

EE 610, Selected Topics:  
Machine Learning Fundamentals

**Introduction**

Dr. W. D. Pan

Dept. of ECE  
U. of Alabama in Huntsville

# Topics

- Overview of Machine Learning (ML) and its relation to AI
- Categories of ML Techniques
  - Supervised Learning
    - Classification (KNN, Bayes Classifier, discriminant analysis, logistic regression)
    - Regression (linear and non-linear models, curve fitting)
    - Neural Networks (perceptron, multilayer network, backpropagation method, deep learning)
  - Unsupervised Learning
    - Cluster Analysis
    - Principal Component Analysis
  - Reinforcement Learning, Evolutionary Learning, etc.
- Math Required
  - Linear Algebra (singular value decomposition, QR decomposition, least square, ...)
  - Multivariate Calculus (gradient, Jacobian, Hessian, etc.)
  - Probability and Statistics (Bayes theorem, maximum likelihood estimation, ...)
  - Optimization Method (gradient decent, ...)
- Implementations
  - Matlab
    - Statistics and Machine Learning Toolbox, Deep Learning Toolbox, Image Processing Toolbox, ...
  - Python
    - Scikit-learn, Keras, Tensorflow
- Examples
  - Naïve Bayes classifier with the Iris dataset, CNN classifier with handwritten digits dataset.

# Example of ML

- Suppose that you have a website selling software that you've written.
- You want to make the website more personalized to the user, so you start to collect data about visitors, such as their computer type/operating system, web browser, the country that they live in, and the time of day they visited the website.
- Once you have collected a large set of such data, the problem you have is one of **prediction**: given the data you have, predict what the next person will buy.
- Because we know what the right answer are for some examples, we can give the **learner** these examples with the right answer.
- The reason this approach might work is that people who seem to be similar often act similarly.
- This is an example of **Supervised Learning**.

# Questions Addressed by ML

- Data Explosion: computers, sensors, smart devices, etc. capture and store massive amount of data every day. The challenge is to do something useful with the data.
- However, the size and complexity of these datasets mean that humans are unable to extract useful information from them.
- Computing machines can **learn** from the data to answer many questions
- This is why machine learning is becoming so popular.

# Learning

- Humans and other animals can display intelligent behaviors by learning from experience.
- By learning, we can adjust and adapt to new circumstances.
- Important parts of learning:
  - Remembering
  - Adapting
  - Generalizing.

# Learning Steps

- Remembering
  - recognizing that last time we were in this situation (saw this data)
  - we tried out some particular action (gave this output) and it worked (was correct), so we'll try it again
- Adapting
  - If it didn't work, we'll try something different.
- Generalizing
  - Recognizing similarity between different situations, so that things that applied in one place can be used in another.
  - This is what makes learning useful, because we can use our knowledge in lots of different places.

# Link to Artificial Intelligence

- We are interested in the most fundamental parts of intelligence — learning and adapting—and how we can model them in a computer.
- There has also been a lot of interest in making computers reason and deduce facts, which was the basis of most early Artificial Intelligence, and is sometimes known as *symbolic processing* because the computer manipulates symbols that reflect the environment.
- In contrast, machine learning methods are sometimes called *subsymbolic* because no symbols or symbolic manipulation are involved.

# Artificial Intelligence (AI)

- Using math and logic, a computer system simulates the reasoning that humans use to learn from new information and make decisions.
- An AI computer system makes predictions or takes actions based on patterns in existing data and can then learn from its errors to increase its accuracy.
- A mature AI processes new information extremely quickly and accurately, which makes it useful for complex scenarios such as self-driving cars, image recognition programs, and virtual assistants.
- Examples of AI
  - Self-driving cars, Bots and digital assistants, Recommendation engines, Spam filters, Smart home technology, Health data analysis, etc.



# Types of AI

- **Narrow**
  - Also called “weak AI” — refers to the ability of a computer system to perform a narrowly defined task better than a human can.
- **General**
  - sometimes called “strong AI” or “human-level AI” — refers to the ability of a computer system to outperform humans in any intellectual task.
- **Artificial super intelligence (ASI)**
  - the ability to outperform humans in almost every field, including scientific creativity, general wisdom, and social skills.

# ML achieves Narrow AI

- Machine learning is a process that computer systems follow to achieve artificial intelligence.
- It uses algorithms to identify patterns within data, and those patterns are then used to create a data model that can make predictions.
- Machine-learning models are trained on subsets of data.
- When the data that's used to train the model accurately represents the full dataset that will be analyzed, the algorithm calculates more accurate results.
- When the machine learning model has been trained well enough to perform its task quickly and accurately enough to be useful and trustworthy, it's achieved narrow AI.

# ML

- Machine learning is about making computers modify or adapt their actions (whether these actions are making predictions, or controlling a robot) so that these actions get more accurate.
- Accuracy is measured by how well the chosen actions reflect the correct ones.
- Generalization
  - Imagine that you are playing a game against a computer, you might beat it every time in the beginning, but after lots of games it starts beating you.
  - The computer can go on and use the same strategies against other players, so that it doesn't start from scratch with each new player -- a form of generalization.

# Computation Complexity

- Computation complexity of ML is particularly important because we might want to use some of the methods on very large datasets.
- Algorithms that have high degree of polynomial complexity in the size of the dataset will be a problem.
- The complexity is often broken into two parts:
  - the complexity of training, and
  - the complexity of applying the trained algorithm (testing).
- Training does not happen very often, and is not usually time critical, so it can take longer.
- However, we often want a decision about a test point quickly, and there are potentially lots of test points when an algorithm is in use, so testing needs to have low computational cost.

# Types of ML Algorithms

- Supervised learning
  - A training set of examples with the correct responses (targets) is provided and, based on this training set, the algorithm generalizes to respond correctly to all possible inputs. This is also called learning from exemplars.
- Unsupervised learning
  - Correct responses are not provided, but instead the algorithm tries to identify similarities between the inputs so that inputs that have something in common are categorized together.
  - The statistical approach to unsupervised learning is known as density estimation.

