LING 506 - TOPICS IN COMPUTATIONAL LINGUISTICS

Introductory Machine Learning

Yan Tang

Department of Linguistics, UIUC

Week 11

Last week…

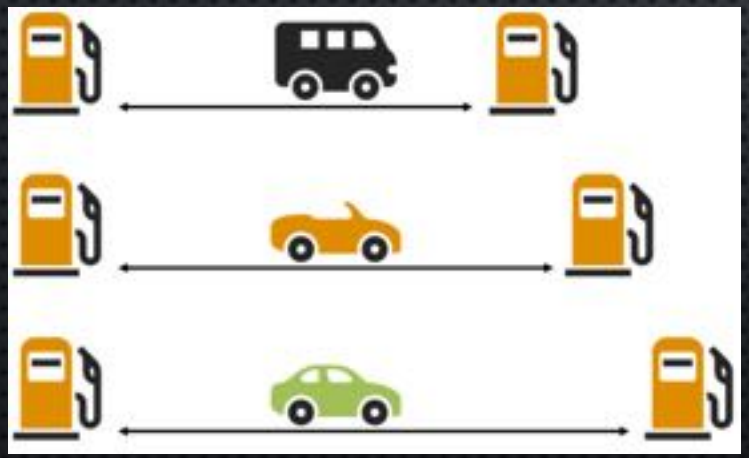
- Feature reduction techniques
  - Feature transformation, e.g. PCA
  - Feature selection, e.g. sequential feature selection
  - Feature elimination
  - Univariate statistics, e.g. ANOVA
- Idea of Ensemble learning
  - Idea: weak learner; ensemble: strong learner
- Hard and soft voting classifiers

Last week…

- Ensemble methods
  - Bagging and pasting
  - Random Forests
  - Extremely-Randomised Trees
  - Boosting
  - Stacking

Regression analysis

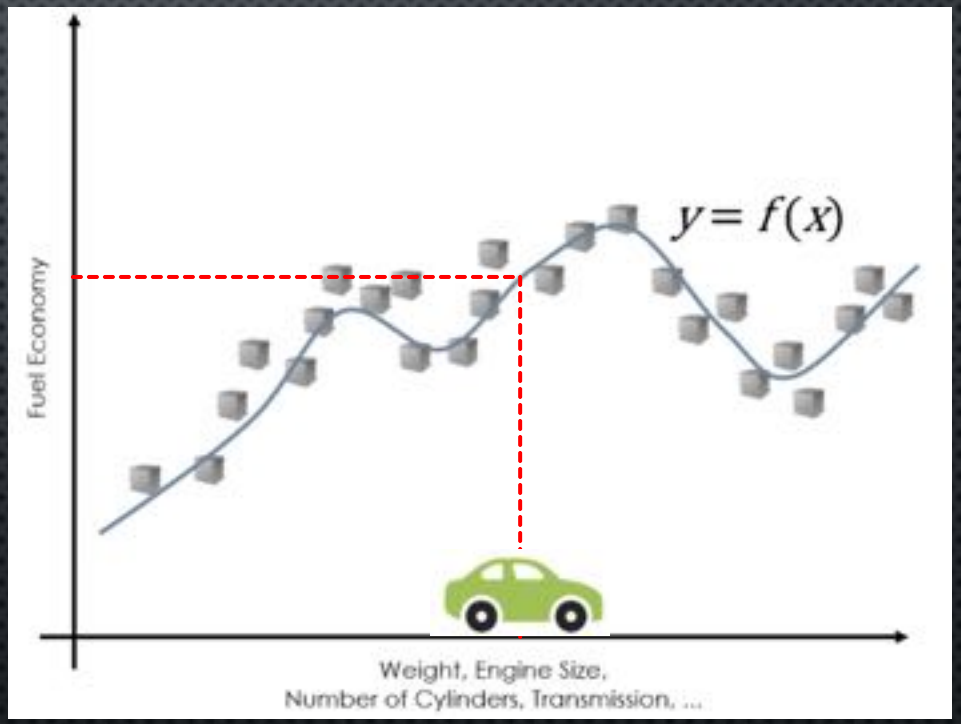
- Regression models
  - Predict target variables on a continuous scale
  - Powerful for addressing many questions in science and industry



