2. Features, Targets, Scalers, Vectors, Matrices, Tensors and Norms

M.A.Z. Chowdhury and M.A. Oehlschlaeger

Department of Mechanical, Aerospace and Nuclear Engineering
Rensselaer Polytechnic Institute
Troy, New York

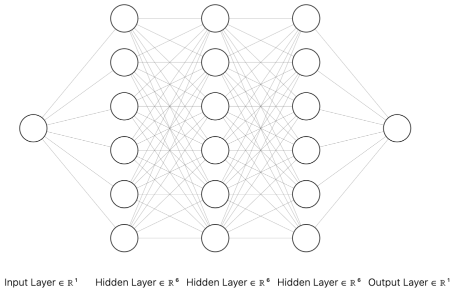
chowdm@rpi.edu
oehlsm@rpi.edu

"In mathematics, you don't understand things. You just get used to them."
- Mathematician John von Neumann to Chemist Felix Smith

MANE 4962 and 6962

Visualizing the neural net from lecture one

```
model = Sequential()
model.add(Dense(6, input_dim=1,
        activation='relu'))
model.add(Dense(6,activation='relu'))
model.add(Dense(6,activation='relu'))
model.add(Dense(1))
opt = optimizers.Adam(learning_rate=0.001)
mse = tf.keras.losses.MeanSquaredError(
    reduction=tf.keras.losses.Reduction.SUM)
model.compile(loss=mse, optimizer=opt)
model.fit(X,y,epochs=2000,batch_size=10, verbose=0)
predictions = model.predict(X)
```



- Sequential tells Python we want to add layers one after another.
- add adds a layer. For first time use, you must tell the size of the first layer or the input layer.
- Dense is the type of layer meaning it takes input from previous neurons and performs Matrix-vector operation.
- Activation means switching a neuron on or off in a particular way.
- Optimizer helps the network learn by methodically minimizing the error measure.
- Learning rate controls how fast the model will try to minimize the error measure.
- Compile put togethers everything and have the model prepared for training.
- Fit trains the net, predict asks the model to predict, epoch is the number of tries. verbose=0 tells the model to be quiet!
- Batch size is the number of data points to process at one go.

We'll learn a lot more about neural networks in later part of the course.
But first we need to learn some fundamentals.

- ☞ Please check LMS announcements for updates to lecture notes and notebooks.
- ☞ In case, you are struggling with Python, please let me know, or come to office hours, or schedule a meeting with me.
- ☞ Submitting the homework assignments as PDFs of Jupyter notebook would help me grade faster.
- ☞ For HW1, prob 6, plot the model output and the petal length to evaluate the predictions made by the network.

Quiz 1

Remember to take quiz 1, on LMS, 15 min. multi-choice on Jan 23, 2022.

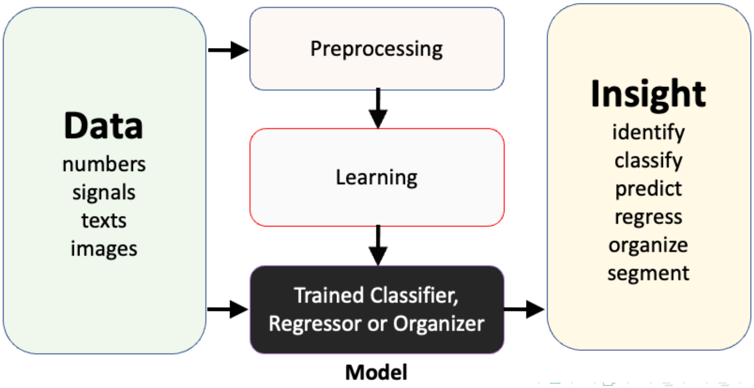
Why machine learning?

Tackle "unprogrammable" problems

Automate and optimize experiments/processes/products

Affordable computing power

Design interpretable predictive models



When is machine learning useful?