- Reinforcement learning
  - somewhere between supervised and unsupervised learning.
  - The algorithm gets told when the answer is wrong, but does not get told how to correct it.
  - It has to explore and try out different possibilities until it works out how to get the answer right.
  - Reinforcement learning is sometime called learning with a critic, because of this monitor scores the answer, but does not suggest improvements.

- Evolutionary learning
  - Biological evolution can be seen as a learning process: biological organisms adapt to improve their survival rates and chance of having offspring in their environment.
  - We can model this in a computer, using an idea of fitness, which corresponds to a score for how good the current solution is.

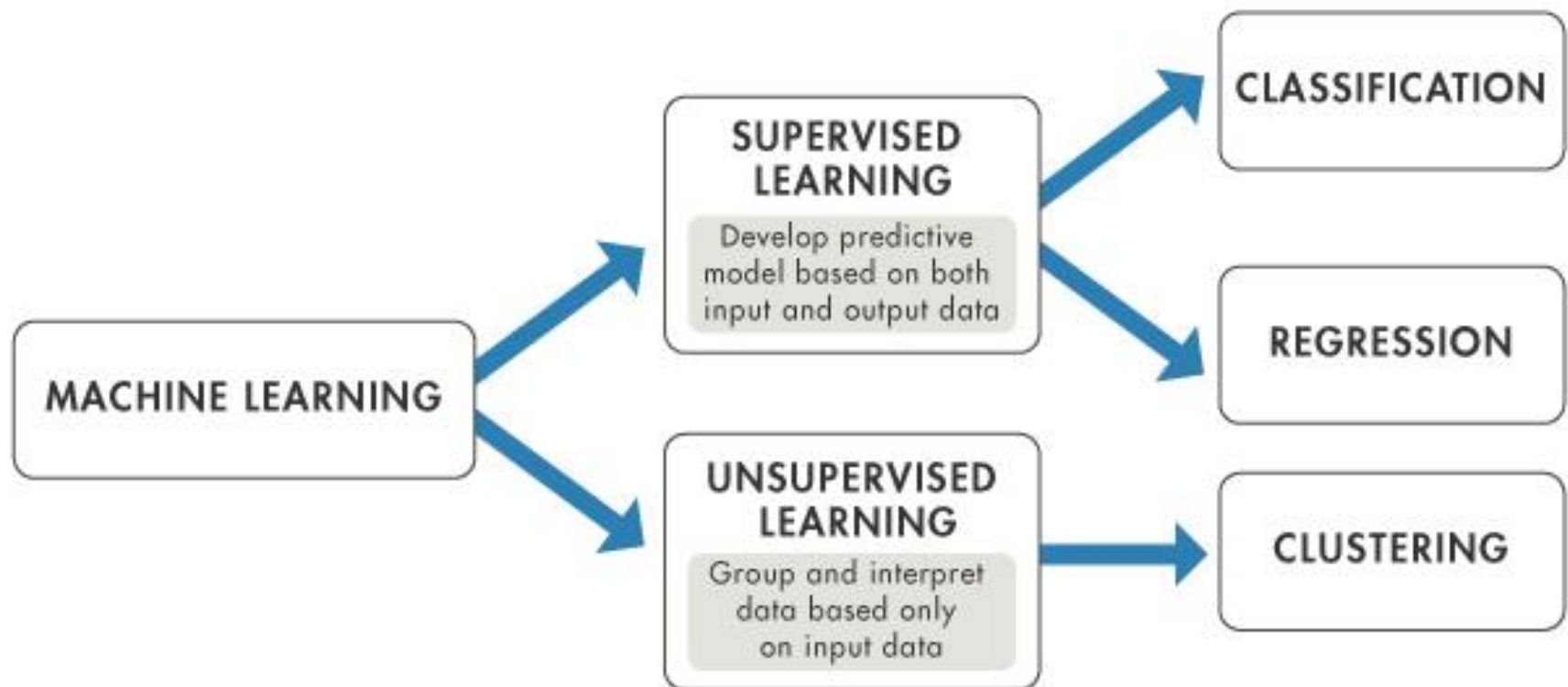
# Supervised Learning

- Classification
  - Taking input vectors and deciding which of  $N$  classes they belong to, based on training from exemplars of each class.
  - The classification problem is discrete — each example belongs to precisely one class, and the set of classes covers the whole possible output space.
- Regression
  - Output has continuous values. The goal to predict a value as close to the actual output value.
  - Evaluation is done by calculating the error value. The smaller the error the greater the accuracy of the regression model.
  - Example applications:
    - predicting changes in temperature, fluctuations in power demand, electricity load forecasting, algorithmic trading, etc.

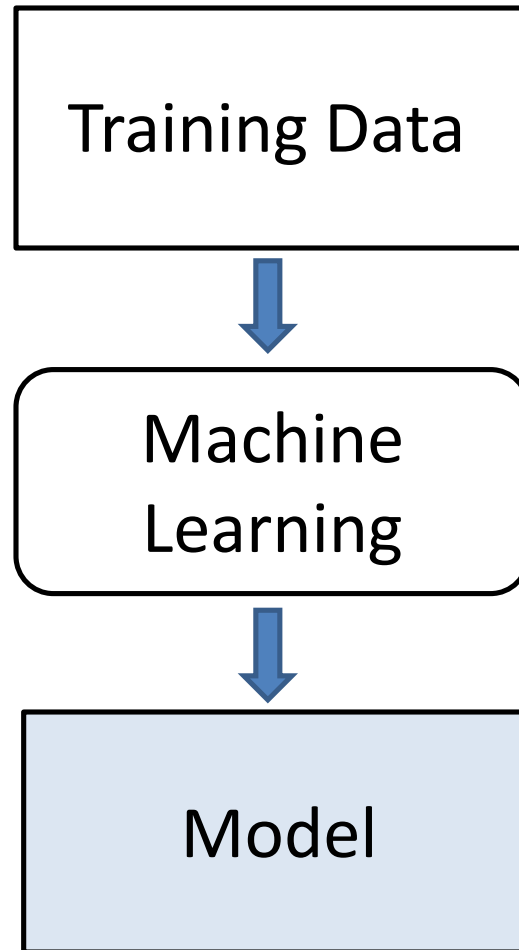


# Unsupervised Learning

- Unsupervised learning finds hidden patterns or intrinsic structures in data.
- It is used to draw inferences from datasets consisting of input data without labeled responses.
- Clustering is the most common unsupervised learning technique. It is used for exploratory data analysis to find hidden patterns or groupings in data.
- Applications for clustering include gene sequence analysis, market research, and object recognition.



