Final Review

2019

Most popular topics

- **Autoencoders**
- **EM**
- **Bayesian Belief nets**
  - Bayes nets, Naïve Bayes, LDA, HMM, MDP
- **Reinforcement learning**
  - model-based, model-free, on-policy, off-policy
  - Q-learning vs SARSA
- **AutoML**

Autoencoders

- PCA
- ICA
- Neural net autoencoder

PCA minimizes Distortion

- **First subtract off the average x from all the x_i**
  - From here, we'll assume this has been done
- **Approximate x in terms of an *orthonormal basis* v**
  - $X = UDV^T = ZV^T$ or $\hat{x}_i = \Sigma_k \; z_{ik} \; v_k$
- **Distortion = Reconstruction Error**
  - $||X - ZV^T||_2 = ||X - (XV) V^T||_2$

$$\sum_{i=1}^{n} ||\mathbf{x}^i - \hat{\mathbf{x}}^i||_2^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} (x_j^i - \hat{x}_j^i)^2$$

Independent Components Analysis (ICA)

- Given **X**, find **S** and **W** such that components $\mathbf{s}_j$ of **S** = **XW** are "as independent of each other as possible"
- Reconstruct $\mathbf{X} \sim \mathbf{S}\mathbf{W}^{+} = (\mathbf{X}\mathbf{W})\mathbf{W}^{+}$
  - **S** like *principal component scores*
  - $\mathbf{W}^{+}$ like *loadings*
  - $\boldsymbol{x} \sim \sum_j s_j \boldsymbol{w}_j^{+}$

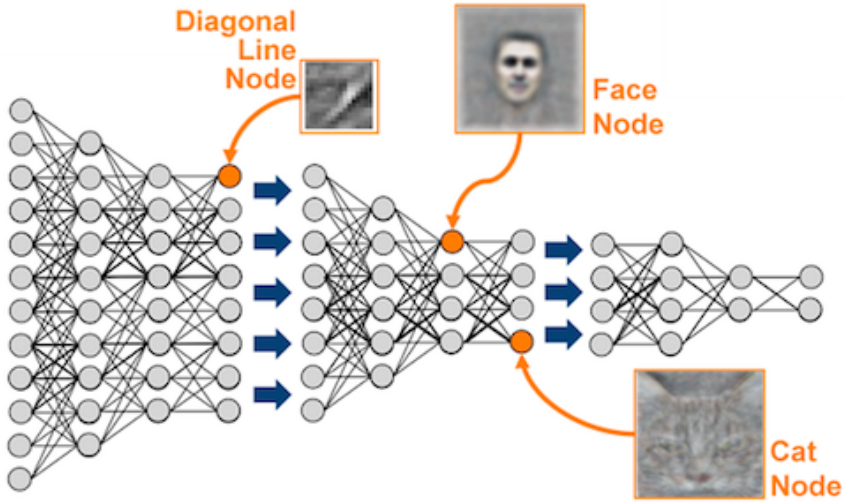
- **Auto-encoder** – nonlinear generalization that "encodes" **X** as **S** and then "decodes" it

Reconstruction ICA (RICA)

- **RICA: find W to minimize *reconstruction error:***
  - $||X - SW^{+}||_2 = ||X - (XW)W^{+}||_2$
- **And minimize**
  - Mutual information between sources S = XW
  - $MI(s_1,s_2,\ldots s_k) = KL(p(s_1,s_2,\ldots s_k) \,||\, p(s_1)p(s_2) \ldots p(s_k))$

