1. Markov Models
X_t given, $X_{t+1} \perp X_0, \cdots X_{t-1}$ (只有 given 上个).

$P(X_0, \cdots X_t) = P(X_0) \prod_t \underbrace{P(X_t | X_{t-1})}$ ← transition model.

$P(X_t) = \sum_{x_{t-1}} P(X_{t-1} = x_{t-1}) P(X_t | X_{t-1} = x_{t-1})$ web link analysis, Gibbs sampling

$P_\infty(X) = P_{\infty+1}(X) = \sum_x P(X|x) P_\infty(x)$. ← stationary distribution

2. HMM
X_t given, $e_t \perp$ everything else. + MM 的假设.

$P(X_0, X_1, E_1, \cdots X_T, E_T) = P(X_0) \prod_{t=1:T} P(X_t | X_{t-1}) \underbrace{P(E_t | X_t)}$ ← emission model

Filtering: $P(X_t | e_{1:t})$ $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

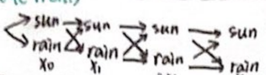
$\boxed{P(X_{t+1} | e_{1:t+1})} = P(X_{t+1} | e_{1:t}, e_{t+1}) = \alpha P(e_{t+1} | X_{t+1}, e_{1:t}) P(X_{t+1} | e_{1:t})$

$= \alpha P(e_{t+1} | X_{t+1}) P(X_{t+1} | e_{1:t})$ $O(|X|^2)$ time

$= \alpha P(e_{t+1} | X_{t+1}) \sum_{x_t} P(x_t | e_{1:t}) P(X_{t+1} | x_t, e_{1:t})$

$= \alpha \boxed{P(e_{t+1} | X_{t+1})} \boxed{\sum_{x_t} \boxed{P(x_t | e_{1:t})} P(X_{t+1} | x_t)}$ ← predict

State trellis



每个 arc $= P(x_t | x_{t-1}) \cdot P(e_t | x_t)$

每条 path weight 乘积 $= P(x_{0:t}, e_{1:t}) \propto P(x|e)$

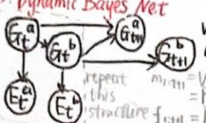
Filtering 求所有可能路径之和:

$P(X_{t+1} | e_{1:t+1}) = \sum_{x_{1:t}} P(x_{1:t+1} | e_{1:t+1})$.

Viterbi 求最好的路. $\arg\max_{x_{0:t}} P(x_{0:t} | e_{1:t})$ (Most likely explanation)

每个结点: 最好的 path 的 weight 之积. $O(|X|^2 T)$ time (上一结点 × arc weight) $O(|X|T)$ space.

3. Dynamic Bayes Net



want to keep track of 多变量 over time. $X_t | X_{t-1}$.

repeat this structure $m_{1:t+1} = \text{VITERBI}(m_{1:t}, e_{t+1})$ $= P(e_{t+1} | X_{t+1}) \max_{x_t} P(X_{t+1} | x_t) m_{1:t}[x_t]$

$f_{1:t+1} = \text{FORWARD}(f_{1:t}, e_{t+1})$

4. Particle Filtering: Approximate inference
Propagate Forward $x_{t+1} \sim P(X_{t+1} | x_t)$ 根据随机数区间 move particle. (consistent)

Observe $W = P(e_t | x_t)$ (如果一直这样更新 = likelihood weight)

Resample 根据 W 重新选 N 个粒子 $w = 1$.

$P(x)$ 根据粒子数估计 $= \frac{n}{N}$. n 可能 $= 0$. $N \ll |X|$ supposed.

---

1. Probablistic Relational Models
Logical language: Frame. (typed relational knowledge) ∈FOL

the type of objs and their valid relations and attributes

Relational skeleton σ: set of objects in each class; relations between them; attribute values (unknown).

BN template for object class: object attributes can depend on attribute of related objs. cp distribution given instantiation

Probablistic Relation Model (BN template) + σ → unrolled BN.