1. When experts are unable to explain their expertise or we want to mimic their expertise
   - ☞ Image/voice/artist recognition
   - ☞ Maneuvering a vehicle
2. Complex problems for which existing solutions require a lot of fine-tuning or long lists of rules or human expertise does not exist
   - ☞ Hazardous environments – Navigating on Mars, Space exploration
3. Solution needs to be adapted to particular cases
   - ☞ User biometrics
   - ☞ Patient specific treatments
   - ☞ Optimizing vehicular trajectories
4. Machine Learning system can adapt to new data in a fluctuating environment.
5. Getting insights about complex problems and large amounts of data so humans can learn better.
6. When cost of incorrect prediction is relatively low.

Types of Machine Learning Methods

Supervised

Data: (x, y)
x is data y is label/target

Goal: Learn function to map x to y

Unsupervised

Data: x
Just data no label

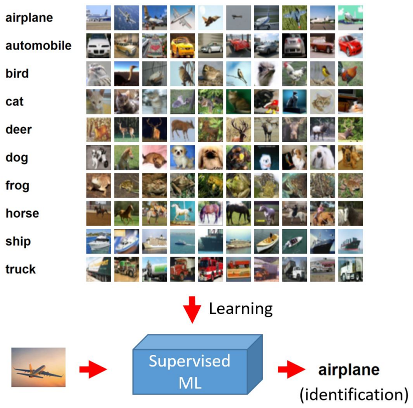
Goal: Learn underlying pattern

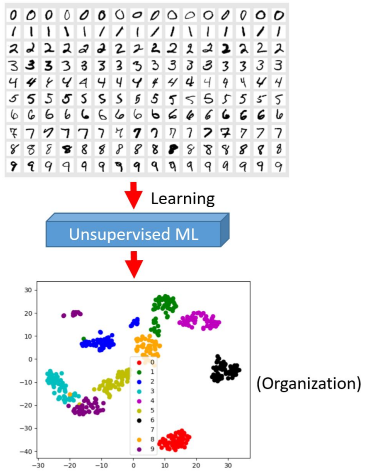
Reinforcement

Data: (s_t, a_t)
Agent interacting with environment

Goal: Maximize reward

There are more types of machine learning, but lets focus on these for the moment.

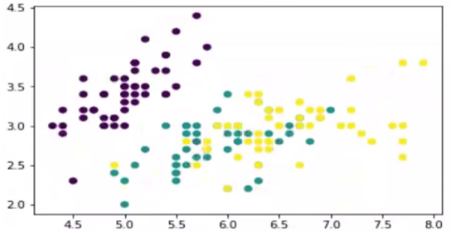




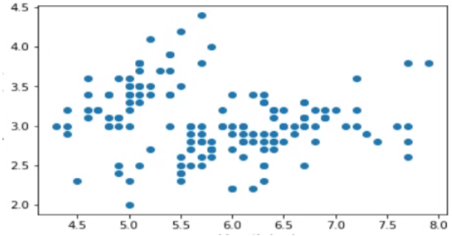
Dataset: CIFAR-10 and MNIST Digits

Our primary focus in this class

- Supervised learning
  1. data is annotated, tagged or labeled by human experts
  2. classifying signals,images, voices, videos
  3. optical character recognition
  4. predicting parameters of interest (pressure, temperature etc.)

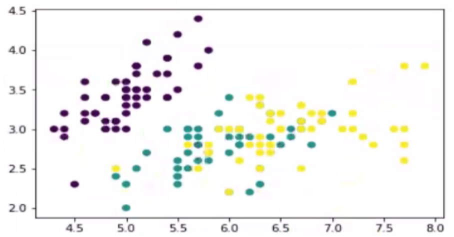


- Unsupervised learning
  1. data does not have labels/annotations/tags
  2. grouping customers
  3. detecting new diseases
  4. anomaly detection

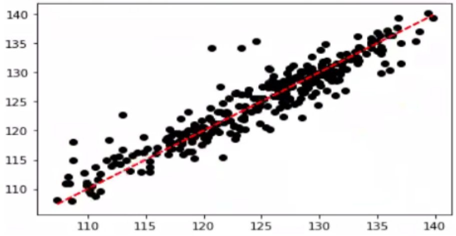


Two key supervised learning tasks

- Classification task
  1. data is splitted by the model showing classes or categories
  2. each data point have discrete class labels (can be integers or hot-encoded)
  3. can be binary, multi-class or multi-label
  4. Example: Classifying tumors



- Regression task
  1. data is fitted by model for making predictions
  2. each data point is associated with a real number target
  3. Example: predicting stock price, airfoil noise, covid cases etc.



