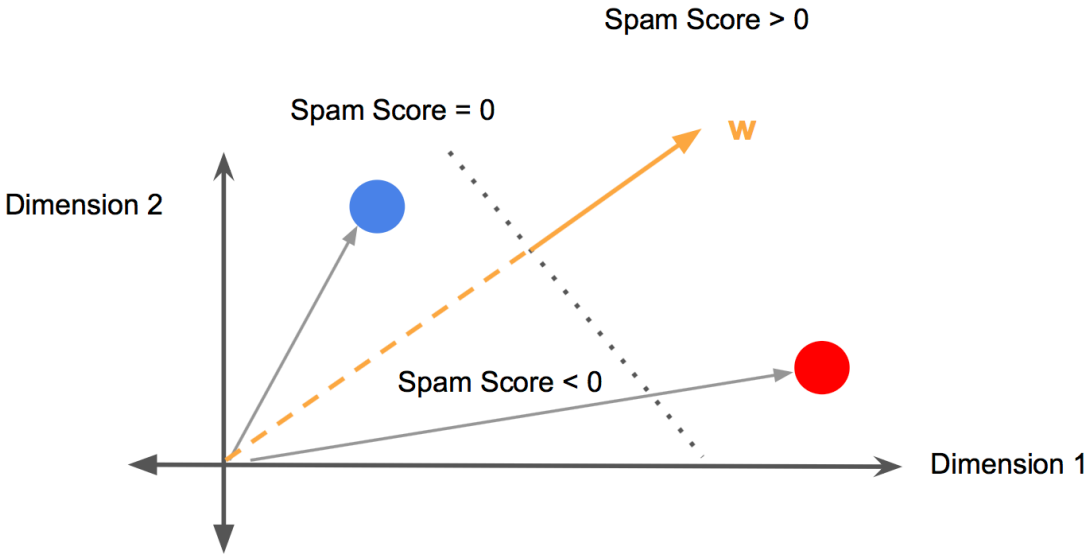
19 Vector Inner Product: Similarity of Vectors



$$\langle \mathbf{v}, \mathbf{w} \rangle = \mathbf{v}^\top \mathbf{w} = \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

$$= \sum_i^D v_i w_i$$

20 Spam Classification as Vector Operation



21 Training Machine Learning Models

- After feature extraction, many data sets can be modeled with the same methods
- The usual procedure:
  - Pick a Machine Learning model (Neural Network, Linear Regression, …)
  - Pick a *loss function* (mean squared error, accuracy, …)
  - Pick a *regularizer* (controls the complexity of a model, more on that later)
  - Pick an *optimizer* (Usually some gradient descent version)

– Minimize the weighted sum of loss and regularizer on some training data
– Evaluate the loss function on some independent test data (cross-validation)

22 Objective Functions of ML Models

Most ML models have objective functions like

$$e(w) = l(x, w) + r(w)$$

where $l(x, w)$ is called **loss** and measures the prediction quality and $r(w)$ is called **regularizer** and measures the complexity of the model

A simple loss function would be

$$||y - \hat{y}||_2^2$$

where $\hat{y}$ is some prediction of some (target) variable

A simple regularizer would be the L_2 or euclidean norm of w

$$||w||_2^2$$

meaning we penalize models with large coefficients in w

23 Minimizing Mathematical Functions: Gradient Descent

Objective functions can be minimized by **gradient descent**

$$w_{t+1} \leftarrow w_t - \eta \nabla e(w)$$

where

- w are the parameters to be optimized
- t is the iteration
- η is the learning rate
- $\nabla e(w)$ is the gradient of the objective function with respect to w

23.1 Derivatives of Univariate Functions

Function $f(x)$	Derivative $f'(x)$
x^n	nx^{n-1}
$cf(x)$	$cf'(x)$
...	...