PRM with AU: Aggregate dependency: sum, min, max, avg, mode, count

$P(I | \sigma, S, \theta) = \prod_{x \in \sigma} \prod_{x.A} P(x.A | \text{parents}_{S,\sigma}(x.A))$ PRM (S, θ)

attributes objects → $x \in \sigma$ $x.A$ ← attributes

Structural Uncertainty → use existence uncertainty (unknown relation) ($P \approx 0$ for impossible)

2. Markov Logic
Logical language: FOL.

Probablistic Language: Markov Network 无向图的联合分布.

$p(x) = \frac{1}{Z} \prod_c \psi_c(x_c)$ c: clique

Markov Logic Network MLN: (F, w) pairs. (formula ∈ FOL, weight ∈ ℝ)

+ constraints, 可以生成: (proposionalization)

- One node for each grounding of each predicate in the MLN
- One clique for each grounding of $F \in$ MLN, potential $= \exp(w)$ for 满足 F 的赋值; 1 otherwise.

MLN is template for ground MN.

$P(x) = \frac{1}{Z} e^{(\sum_i w_i n_i(x))}$ → # of true groundings of formula i in x.

Weight ↗, stronger constraint. soft constraint, 而非 logical KB 的 hard constraint: 一旦一个公式不成立, 就不可能

MLN (MN template) + propositionalized KB = unrolled MN

Relation to FOL ① weight $= \infty \Rightarrow$ FOL. $P(x) > 0 \Leftrightarrow x$ 满足 KB.

② Markov Logic allows contradiction between formulas.

Relation to PRM ① MLN general ++, flexible ++.

② PRM → MLN 将 CPT 每一行转化成一个 FOL, $w = \log(P)$.

e.g. $\log 0.1$: $\forall x, y$ hasReview$(x,y) \wedge Q(x,F) \wedge M(y,F) \wedge A(x,T)$

3. Inference
大, 高度相连

Naive: unroll to a BN/MN and run inference algo (e.g. VE)

Lifted Inference: group similar r.v. at the FOL level, handle them at the same time.

---

1. Sequence labeling
known: $C = \{c_1, c_2, \cdots, c_j\}$ labels

Input: sentence $s = \{x^1, x^2, \cdots, x^n\}$ Output: $y^i \xrightarrow{\text{predict}} c_i \in C$

① Simplest: most frequent label 未考虑附近词的关系.

② HMM evidence: x^i (words) value: y^i (labels) VITERBI 求解.

$P(y_1) = P(y_1 | Y_0 = \text{START})$ $P(y_n)$ $Y_n \to Y_{n+1} = \text{STOP}$.

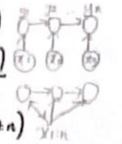
Pro: adjacent relation ++ Cons: contextual info 无.

③ MEMM (Max-Entropy Markov models)

普通 HMM:

$P(y_{1:n} | x_{1:n}) = P(y_1 | x_1) \prod_{t=2}^{n} P(y_t | y_{t-1}, x_t)$

$P(y_t | y_{t-1}, x_t) = \frac{\exp(w^T f(y_{t-1}, y_t, x_t))}{Z(y_{t-1}, x_t)}$.



MEMM:

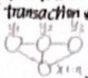
$P(y_{1:n} | x_{1:n}) = P(y_1 | x_{1:n}) \prod_{t=2}^{n} P(y_t | y_{t-1}, x_{1:n})$

$P(y_t | y_{t-1}, x_{1:n}) = \frac{\exp(w^T f(y_{t-1}, y_t, x_{1:n}))}{Z(y_{t-1}, x_{1:n})}$

Pro: contextual info, relations adjacent lablels ++

Con: Label Bias Problem: prefer # of transaction ↓ 的状态

solution: local p → local potentials.



无向图

④ CRF (Conditional Random Field)

$P(y_{1:n} | x_{1:n}) = \frac{1}{Z(x_{1:n})} \prod_{t=1}^{n} \exp(w^T f(y_{t-1}, y_t, x_{1:n}))$

⇒ Global normalization instead of local. Viterbi for inference.