$$Fuel\ efficiency(size, volume, weight \ldots) =$$

Regression analysis: example

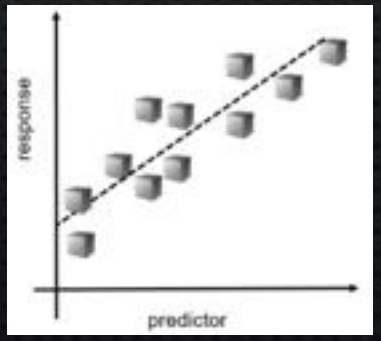
- Understand the relationship between the target variable and the known variables that affect the target variable
- Quantify the problem by modelling the relationship – a vigorous mathematic relationship



Linear regression

- Linear regression is a parametric regression technique
- Linear regression is to model the relationship between one or multiple predictor variables and a continuous target variable
  - Simple but powerful
- Example:

$$Fuel\ efficiency = c_0 + c_1 \cdot Weight + c_2 \cdot HorsePower + c_3 \cdot AxleRatio + \cdots + c_n \cdot etc$$



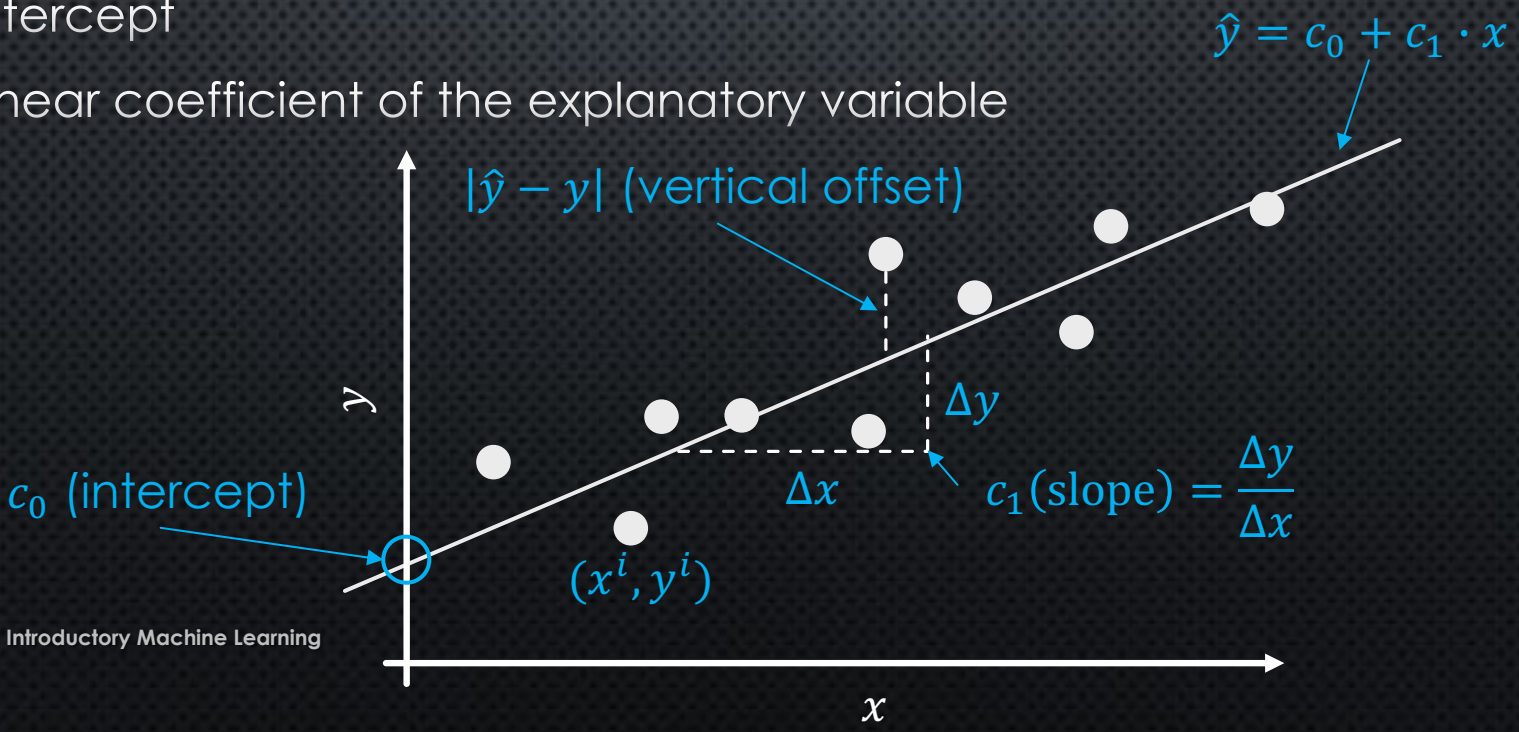
Univariate linear regression

- Univariate linear regression models the relationship between a single predictor variable (*explanatory variable*, $\boldsymbol{x}$) and a continuous-valued target (*response variable*, $\boldsymbol{y}$)

$$y = c_0 + c_1 \cdot x$$

c_0: intercept

c_1 : linear coefficient of the explanatory variable



Multiple linear regression

- A generalized form of linear regression model:

$$y = c_0 \cdot x_0 + c_1 \cdot x_1 + \ldots + c_N \cdot x_N$$

N: number of variables(features)

- A vectorised form

$$\hat{y} = \sum_{i=0}^{N} c_i \cdot x_i = c^T x$$

$\hat{y}$: predicted response

c: the linear coefficient vector (the model parameters), containing the intercept c_0 (the bias term) and the variable weights c_1 to c_N

X: variable vector (feature vector), containing x_1 to x_N, with x_0 equal to 1

Root Mean Square Error

- The **Room Mean Square Error** (RMSE): the most common performance metric for regression models

vertical offset

$$RMSE(X) = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(c^T x^i - y^i)^2}$$

- M: the total number of samples
- y^i: the ground-truth for sample i

- The **Mean Square Error** (MSE)

$$MSE(X) = \frac{1}{M}\sum_{i=1}^{M}(c^T x^i - y^i)^2$$

The normal equation

- The Normal Equation: a closed-from solution to find the value of $\hat{c}$ that minimise the cost function (the performance metric)

$$\hat{c} = (X^T X)^{-1} \cdot (X^T y) = X^+ y$$

X: Input feature matrix

y: the vector of ground-truth

X^+: the pseudoinverse (the Moore-Penrose inverse) of X

- Singular Value Decomposition (SVD)
  - More efficient than the normal equation
  - Can handle when $X^T X$ is singular