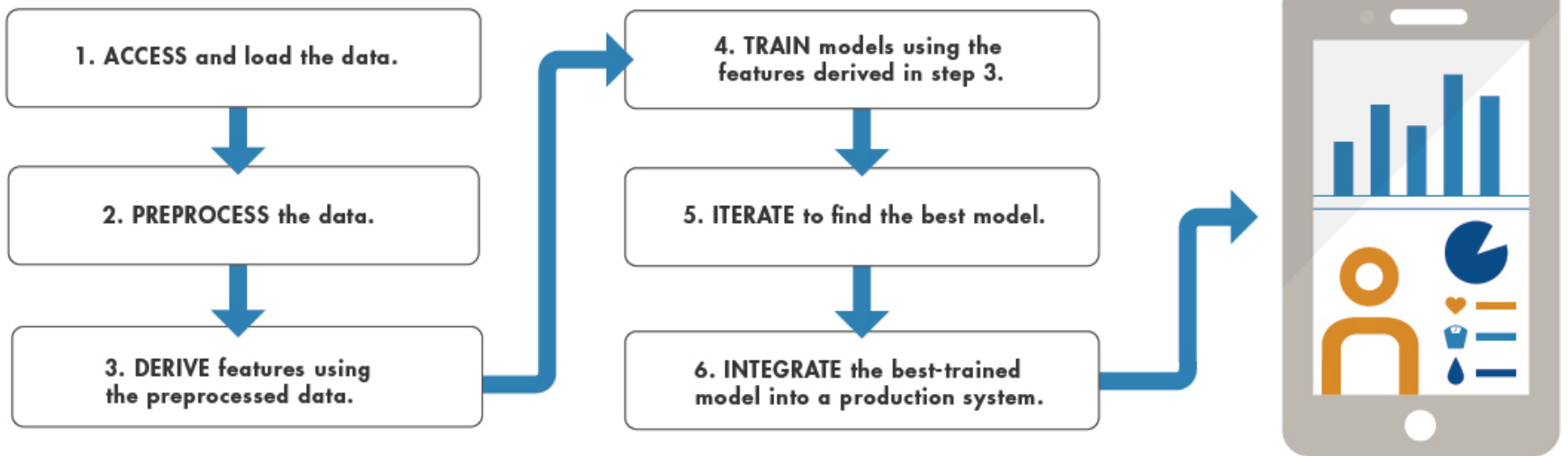
# ML Process



- Data Collection and Preparation
  - For supervised learning, target data (labels) are also needed, which can require the involvement of experts in the relevant field and significant investments of time.
- Feature Selection
  - Identifying the features that are most useful for the problem under examination.
  - This invariably requires prior knowledge of the problem and the data.

- Algorithm Choice
  - Given the dataset, the knowledge of the underlying principles of each algorithm and examples of their use is needed.
- Parameter and Model Selection
  - For many of the algorithms there are parameters that have to be set manually, or that require experimentation to identify appropriate values.

- Training
  - Given the dataset, algorithm, and parameters, training is the use of computational resources in order to build a model of the data in order to predict the outputs on new data.
- Evaluation (testing)
  - Before a system can be deployed it needs to be tested and evaluated for accuracy on data that it was not trained on.
  - This can often include a comparison with human experts in the field, and the selection of appropriate metrics for this comparison.



# Iris Flower Dataset

Fisher's iris dataset consists of 150 samples of iris flower measurements (cm)

- sepal length
- sepal width
- petal length
- petal width

There are three species, with 50 samples/specie.



Iris setosa



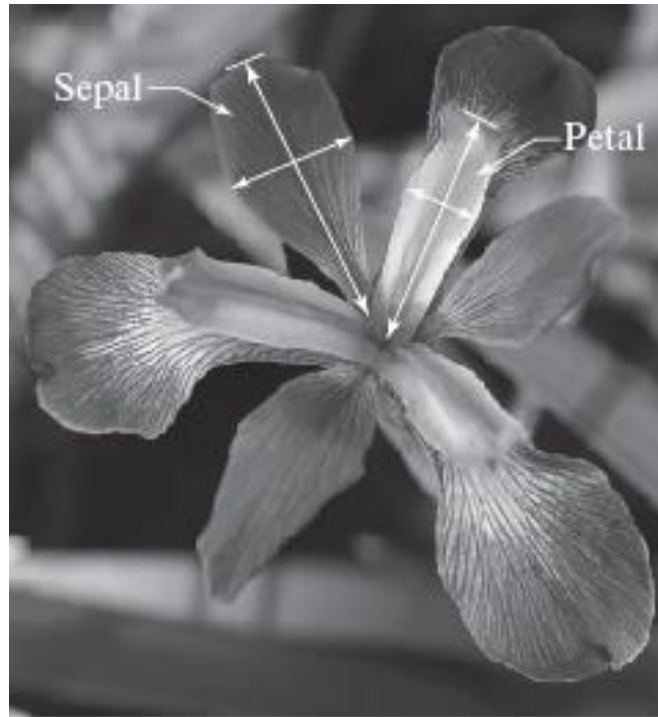
Iris versicolor



Iris virginica



# Feature Vector (Predictors)



$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$x_1$  = Petal width  
 $x_2$  = Petal length  
 $x_3$  = Sepal width  
 $x_4$  = Sepal length

Classes	3
Samples per class	50
Samples total	150
Dimensionality	4
Features	real, positive

# Dataset Content

[https://en.wikipedia.org/wiki/Iris\\_flower\\_data\\_set#Data\\_set](https://en.wikipedia.org/wiki/Iris_flower_data_set#Data_set)

```
>> load fisheriris
```

```
    meas          150x4 double
    species 150x1 cell
```

```
>> meas(1,:)
```

```
ans =
```

```
5.1000  3.5000  1.4000  0.2000
```

```
>> species(1)
```

```
ans =
```

```
1x1 cell
```

```
array
```

```
    {'setosa'}
```

```
>>
```

```
species(51)
```

```
ans =
```

```
1x1 cell
```

```
array
```

```
    {'versicolor'}
```

```
>>
```

```
species(101)
```

```
ans =
```

```
1x1 cell
```

```
array
```

```
    {'virginica'}
```

# Matlab (Statistical and Machine Learning Toolbox)

```
>> load fisheriris
>> X = meas;
>> Y = species;
>> Mdl = fitcnb(X,Y)    % Train a naïve Bayes classifier
Mdl =
  ClassificationNaiveBayes
      ResponseName: 'Y'
      ...
  ClassNames: {'setosa' 'versicolor' 'virginica'}
  NumObservations: 150
  DistributionNames: {'normal' 'normal' 'normal' 'normal'}

>> L = resubLoss(Mdl)
L =
  0.0400
```

The naive Bayes classifier misclassifies 4% of the training observations.