2. Formal Grammer
① Constituency (成分): 可以被认为是一个单元 act 的 words.

相同 unit: internal, externel (与其他 unit 相关)

e.g. noun phrase: determiner + modifier + noun + modifier(s). (限定词 修饰语 名词 修饰语)

有 percedes verbs (ex)

② Grammer {constituents} {rules of combine cs}

③ Context-free grammers (CFGs) / Phrase structure grammer

simplest form of grammer formalisms.

- a set Σ of terminals (words)
- a set N of nonterminals (phrases)
- a start symbol $S \in N$
- a set R of production rules: specify how a nonterminal can produce a string of terminals and/or nonterminals.

④ Sentence Parsing a string + a gramer ↦ 1/more parse tree

If no parse tree is found: the string ∉ the language.

⑤ Probablistic Grammers (stochastic grammers)

Each rule + p. $\alpha \to \beta$: $P(\alpha \to \beta | \alpha)$.

p of a parse tree: product of p of all rules used in generating the parse tree.

⑥ Chomsky Normal Form (CNF) Intermidiate non-terminals: doesn't occur anywhere else in the grammer.

- $A \to BC$ e.g. CFG: $S \to ABC$ CNF: $S \to XC$, $X \to AB$.
- $A \to w$ e.g. $S \to$ book | include | prefer ($w=0.2$, $w=0.2$, $w=1$)

parse tree 不同了, → binary tree.

3. Parsing
brute-force: n words → C_{n-1} exponetial.

① DP subproblem: Parse$(x_{i:j})$

CYK (Cocke-Younger-Kasami Algo) 适用于 SFG in CNF.

Base case: $A \to w_i$ at $[i-1, i]$ [顺序要对]

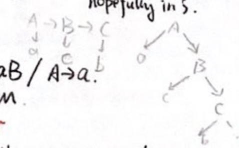
Recursion: A in $[i, j] \Leftrightarrow A \to BC$, B in $[i, k]$, $[k, j]$

ambiguity: 选概率最大的

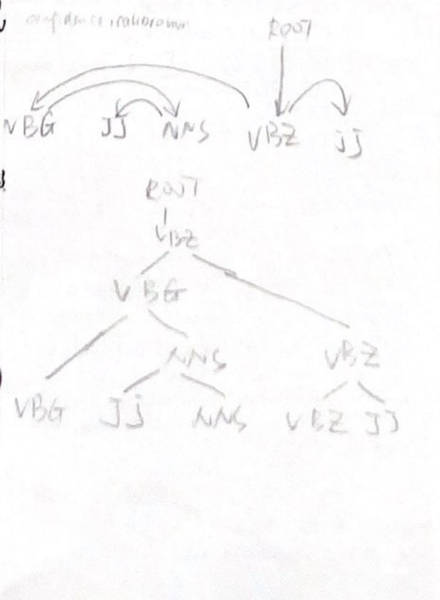
$[0, n]$ 结果. hopefully in S.

4. Regular Grammer
Production rules $A \to aB$ / $A \to a$.

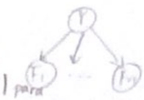
Probablistic RG = HMM



5. Dependency Grammer
Focus on binary relations among words in a sentence



1. Classification: NB
① Naive Bayes feature 间



$P(Y, F_1, \cdots F_n) = P(Y) \prod_i P(F_i | Y)$

$|Y| \times |F|^n$ values $|Y|$ para $n \times |F| \times |Y|$ para

Total para ∝ n, tree structure: linear infer time.

$P(Y, F_1, \cdots F_n) = \begin{bmatrix} P(y_1, f_1, \cdots f_n) \\ P(y_2, f_1, \cdots f_n) \\ \vdots \\ P(y_k, f_1, \cdots f_n) \end{bmatrix} \Rightarrow \begin{bmatrix} P(y_1) \prod_i P(f_i | y_1) \\ \vdots \\ P(y_k) \prod_i P(f_i | y_k) \end{bmatrix}$

⇒ normalize by $P(f_1, \cdots f_n) \Rightarrow P(Y | f_1, \cdots f_n)$

$\theta = (\underbrace{P(Y)}_{\text{prior over labels}}, P(F_i | Y))$ — evidence vars

estimate local CPTs.

