Probabilistic Artificial Intelligence

Multivariate Gaussians $\mathcal{N}(y;\Sigma,\mu)$

$$\frac{1}{(2\pi)^{n/2}\sqrt{|\Sigma|}}\exp\left(-\frac{1}{2}(y-\mu)^\top\Sigma^{-1}(y-\mu)\right)$$

$\sigma_{ij} = \mathbb{E}[(X_i-\mu_i)(X_j-\mu_j)]$, $\sigma_i^2 = \mathbb{V}(X_i)$, $X_i \perp X_j \Leftrightarrow \sigma_{ij}=0$, Joint distribution over n variables requires $\mathcal{O}(n^2)$ parameters

Gaussian Conditional Distribution

$p(X_A|X_B=x_b)=\mathcal{N}(\mu_{A|B},\Sigma_{A|B})$

$\mu_{A|B}=\mu_A+\Sigma_{AB}\Sigma_{BB}^{-1}(x_B-\mu_B)$ $\quad\Sigma_{A|B}=\Sigma_{AA}-\Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}$

Mutliples of Gaussians

$X\sim\mathcal{N}(\mu_V,\Sigma_{VV})$, $Y=MX$, $M\in\mathbb{R}^{m\times d}$, then $Y\sim\mathcal{N}(M\mu_V,M\Sigma_{VV}M^T)$

Sums of Gaussians

$X\sim\mathcal{N}(\mu_V,\Sigma_{VV}), X'\sim\mathcal{N}(\mu'_V,\Sigma'_{VV}), Y=X+X'$ then $Y\sim\mathcal{N}(\mu_V+\mu'_V,\Sigma_{VV}+\Sigma'_{VV})$ $\rightarrow$ not the case for product. **Sum of Gaussian** distributed RV is Gaussian distributed. The product of Gaussian distributed RV is NOT Gaussian distributed (but the product of Gaussian PDF is Gaussian).

Useful Math

Probabilities

$\mathbb{E}[X]=\int x\cdot p(x)dx$ or $\sum_x x\cdot p(x)$

Tower Law $\mathbb{E}[X]=\mathbb{E}_y[\mathbb{E}_x[X|Y]]$

$\mathbb{V}[X]=\mathbb{E}[(X-\mu_X)^2]=\mathbb{E}[X^2]-\mathbb{E}[X]^2$

$\mathbb{V}_x[b+cX]=c^2\mathbb{V}_x[X]$

$\mathrm{Cov}[X,Y]=\mathbb{E}[(X-\mathbb{E}(X)(Y-\mathbb{E}(Y))]=\mathbb{E}[XY]-\mathbb{E}[X]\mathbb{E}[Y]$

$\mathbb{V}[X+Y]=\mathbb{V}[X]+\mathbb{V}[Y]+2\mathrm{Cov}[X,Y]$

Change of variables: $f_y(y)=f_x(x)|dg(x)/dx|^{-1}$

Jensen's inequality $f(\mathbb{E}[X])\leq\mathbb{E}[f(X)]$ f convex

$f(\mathbb{E}[X])\geq\mathbb{E}[f(X)]$ f concave

Robbins Monro (RM) conditions If sequence ε_t satisfies $\sum_t\varepsilon_t=\infty$ and $\sum_t\varepsilon_t^2<\infty$ then it will converge to optimum with probability 1. Can be used to check if **learning rate** α will lead to optimum

Entropy and Mutual Information

$H(X)=\mathbb{E}_{x\sim p(x)}[-\log P(X)]=-\sum_{i=1}^n P(x_i)\log P(x_i)$

$H(X,Y)=H(Y,X)$ $\quad H(X\cdot Y)=H(X)+H(Y)$

$H(X,Y)\leq H(X)+H(Y)$ $\quad H(X)=\mathbb{E}_X[I(x)]$

$H(X|Y)=\mathbb{E}_{y\sim p(y)}[H(X|Y=y)]=H(X,Y)-H(Y)$

$I(X;Y)=KL(p(X,Y)\,||\,p(X)p(Y))$

$I(X;Y)=H(X)-H(X|Y)=I(Y;X)\leq I(X;Y,Z)$

If q is Gaussian: $H(q)=\frac{1}{2}\ln|2\pi e\Sigma|$ $\quad q\sim\mathcal{N}(\mu,\Sigma)$