“Deep” Autoencoders

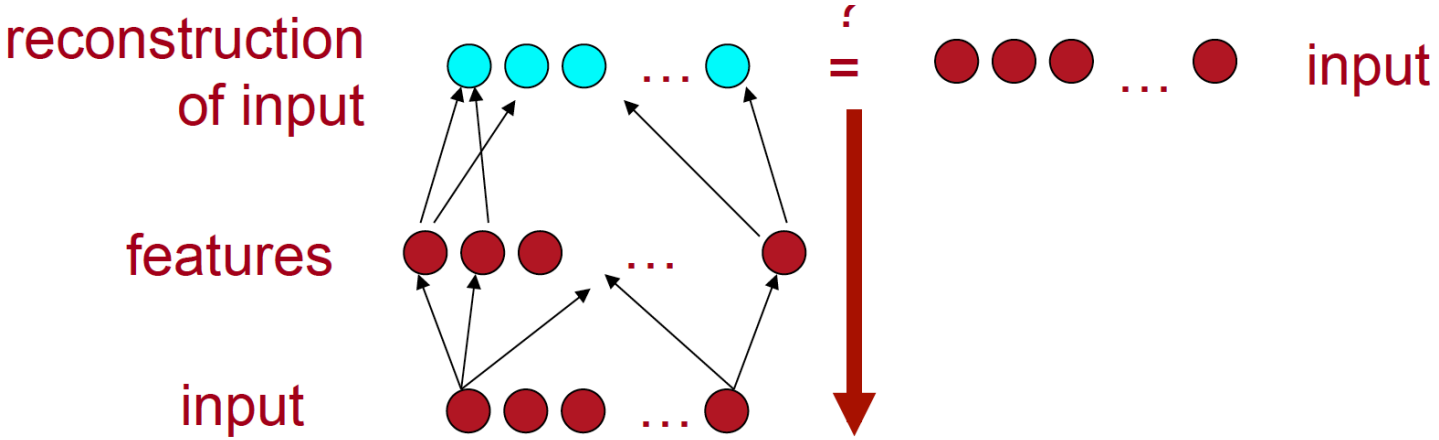
- **Take same image as input and output**
  - often adds noise to the input
    (*denoising auto-encoder*)
    remove pixels or words
- **Learn weights to minimize the reconstruction error**
- **This can be done repeatedly (”stacked”)**
- **Use for semi-supervised learning**



from Socher and Manning

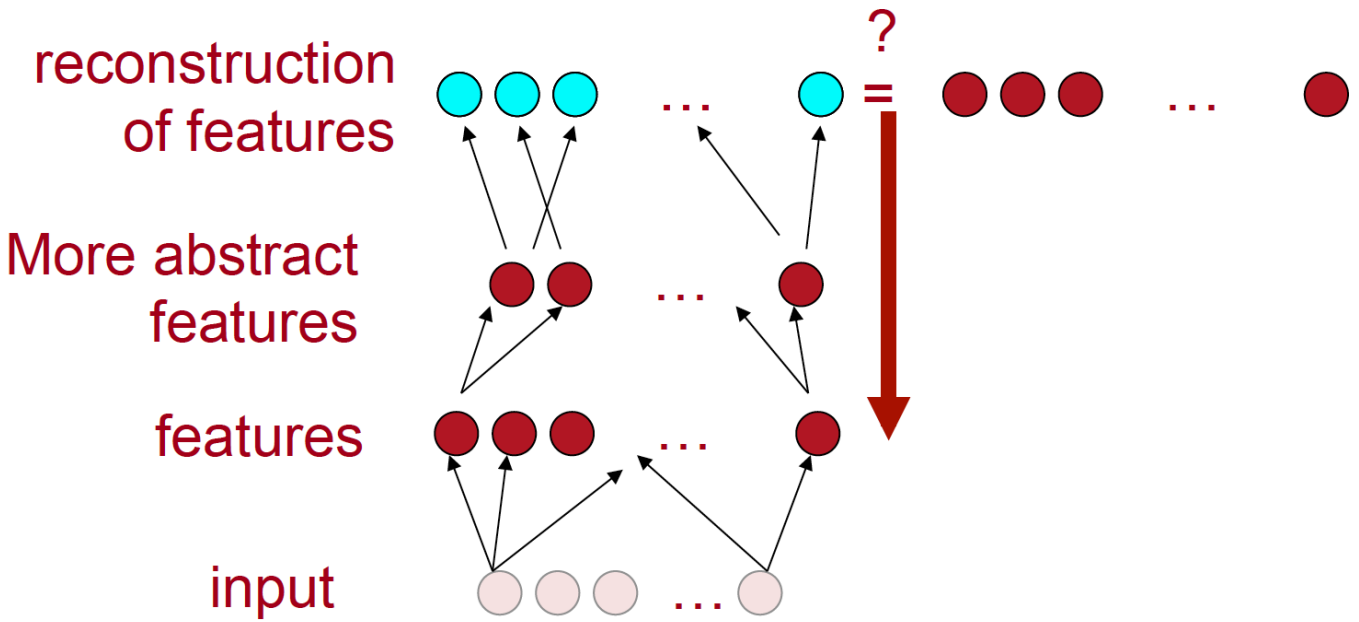
Can increase or decrease input dimensionality