23.2 Derivatives of Vector Values Functions

For vector values functions $f(\mathbf{x}) : R^D \rightarrow R^U$

the gradient $\nabla f(\mathbf{x})$ at position $\mathbf{x}$ is

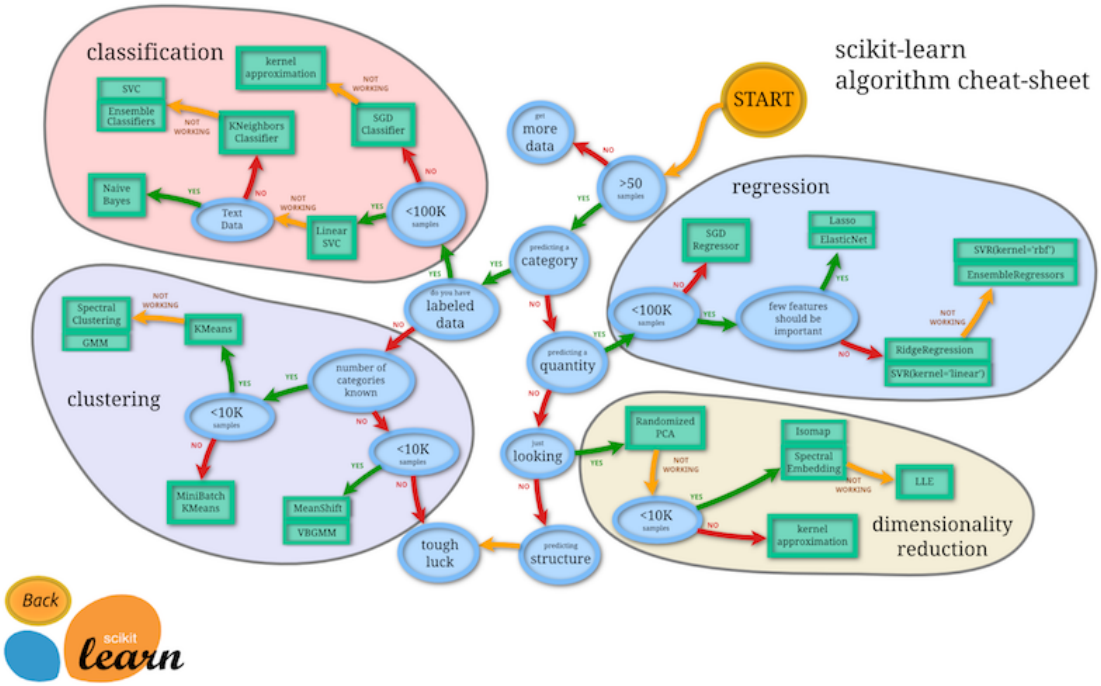
$\nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, ...\right]$

Function $f(\mathbf{x})$	Derivative $\nabla f(\mathbf{x})$
$\mathbf{x}^\top \mathbf{w}$	$\mathbf{w}$
$\mathbf{x}^\top \mathbf{x}$	$2\mathbf{x}$
$\mathbf{x}^\top A$	A
...	...

24 Types of Machine Learning Models

- Unsupervised Learning:
  - Learns to reconstruct data
  - Used to find patterns to focus on or to discard
  - Does not require labels
  - Examples: PCA, Autoencoder Neural Networks, Clustering
- Supervised learning:
  - Learns to predict target variable
  - Examples for continuous target variables (Regression): Linear Regression, Kernel Regression, Neural Networks
  - Examples for categorical target variables (Classification): Logistic Regression, Support Vector Machines, Neural Networks

25 Choosing Machine Learning Models



Taken from sklearn, figure by Andreas Mueller.

