Getting started with Machine Learning

(One-hour version)

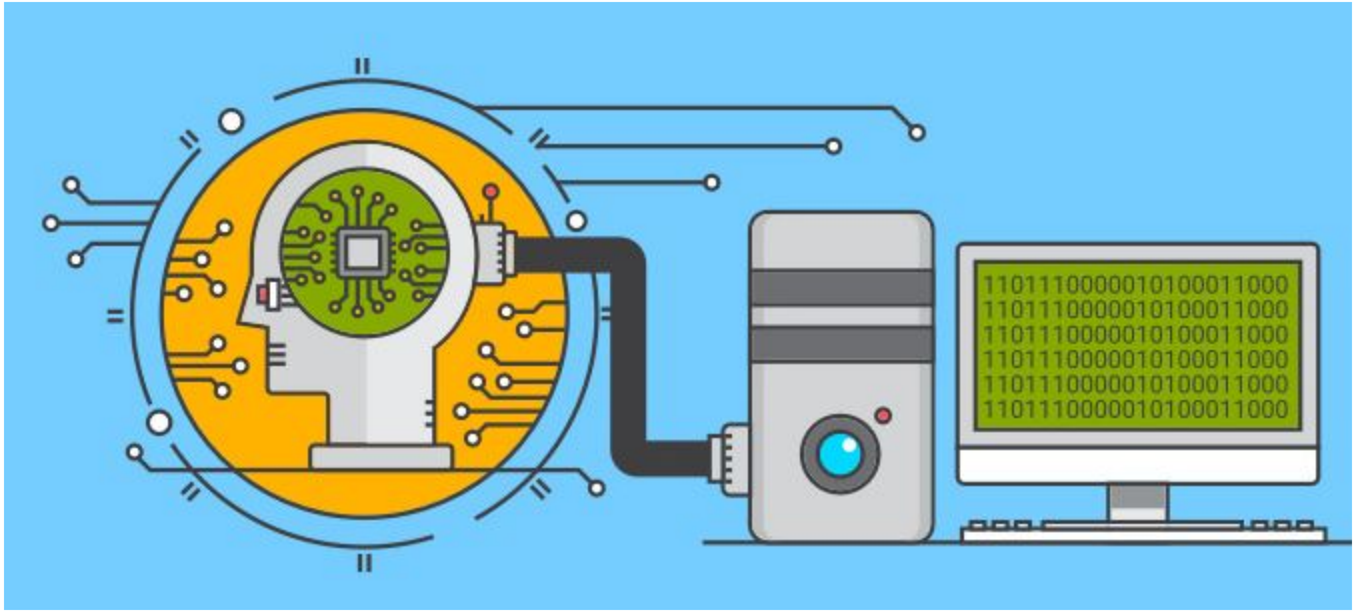
Hannah Wang (credits to Prof. John Lafferty)

Outline

- Machine learning concepts
  - Paradigm: Supervised / Unsupervised
  - Task: Classification / regression
  - Stochastic gradient descent
  - Cross-validation
- Demo and pipeline

Machine Learning

- Machine (Computers) + Learn, analogous to human + Learn
- Learn through Data and Algorithms to imitate what human beings do

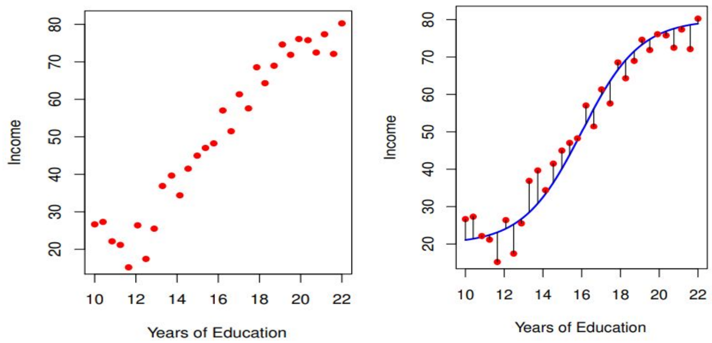


Machine learning concept - Paradigm

(**Supervised** / **Unsupervised** / Reinforcement learning / etc.)

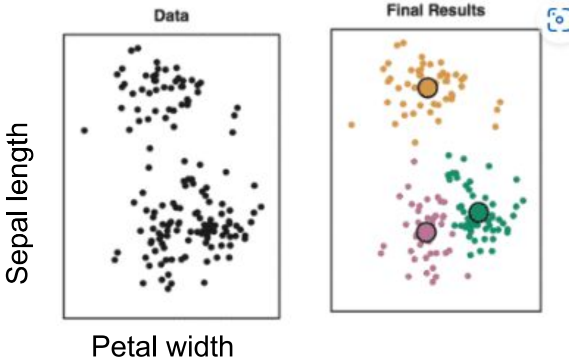
Supervised Learning (learn with labeled data):

- Given a set of data points (x, y), learn the function to predict label y using x, where x would have some features
- E.g., income prediction based on year of education (feature)



Unsupervised Learning (learn with unlabeled data):

- Given a set of data points with x (no labels y), learn underlying structure of the data or relationships of x
- E.g., Cluster flowers into categories based on petal width & sepal length (i.e., each data point x has two features)



Machine learning concept - Task

(Regression / Classification, both can be supervised or unsupervised, here, for simplicity: supervised)

The `Income` dataset:

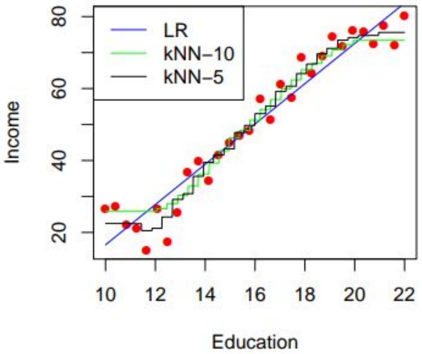
	Education	Seniority	Income	Income group
x_1	21.58621	113.1034	99.91717	1
x_2	18.27586	119.3103	92.57913	1
	12.06897	100.6897	34.67873	0
	17.03448	187.5862	78.70281	1
	19.93103	20.0000	68.00992	1
	18.27586	26.2069	71.50449	1

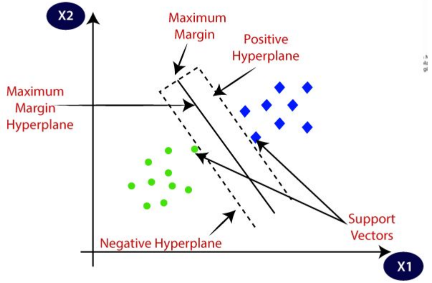
Information for 30 *simulated individuals*.

Collect data as a matrix: $X = (x_1\ x_2 \ldots x_{30})$, each x_i has 2 features (education, seniority). Given X:

Regression:

- Predict real or continuous response y, e.g., Income
- Assume relationship expressed as: y = f(X) + error
- f(.) can be any function/model:
  - linear regression: fit straight line through data
  - k-nearest neighbor: average together the y_i for x_i close to a fix x
  - etc.



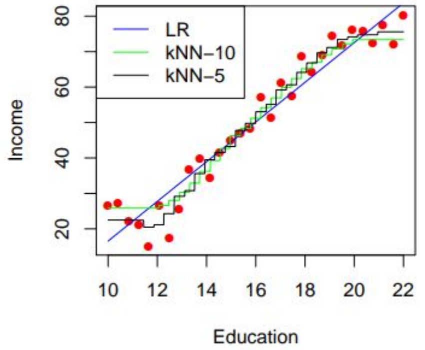


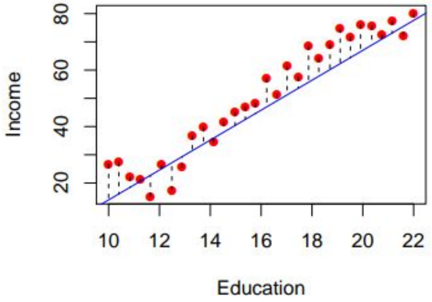
Classification: (more later!)

- Predict discrete response y (e.g, high income group, label 1; 0 o/w)
- With classification rule (h): input X to h (i.e., h(X)) →assign a class to each data point according to your rule
- h(.) can be any rule/model:
  - Bayes decision rule
  - Support vector machine
  - etc.

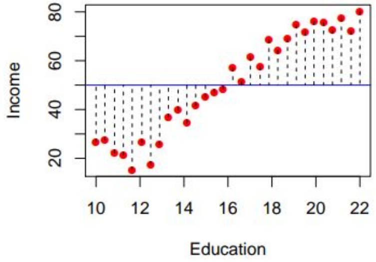
Machine learning concept - Models..

- f(.) and h(.) are some sort of models, determined by sets of parameters (coefficients)
- E.g., y = f(.) = linear model = $\beta_0 + \beta_1 x$; coefficients are: β_0, β_1
- Questions:
  - Given a dataset, many possible models to choose from, which should we choose?
    - → cross validation
  - Within one type of model (e.g., linear), there are many possible combinations of coefficients (e.g., β_0, β_1), how to get the optimal ones?
    - → optimization techniques, e.g, Stochastic gradient descent (SGD, basis of complicated algorithms)





$\hat{\beta}_0 = -39, \hat{\beta}_1 = 5.3$

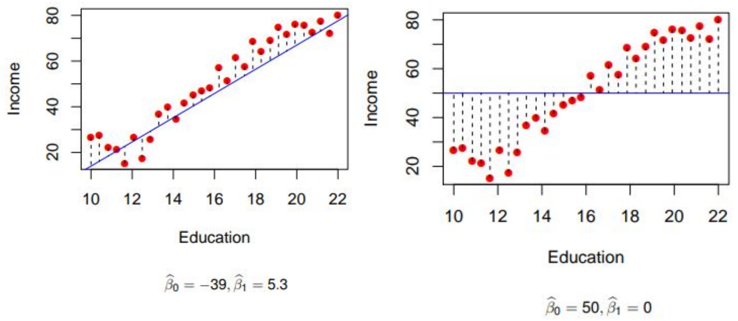


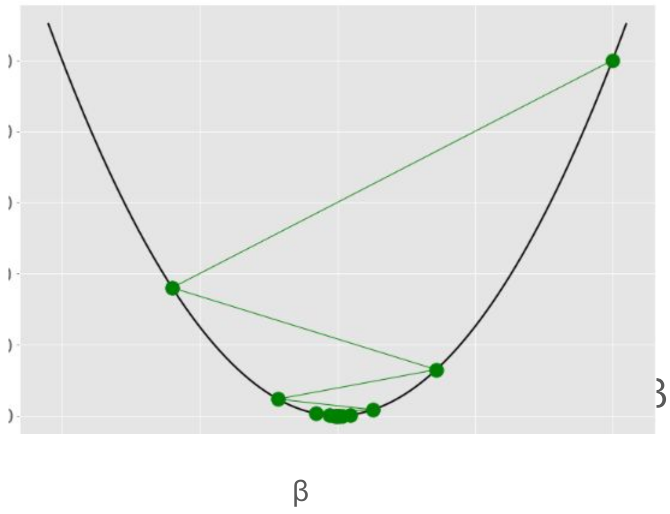
$\hat{\beta}_0 = 50, \hat{\beta}_1 = 0$

Machine Learning concept - SGD

- Example: linear regression
- Objective: minimize the discrepancies between predicted label ($\hat{y}$, blue) and observed label (y, red)
- One way to write:
  - $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$
  - Want betas to Minimize $\frac{1}{n}\Sigma_{i=1}(y_i - \hat{y}_i)^2$ (**mean squared error**, depending on **complexity** of model)
  - To find the betas: SGD algorithm
- SGD:

1. Read data item x
2. Make a prediction $\hat{y}(x) = \sum_{j=1}^{p} \beta_j x_j$
3. Observe the true response/label y
4. Update the parameters β so $\hat{y}$ is closer to y





What we've done…

- Train a model that performs (fits) well on predicting for the data we have
- Issue: might be **overfitting**, meaning the model fits training data well and generalize poorly on unseen data
  - Because it minimizes the discrepancies between training data's observed label y and predicted y
- One way to guard against overfitting: cross validation, it helps
  - Choose parameters (coefficients) for the models
  - Select between different models
  - Assess model performance (cross-validation err)

Machine Learning concept: Cross - validation

We've been doing this:

1. Divide dataset randomly into a training set and a validation set.
2. Fit the model on the training set.
3. Use the validation set to obtain estimated test error.
4. Repeat!