Stacking for deep autoencoders



from Socher and Manning

Stacking for deep autoencoders

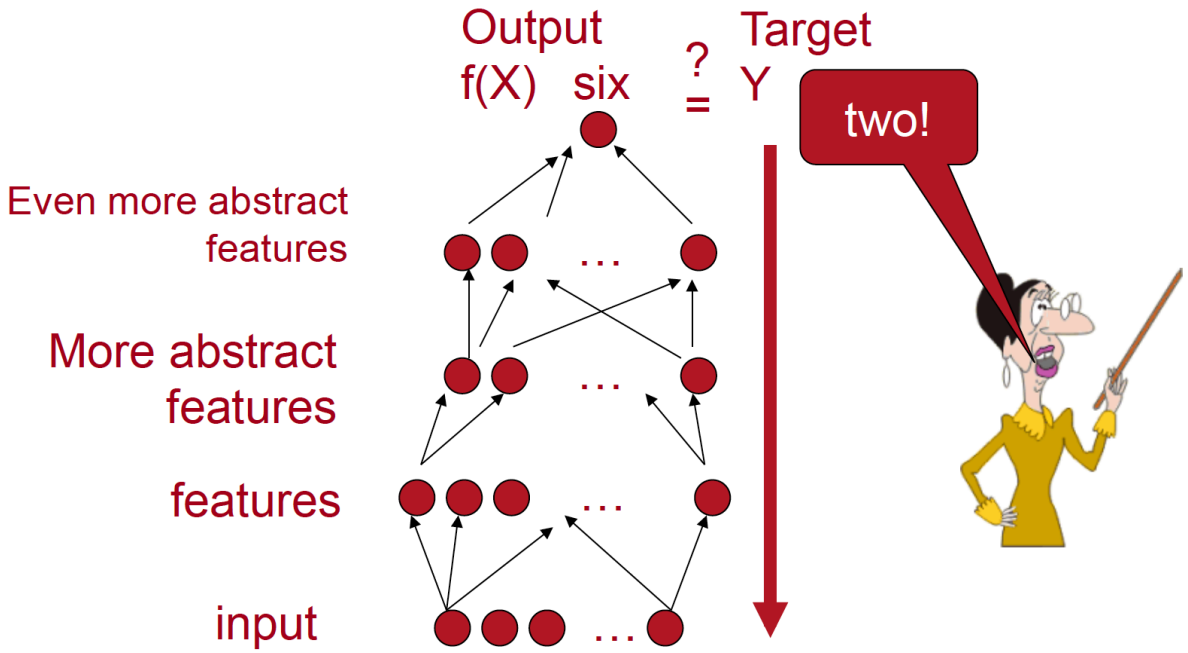


Now learn to reconstruct the features (using more abstract ones)