26 Some Challenges of Machine Learning

- ML systems don't require rules, just data
  - Data implicitly defines the rules to learn
  - Scales well, provided there is training data
- ML systems can be difficult:
  - Not all parameters can be easily optimized (e.g. model *hyperparameters*, such as: which model or features should I use)
  - Often training data is not available
  - Overfitting to training data
  - Behaviour changes with data quality
  - Difficult to test

27 The Perceptron Learning Algorithm

28 The Perceptron Learning Algorithm

- Frank Rosenblatt was an American Psychologist
- He invented the Perceptron Algorithm - the first neural network
- Perceptrons were shortly afterwards criticized for their simplicity
- Took until the 80ies until they were rediscovered. And then again in 2012.

29 The Perceptron Learning Algorithm

Perceptron error $\mathcal{E}_{\mathcal{P}}$ is a function of the weights $\mathbf{w}$

$$\underset{w}{\text{argmin}} \qquad \left(\mathcal{E}_{\mathcal{P}}(\mathbf{w}) = -\sum_{m \in \mathcal{M}} \mathbf{w}^{\top} \mathbf{x}_m y_m \right) \tag{4}$$

where $\mathcal{M}$ denotes the index set of all *misclassified* data $\mathbf{x}_m$

Eq. 4 can be minimized *iteratively*
using **stochastic gradient descent**
[Bottou, 2010; Robbins and Monro, 1951]

1. Initialize $\mathbf{w}^{\text{old}}$ (randomly, $1/n$, ...)
2. While there are misclassified data points $\mathbf{x}_m$
   Pick a random misclassified data point $\mathbf{x}_m$

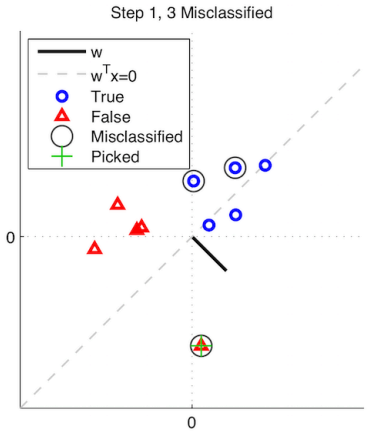
$$\mathbf{w}^{\text{new}} \leftarrow \mathbf{w}^{\text{old}} - \eta \nabla \mathcal{E}_{\mathcal{P}}(\mathbf{w}) = \mathbf{w}^{\text{old}} + \eta \mathbf{x}_m y_m \tag{5}$$

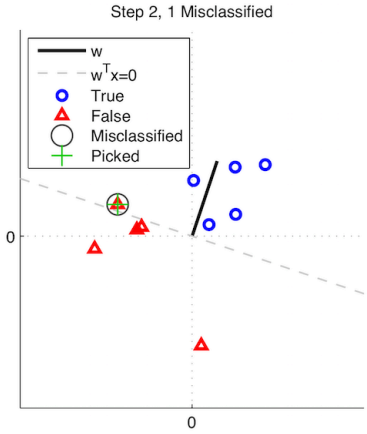
Algorithm 1 Perceptron learning

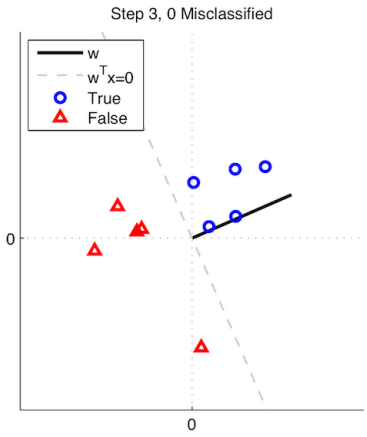
Require: Data $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$, $x_i \in \mathbb{R}^D$, Labels $y_1, \ldots, y_N \in \{-1, +1\}$, iterations *its*, learning rate η

Ensure: $\mathbf{w}$

$\mathbf{w} = \mathbb{1}_D/(D)$
for $it = 1, \ldots, its$ **do**
    # Pick a random example
    i = ceil(rand*N)
    # If this example is not correctly classified
    if $sign(\mathbf{w}^\top \mathbf{x}_i)$ $!= y_i$ **then**
        # Update weight vector
        $\mathbf{w} \leftarrow \mathbf{w} + \eta/it\ \mathbf{x}_i y_i$
    end if
end for



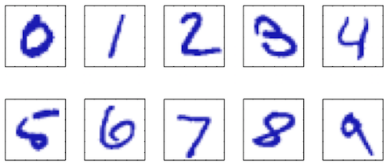




30 Application Example: Handwritten Digit Recognition

Since the beginning of AI, handwritten digit recognition was used as a benchmark test.

