Supervised Learning

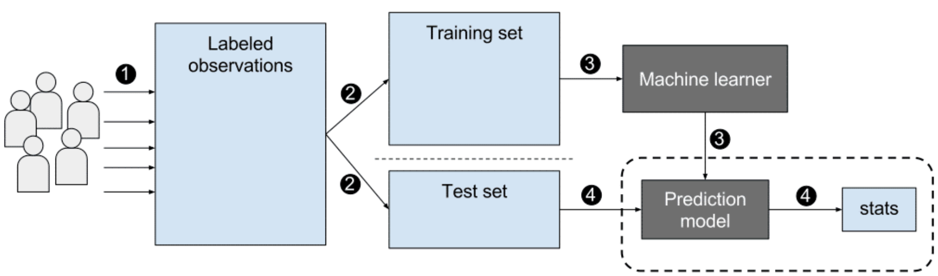
CSC 461: Machine Learning

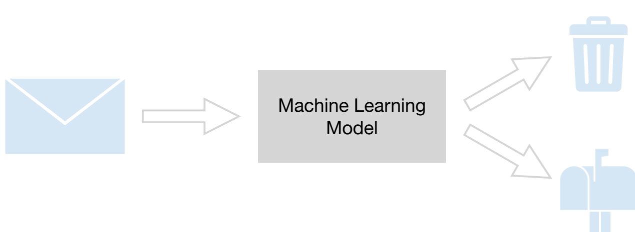
Fall 2022

Prof. Marco Alvarez
University of Rhode Island

Supervised Learning

Example: spam filtering





Components of supervised learning

- Data instance $(x, y), \quad x \in \mathscr{X}$ and $y \in \mathscr{Y}$
  - ✓ Input space (set) $\mathscr{X}$
  - ✓ Output space (set) $\mathscr{Y}$
- Dataset $\{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathscr{X} \times \mathscr{Y}$
- Hypothesis $h : \mathscr{X} \mapsto \mathscr{Y}, h \in \mathscr{H}$

Dataset

- Samples (data instances) are independently drawn from an **unknown joint distribution** $P(X, Y)$

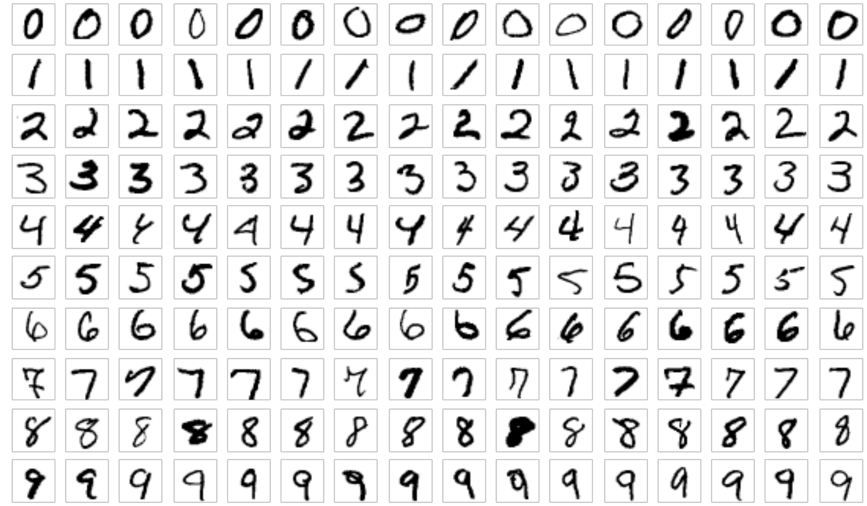
$$\mathscr{D} = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$$

in general $\mathscr{X} = \mathbb{R}^d$

$$(\mathbf{x_i}, y_i) \sim P$$

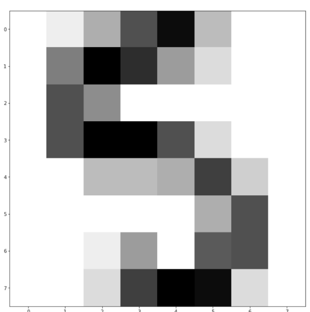
unknown

Example: MNIST dataset



https://en.wikipedia.org/wiki/MNIST_database

Example: MNIST dataset



Image

```
[[ 0.   1.   5.  11.  15.   4.   0.   0.]
 [ 0.   8.  16.  13.   6.   2.   0.   0.]
 [ 0.  11.   7.   0.   0.   0.   0.   0.]
 [ 0.  11.  16.  16.  11.   2.   0.   0.]
 [ 0.   0.   4.   4.   5.  12.   3.   0.]
 [ 0.   0.   0.   0.   0.   5.  11.   0.]
 [ 0.   0.   1.   6.   0.  10.  11.   0.]
 [ 0.   0.   2.  12.  16.  15.   2.   0.]]
```