# Prediction (Resubstitution)

```
>> X = meas;  
>> Y = species;  
>> Mdl = fitcnb (X, Y);           % Train the classifier with (X, Y)  
  
>> label = predict (Mdl, X);     % Predict using the same training data X  
>> diff = strcmp (label, Y);  
>> length(find(diff ==0))         % Number of prediction errors  
ans =  
    6  
>> 6/150  
ans =  
    0.0400
```

# *scikit-learn* (sklearn): ML in Python

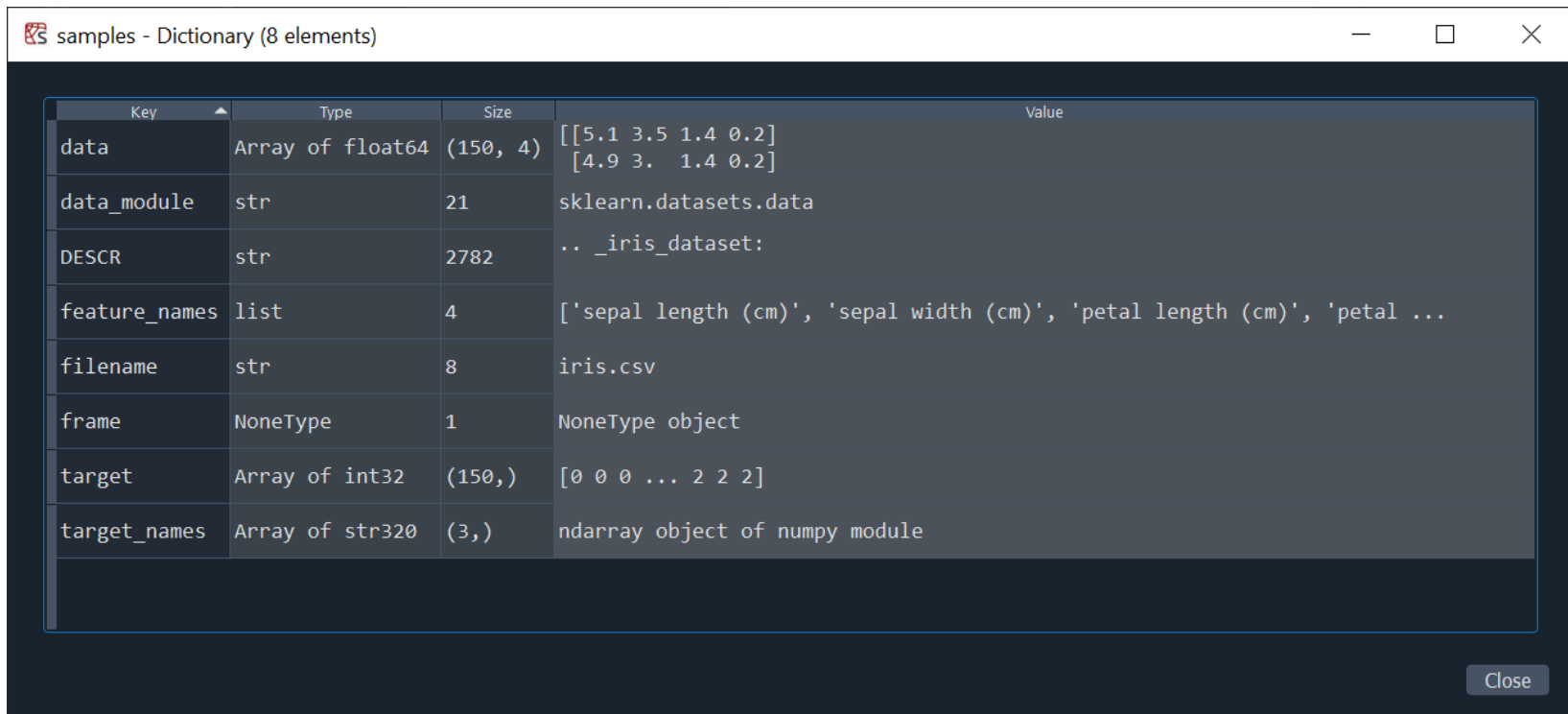
- Python IDE Installation
  - Transition from Matlab to Python
    - <https://realpython.com/matlab-vs-python/>
  - Anaconda Data Science Platform:  
<https://www.anaconda.com>
  - Installation includes Python, NumPy, and many other commonly used packages for scientific computing and data science.
  - Includes the Spyder (with IPython, or interactive Python) development environment that supports advanced editing, analysis, debugging, etc.
- scikit-learn is an open source machine learning library that supports supervised and unsupervised learning.
  - <https://scikit-learn.org/stable/index.html>

# *scikit-learn* (sklearn): ML in Python

- Transition from Matlab to Python/Numpy
  - <https://realpython.com/matlab-vs-python/>
  - <https://docs.python.org/3.9/tutorial/>
  - <https://numpy.org/doc/stable/user/quickstart.html>
- Python IDE Installation
  - Anaconda Data Science Platform:  
<https://www.anaconda.com>
  - Installation includes Python, NumPy, and many other commonly used packages for scientific computing and data science.
  - Includes the Spyder (with IPython, or interactive Python) development environment that supports advanced editing, analysis, debugging, etc.
- scikit-learn is an open source machine learning library that supports supervised and unsupervised learning.
  - <https://scikit-learn.org/stable/index.html>