31 Application Example: Handwritten Digit Recognition

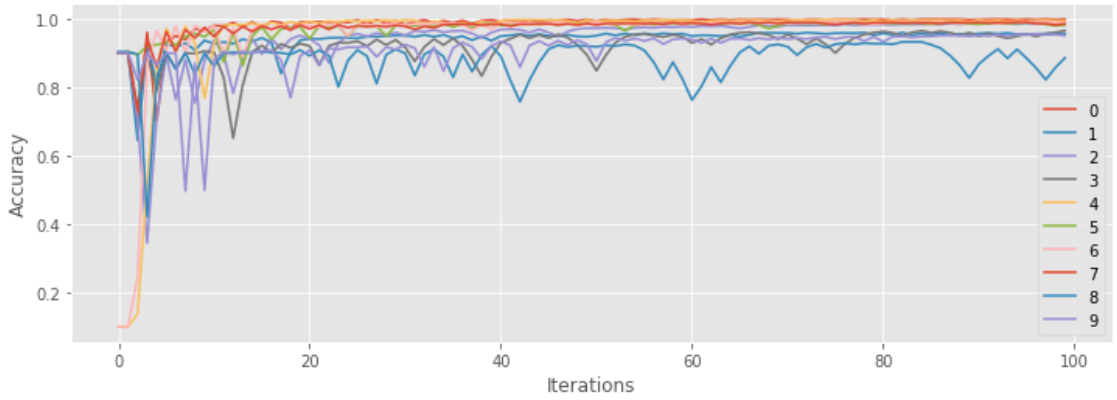


MNIST dataset: - 10 classes - Here: subsampled to 16 by 16 pixels, centered symbols - We train a binary perceptron classifier to distinguish one digit from all others

[3]:
```
import scipy as sp
import matplotlib.pyplot as plt
plt.style.use('ggplot')
from sklearn.datasets import load_digits
from assignment_perceptron_solution import perceptron_train

# load the digits dataset from sklearn
X, Y = load_digits(n_class=10, return_X_y=True)

accs = []
for digit_to_recognize in range(10):
    # transform the 10-class labels into binary form
```

```
    y = sp.sign((Y==digit_to_recognize)* 1.0 - .5)

    # train a perceptron to recognize the digit 5
    _, acc = perceptron_train(X,y)
    accs.append(acc)
```

[4]:
```
# plot the learning curve
plt.figure(figsize=[12,4])
plt.plot(sp.vstack(accs).T)
plt.xlabel("Iterations");plt.ylabel("Accuracy");
plt.legend(range(10));
```



32 Assignment 8

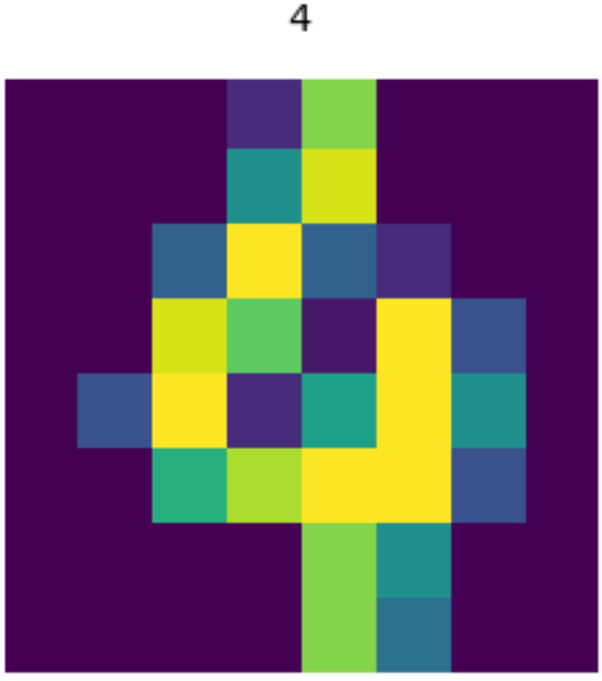
In this exercise you will implement a simple perceptron for handwritten digit recognition