Machine Learning concept: Cross - validation (K-fold)

- Randomly divide the dataset into k *folds*.
- For $b = 1, \dots, k$:
  - Use b-th fold ("batch") as validation set.
  - Use everything else as training set.
  - Compute validation error on b-th fold.
- Estimate test error using:

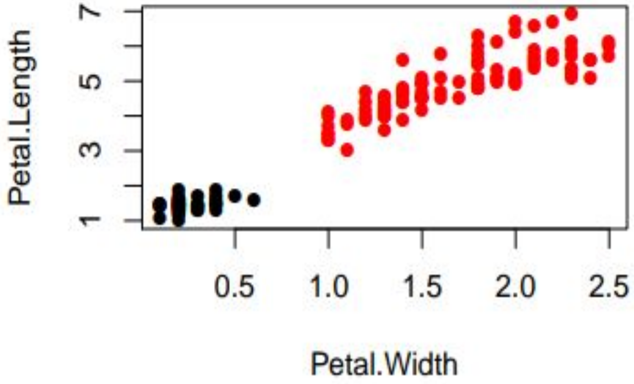
$$CV_{(k)} = \sum_{b} \frac{n_b}{n} MSE_b,$$

where n_b is the total # observations in the b-th fold, and n is the total # observations in the entire dataset.

	Iteration						
Obs	1	2	3	4	...	k	
1	valid	train	train	train	...	train	fold 1
2	valid	train	train	train	...	train	fold 1
3	valid	train	train	train	...	train	fold 1
4	train	valid	train	train	...	train	
...	...	...	...	...	...	...	
$n-2$	train	train	...	...	...	valid	fold k
$n-1$	train	train	...	...	...	valid	fold k
n	train	train	...	...	...	valid	fold k
MSE	MSE_1	MSE_2	MSE_3	MSE_4	...	MSE_k	

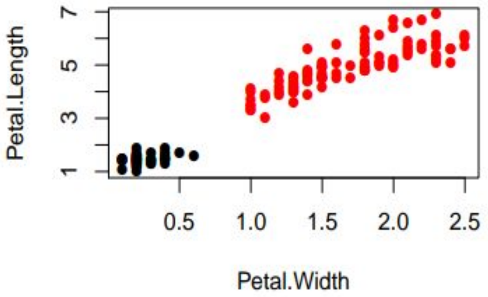
Same idea applies to classification

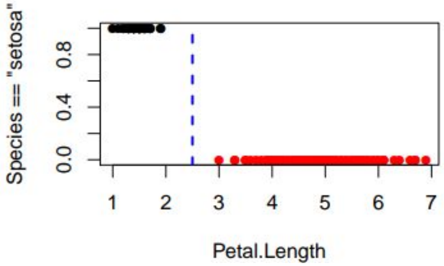
Iris setosa (Left), *Iris versicolor* (Middle), and *Iris virginica* (Right).



- Example: predict for 2 classes, label: {0: non-setosas, red, 1: setosas, black}
- Recall
  - Need classification rule (e.g., Bayes rule)
  - Desirable property for the Bayes rule:
    - at the boundary for classification (called: **decision boundary**), the data is **equally likely** to be classified into class 0 and class 1
    - To the left of the decision boundary: likely to be classified to class 1
    - To the right of the decision boundary: likely to be classified to 0
- One way to map data to the “Likelihood”:
  - Logistic regression

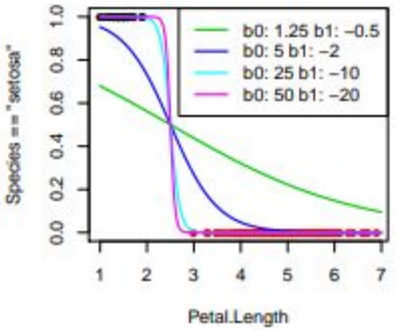
Logistic regression





- Map data to likelihood (softmax)
- Define p(x) = P(Y = 1|X=x)

$$\widehat{p}(x) = \frac{e^{\widehat{\beta}_0+\widehat{\beta}_1 x}}{1 + e^{\widehat{\beta}_0+\widehat{\beta}_1 x}}$$



- This softmax has decision boundary $\beta_0 + \beta_1 x = 0$, because:

$$\text{When } \widehat{\beta}_0 + \widehat{\beta}_1 x = 0,\ \frac{\widehat{p}}{1-\widehat{p}} = 1, \text{ so } \widehat{p} = 0.5.$$

- To the left of the decision boundary: likely to be classified to class 1 (black, setosas); right of the decision boundary, likely to be classified to class 0 (red, non-setosas)
- x = 2.5; many possible betas, need parameter tuning like before

Demo (& Pipeline)

- Google colab: shorturl.at/clNPS