from Socher and Manning

Stacking for deep autoencoders

- Recurse – many layers deep
- Can be used to initialize supervised learning

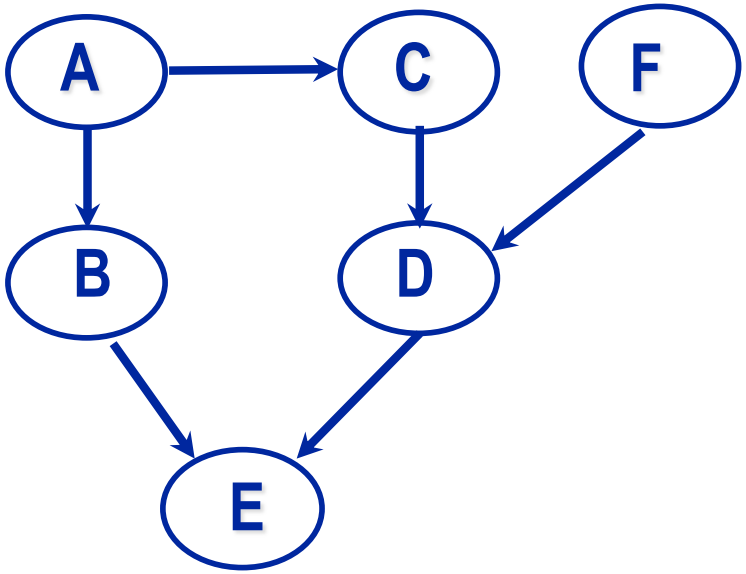


from Socher and Manning

Bayesian Belief Nets

- **NB**
  - Binary or real-valued X's;
- **GMM**
  - Different model forms
- **LDA**
- **HMM/MDP**

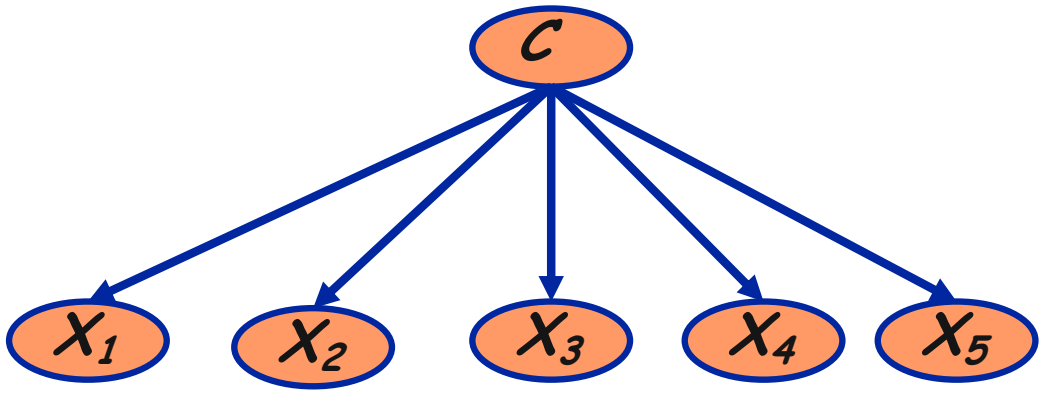
Example Belief Net



$B \perp C \mid A?$

$B \perp D \mid E?$

The Naïve Bayes Classifier

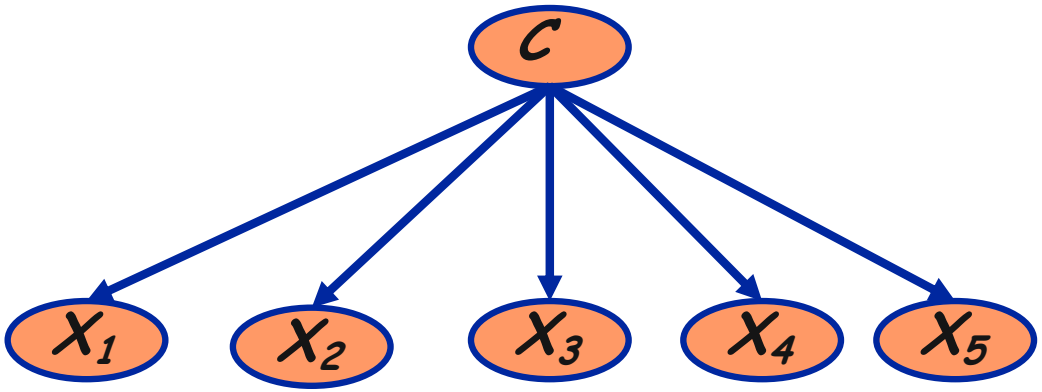


- Conditional Independence Assumption: Features are independent of each other given the class:

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

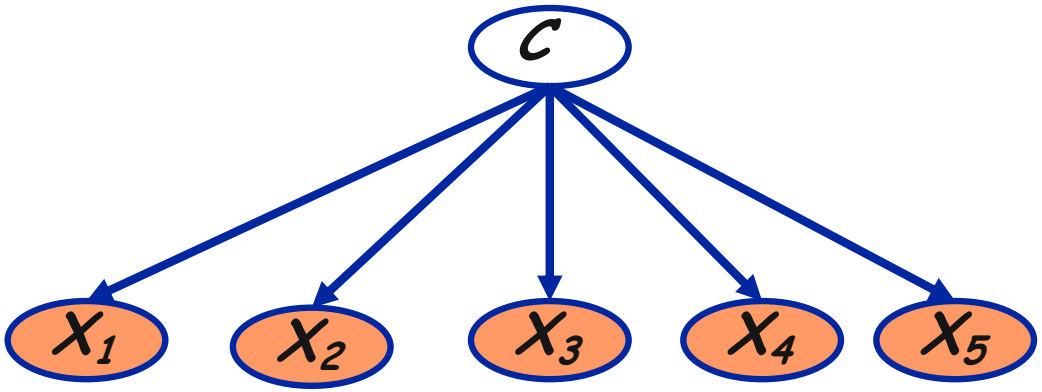
- For language, assume $P(\sim X_j|C) = 1$
- Use MLE or MAP to estimate the parameters

