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

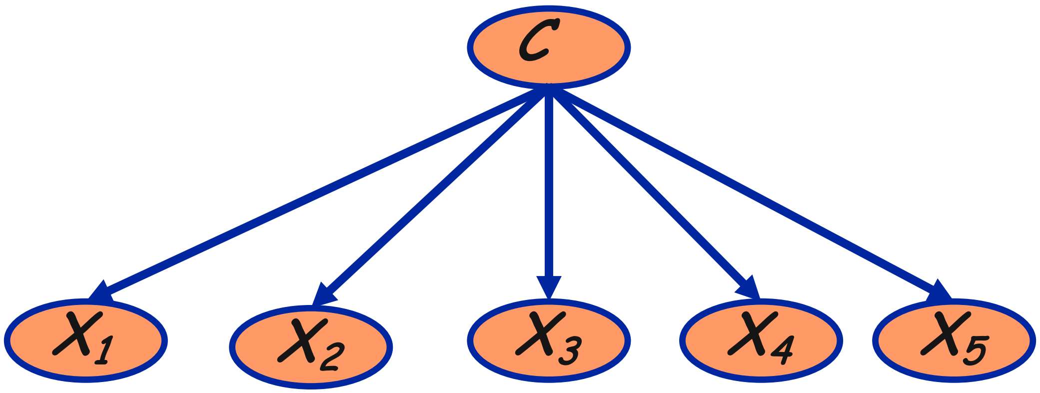
KL-divergence

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

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

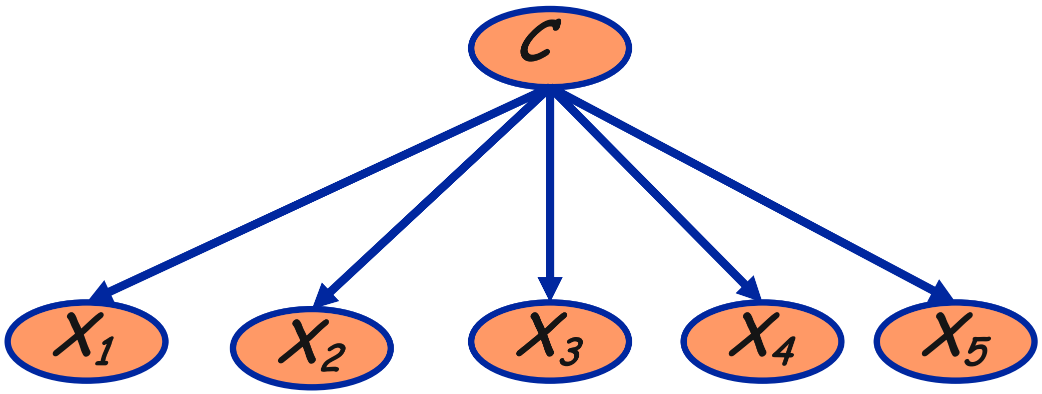


- **Conditional Independence Assumption:** Features are independent of each other given the class:

  $P(C|\mathbf{X}) \sim P(\mathbf{X}|C)\, P(C) = P(X_1|C)\, P(X_2|C) \ldots P(X_5|C)\, P(C)$

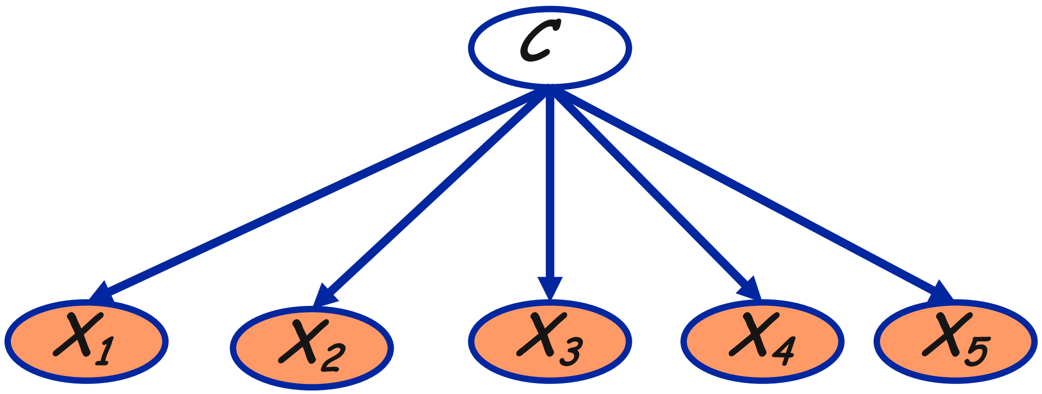
- For language, assume $P(\sim X_j|C) = 1$
- Use MLE or MAP to estimate the parameters