# Fisher Iris Dataset

```
>>> from sklearn.datasets import load_iris
>>> samples = load_iris()
>>> samples.target_names
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```



Key	Type	Size	Value
data	Array of float64	(150, 4)	[[5.1 3.5 1.4 0.2] [4.9 3. 1.4 0.2]
data_module	str	21	sklearn.datasets.data
DESCR	str	2782	.. _iris_dataset:
feature_names	list	4	['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal ...
filename	str	8	iris.csv
frame	NoneType	1	NoneType object
target	Array of int32	(150,)	[0 0 0 ... 2 2 2]
target_names	Array of str320	(3,)	ndarray object of numpy module

Close

# sklearn

```
# 'naive_bayes_demo.py'  
  
from sklearn.naive_bayes import GaussianNB  
clf = GaussianNB()  
  
from sklearn.datasets import load_iris  
train_samples = load_iris()  
X = train_samples.data  
Y = train_samples.target  
  
clf.fit(X, Y)  
  
clf.score(X, Y)  
  
# Verify the classification accuracy  
Y_pred = clf.predict(X)  
  
num_correct = np.sum(Y == Y_pred)  
num_sample = np.size(Y_pred)  
num_correct/num_sample
```

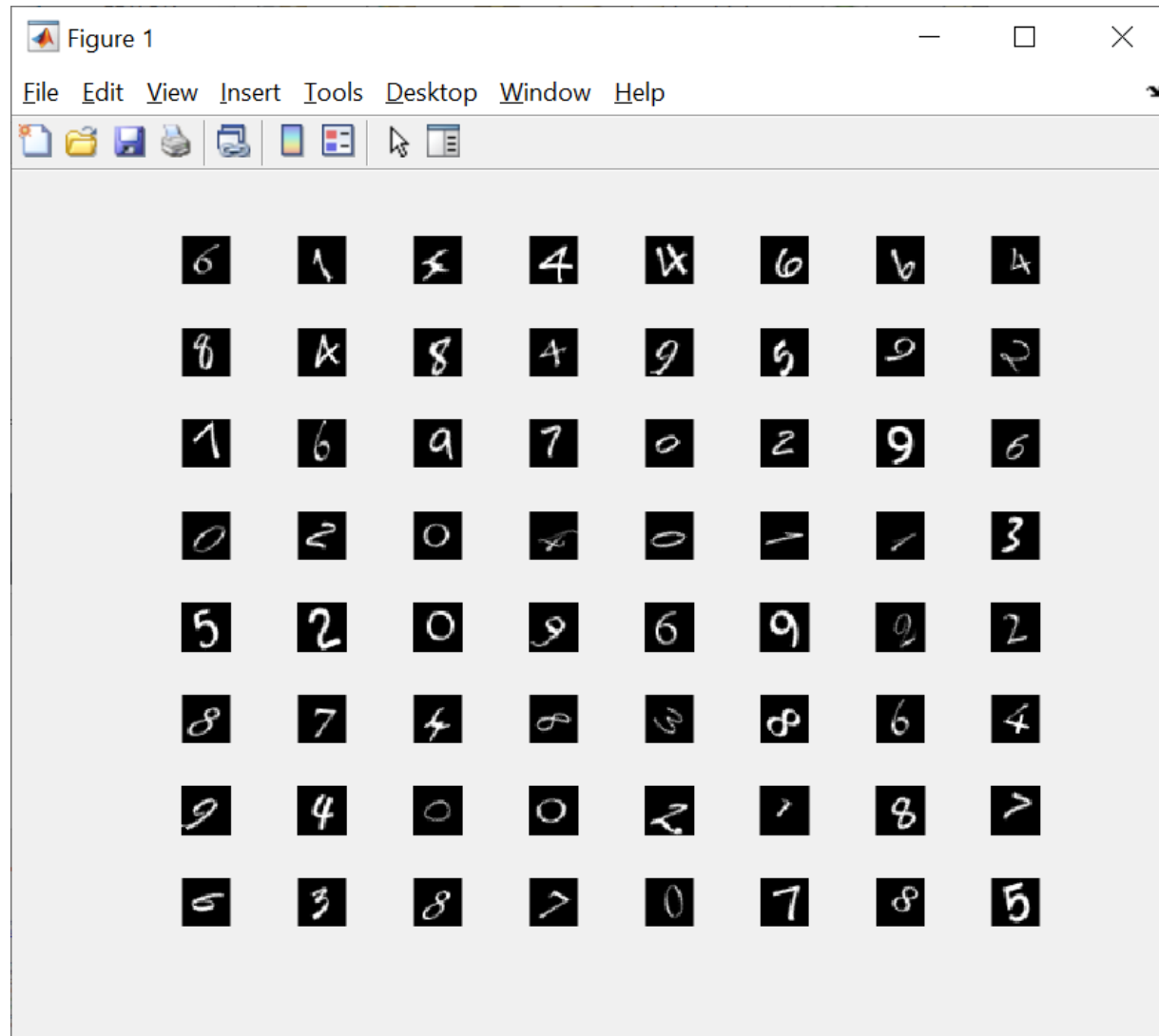


# Example: CNN

- Intro. to Deep Learning
  - [What are Convolutional Neural Networks](#) (CNNs)?
- Dataset
  - [XTrain, YTrain] = digitTrain4DArrayData; loads the digit training set as 4-D array data.
  - XTrain is a 28-by-28-by-1-by-5000 array, where:
    - 28 is the height and width of the images.
    - 1 is the number of channels.
    - 5000 is the number of synthetic images of handwritten digits (0, 1, 2, ..., 9).
  - YTrain is a categorical vector containing the labels for each observation.