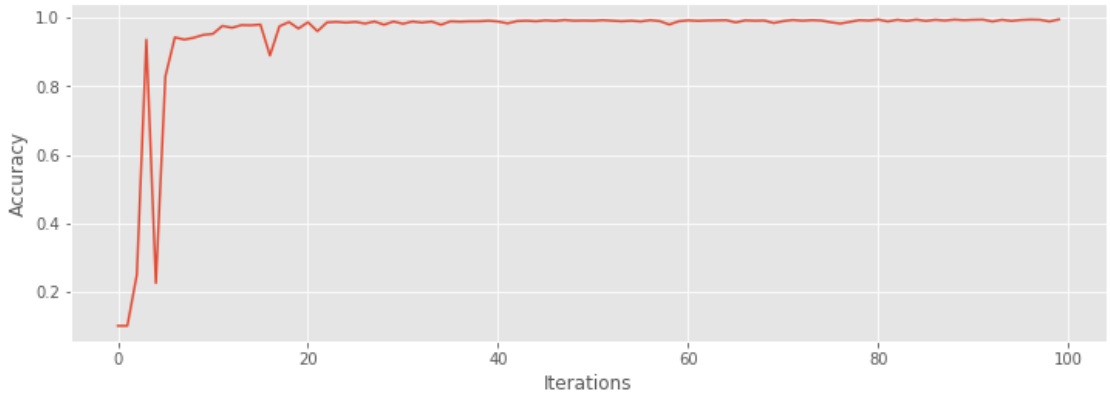
[5]:
```
import scipy as sp
from sklearn.datasets import load_digits

def assignment_07_06(digit_to_recognize=5):
    # load the usps digits dataset from sklearn repository
    n_example = 100
    X, Y = load_digits(n_class=10, return_X_y=True)
    plt.matshow(X[n_example,:].reshape(8,8));
    plt.xticks([]);plt.yticks([]);
    plt.title(Y[n_example])
    plt.savefig("usps_example.png")
    # transform the 10-class labels into binary form
    y = sp.sign((Y==digit_to_recognize)* 1.0 - .5)
    _, acc = perceptron_train(X,y)
    plt.figure(figsize=[12,4])
```

```
    plt.plot(acc)
    plt.xlabel("Iterations");plt.ylabel("Accuracy");
    plt.savefig("learning_curve.png")
```

```
def perceptron_train(X,Y,iterations=100,eta=.01):
    '''
    Trains a perceptron and returns the weights and learning curve
    '''
    acc = sp.zeros(iterations)
    # initialize weight vector
    weights = sp.random.randn(X.shape[1]) * 1e-5
    for it in sp.arange(iterations):
        # indices of misclassified data
        wrong = (sp.sign(X @ weights) != Y).nonzero()[0]
        if wrong.shape[0] > 0:
            # pick a random misclassified data point
            # rand_ex = ...
            # update weight vector
            # weights += ...
            # compute accuracy
            acc[it] = sp.double(sp.sum(sp.sign(X @ weights)==Y))/X.shape[0]
    # return weight vector and accuracy
    return weights,acc
```

[61]:
```
assignment_07_06()
```




1.0
0.8
0.6
0.4
0.2
Accuracy
0
20
40
60
80
100
Iterations

[ ]: