Review

Assignment 1 – Task 1 - Clarifications

- You can use your images with and without glasses
- You can use images of celebrities as “unknown” (other than you) class.
- Ensure class ratios in train and test splits (recall stratified split).
- You should resize all the images to same resolution
  - 32x32 is a good start for now. **Why?**
  - How many features would be there if the image is grayscale and the resolution is 32x32?
  - How many features would be there for colored image with resolution of 32x32?

- **Implementation tip: You can use .flatten for any numpy array to… flatten it! (aka converting to a vector)**

Feature Space: Image Data

- Images are nothing but a **2D/3D arrays** with values of color intensities, typically ranging $\mathbf{0-255}$

Do this for all of your images and now each record is a vector!

10	90	16	16
0	11	11	11
18	30	33	33
18	18	18	18

10	90	16	16	0	11	11	11	18	30	33	33	18	18	18	18

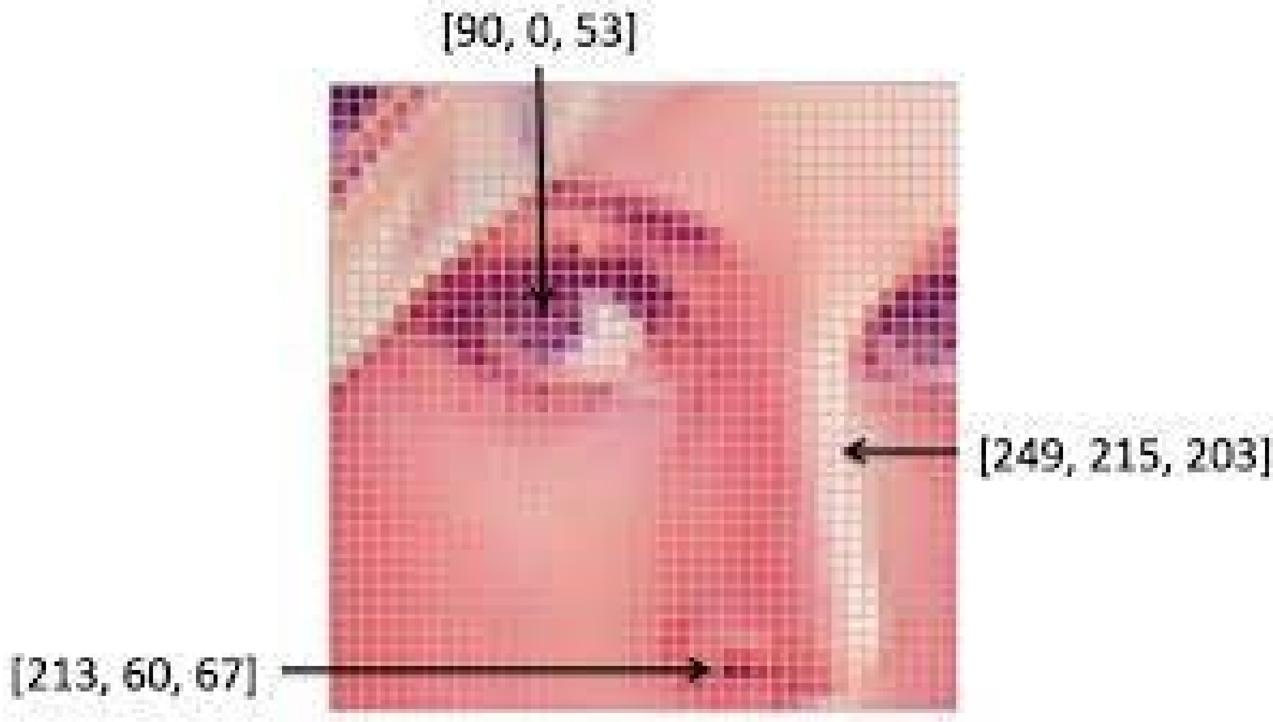
The label is stored separately for corresponding record.

If label is “Me” and “Not Me”, it’s classification.
If label is “Age”, its regression.

Implementation Tip:
Use numpy reshape.

Feature Space: Image Data

- The color Image is 3D array $(Width \times Height \times Channels)$
- Color image has 3 channels while grayscale image has 1 channel.



How would you convert color image to 1D array?

Feature Space: Text Data

- Suppose you are given labeled textual data in excel sheet

	Document#	Text	Class
Training	1	The Best movie best	Pos
	2	The Best best ever	Pos
	3	The Best film	Pos
	4	The Worst cast ever	Neg
Testing	5	The Best best best worst ever	?

the	best	movie	ever	film	worst	cast
1	1	1	0	0	0	0
1	1	0	1	0	0	0
1	1	0	0	1	0	0
1	0	0	1	0	1	1

label
1
1
1
0

These are called "Binary Occurrences" features.

1	1	0	1	0	1	0

?

Feature Space: Text Data

- Suppose you are given labeled textual data in excel sheet

	Document#	Text	Class
Training	1	The Best movie best	Pos
	2	The Best best ever	Pos
	3	The Best film	Pos
	4	The Worst cast ever	Neg
Testing	5	The Best best best worst ever	?

the	best	movie	ever	film	worst	cast	label
1	2	1	0	0	0	0	1
1	2	0	1	0	0	0	1
1	1	0	0	1	0	0	1
1	0	0	1	0	1	1	0
1	3	0	1	0	1	0	?

These are called "Term Frequency" features.

Advanced TFIDF features (we'll skip details)

Supervised Learning

- Predicting the labels for unseen data based on labelled instances.
- **Quick recap!**

Path	Label
data/1.jpg	me
data/2.jpg	not me
data/3.jpg	me
data/4.jpg	me
.	.
.	.
.	.

Classification or Regression?

Classification. Specifically, binary-class classification.

Path	Label
data/1.jpg	me
data/2.jpg	not me
data/3.jpg	my friend
data/4.jpg	me
.	.
.	.
.	.

Classification or Regression?

Classification. Specifically, multi-class classification.

Path	Label	Label 2
data/1.jpg	me	smiling
data/2.jpg	not me	not smiling
data/3.jpg	my friend	smiling
data/4.jpg	me	smiling
.	.	.
.	.	.
.	.	.

Classification or Regression?

Classification. Specifically, multi-label classification.

Rules vs. Learning

- Suppose we are working on classification of emails into “spam” and “ham” (not spam)
- **We can write a complicated set of rules**
  - Works well for a while
  - Cannot adapt well to new emails
  - Program could be reverse-engineered and circumvented
- **Learn the mapping between an email and its label using past labelled data**
  - Can be retrained on new emails
  - Not easy to reverse-engineer and circumvent in all cases
  - Easier to plug the leaks

Formalizing the Setup

$$\boldsymbol{D} = \{(\boldsymbol{x}^1, \boldsymbol{y}_1), (\boldsymbol{x}^2, \boldsymbol{y}_2), \dots, (\boldsymbol{x}^n, \boldsymbol{y}_n) \subseteq \boldsymbol{X} \times \boldsymbol{Y}$$

Feature vector

$$\boldsymbol{D} = \{(\overrightarrow{\boldsymbol{x}_1}, \boldsymbol{y}_1), (\overrightarrow{\boldsymbol{x}_2}, \boldsymbol{y}_2), \dots, (\overrightarrow{\boldsymbol{x}_n}, \boldsymbol{y}_n) \subseteq \boldsymbol{X} \times \boldsymbol{Y}$$

- Where,
  - D is the dataset
  - X is the d-dimensional feature space $(\mathbb{R}^d)$
  - $\overrightarrow{x_i}$ or x^i is the input vector of the ith sample/record/instance
  - Y is the label space

Any categorical attribute can be converted to numerical representation.

The data points are drawn from an **unknown** distribution P

$$(\overrightarrow{\boldsymbol{x}_i}, \boldsymbol{y}_i) \sim \boldsymbol{P}(\boldsymbol{x}, \boldsymbol{y})$$

If we don't know the distribution, lets approximate that using samples we gathered!

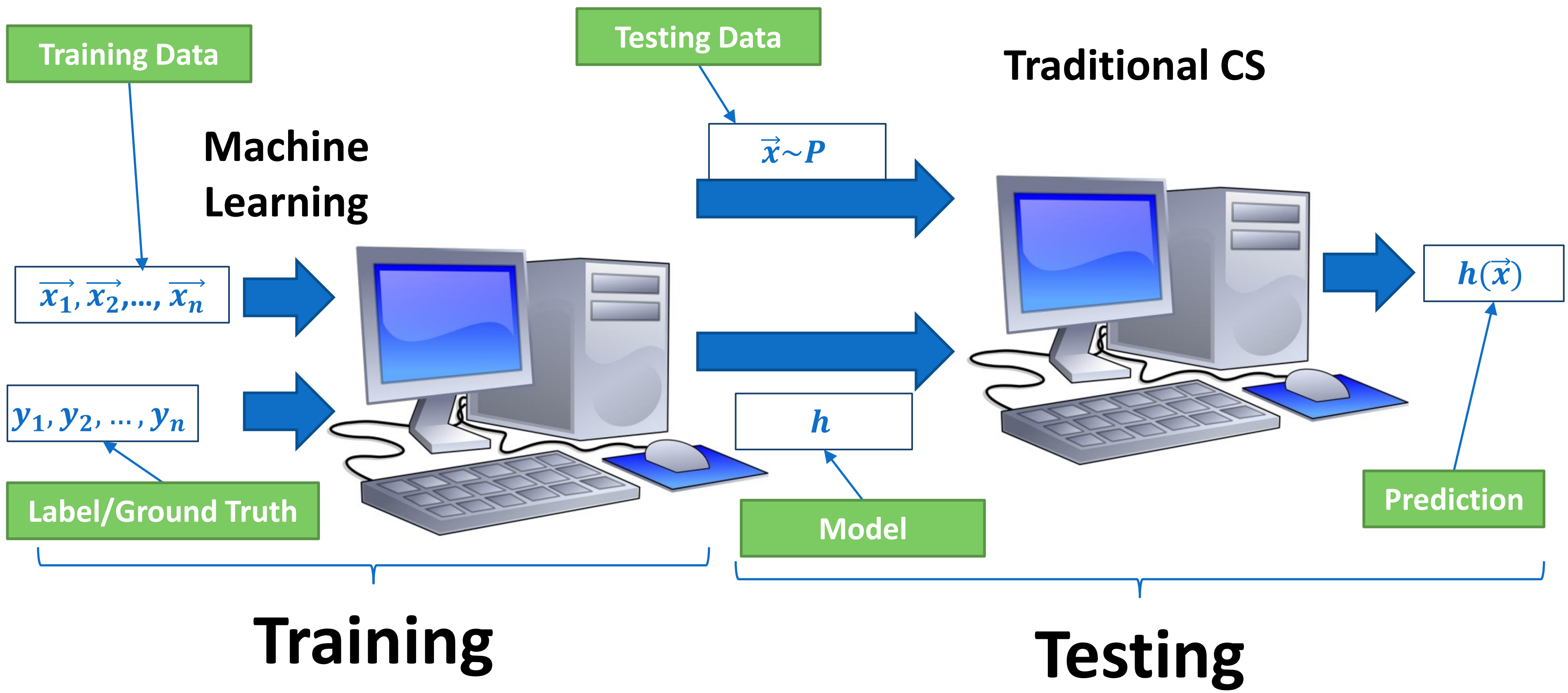
We want to learn a function $h \in H$, such that for a new instance $(\vec{x}, y) \sim P$

$\boldsymbol{h}(\vec{\boldsymbol{x}}) = \boldsymbol{y}$ with a high probability or at least $\boldsymbol{h}(\vec{\boldsymbol{x}}) \approx \boldsymbol{y}$

This also have to be from the same distribution as $\overrightarrow{x_i}$

In plain words, don't train on dogs and ask prediction for cats.

Training and Testing: Formally



$h(\vec{x}) = y$ **(Ideal)**
$h(\vec{x}) \approx y$ **(Plausible)**

Label Space

- **Binary (Binary classification)**
  - Sentiment: positive / negative
  - Email: spam / ham
  - Online Transactions Fraud: Yes / No
  - Tumor: Malignant / Benign
  - $y \in \{0,1\}$
  - $y \in \{-1, 1\}$
- **Multi-class (multi-class classification)**
  - Sentiment: Positive / Negative / Neutral
  - Emotion: Happy / Sad / Surprised / Angry / …
  - Parts of Speech Tag: Noun / Verb / Adjective / Adver / …
  - $y \in \{0,1,2, \dots\}$
- **Real-valued (Regression)**
  - Temperature, height, age, length, weight, duration, price, …

Hypothesis Space

- The hypothesis h is sampled from a hypothesis space H

$\boldsymbol{h \in H}$ $\boldsymbol{H \in \{H_D, H_R, H_{SVM}, H_{DL}, \ldots\}}$

- H can be thought of to contain types of hypotheses, which share sets of assumptions like:
  - Support Vector Machines $\boldsymbol{H_{SVM} \in \{H_1, H_2, \ldots\}}$
  - Decision Tree $\boldsymbol{H_D \in \{H_1, H_2, \ldots\}}$
  - Perception $\boldsymbol{H_P \in \{H_1, H_2, \ldots\}}$
  - Neural Networks $\boldsymbol{H_{NN} \in \{H_1, H_2, \ldots\}}$
  - …

$\boldsymbol{h \in H_D}$

Selection done manually.

Selection done automatically.

- For example: $h \in H$ for H decision trees:
  - Would be instance of decision trees of different height, arity, thresholds etc.

So, how do we choose our h?

- ❑ **Randomly?**
- ❑ **Exhaustively?**

How do we evaluate h?

How to choose h?

Randomly

- May not work well
- Like using a random program to solve your sorting problem!
- May work if H is constrained enough

Exhaustively

- Would be very slow!
- The space H is usually very large (if not infinite)

H is usually chosen by ML Engineers (You!) based on their experience

- $h \in H$ is estimated efficiently using various optimization techniques **(math alert!)**

Before moving to finding h, let's first evaluate the labels.

Book Reading

- Murphy – Chapter 1