In most ML-engineering problems, you need to figure out what is the “Learning Task”, i.e., what should be the inputs and outputs.

Features

- are n-dimensional vector. (Actually, tensor)
- quantify what we know about the problem we want to solve.
- measurable property or some characteristic of a problem or a phenomena.
- also called descriptor/predictor/attribute in different fields and applications.
- independent variable. Values are determined by data.
- are a representation of the data. You can represent the same data in many ways with different features.
- can be calculated/extracted from original features via process called feature extraction. Feature extraction derives new improved features.

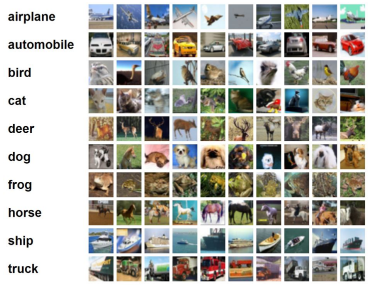
1. In the Iris example, how many features were there?
2. In the square root example, how many features were there?
3. In a digit classification problem using MNIST digits data, what will be the features and how many features will be there?

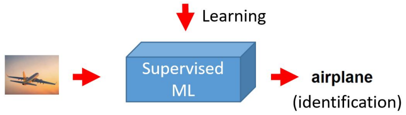
Targets

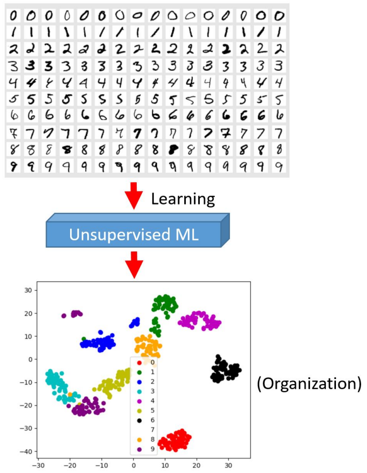
- are dependent variable.
- are the things we want to know.
- are discrete variables if we want to identify/categorize/classify something. (classification task)
- are continuous variables if we want to predict/regress/estimate something. (regression task)

1. In the Iris example, what was the target and how many targets were there?
2. In the square root example, what was the target and how many targets were there?
3. In a digit classification problem using MNIST digits data, what will be the target and how many targets will be there?

Features for image data







Dataset: CIFAR-10 and MNIST Digits

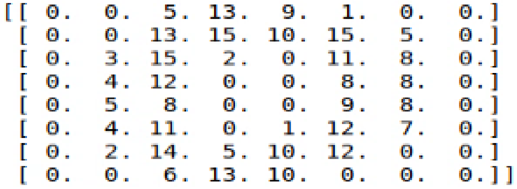
Let's say we wish to identify images of different types. How can we define features?

Image data as matrices

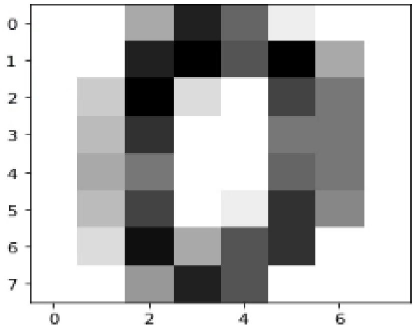
Images are represented as matrices and is a great form of data because we can visualize them.

```
from sklearn import datasets
# load the data
digits = datasets.load_digits()

images = digits.images
targets = digits.target

print("shape of image object", images.shape)
print("shape of target object", targets.shape)

# view a single image and its matrix representation

import matplotlib.pyplot as plt
plt.imshow(images[0], cmap=plt.cm.gray_r, interpolation="nearest")
print(images[0])
```