Gaussian Naïve Bayes Classifier



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

Gaussian Mixture Model

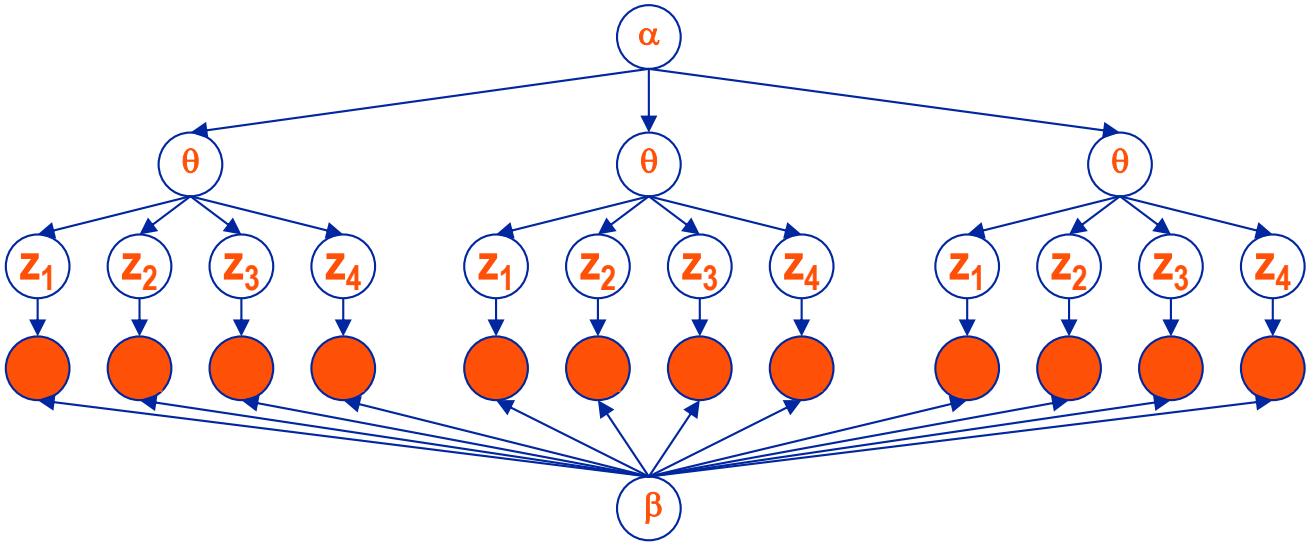


- $X \sim N(\mu_C, \Sigma_C)$
- Now, C is not observed and Σ_C can have any form

EM

- **Used for:**
  - Clustering (GMM, LDA)
  - Missing data
- **E Step**
  - Estimate the values of the missing data
- **M Step**
  - Do MLE (or MAP) estimation of the parameters

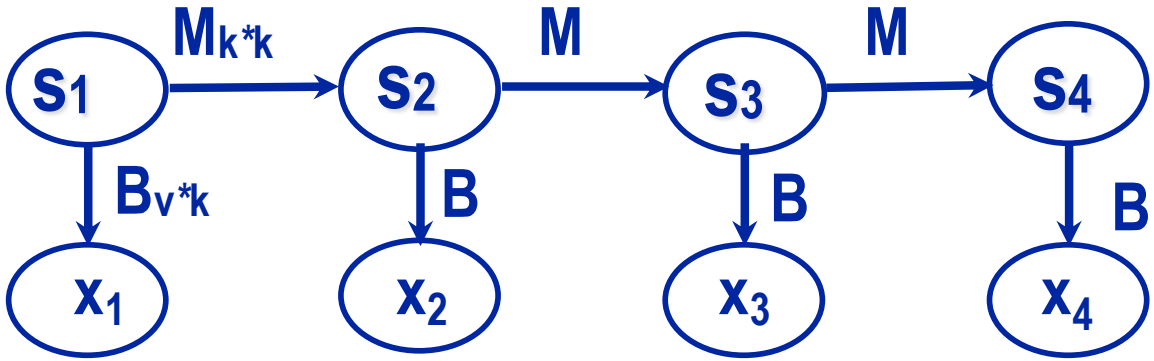
The LDA Model (for 3 docs)