If $Y=X+\varepsilon$ then $H(Y|X)=H(\varepsilon)$

Bayesian Learning

We can write $X=aY+b+\varepsilon$ if X,Y are jointly Gaussian

Ridge Regression

Linear and Ridge Regression fail when multicolinearity, i.e. when more than two explanatory variables are highly linearly related

$\hat{w}=\mathrm{argmin}_w\sum_{i=1}^n(y_i-w^Tx_i)^2+\lambda||w||_2^2$

$\hat{w}=(X^TX+\lambda I)^{-1}X^Ty$

Ridge is the same as finding the **MAP parameter estimate** assuming noise is (cond.) iid Gaussian and prior is Gaussian. Ridge can be derived from BLR by only taking the mode of the posterior

$\hat{w}=\mathrm{argmax}_w P(w)\Pi_i P(y_i|x_i,w)$

$w_{MAP}=\sigma_y^{-2}(\sigma_w^{-2}I+\sigma_y^{-2}X^TX)^{-1}X^Ty$

Bayesian Linear Regression (BLR) $\mathcal{O}(nd^2)$

Pr: $p(w)=\mathcal{N}(0,\sigma_p^2I)$ $\qquad\varepsilon=\mathcal{N}(0,\sigma_n^2I)$

Li: $p(y|x,w,\sigma_n)=\mathcal{N}(y;w^Tx,\sigma_n^2)\equiv y=w^Tx+\varepsilon$

Po: $p(w|X,y)=\mathcal{N}(w;\bar{\mu},\bar{\Sigma})$ has **closed form**

$\bar{\mu}=(X^TX+\sigma_n^2I)^{-1}X^Ty$ $\quad\bar{\Sigma}=(\sigma_n^{-2}X^TX+I)^{-1}$

BLR is the same as averaging all w acc. to posterior

For test point x^*, $f^*=w^Tx^*$

$p(f^*|X,y,x^*)=\mathcal{N}(\bar{\mu}^Tx^*,x^{*T}\bar{\Sigma}x^*)$

$p(y^*|X,y,x^*)=\mathcal{N}(\bar{\mu}^Tx^*,x^{*T}\bar{\Sigma}x^*+\sigma_n^2)$ *Epistemic* (lack of data) and *Aleatoric* (noise) uncertainties captured

Online updating: $X_{new}^TX_{new}=X^TX+x_{i+1}x_{i+1}^T$ $X_{new}y_{new}=Xy+y_{t+1}x_{t+1}$

Choosing **hyperparameters** $\hat{\lambda}=\hat{\sigma}_n^2/\hat{\sigma}_p^2$ via cross-validation. Estimate $\hat{\sigma}_n^2=\frac{1}{n}\sum_{i=1}^n(y_i-\hat{w}^Tx_i)^2$, then solve for $\hat{\sigma}_p^2$. Otherwise, use marginal likelihood.

Kalman Filters (KF)

Assume conditional linear Gaussian dependencies between states (X) and observations (Y). $X_{t+1}\perp y_{1:t}|X_t$

Motion model: $X_{t+1}=FX_t+\varepsilon_t$ $\quad\varepsilon_t\sim\mathcal{N}(0,\Sigma_x)$

Sensor model: $Y_t=HX_t+\eta_t$ $\quad\eta_t\sim\mathcal{N}(0,\Sigma_y)$

Bayesian Filtering in KFs

Update: $\mu_{t+1}=\frac{\sigma_y^2\mu_t+(\sigma_t^2+\sigma_x^2)y_{t+1}}{\sigma_t^2+\sigma_x^2+\sigma_y^2}$ $\quad\sigma_{t+1}^2=\frac{(\sigma_t^2+\sigma_x^2)\sigma_y^2}{\sigma_t^2+\sigma_x^2+\sigma_y^2}$

BLR is a form of KF

Gaussian Processes (GP)

GP is a normal distribution over functions

GP with linear kernel = BLR

Prediction gives closed form formulas. Posterior covariance doesn't depend on the observations y. Exact computation requires $\mathcal{O}(n^3)$ but speedup with parallelism, local GP methods, kernel function approximations, inducing point methods