Digits dataset from sklearn

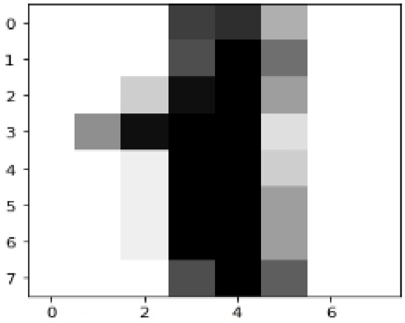


```
[[ 0.  0.  5. 13.  9.  1.  0.  0.]
 [ 0.  0. 13. 15. 10. 15.  5.  0.]
 [ 0.  3. 15.  2.  0. 11.  8.  0.]
 [ 0.  4. 12.  0.  0.  8.  8.  0.]
 [ 0.  5.  8.  0.  0.  9.  8.  0.]
 [ 0.  4. 11.  0.  1. 12.  7.  0.]
 [ 0.  2. 14.  5. 10. 12.  0.  0.]
 [ 0.  0.  6. 13. 10.  0.  0.  0.]]
```

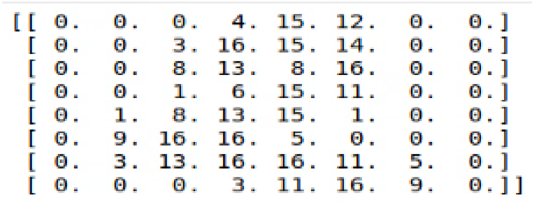


Digits dataset from sklearn

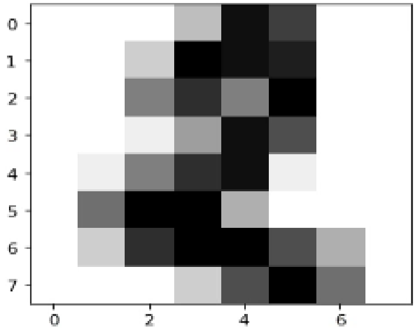
```
[[ 0.  0.  0. 12. 13.  5.  0.  0.]
 [ 0.  0.  0. 11. 16.  9.  0.  0.]
 [ 0.  0.  3. 15. 16.  6.  0.  0.]
 [ 0.  7. 15. 16. 16.  2.  0.  0.]
 [ 0.  0.  1. 16. 16.  3.  0.  0.]
 [ 0.  0.  1. 16. 16.  6.  0.  0.]
 [ 0.  0.  1. 16. 16.  6.  0.  0.]
 [ 0.  0.  0. 11. 16. 10.  0.  0.]]
```



Digits dataset from sklearn



```
[[ 0.  0.  0.  4. 15. 12.  0.  0.]
 [ 0.  0.  3. 16. 15. 14.  0.  0.]
 [ 0.  0.  8. 13.  8. 16.  0.  0.]
 [ 0.  0.  1.  6. 15. 11.  0.  0.]
 [ 0.  1.  8. 13. 15.  1.  0.  0.]
 [ 0.  9. 16. 16.  5.  0.  0.  0.]
 [ 0.  3. 13. 16. 16. 11.  5.  0.]
 [ 0.  0.  0.  3. 11. 16.  9.  0.]]
```

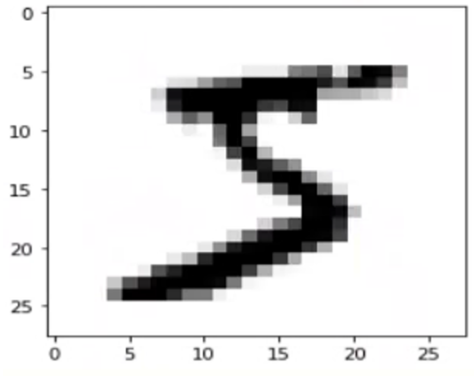


MNIST Digits images as matrices

MNIST image dataset is a standard dataset for studying ML algorithms.

```
from keras.datasets import mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()

print("training images shapes: ", x_train.shape)
print("testing images shapes: ", x_test.shape)
print("training targets shapes: ", y_train.shape)
print("testing targers shapes: ", y_test.shape)

plt.imshow(x_train[0], cmap=plt.cm.gray_r, interpolation="nearest")
print(x_train[0])
```