- **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

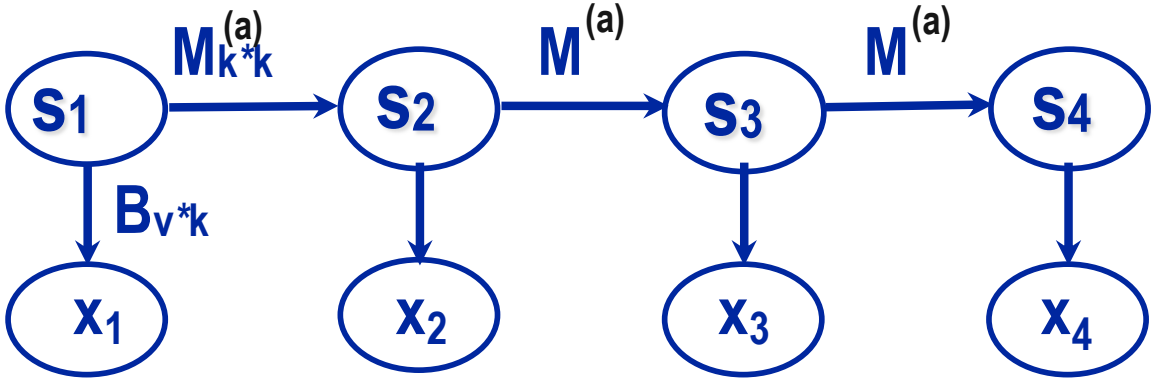
MDPs generalize Markov Models

- **MM**



M = Markov transition matrix
B = emission matrix

- **MDP**

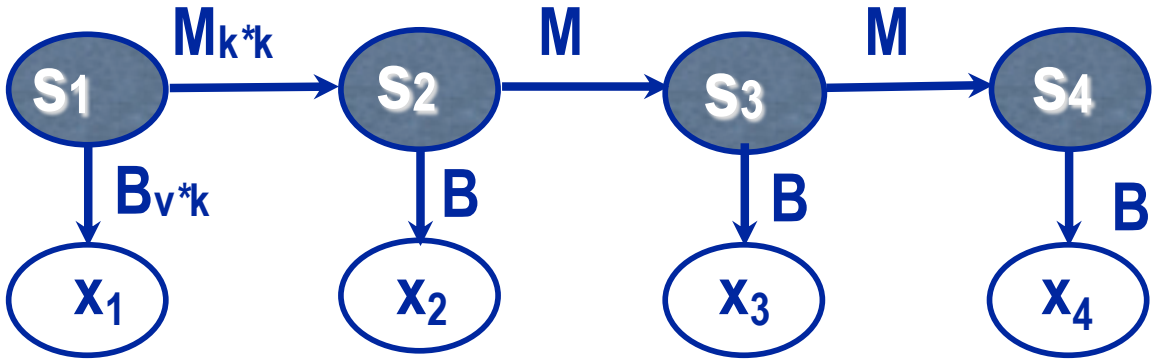


$\mathbf{M}^{(a)}$ Different transition matrix for each action, a

Emission, $\mathbf{x}_t$, includes reward, R_t

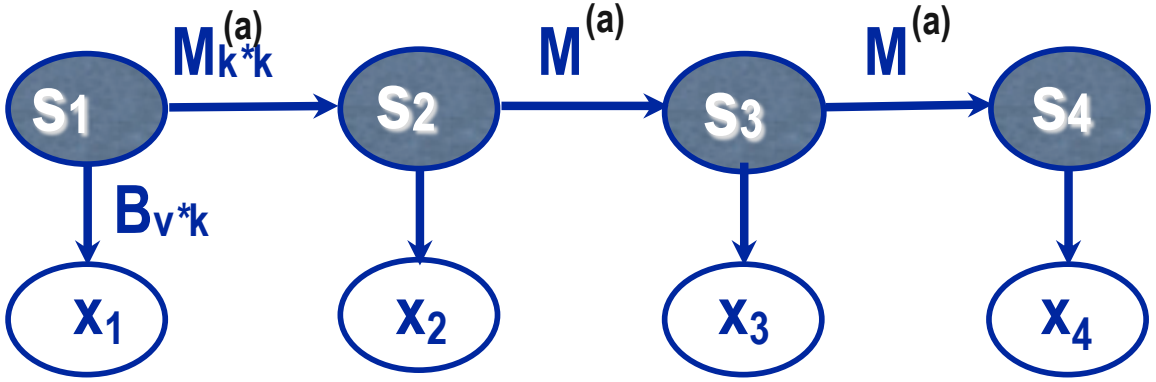
POMDPs generalize HMMs

- **HMM**



M = Markov transition matrix
B = emission matrix

- **POMDP**



$\mathbf{M}^{(a)}$ Different transition matrix for each action, a

Emission, $\mathbf{x}_t$, includes reward, R_t

HMM Esimation

- What is hidden in an HMM?
- What are the parameters?
- How is they estimated?
- What makes the estimation different from other belief nets?