For text, F_i 是第 i 个位置的 word.

Bag of words model: $P(F_i | Y) = P(W_i | Y)$ 是相同的.

each position has the same distribution

⇒ a model insensitive to word order or reorder.

② Testing and Training (train and test, 多个 held out)

Train - test - held out ← tune hyperparas: $k, \alpha, \cdots$

↑ learn paras ↑ compute accuracy (never peek at test set!)

relative frequency: overfit. unseen → 0 不合理

Para estimation: Maximum-likelihood estimation

$P(D | \theta) = \theta^{\alpha_H} (1-\theta)^{\alpha_T}$ Train test held-out

$\hat\theta = \arg\max_\theta \ln P(D|\theta) = \arg\max_\theta \ln \theta^{\alpha_H} (1-\theta)^{\alpha_T}$

$\frac{d \ln P(D|\theta)}{d\theta} = \frac{d}{d\theta} [\alpha_H \ln\theta + \alpha_T \ln(1-\theta)] = \frac{\alpha_H}{\theta} - \frac{\alpha_T}{1-\theta} = 0$

$\therefore \hat\theta_{MLE} = \frac{\alpha_H}{\alpha_H + \alpha_T}$

Likelihood of data (r) (r) (b) $P_{ML}(r) = \frac{2}{3}$

$L(x, \theta) = \prod_i P_\theta(x_i) = \theta \cdot \theta (1-\theta)$

③ Laplace Smoothing pretend 你看到每个 outcome 多了 k 次

$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$ k: strength of the prior

$P_{LAP,k}(x|y) = \frac{c(x,y) + k}{c(y) + k|X|}$ $|X|, |Y|$ 很大时表现不好. 也从数据中学 $P(X)$.

④ Linear Interpolation

$P_{LIN}(x|y) = \alpha \hat P(x|y) + (1-\alpha) \hat P(x)$

⑤ Confidence from a classifier

Posterior over the top label:

confidence$(x) = \max_y P(y|x)$.

2. Classification: NN
① Perception (linear classifier) $y = \arg\max_y w_y f(x)$ weight vector: hyperplane feature vector

activation$_w(x) = \sum_i w_i f_i(x) = w f(x)$. $\begin{cases} > 0 & + \\ < 0 & - \end{cases}$

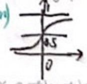
$y = y^*$: no change

$y \neq y^*$: $w = w + y^* \cdot f$ (加/减去 feature vector) 理应是 negative 时

mistakes bound: mistakes $< \frac{k}{\delta^2}$ ← margin/degree of separability

② + Probablistic (logistic regression)

$z = w \cdot f(x)$, $\phi(z) = \frac{1}{1 + e^{-z}}$



$z_1 \to \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$ ← softmax activation

original activation

$\max_w ll(w) = \max_w \sum_i \log P(y^{(i)} | x^{(i)}; w)$.

$P(y^{(i)} | x^{(i)}; w) = \frac{e^{w_y f(x^{(i)})}}{\sum_y e^{w_y \cdot f(x^{(i)})}}$ MLL = Logistic Regression

③ Solving $\max_w ll(w)$ $\max g(w + \Delta w)$

Gradient ascent: $w \leftarrow w + \alpha \nabla_w g(w)$ (gradient direction = steepest direction) gradient $\nabla_w g(w) = \begin{bmatrix} \frac{\partial g}{\partial w_1}(w) \\ \frac{\partial g}{\partial w_2}(w) \end{bmatrix}$

On max likelihood:

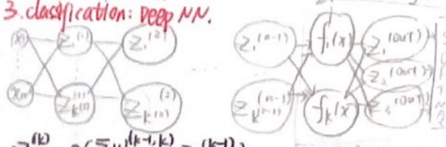
```
init w
for iter = 1, 2, ...
    w ← w + α · Σ ∇ log P(y^(i) | x^(i); w)
```

→ $P(y_1 | x_1; w)$ → $P(y_2 | x_2; w)$ → $P(y_3 | x_3; w)$

stochastic: 一次只 pick random j

mini-batch: 一次 pick random subset

3. Classification: Deep NN.



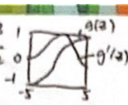
$z_i^{(k)} = g(\sum_j W_{i,j}^{(k-1,k)} z_j^{(k-1)})$

g: nonlinear activation f.

sigmoid $g(z) = \frac{1}{1 + e^{-z}}$ $g'(z) = g(z)(1 - g(z))$

Hyperbolic Tangent $g(z)=\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$

$g'(z)=1-g(z)^2$



Rectified Linear Unit (ReLU)

$g(z)=\max(0,z)$ $g'(z)=\begin{cases}1, & z>0\\ 0, & \text{otherwise}\end{cases}$

CNN ① local connectivity of hidden units
② share weights across certain units
③ pool hidden units in the same neighborhood
④ interleave feature extraction and pooling

Gradient Descent

① Pure analytically: give simbolic answer. hard to compute.
② Finite difference approximation: $O(NM)$
③ Back Propagation: exact answer. hard to implement $O(NM)$
分成 $g=b+c$, $b=aw_2$, $a=w_1^3$, $c=3w_1$ 逐个计算.

4. Regression

Linear Regression

prediction $h_w(x)=w_0+w_1x$

L2 loss function: $L(w)=\sum_i (y_i-h_w(x_i))^2=\sum_i (y_i-w^Tx_i)^2$

$w^*=(X^TX)^{-1}X^Ty$

Regularization alleviate overfitting

LASSO (Least absolute shrinkage and selection operator) (L1)

$L(w)=\sum_i (y_i-w^Tx_i)^2+\lambda\sum_k |w_k|$ 0 weight + sparse, large weights, Laplace as weight prior

Ridge Regression (L2)

$L(w)=\sum (y_i-w^Tx_i)^2+\lambda\sum_k w_k^2$ small weights, Gaussian

Unsupervised ML

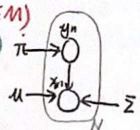
1. k-means

the point y with min squared Euclidean distance to a set of point is their mean.

问题: overlap clusters, 有些 cluster 会很宽, dist 不好处理.

2. Expectation-Maximization (EM)

① Gaussian Mixture Model (GMM)



$p(x)=\sum_{k=1}^{K}\pi_k\,N(x\mid u_k,\Sigma_k)$ mixing coefficients

$\forall k: \pi_k\geq 0$ $\sum_{k=1}^{K}\pi_k=1$.

P(Y): Distribution over k components (clusters)

P(X|Y): Each component generates data from a multi-var Guassian with u_i and Σ_i.

Each point is sampled from a generative process.
① choose component i with $p=\pi_i$ (random)
② Generate data point from $N(x\mid u_i,\Sigma_i)$

② Don't know label y:

$\max_\theta U=\prod_j P(x_j)=\prod_j\sum_i P(y_j=i,x_j)=\prod_j\sum_i \pi_i N(x_j\mid u_i,\Sigma_i)$

③ EM (Expectation-Maximization)
- Pick k random cluster models (Gaussians)
- Loop until convergence:
  - Assign data instances proportionately to different models
  - Revise each model based on its points

Object Function: $\arg\max_\theta \prod_j\sum_{i=1}^{k} P(y_j=i,x_j\mid\theta)$ → $\sum_j \log$

Data: $\{x_j \mid j=1\cdots n\}$

E-step: compute Expectation to 'fill in' missing y values according to current para: θ.
⇒ 每个 sample 算 $P(y_j=i\mid x_j,\theta)$.