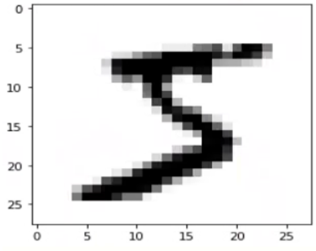
MNIST Digits images as matrices



```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  35 241 225 160 108   1
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119
   25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253
  150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252
  253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249
  253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253
  253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253
  250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201
   78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]]
```

Image data are represented by a matrix. Each pixel has an associated value depending on the image format (jpg, tiff, png etc.)

Defining Features



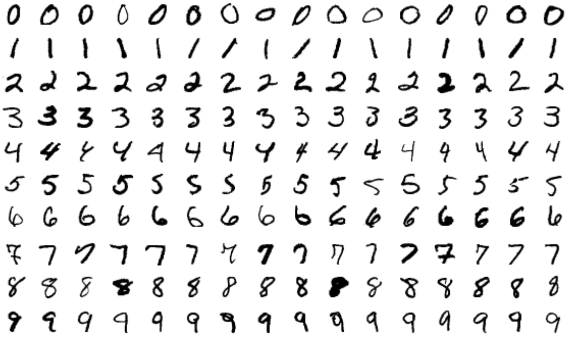
```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  35 241 225 160 108   1
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119
   25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253
  150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252
  253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249
  253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253
  253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253
  250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201
   78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]]
```

Let's define a feature vector $\underline{X}$, for $n = 1, 2, \ldots 784$,

$$\underline{X} = (x_1, x_2, \ldots x_n)$$

where x is the pixel intensity value for a single pixel.

Defining Features

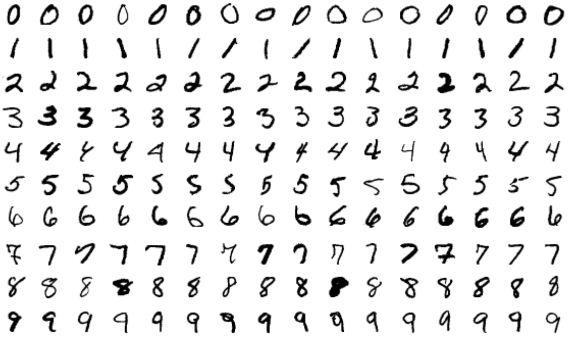


The first digit image will be represented by the feature vector

$$\underline{X_1} = (x_1^1, x_2^1, \ldots x_n^1)$$

where the superscript keeps track of the image each individual feature belongs to.

Defining Features



The second digit image will be represented by the feature vector

$$\underline{X_2} = (x_1^2, x_2^2, \ldots x_n^2)$$

where the superscript keeps track of the image each individual feature belongs to.

Defining Features

The N-th digit image will be represented by the feature vector

$$\underline{X_N} = (x_1^N, x_2^N, \ldots x_n^N)$$

where the superscript keeps track of the image each individual feature belongs to.

Data in matrix format

$$X_{input} = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_n^1 \\ x_1^2 & x_2^2 & \dots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^N & x_2^N & \dots & x_n^N \end{bmatrix}$$

$N \times n$ matrix

n is the number of features and N is the number of individual training examples (digit images).

For digits data, each row represents all the pixels in the image.

The good, the bad, and the ugly

Let’s analyze this ...

Good: We are using all information from the data we have.

Bad: We may be using irrelevant information. We are considering some pixels which are zero intensity and may not give us any information.