Matrix representation

$$\mathscr{X} = \mathbb{R}^{64}$$

```
[ 0.  1.  5. 11. 15.  4.  0.  0.  0.  8. 16. … 11.  0.  0.  0.  2. 12. 16. 15.  2.  0.]
```

Vector representation

Output space

Binary classification

$\mathscr{Y} = \{0,1\}$

$\mathscr{Y} = \{-1, +1\}$

Multiclass classification

$\mathscr{Y} = \{0,1,\ldots,k-1\}$

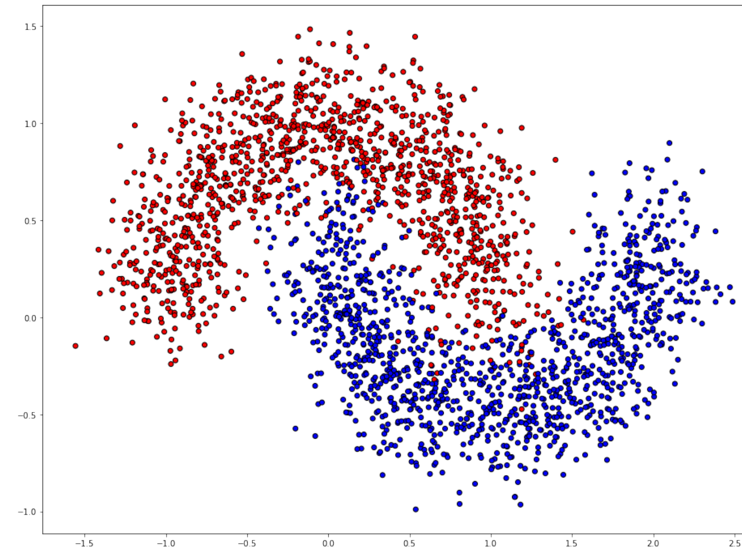
Regression

$\mathscr{Y} = \mathbb{R}$

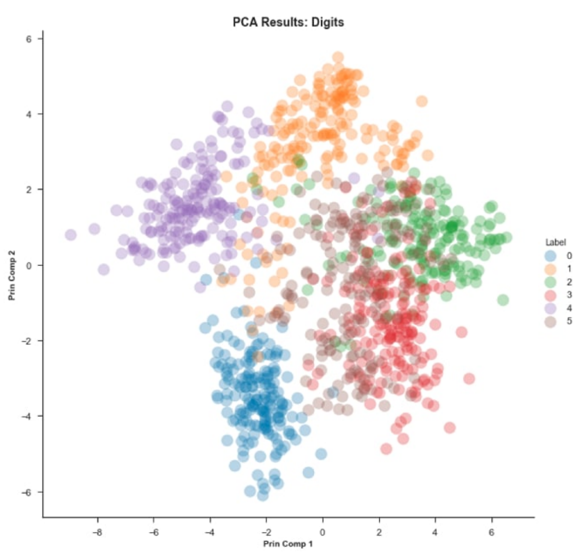
Example: binary classification

```
array([[ 0.24277092,  0.89098144],       array([[0],
       [-0.57961074,  0.50618765],              [1],
       [ 0.24259841,  0.12209649],              [1],
       [ 1.68348295, -0.10059047],              [1],
       [ 2.00696736, -0.79306007],              [1],
       [ 1.56891881,  0.30515286],              [0],
       [ 0.1314049 , -0.35704446],              [1],
       [ 2.14017386,  0.33933491],              [1],
       [-1.03087047,  1.52609949],              [0],
       [-0.38504321,  1.24209655],              [0],
       [-1.20252537,  0.56167652],              [0],
       [ 0.08590311,  0.68265315],              [1],
       [ 0.88074085, -0.11759523],              [1],
       [ 0.32558238,  0.4181143 ],              [1],
       [-0.74202798,  0.68847344]])            [0]])
```

Example: binary classification

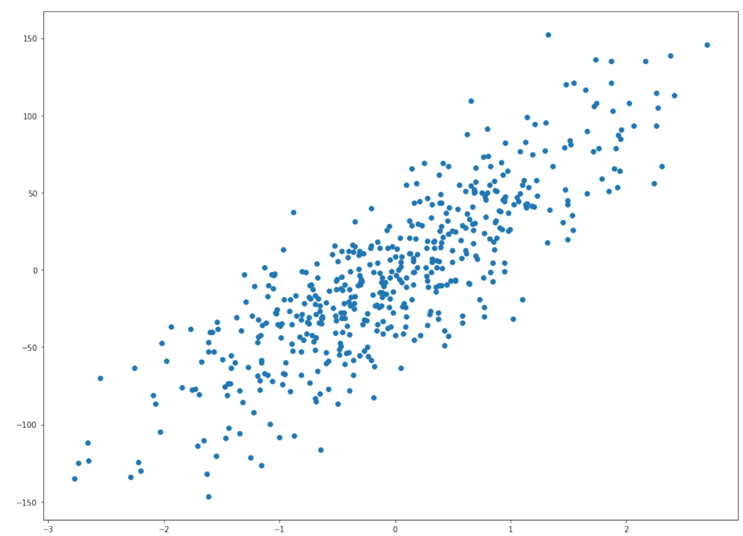


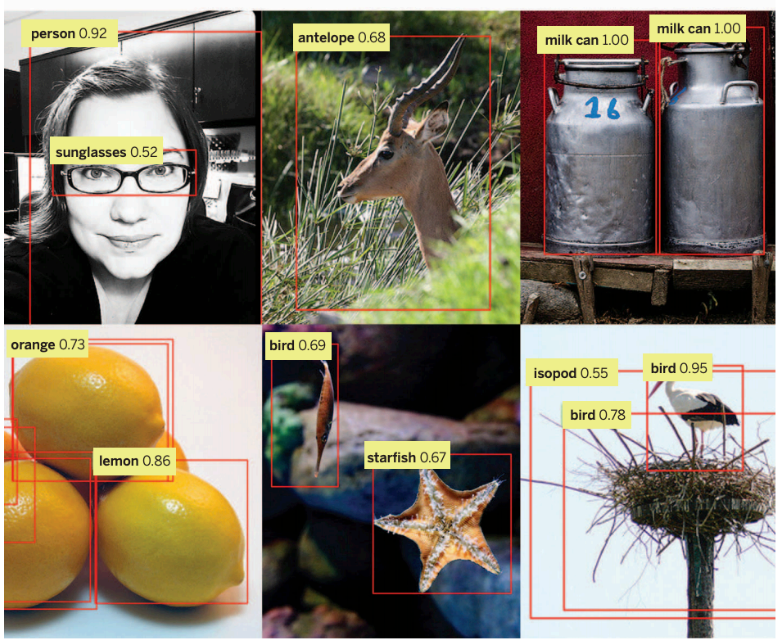
Example: multiclass classification



https://www.kdnuggets.com/2018/08/introduction-t-sne-python.html

Example: regression





"Machine learning: Trends, perspectives, and prospects", M. I. Jordan and T. M. Mitchell

Hypothesis spaces

- Hypotheses are functions that belong to a **hypothesis space**
  - ✓ space is defined by the machine learning technique, i.e., decision trees, neural networks, support vector machines, etc.
- How to perform machine learning?
  - ✓ define the hypothesis space $\mathscr{H}$ (restricts space of all possible functions)
  - ✓ find the best function within this space, $g \in \mathscr{H}$
    - ✓ a **loss function** is used to evaluate and select hypotheses

Example

$h_1 \in \mathscr{H}$

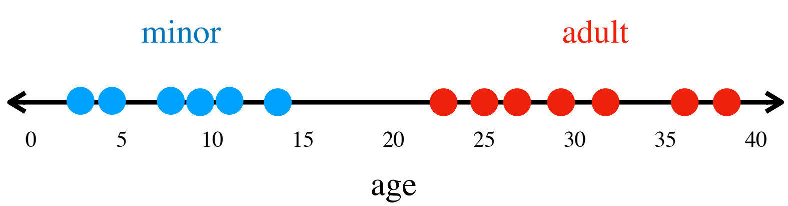
$h_2 \in \mathscr{H}$

$h_3 \in \mathscr{H}$

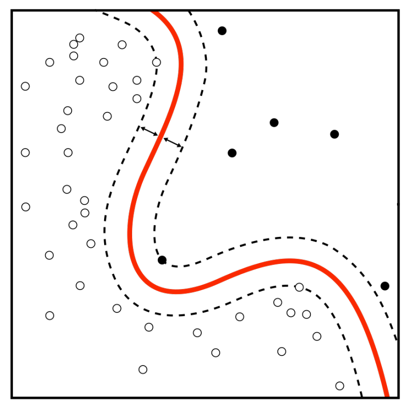
$\dots$

can you define the hypothesis space $\mathscr{H}$?

how to pick a hypothesis that makes you happy?

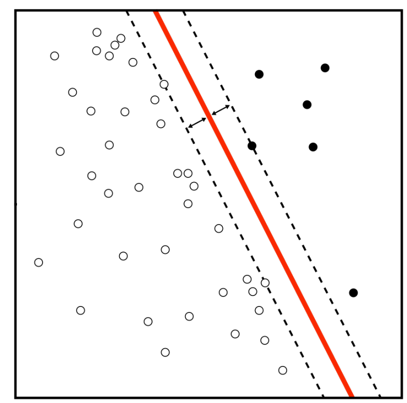


Example of hypotheses



$h(\mathbf{x}) = \mathrm{sgn}\left(\mathbf{w}^T \mathrm{relu}\left(W\mathbf{x}\right)\right)$

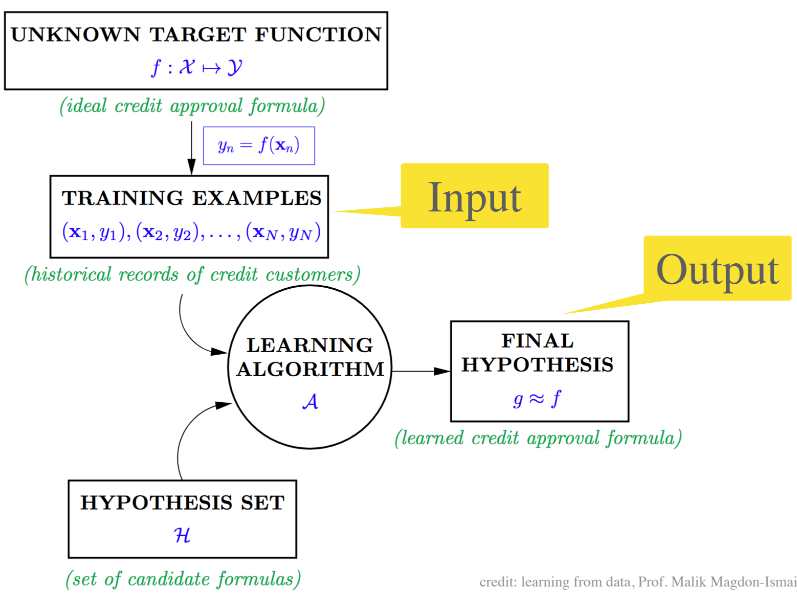
Hypothesis space: all possible single-layer neural nets



$h(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}^T\mathbf{x})$

Hypothesis space: all possible linear functions

Learning setup



credit: learning from data, Prof. Malik Magdon-Ismail

Loss Functions

What is the goal of (**supervised**) learning?

- Finding a **hypothesis** (**classifier/regressor**) that best approximates the **target** function

for $h \in \mathscr{H}$ and $\forall (x_i, y_i) \sim P$, we want $h(x_i) \approx f(x_i)$

ML uses **search** and **optimization**
(to **minimize expected loss**)

Expected Loss

$$\mathbb{E}\left[l\left(h, x_i, y_i\right)\right]_{(x_i, y_i) \sim P}$$

We cannot calculate this term, but we can approximate it

Approximating the expected loss?

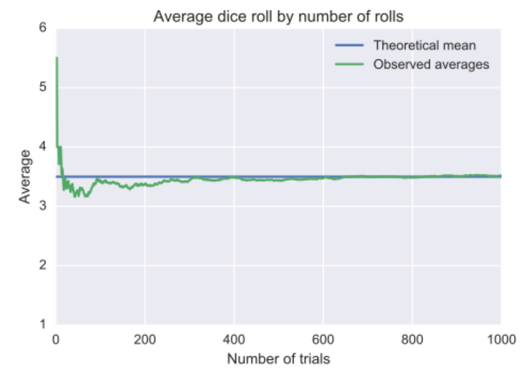
$$\mathbb{E}\left[l\left(h, x_i, y_i\right)\right]_{(x_i,y_i)\sim P}$$

$$\approx L = \frac{1}{n}\sum_{i=1}^{n} l\left(h, x_i, y_i\right)$$

the **law of large numbers** states that the arithmetic mean of the values almost surely converges to the expected value as the number of repetitions approaches infinity

Law of large numbers

$$Pr\left(\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n} x_n = \mathbb{E}[x]\right) = 1$$



credit: wikipedia

0/1 loss

$$l_{0/1}(h, x_i, y_i) = I\left(h(x_i) \neq y_i\right)$$

indicator function

Prediction	Target
5	5
1	9
2	2
7	7
8	0
0	0
0	8
3	3
6	6
4	4

Expected loss?

Squared loss

$$L_{sq}(h, x_i, y_i) = \left(h(x_i) - y_i\right)^2$$

positive loss and penalizes big mistakes

Prediction	Target
1.2	1.4
2.3	2.3
1.1	1.2
3.4	4.1
2.3	2.5
1.1	1.1
2.5	2.6
3.1	3.2
1.7	1.8
2.3	2.3

Expected loss?

Absolute loss

$$L_{abs}(h, x_i, y_i) = \left| h(x_i) - y_i \right|$$

Prediction	Target
1.2	1.4
2.3	2.3
1.1	1.2
3.4	4.1
2.3	2.5
1.1	1.1
2.5	2.6
3.1	3.2
1.7	1.8
2.3	2.3

Expected loss?

Learning

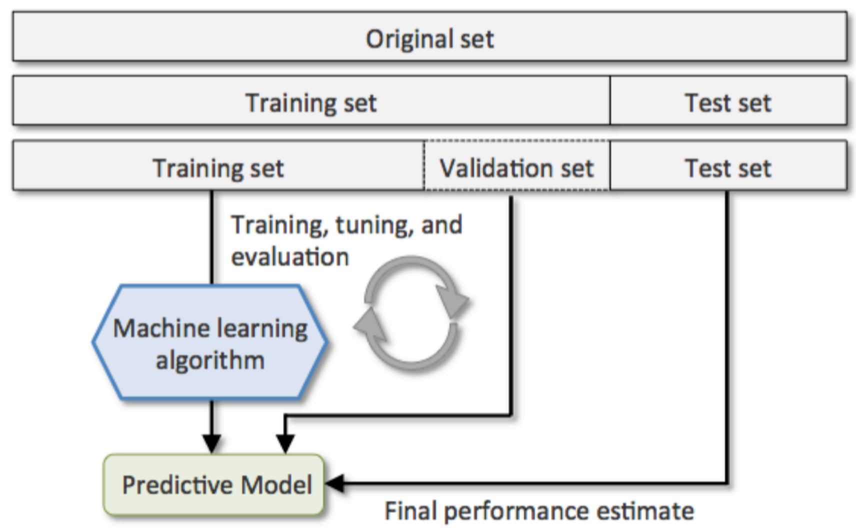
Generalization

- We can use a ML method to calculate:

$$g = \arg\min_{h \in \mathcal{H}} L(h, \mathcal{D})$$

- Problem: it may **overfit** the training data $\mathcal{D}$
  - ✓ we want better **generalization**
- Solution: split your data in train, validation, test
  - ✓ use train and validation to **select the best hypothesis**
  - ✓ use test for **final evaluation and report**

Train, validation, and test sets



from INFO-4604: Applied Machine Learning, Fall 2017, Michael Paul, Univ. of Colorado

```
>>> import numpy as np
>>> from sklearn.model_selection import train_test_split
>>> X, y = np.arange(10).reshape((5, 2)), range(5)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
>>> list(y)
[0, 1, 2, 3, 4]


>>> X_train, X_test, y_train, y_test = train_test_split(
...     X, y, test_size=0.33, random_state=42)
...
>>> X_train
array([[4, 5],
       [0, 1],
       [6, 7]])
>>> y_train
[2, 0, 3]
>>> X_test
array([[2, 3],
       [8, 9]])
>>> y_test
[1, 4]
```

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

train_test_split

Parameters:

arrays : *sequence of indexables with same length / shape[0]
Allowed inputs are lists, numpy arrays, scipy-sparse matrices or pandas dataframes.

test_size : *float or int, default=None*
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples. If None, the value is set to the complement of the train size. If `train_size` is also None, it will be set to 0.25.

train_size : *float or int, default=None*
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the train split. If int, represents the absolute number of train samples. If None, the value is automatically set to the complement of the test size.

random_state : *int, RandomState instance or None, default=None*
Controls the shuffling applied to the data before applying the split. Pass an int for reproducible output across multiple function calls. See Glossary.

shuffle : *bool, default=True*
Whether or not to shuffle the data before splitting. If shuffle=False then stratify must be None.

stratify : *array-like, default=None*
If not None, data is split in a stratified fashion, using this as the class labels. Read more in the User Guide.

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Stratified split

Example using MNIST

https://colab.research.google.com/drive/1m_h-c2sSC4fNhRRNR2q-Dfk2ji5V6ILQ?usp=sharing