M-step: reestimate the paras with weighted MLE
set $\theta=\arg\max_\theta$ (obj) marginal likelihood

④ EM → kmeans when
① All Guassians are spherical, w_i, Σ_i same. ($\theta=\{u_i\}$)
② hard assignment of points to Guassians. ($\Sigma_i\to 0$)

$Q(s_t,a_t)\leftarrow(1-\alpha)Q(s_t,a_t)+\alpha(r_t+\gamma\max_{a'}Q(s_{t+1},a'))$

Markov Decision Process

1. MDP Model

a set of states S, a set of actions A,
a transition function $T(s,a,s')=P(s'\mid s,a)$, Also called models/dynamics
a reward function $R(s,a,s')$ ($/R(s)/R(s')$),
a start state, sometimes a terminal state.

2. Quantities

Policy $\pi: S\to A$ π^* optimize expected utilities.

Utility of states: $V^*(s)$ = expected utility starting in s and acting optimally (V value)

Utility of q state (s,a): $Q^*(s,a)$: $s\xrightarrow{a}\{s_1, s_2, \ldots\}$ 之后 act optimal 的 exp utility (Q value) optimal policy

$V^*(s)=\max_a Q^*(s,a)$

$Q^*(s,a)=\sum_{s'}T(s,a,s')[R(s,a,s')+\gamma V^*(s')]$ Bellman equation

$V^*(s)=\max_a\sum_{s'}T(s,a,s')[R(s,a,s')+\gamma V^*(s')]$

3. Value Iteration 慢 $O(S^2A)$; Policy 比 value 更早 converge. (max rarely change)

$V_{k+1}(s)\leftarrow\max_a\sum_{s'}T(s,a,s')[R(s,a,s')+\gamma V_k(s')]$

必会 converge to optimal (unique).

4. Policy extraction

$\pi^*(s)=\arg\max_a\sum_{s'}T(s,a,s')(R(s,a,s')+\gamma V^*(s'))$

$\pi^*(s)=\arg\max_a Q^*(s,a)$ ← (more useful, compute fast)

5. Policy Iteration optimal, converge faster, repeat ①②

① Policy evaluation $O(S^2A)$ 每次 iteration

a. iterative update

$V_0^\pi(s)=0$, $V_{k+1}^\pi(s)\leftarrow\sum_{s'}T(s,\pi(s),s')[R(s,\pi(s),s')+\gamma V_k^\pi(s')]$

直到 converge. 每次迭代 $O(S^2)$ [converge, 但不一定 optimal]

b. 无 max. Bellman equation 可直接用 linear system solver 解.

$V^\pi(s)=\sum_{s'}T(s,\pi(s),s')[R(s,\pi(s),s')+\gamma V^\pi(s')]$ ← 上一次算的

② Policy improvement one step look ahead:

$\pi_{i+1}(s)=\arg\max_a\sum_{s'}T(s,a,s')[R(s,a,s')+\gamma V^{\pi_i}(s')]$

6. 比较.

Reinforcement Learning

1. Model: MDP 无 T, R, online.

2. Model-based learning
① Learn Empirical MDP: 求 $\hat{T}(s,a,s')$, 然后 $\hat{R}(s,a,s')$.
② 解 MDP.

3. Model Free Learning

① Passive RL: policy evaluation. (values of)
- Direct evaluation: goal: compare states under π.
  Idea: average over samples.
  根据 policy π 走, everytime you visit a state, write down what the sum of discounted rewards turned out to be.
  Pro: eventually compute correct average values.
  Con: waste info about state connections, each state should be learned seperately, time long.
- Temporal Differente Learning: learn 马上 from every experience.
  $sample=R(s,\pi(s),s')+\gamma V^\pi(s')$
  $V^\pi(s)\leftarrow(1-\alpha)V^\pi(s)+\alpha\, sample=V^\pi(s)+\alpha(sample-V^\pi(s))$

+ exponetial moving avg

$\bar{x}_n=\frac{x_n+(1-\alpha)x_{n-1}+(1-\alpha)^2x_{n-2}+\cdots}{1+(1-\alpha)+(1-\alpha)^2+\cdots}$ $\alpha\downarrow$, converge.