Ugly:All pixels are given equal weight or preference. The model requires data from every single pixel to work. So it Will be very data-hungry.
Will it work? Maybe.

Defining Features

Let's define a feature called average intensity,

$$\bar{A} = \frac{\sum \text{pixel-intensity}}{n_{pixel}}$$

Each digit image containing different digits have slightly different intensities at each pixel. Thus we will have a good feature to distinguish between each digit, right?

Answer: Not necessarily. We will only have one feature instead of 784, which is good for us to understand, but that single feature alone may not be sufficient for a ML model to learn. We may need to come up with even more features, for example, is the digit symmetric? or is parts of the image requires more pressure from the writer to generate.

Mathematics and Machine Learning

1. Input and the output to ML/DL models are both represented as vectors (tensors).
2. Mathematical knowledge helps us expresses qualitative inputs (such as a image, video, sound, color, text, signal etc) into a number or an arrangement of numbers.

Scaler, Vector, Matrix, Tensors

1. Scalar (0th order tensor) : Single number.
   Example : Learning rate, $\alpha \in \mathbb{R}$, number of hyperparameters $n_h \in \mathbb{N}$
   In engineering: density, speed
2. Vector (1st order tensor) : In ML, array of numbers. ($n \times 1$ matrix)
   Example : Input vector $x \in \mathbb{R}^n$
   In engineering: velocity, heat flux, lift force
3. Matrix (2nd order tensor) : In ML, 2-D array of numbers.
   Example : Weight matrix, $W \in \mathbb{R}^{m \times n}$
   In engineering: Nine Stress components τ_{ij} in continuum and fluid mechanics, components of thermal conductivity tensor
4. Tensors (3rd or higher order tensor) : In ML, array of numbers with dimensions greater than two
   Example : $A_{i,j,k}$ Example, video data, color images
   In engineering: Stress tensor $\tau = \tau_{ij}\hat{e}_i\hat{e}_j$ in continuum and fluid mechanics, thermal conductivity tensor.

Limitations and Advantages Tensors

1. Tensors are high dimensional. So machine learning algorithms must be able to process high dimensional data efficiently.
2. Tensors are much harder to comprehend compared to vectors due to their rank.
3. Tensor-tensor premultiplication and postmultiplication are different.
4. Converting tensors from one rank to another is much harder to do intuitively.
5. But vectors can be transformed into new vectors via multiplication with matrix.
6. Since vectors and matrix are both tensors so ultimately we are using reduced rank tensors to do machine learning and keeping the math simple. (I mean relatively)

Basic Tensor Operation

1. Addition, Broadcasting
   - Addition: $A_{ij} + B_{ij} = C_{ij}$
   - Broadcasting: $A_{ij} + B_j = C_{ij}$ [Adding a vector to a matrix by repeating the vector]
   - MATLAB and Python NumPy package automatically do broadcasting.
   - Vectorizing matrices speed up your code.
2. Multiplication :
   - Matrix Product : $C = AB$ or $c_{ij} = \sum_k a_{ik} b_{kj}$ [A B dimensions need to match.]
   - Dot Product : $\underline{a} \cdot \underline{b} = \alpha$ or $\alpha = a^T b = b^T a = \sum_i a_i b_i$
   - Hadamard Product : $C = A \odot B$ (Elementwise multiply)
3. Transpose : $B = A^T$ or $b_{ij} = a_{ji}$
4. Inverse :
   - $I = AA^{-1} = A^{-1}A$
   - A is a square matrix
   - Nonsquare matrix have pseudoinverse
   - Some matrices may not have inverse matrices.

Norms

Many machine learning algorithms use tensors as buliding block. Tensors provide a flexible unit of representation. In most applications, tensors appear in the form of vectors and matrices.

Norms

Norm is a measurement of the size of a tensor and is considered to be generalization of absolute value. It is a generalized representation of length to vectors, matrices and tensors.