```
>> size(XTrain)
ans =
    28    28     1   5000
```

```
>> size(YTrain)
ans =
    5000     1
```



```
perm = randperm(5000, 64);  
for i = 1:64  
    subplot(8,8,i);  
    imshow(XTrain(:, :, :, perm(i)));  
end
```

# Dataset Partitioning

```
>> size(XTrain)
ans =
    28    28     1   5000
```

```
% Set aside 1000 of the images for testing
after the network is trained
```

```
idx = randperm(size(XTrain,4),1000);
XTest = XTrain(:,:, :,idx);
XTrain(:,:, :,idx) = [];
YTest = YTrain(idx);
YTrain(idx) = [];
```

```
>> size(XTrain)
ans =
    28    28     1   4000
```

```
% Set aside 1000 of the training images
for validation during training.
```

```
idx = randperm(size(XTrain,4),1000);
XValidation = XTrain(:,:, :,idx);
XTrain(:,:, :,idx) = [];
YValidation = YTrain(idx);
YTrain(idx) = [];
```

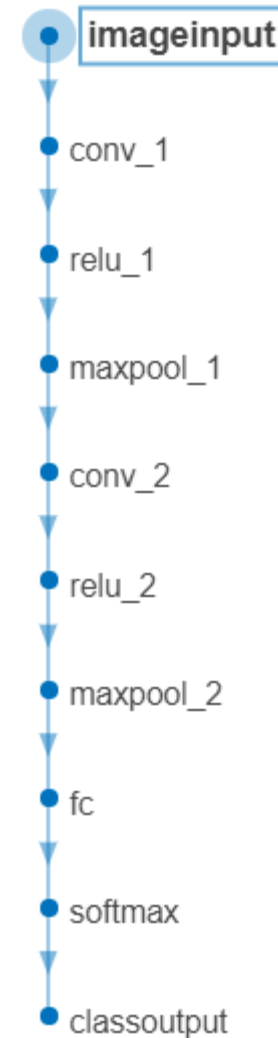
```
>> size(XTrain)
ans =
    28    28     1   3000
```

**Partitioning:** Training (3000), Validation (1000), Testing (1000)

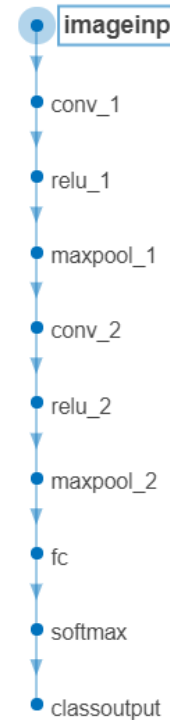
# Network Structure and Training Options

```
layers = [  
    imageInputLayer([28 28 1])  
  
    convolution2dLayer(3,32)  
    reluLayer  
    maxPooling2dLayer(2, 'Stride',2)  
  
    convolution2dLayer(3,64)  
    reluLayer  
    maxPooling2dLayer(2, 'Stride',2)  
  
    fullyConnectedLayer(10)  
    softmaxLayer  
    classificationLayer];
```

```
options = trainingOptions('adam', ...  
    'InitialLearnRate',0.01, ...  
    'MaxEpochs',5, ...  
    'Shuffle','every-epoch', ...  
    'MiniBatchSize',64, ...  
    'ValidationData', {XValidation,YValidation}, ...  
    'ValidationFrequency', 20, ...  
    'Verbose',false, ...  
    'Plots','training-progress');
```



# Lots of Parameters to Learn!



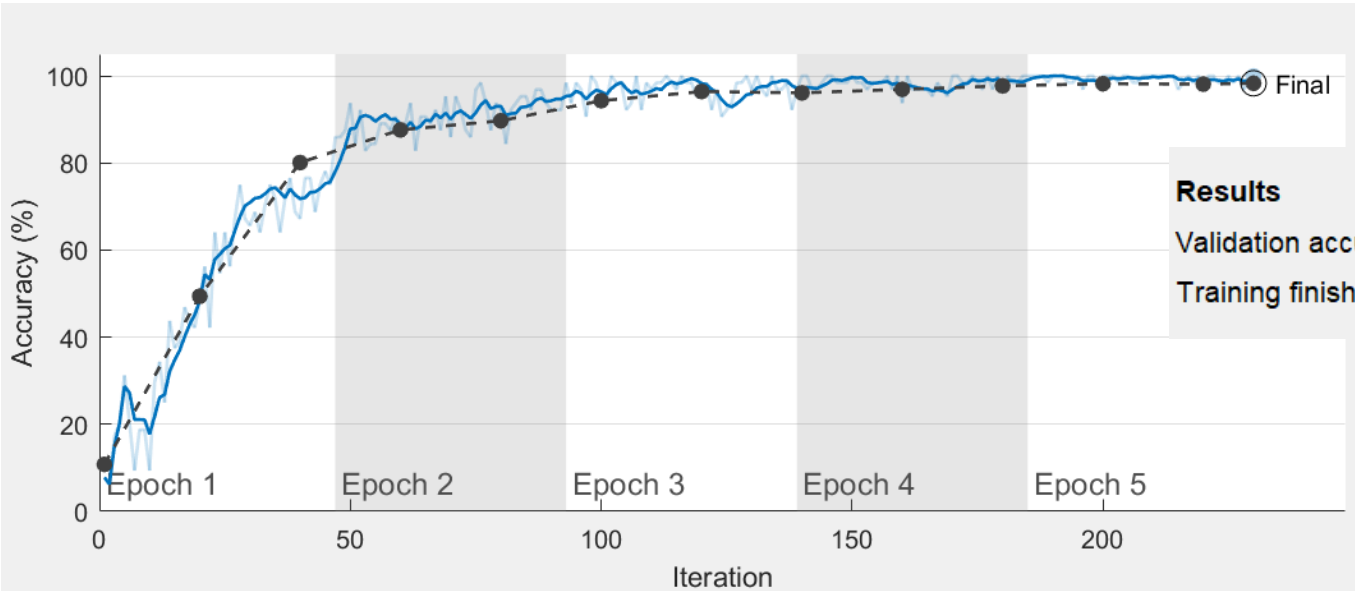
## ANALYSIS RESULT

	Name	Type	Activations	Learnables
1	<b>imageinput</b> 28x28x1 images with 'zerocenter' normalization	Image Input	28x28x1	-
2	<b>conv_1</b> 32 3x3x1 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	26x26x32	Weights 3x3x1x32 Bias 1x1x32
3	<b>relu_1</b> ReLU	ReLU	26x26x32	-
4	<b>maxpool_1</b> 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	13x13x32	-
5	<b>conv_2</b> 64 3x3x32 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	11x11x64	Weights 3x3x32x64 Bias 1x1x64
6	<b>relu_2</b> ReLU	ReLU	11x11x64	-
7	<b>maxpool_2</b> 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	5x5x64	-
8	<b>fc</b> 10 fully connected layer	Fully Connected	1x1x10	Weights 10x1600 Bias 10x1
9	<b>softmax</b> softmax	Softmax	1x1x10	-
10	<b>classoutput</b> crossentropyex with '0' and 9 other classes	Classification Output	-	-