Gaussian Naïve Bayes Classifier



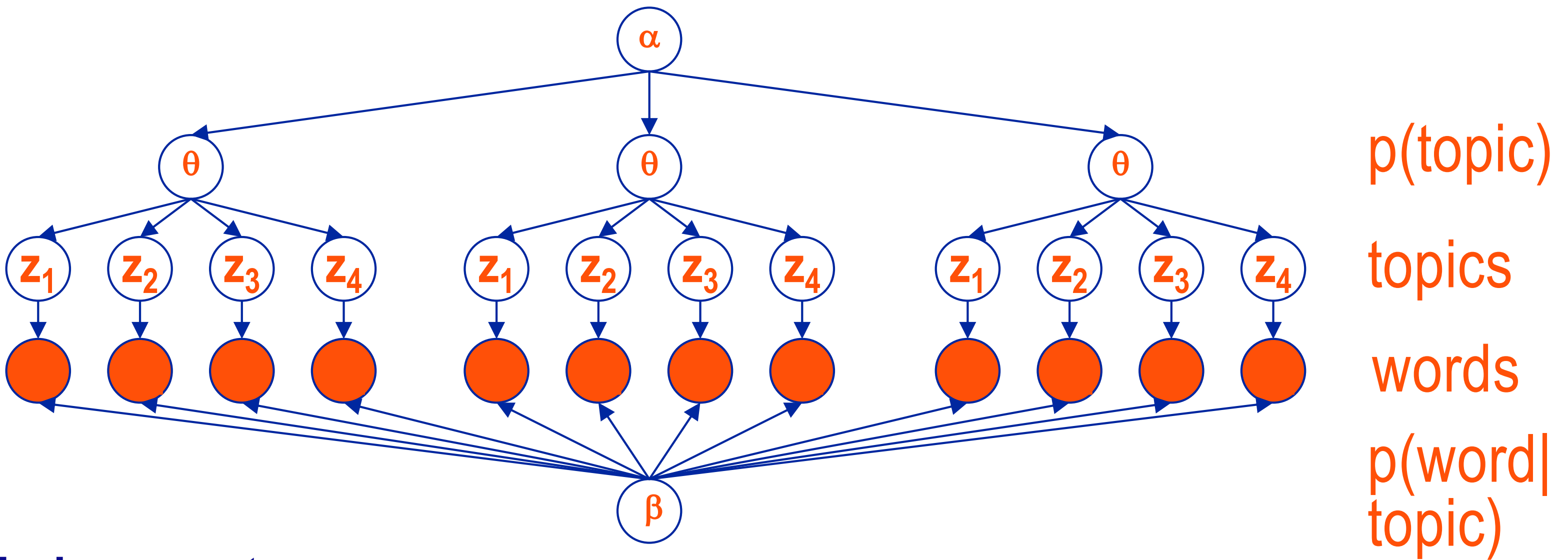
- $P(\mathbf{X}|C)$: $N(\boldsymbol{\mu}_C, \Sigma_C)$
  - Σ_C is diagonal (= conditional independence)
- $P(C|\mathbf{X}) \sim P(\mathbf{X}|C)\, P(C) = P(X_1|C)\, P(X_2|C) \ldots P(X_5|C)\, P(C)$

Gaussian Mixture Model



- $\boldsymbol{X} \sim N(\mu_C, \Sigma_C)$
- Now, C is not observed and Σ_C can have any form
- What does the E step compute?
- What does the M step compute?