Reinforcement Learning

- **Model-based vs. model-free**
  - Model-based = MDP
- **On-policy vs. off-policy**
  - Why learn off-policy?
- **One step ahead vs. Monte Carlo**
  - TD(0) = take one step, make new prediction
- **SARSA vs. Q-learning**
  - SARSA: ϵ-greedy, on policy
  - Q-learning: (e.g.) ϵ-greedy action; then optimal (greedy)

Q-learning

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{(1-\alpha)}_{\text{We just used 1 here}} \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

After we take an ϵ-greedy action

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)

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

Update this

To be closer to new value estimate

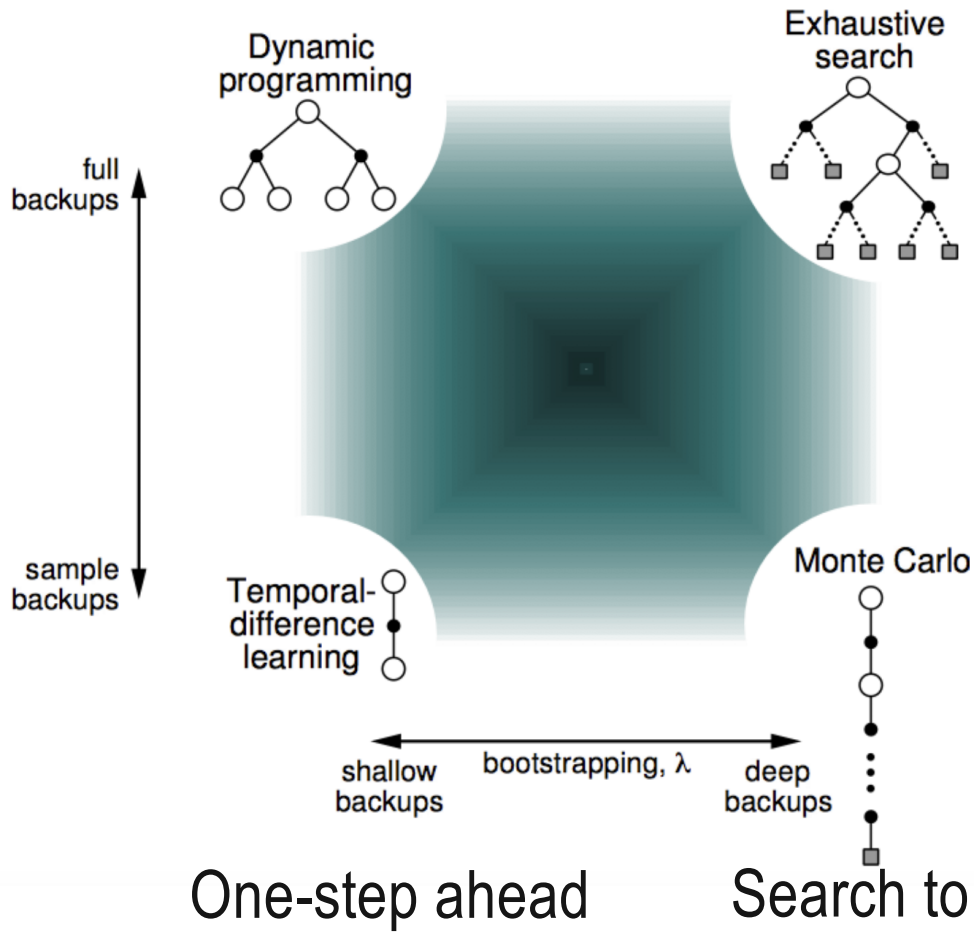
Represent *Q* with a neural net

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

Summary

Response to all possible actions

Response to one possible actions



Model-based

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

Auto-Sklearn

- **Combined Algorithm Selection and Hyperparameter (CASH) Optimization**
  - Fit a random forest model predicting performance from hyperparameters
    - Based on 38 metafeatures of 140 datasets
    - Can also use text of problem desciption
  - Use it to find the optimum
    - Search starting from this ‘optimumum’
- Use **Ensemble** of the 50 best classifiers
  - Stagewise selection: ‘unit weight’ (with repeats)