```
>> net = trainNetwork (XTrain, YTrain, layers, options);  
>> analyzeNetwork (net)
```

# Training and Validation Progress

```
>> net = trainNetwork (XTrain, YTrain, layers, options);
```

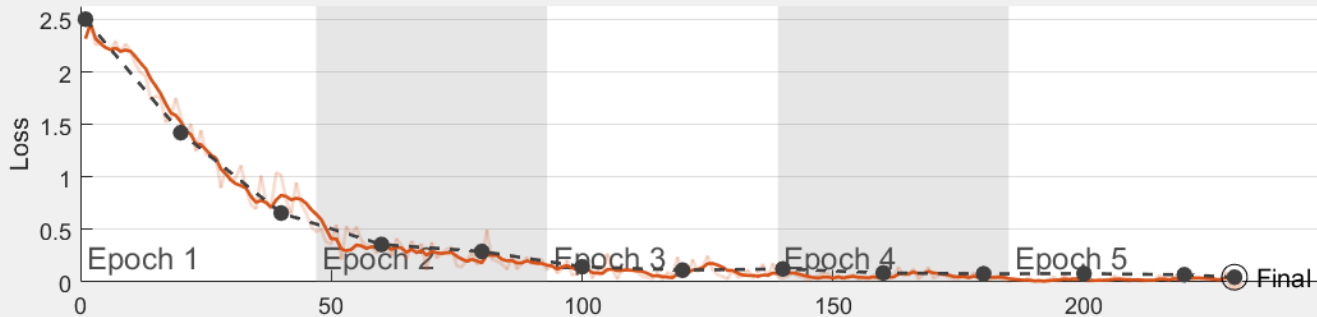


## Results

Validation accuracy: 98.30%  
Training finished: Reached final iteration

## Training Cycle

Epoch: 5 of 5  
Iteration: 230 of 230  
Iterations per epoch: 46  
Maximum iterations: 230



## Accuracy

- Training (smoothed)
- Training
- - ● - - Validation

## Loss

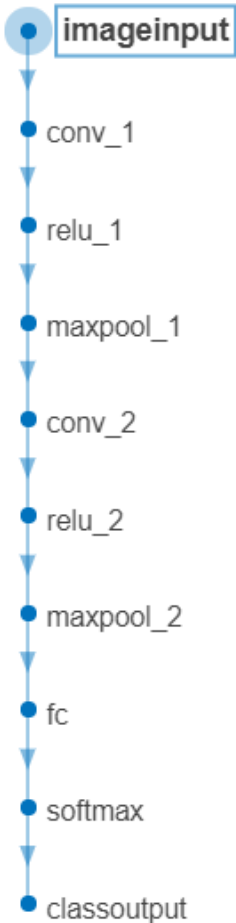
- Training (smoothed)
- Training
- - ● - - Validation

# Testing

```
>> YPred = classify (net, XTest);  
accuracy = sum(YPred == YTest)/numel(YPred)
```

```
accuracy =  
    0.9790
```

# Layers and Weights



```
>> net.Layers(1)
ans =
ImageInputLayer with properties:
```

```
    Name: 'imageinput'
  InputSize: [28 28 1]
```

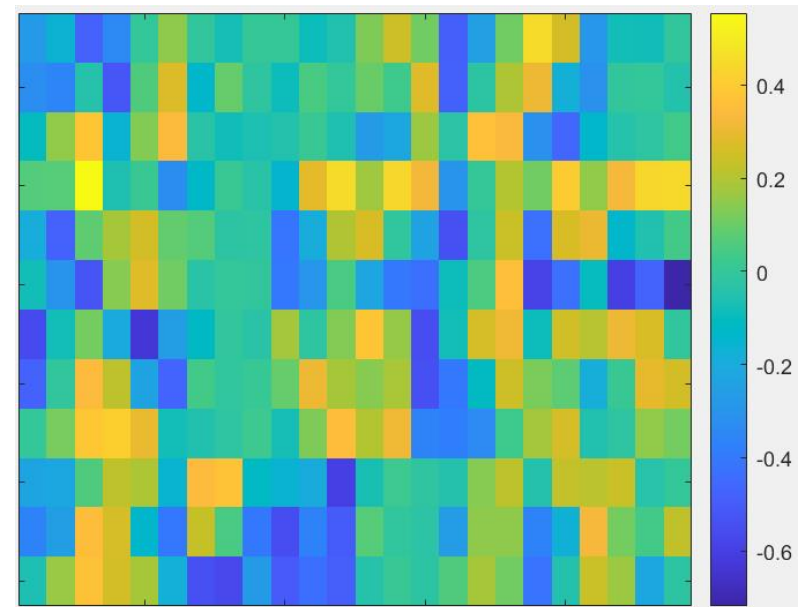
```
Hyperparameters
```

```
DataAugmentation: 'none'
Normalization: 'zerocenter'
NormalizationDimension: 'auto'
Mean: 0.0886
```