- Size (magnitude) of the tensor.
- How close or similar two tensors are.
- Norm analysis is required to tell if methods will converge or not.
- Both NumPy and Torch has functions to calculate norms.
- numpy.linalg.norm() or torch.linalg.norm()

Mathematically, norm is a function, f such that

1. $f(x) = 0 \implies x = 0$
2. $f(x+y) \leq f(x) + f(y)$ [triangle inequality]
3. $\forall \alpha \in \mathbb{R} : f(\alpha x) = ||\alpha|| f(x)$ [scaler multiplicavity]
4. $||x^T y|| \leq ||x|| ||y||$ [Cauchy-Schwarz-Buniakovsky inequality]

Vector Norms

Consider a vector, $\underline{v} = (v_1, v_2, v_3, \ldots, v_n)$

1-Norm or taxicab or Manhattan norm

$$||\underline{v}||_1 = ||v_1|| + ||v_2|| + ||v_3|| + \ldots ||v_n|| = \sum_{i+1}^{n} v_i$$

2-Norm or Euclidean norm

$$||\underline{v}||_2 = (||v_1||^2 + ||v_2||^2 + ||v_3||^2 + \ldots ||v_n||^2)^{\frac{1}{2}} = \sqrt{(\sum_{i+1}^{n} v_i^2)}$$ [common distance measure]

Vector Norms

p-Norm

$$||\underline{v}||_p = (||v_1||^p + ||v_2||^p + ||v_3||^p + \ldots ||v_n||^p)^{\frac{1}{p}} = \sqrt[p]{(\sum_{i+1}^{n} v_i^p)}$$

p = 1 and p = 2 gives Manhattan and Euclidean norms respectively.

Max-norm or ∞-norm

$$||\underline{v}||_\infty = max(||v_1|| + ||v_2|| + ||v_3|| + \ldots ||v_n||) = \max_i (v_i)$$

for very large values of p we get the ∞-norm.

Matrix Norms

Let, $|\cdot|_v$ be a vector norm. Then the real-valued function $|\cdot|$ defined for $\forall A \in \mathbb{R}^{n\times n}$, is called "**consistent**" norm of A.

$$||A|| = \max_{||x||_v \neq 0} \left(\frac{||A\underline{x}||_v}{||\underline{x}||_v}\right)$$

Frobenius Norm:

$$||A||_F = \left(\sum_{i,j} a_{ij}^2\right)^2 = (trace(A * A))^{\frac{1}{2}}$$

$$trace(A) = \sum_{i=1}^{n} a_{ii}, \quad \forall A \in \mathbb{R}^{n\times n}$$

Matrix Norms

1-Norm: Sum of absolute values of entries in j-th column of A which is a $n \times n$ matrix.

$$||A||_1 = \max_{1 \leq j \leq n} \sum_{i,j} ||a_{ij}||$$

2-Norm: Also called **spectral norm.** ρ is the spectral radius of A.

$$||A||_2 = \sqrt{\rho(A^T A)}$$

∞**-Norm:** Sum of absolute values of entries in i-th row of A which is a $n \times n$ matrix.

$$||A||_\infty = \max_{1 \leq i \leq n} \sum_{i,j} ||a_{ij}||$$

Properties of Matrix Norms

1. $||A|| \geq 0$
2. $||A|| = 0 \iff a_{ij} = 0 \quad \forall i, \forall j$
3. $||A + B|| \leq ||A|| + ||B||$ [triangle inequality]
4. $||sA|| = ||s||||A||$ [scaler multiplicavity]
5. $||AB|| \leq ||A||||B||$ [sub-multiplicavity]

Spectral Radius ρ

Spectral radius of matrix A whose eigenvalues are λ is defined as

$$\rho(A) = \max_{1 \leq i \leq n} |\lambda|$$

Spectral radius is the maximum of the absolute values of the eigenvalues of a matrix. The matrix must be square.

Given, an n-element column vector x and $n \times n$ matrix A,
if $|Ax| \leq |A||x|$, then the matrix and vector norms are called "consistent".

For all consistent norms,

$$\rho(A) \leq ||A||$$

A circle with spectral radius will contain all the eigenvalues of the matrix.