② Passive RL: Q-learning

learn Q(s,a) as you go:
- Receive a sample (s,a,s',r)
- Consider old estimate $Q(s,a)$
- consider new estimate: $sample=R(s,a,s')+\gamma\max_{a'}Q(s',a')$
- $Q(s,a)\leftarrow(1-\alpha)Q(s,a)+\alpha[sample]$

off-policy planning: converges to optimal policy even acting suboptimally. [explore enough, 随 t α slowly↓]

4. Exploration v.s Exploitation

① ε-Greedy: random actions with ε, policy act 1-ε.
问题: explore the space but thrashing around.
solution: ε 随时间↓ / exploration functions

② Exploration Functions: select actions based on 探索 Q value

$f(u,n)=u+\frac{k}{n}$ u: Q-value estimation, n: visit count, returns an optimistic utility

$Q(s,a)\leftarrow\alpha R(s,a,s')+\gamma\max_{a'}f(Q(s',a'),N(s',a'))$

5. Regret: 最优的与实际最优 policy. $R-R^*$.
ε greedy > exploration function.

Approximate Q-learning

$V(s)=w_1f_1(s)+w_2f_2(s)+\cdots w_nf_n(s)$

$Q(s,a)=w_1f_1(s,a)+\cdots+w_nf_n(s,a)$

$Q(s,a)\leftarrow Q(s,a)+\alpha[\text{diffence}]$.

$w_i\leftarrow w_i+\alpha[\text{difference}]f_i(s,a)$

$\text{difference}=[r+\gamma\max_{a'}Q(s',a')]-Q(s,a)$

$\text{transition}=(s,a,r,s')$

Policy Search
- Observation: often the feature-based policies that work well (win games, maximize utilities) aren't the ones that approximate V / Q best
- Idea: learn policies that maximize rewards, not the values that predict them
- Policy search: start with an OK solution (e.g., approximate Q-learning), then fine-tune feature weights to find a better policy
- Simplest policy search:
  - Start with an initial linear value function or Q-function
  - Change each feature weight up and down and see if your policy is better than before
- Problems:
  - How do we tell the policy got better?
  - Need to run many sample episodes!
  - If there are a lot of features, this can be impractical

Dependency Parse

} Advantages
} Deals well with free word order languages where the constituent structure is quite fluid
} Ex: Czech, Turkish
} Parsing is much faster than CFG-bases parsers
} Dependency structure often captures the syntactic relations needed by later applications
} CFG-based approaches often extract this same information from trees anyway

Dependency Parsing

} Parsing: taking a string and a grammar and returning one or more parse tree(s) for that string
} There are two modern approaches to dependency parsing
} Graph-based approach: finding the (maximum) spanning trees of the complete graph over words
} Transition-based (shift-reduce) approach: reading words from left to right and taking a sequence of actions to construct a tree

EM in General
- Can be used to learn any model with hidden variables (missing data)
- Alternate:
  - Compute distributions over hidden variables based on current parameter values
  - Compute new parameter values to maximize expected log likelihood based on distributions over hidden variables
- Stop when no changes
- EM degrades to k-means if we assume
  - All the Gaussians are spherical and have identical weights and covariances
    - i.e., the only parameters are the means
  - The label distributions computed at E-step are point-estimations
    - i.e., hard-assignments of data points to Gaussians
  - Alternatively, assume the variances are close to zero

Transition matrix

X_{t-1}	$P(X_t\mid X_{t-1})$	
	sun	rain
sun	0.9	0.1
rain	0.3	0.7

$\to T=\begin{pmatrix}0.9 & 0.3\\ 0.1 & 0.7\end{pmatrix}$

$P(X_2)=0.6\langle 0.9,0.1\rangle+0.4\langle 0.3,0.7\rangle$
$=T\begin{pmatrix}0.6\\0.4\end{pmatrix}$