Final Review

2020
Most requested topics

- KL-divergence
- EM
- Reinforcement learning
- Other questions
- FYI: we didn’t cover PAC learning.
KL-divergence

◆ What does it measure?

◆ Where did we use it?
  - Decision trees: pick most informative feature
  - Neural net loss function: maximize estimated probability of true label
  - Active learning: label the most informative $x$
  - ICA: make sources as independent as possible
KL-divergence

- Measure change in distribution \( p(y|x) = f(x) \)
  - **Decision trees**: pick most informative feature \( x_j \)
    
    \[
    \argmax_{x_j} KL( f(x|X_j, x_j, y) \| f(x|X_j, y) )
    \]
  - **Neural net loss function**: maximize estimated probability of true label
    
    \[
    \argmax_\theta KL( p(y|x) \| f(x; \theta) )
    \]
    
    maximizes cross-entropy
  - **Active learning**: label the most informative \( x_i \)
    
    \[
    \argmax_{x_i} KL( f(x|X, x_i) \| f(x|X) )
    \]
- \( \min \ MI(s_1, s_2, \ldots s_k) = KL(p(s_1, s_2, \ldots s_k) \| p(s_1)p(s_2) \ldots p(s_k)) \)
EM

Used for:
- Missing data
- Clustering (GMM, LDA)

E Step
- Estimate the values of the missing data

M Step
- Do MLE (or MAP) estimation of the parameters
The Naïve Bayes Classifier

- **Conditional Independence Assumption:** Features are independent of each other given the class:
  \[ P(C|X) \sim P(X|C) \, P(C) = P(X_1|C) \, P(X_2|C) \cdots P(X_5|C) \, P(C) \]

- For language, assume \( P(\sim X_j|C) = 1 \)
- Use MLE or MAP to estimate the parameters
Gaussian Naïve Bayes Classifier

- $P(X|C): \mathcal{N}(\mu_C, \Sigma_C)$
  - $\Sigma_C$ is diagonal (= conditional independence)
- $P(C|X) \sim P(X|C) P(C) = P(X_1|C) P(X_2|C) \ldots P(X_5|C) P(C)$
Gaussian Mixture Model

- \( X \sim N(\mu_C, \Sigma_C) \)
- Now, \( C \) is not observed and \( \Sigma_C \) can have any form
- What does the E step compute?
- What does the M step compute?
The LDA Model (for 3 docs)

- For each document,
  - Choose the topic distribution $\theta \sim \text{Dirichlet}(\alpha)$
  - For each of the $N$ words $w_n$:
    - Choose a topic $z \sim \text{Multinomial}(\theta)$
    - Then choose a word $w_n \sim \text{Multinomial}(\beta_z)$
      - Where each topic has a different parameter vector $\beta$ for the words

$p(\text{topic})$  
topics  
words  
p(\text{word}|\text{topic})$
Reinforcement Learning

- Model-based: Response to all possible actions
- Model-free: Response to one action
- One-step ahead: Search to end

From David Silver UCL Course on RL: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
Reinforcement Learning

- Model-based (e.g., MDP) vs. model-free
- One step ahead (e.g., TD(0)) vs. Monte Carlo
- $V_\pi(s)$ vs. $Q_\pi(s,a)$
- On-policy vs. off-policy
  - SARSA: $\epsilon$-greedy, on policy
  - Q-learning: $\epsilon$-greedy action; then optimal (greedy)
Bellman’s Equation

\[ v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma v_\pi(s')] , \forall s \in S \]

- \( s \) state
- \( v_\pi(s) \) value
- \( \pi(a|s) \) policy (stochastic)
- \( \gamma \) discount factor
- \( r \) reward
- \( p(s'|s,a) \) model
- \( s' \) next state
Bellman’s Equation

\[ v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a) \left[ r + \gamma v_\pi(s') \right], \forall s \in S \]

\[ q_\pi(s, a) = \sum_{s', r} p(s', r \mid s, a) \left[ r + \gamma v_\pi(s') \right], \forall s \in S, \forall a \in A(s) \]
Bellman’s Equation

\[ v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[ r + \gamma v_\pi(s') \right] , \forall s \in S \]

\[ = \sum_{s',r} p(s',r|s,a) \left[ r + \gamma v_\pi(s') \right] a = \pi(s) \text{ if deterministic} \]

\[ v_\pi(s) := v_\pi(s) + \eta \left( r + \gamma v_\pi(s') \right) \text{ TD(0) estimation} \]
Q-learning

Pick action $a$ using $\epsilon$-greedy selection over $Q(s,a)$

Update

$$Q(s,a) := Q(s,a) + \eta \left( r + \gamma \max_a Q(s',a) \right)$$

For any MDP, given infinite exploration time and a partly-random policy, Q-learning will find an optimal policy: one that maximizes the expected value of the total reward over all successive steps.
Deep Q-Learning (DQL)

\[ \arg\min_\theta \left[ Q(s, a; \theta) - \left( r(s, a) + \gamma \max_{a} Q(s', a; \theta) \right) \right]^2 \]

Update this

To be closer to new value estimate

Represent \( Q \) with a neural net

\( s, a \) can be one-hot or real valued