The LDA Model (for 3 docs)

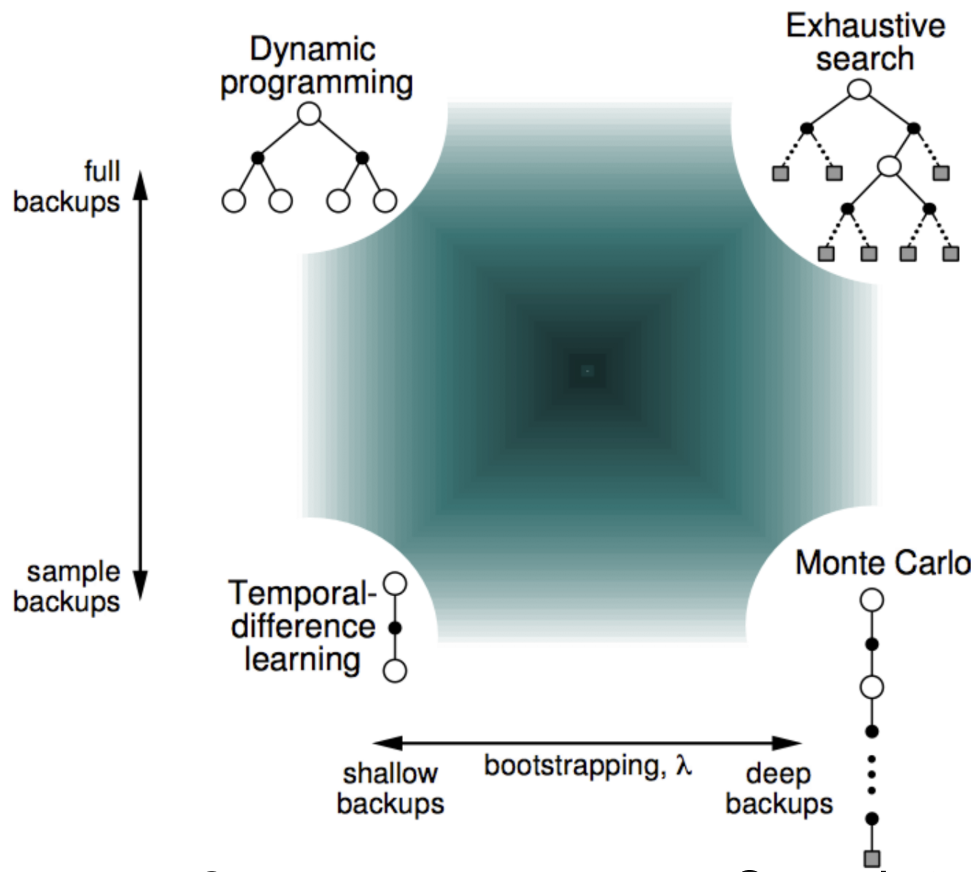


- **For each document,**
  - Choose the topic distribution $\theta \sim \text{Dirichlet}(\alpha)$
  - For each of the N words w_n:
    - Choose a topic $z \sim \text{Multinomial}(\theta)$
    - Then choose a word $w_n \sim \text{Multinomial}(\beta_z)$
      - Where each topic has a different parameter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

Reinforcement Learning

Response to all possible actions

Model-based



Response to one action

Model-free

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:** ϵ-greedy, on policy
  - **Q-learning:** ϵ-greedy action; then optimal (greedy)

Bellman's Equation

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

- s **state** s' **next state**
- $v_\pi(s)$ **value**
- $\pi(a|s)$ **policy (stochastic)**
- γ **discount factor**
- r **reward**
- $p(s'|s,a)$ **model**

Bellman's Equation

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

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

Bellman's Equation

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

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

$a = \pi(s)$
if deterministic

$v_\pi(s) := v_\pi(s) + \eta\,(r + \gamma\, v_\pi(s'))$ *TD(0)* estimation

Q-learning

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

Update

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

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.

wikipedia

Deep Q-Learning (DQL)

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

Update this

To be closer to new value estimate

Represent *Q* with a neural net

s, a **can be one-hot or real valued**