Week 4

Friday, June 26, 2020 2:26 PM

Supervised Learning

Def :

Given a data set of input-output pairs,
Learn a function to map input to output

Supervised learning tasks :

- Classification :
  - Supervised learning task of learning a function mapping an input point to a discrete category
  - We use a hypothesis function :
    - h()

- Nearest-neighbor classification :
  - **Algorithm that, given an input chooses**
    The class of the nearest data point to the input
  - K-nearest-neighbor classification :
    - **Algorithm that, given an input chooses the most common class of the nearest K data point to the input**

- the hypothesis function:
  we have :
  - Weight vector (w_0, w_1, $w_{2,}$ ....)
  - Input vector (1, x_1, $x_{2,}$ ....)

  $h_w(X)$ = 1 if : W . X >= 0
  0 otherwise

- Calculating the weight W :
  - Perceptron learning rule :
    - Given data input(x, y), update each weight according to :
      - $W_i = W_i + \alpha(y - h_w(x)) * X_i$
        - Y : actual value
        - $h_w(x)$: estimated value
        - α: learning rate
    - ---
    - We end up with a threshold function :
      - Threshold type :
        - Hard threshold :
          - 1 or 0 and there is no notion of how strong the prediction is
        - Soft thresholder :
          - By taking advantage of logistic regression We can use a logistic function
          - Prediction between 0 : 1
        - ---
- ---

Support vector Machine

Def :

Design to try to find the maximum margin separator

- Maximum margin separator :
  - Boundary that maximize the distance between any of the data points

---

Regression

Def :

Supervised learning task of learning a function that map an input point to a continuous value

---

Evaluating Hypotheses

Loss function :

Function that express how poorly our hypothesis perform

- Examples :

- 0-1 loss function :
  - L(actual, predicted) =
    - 0 if actual = predicted
    - 1 otherwise
- L_1 Loss function :
  - L(actual, prediction) = |actual - prediction|
- L_2 Loss function :
  - L(actual - prediction) = $(\text{actual - prediction})^2$
- --

Overfitting

Def :

A model that fit too closely to a particular data set therefor
May fail to generalize to future data

- To avoid this :
  - Use regularization :
    - Penalizing hypothesis that are more complex to
      favor sampler, more general hypothesis
    - Cost(h) = Loss(h) + λ complexity(h)
  - --
  - Use Handout cross-validation :
    - Splitting data into two sets :
      - Training set
      - Test set

        Such that learning happen on the training set
        And is evaluated on the test set
      - ----------------------------
  - Use K-fold cross-validation :
    - Splitting data into k sets,
      And experiment k times, using each set as
      Test set once, and using reaming data
      as a training set

Reinforcement learning

Def :

Given a set of rewards and punishment,
Learn which action to take in the future

- Markov Decision Process :
  - Model for decision-making, representing
    States, actions and their rewards
    - Set of states (s)
    - Set of actions Actions(s)
    - Transition model P(s` | s, a)
    - Reward function R(s, a, s`)
- ---
- Example :
  - **Q-learning :**
    - Method for learning a function Q(s, a)
      Estimate of the value of preforming action a
      In state s
    - Overview :
      - Start with Q(s, a) = 0 for all s, a
      - When taken an action and receive a reward :
        - Estimate the value of Q(s, a) based on
          current reward and expected future rewards
        - Update Q(s, a) to take in account our old estimate as well as
          Our new estimate
    - **Sudo code** :
      - Start with Q(s, a) = 0 for all s, a
      - Every time we take an action a in state s and observe a reward r,
        We update :
        Q(s, a) = Q(s, a) + α (new value estimate - old value estimate)
        - Old estimate : Q(s, a)

- New estimate : {r + future reward estimate}
  - Future reward estimate : $max_{a`}(s`, a`)$

So :

Q(s, a) =

Q(s, a) + α ([r + γ max_{a}`(s`, a`)] - Q(s, a))

- --
- Greedy decision-making :
  - When in a state s, choose the action that give the highest Q(s, a)
  - -------------------------------------
- $\varepsilon - greedy$:
  - Set ε $equal\ to\ how\ often\ we\ want\ to\ make\ a\ randome\ move$
  - With P-->(1 - ε) choose the best move
  - With p-->(ε) choose a randome move
- ---
- Function approximation :

  Approximating Q(s, a), often by combing various features,
  Rather than storing one value for every state action pair

Unsupervised learning

Def :

Given input without any additional feedback, learn patterns

- Clustering :
  - Organizing a set of objects into groups in such a way that similar
    Objects tend to be in the same group
- K-means clustering :
  - Algorithm for clustering data based on repeatedly assigning points to clusters
    And updating those clusters center