```
>> W2 = net.Layers(2).Weights;
```

```
>> whos W2
```

Name	Size	Bytes	Class	Attributes
W2	3x3x1x32	1152	single	



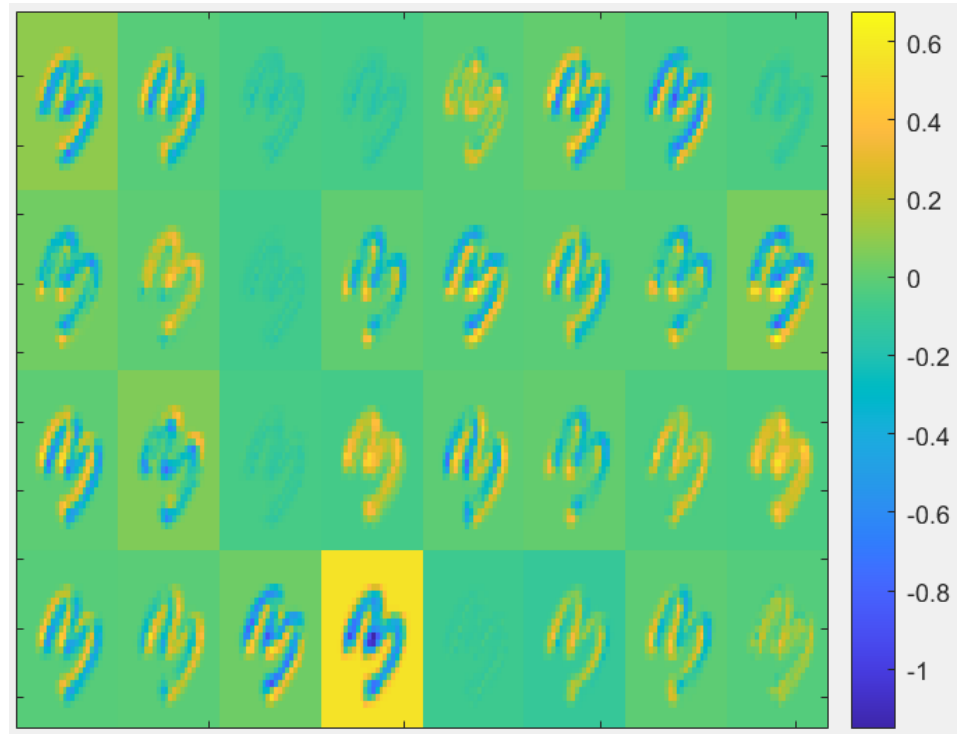
```
I_weight1 = imtile(W2, 'GridSize', [4 8]);
figure; imagesc(I_weight1); colorbar
```



# Visualization of Features and Outputs

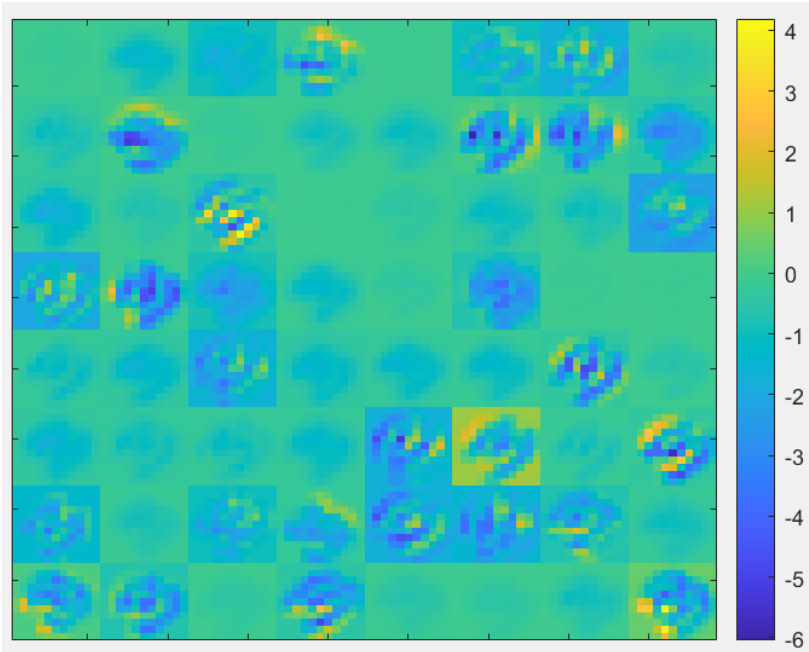


```
im = XTrain(:,:, :,1);  
figure; imshow(im)
```

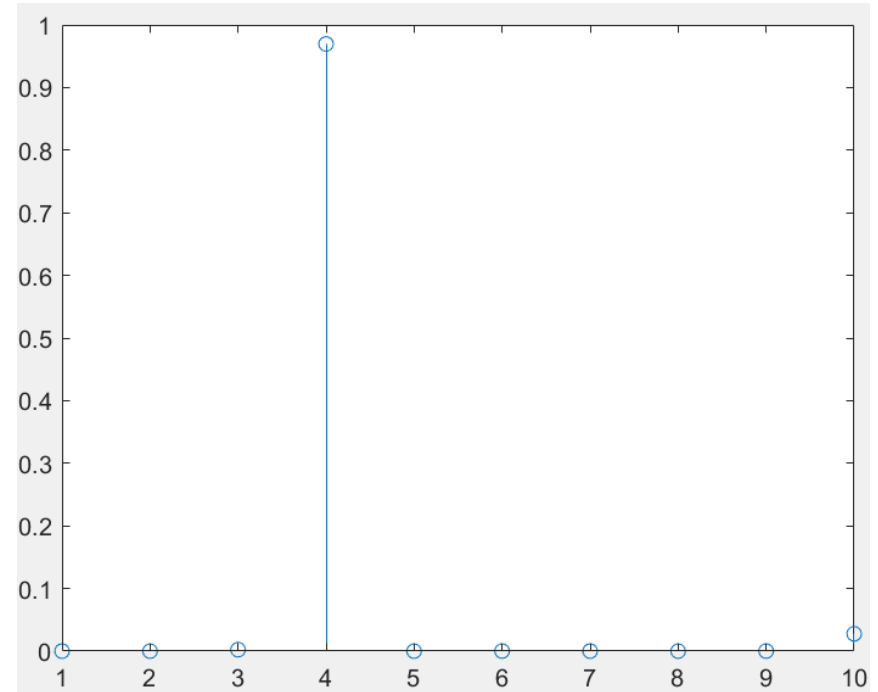


```
act1 = activations(net,im,'conv_1');  
I1 = imtile(act1, 'GridSize',[4 8]);  
figure; imagesc(I1); colorbar
```

# Probability of the Digits



```
act2 = activations(net,im,'conv_2');  
I2 = imtile(act2, 'GridSize',[8 8]);  
figure; imagesc(I2); colorbar
```



```
act9 = activations(net,im,'softmax');  
figure;  
stem(reshape(act9(1,1,:), 10, 1)); grid
```