Computational complexity

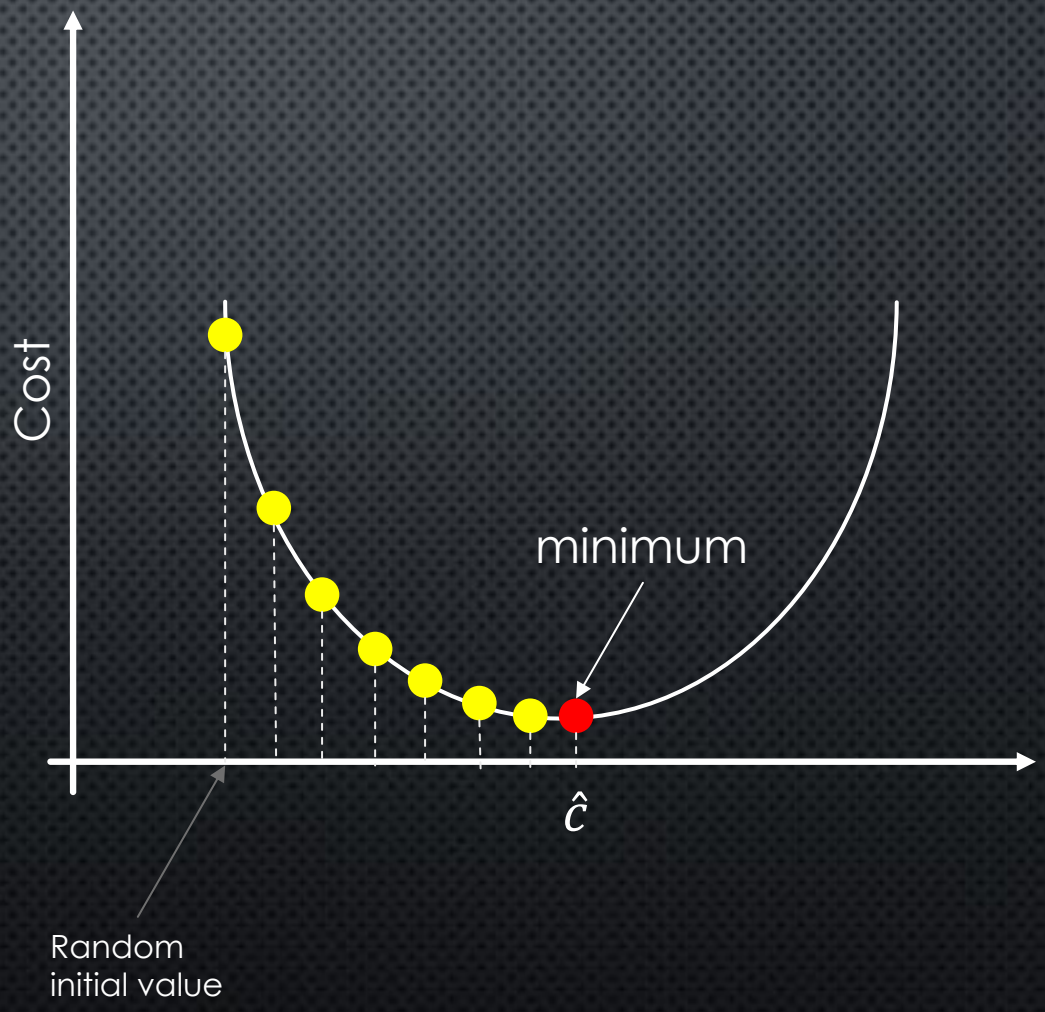
- The computational complexity of the Normal equation is approximately $O(n^{2.4})$ to $O(n^3)$
  - n is the number of features
  - Increasing the number of features by k times increases the computational time roughly $k^{2.4}$ to k^3 times
- The computational complexity of SVD is approximately $O(n^2)$
- Both approach can be slow when the number of features is large
- Making predictions are fast
  - The computational complexity is linear to both the number of instance and the number of features

Gradient Descent (GD)

- A generic optimisation algorithm for finding optimal solution for a problem
- The Idea of GD: iteratively altering parameters towards the possible direction that leads to decrease of loss
- The size of the step: an important parameter in GD
  - It decides how much each time the parameter should change

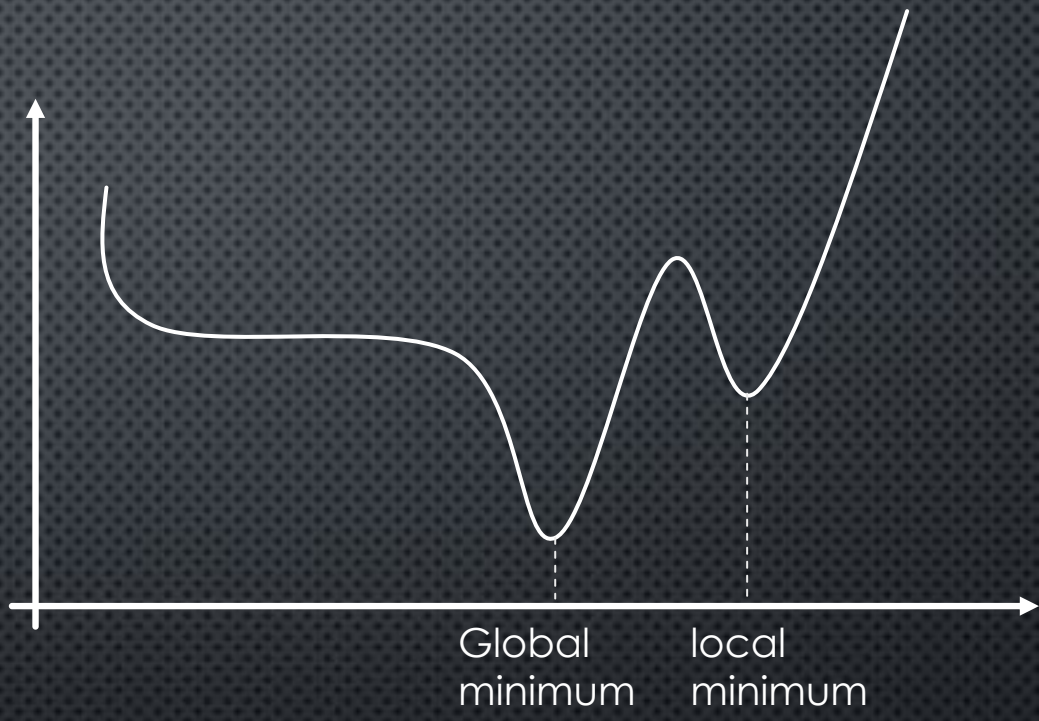
Gradient Descent: principle

1. Starts with a random set of values for the parameter vector c
2. Measures the local gradient of the cost function about c
3. Take a step towards the direction of descending gradient
4. Repeat 2 and 3 until the minimum is reached



Gradient Descent: local vs global minimum

- *Local minimum* and *plateau* are the two main challenges to GD
- The MSE cost function for Linear Regression model is a *convex* function
  - No local minimum
  - A continuous function with moderate slope



GD always converges if:
1. The step size is small enough
2. Sufficient number of iterations are allowed

Gradient Descent: implementation

- Partial derivative: calculation of local gradients to possible changes of c.

- The local gradient vector, $\nabla MSE(c)$, is given as,

$$\nabla MSE(c) = \frac{2}{M} X^T (Xc - y)$$

M: number of samples in training data

- The calculation is performed over the entire training data, hence the name **Batch Gradient Descent** (BGD)

Gradient Descent: implementation

- Given the local gradients $\nabla MSE(c)$, the GD step size is proportional to the learning rate η

- The next parameter candidates c' is therefore,

$$c' = c - \eta \nabla MSE(c)$$

- The optimal parameter $\hat{c}$ can be found by updating c' iteratively until the local gradient vector is smaller than a tolerance (a very small number!)

Stochastic Gradient Descent

- **Stochastic Gradient Descent** (SGD) picks a random sample in the training set at every step and computes the gradients based on only that *single* sample
  - Suitable for very large training sets

- Descending path can be irregular due to randomness; cost function may fluctuate over time
  - The final solution is *not* optimal
  - Greater chance to converge at the global minimum than BGD

Stochastic Gradient Descent

- **Learning schedule** for SGD: a function the determines the learning rate at each iteration
  - e.g. $\eta = \frac{\eta_0}{\eta_1 + t}, \eta \text{ and } \eta_1$ are the parameters
  - Decreases learning rate when approaching to minimum
  - Helps SGD converge as close to the global minimum as possible

- SGD usually iterates for a given number of *epochs*, each of which consists of M iterations
  - M: number of training samples
  - Epoch number decides the computational time of fitting

Mini-batch Gradient Descent

- **Mini-batch Gradient Descent (MGD)** computes the gradients on small random sets of instances, i.e. mini batches

- MGD vs SGD
  - Performance of MGD can be boosted by hardware
  - Parameter space is less erratic than SGD
  - The final solution is closer to the minimum; more likely converge at a local minimum

Comparison of algorithms for Linear Regression

	Large No. of samples	Out-of-core support	Large No. of FTs	No. of parameters	Data scaling required
Normal Equation	Fast	No	Slow	0	No
SVD	Fast	No	Slow	0	No
BGD	Slow	No	Fast	2	Yes
SGD	Fast	Yes	Fast	≥2	Yes
MGD	Fast	Yes	Fast	≥2	Yes

Chapter 4, *Aurélien Géron, 2019*