Covariance (kernel) functions

$K_{x,x'}=\phi(x)^T\phi(x')$ $\qquad k(x,x')=Cov(f(x),f(x'))$

Symmetric: $k(x,x')=k(x',x)\ \forall x,x'$

Positive semi-definite: $x^TKx\geq0\Leftrightarrow\lambda_K\geq0\forall x\in\mathbb{R}^{|A|}$

Composition rules 1. $k(x,x')=k_1(x,x')+k_2(x,x')$ 2. $k(x,x')=k_1(x,x')\cdot k_2(x,x')$ 3. $k(x,x')=c\cdot k_1(x,x')$ for $c>0$ 4. $k(x,x')=f(k_1(x,x'))$ with f a polynomial (with positive coefficient) or exponential

Stationary if: $k(x,x')=k(\tau)$ with $\tau=x-x'$

Isotropic if: $k(x,x')=k(\tau)$ with $\tau=||x-x'||_2$

RBF kernel is both Stationary and Isotropic

If RBF bandwidth h large $\rightarrow$ **smooth** samples

Optimizing kernel hyperparameters

Max. the marginal likelihood $\hat{\theta}=\mathrm{argmax}_\theta P(y|X,\theta)$

Integrate instead of optimizing prevents overfitting

Kernel Function Approximation $\mathcal{O}(nm^2+m^3)$

Random Fourier Features interpret kernel as expectation. Requires a stationary kernel and a randomized feature map. Such a kernel has a Fourier transform. Approximates kernel function uniformly well, wasteful.

Bochner theorem A stationary kernel is positive-definite $\Leftrightarrow$ its Fourier transform is non-negative

Inducing point methods $\mathcal{O}(n)$

Assumes $f^*\perp f|u$, training f, response f^*

$p(f^*,f)\approx q(f^*,f)=\int q(f^*|u)q(f|u)p(u)du$

SoR approximation $q_{SoR}(f|u)$ has the same mean as $p(f|u)$ but variance is set to 0

Approximate Inference

Laplace Approximation uses a Gaussian approximation to the posterior distribution obtained from a second-order Taylor expansion around the posterior mode. Lead to **poor** approximation if p has multiple modes.

Stochastic Gradient Descent converges to (local) minimum if learning rate satisfies RM conditions

Variational Inference (VI)

Approx. unnormalized distribution p by a simple (tractable) distribution q which depends on some parameters λ: $p(\theta|y)=\frac{1}{Z}p(\theta,y)\approx q(\theta|\lambda)$

Only twice as expensive as MAP inference for diagonal q. Only need to be able to differentiate the (unnormalized) joint proba. density p and q. The quality of inference is hard to analyze.

Kullback–Leibler Divergence

$KL(q\,||\,p)=\mathbb{E}_{\theta\sim q}[\log\frac{q(\theta)}{p(\theta)}]=\int q(\theta)\log\frac{q(\theta)}{p(\theta)}d\theta\geq0$

$KL(q\,||\,p)=0\Leftrightarrow p=q$

$KL(q\,||\,p)\neq KL(p\,||\,q)\Rightarrow KL$ is not a distance

$\mathbf{argmin}_q KL(q\,||\,p)=\mathrm{argmax}_q\mathcal{L}(\lambda)$ (c.f. ELBO)

$KL(q\,||\,p)$: *backw.*, **exclusive** (underestimates, if we have two modes, only one of them will be used)

$KL(p\,||\,q)$: *forw.*, **inclusive** (overestimates)

Evidence Lower Bound (ELBO)

$\mathcal{L}(\lambda)=\mathbb{E}_{\theta\sim q(\cdot|\lambda)}[\log p(y|\theta)]-KL(q_\lambda||p(\cdot))$

$\mathcal{L}(\lambda)\leq\log p(y)$, where $p(y)$ is the evidence

Can calculate $\nabla\mathcal{L}$ via the following trick

Reparametrization Trick We want the expectation of the ELBO to depend on an assumed distribution ϕ and not on λ such that: $\nabla_\lambda\mathbb{E}_{\theta\sim q_\lambda}[f(\theta)]=\mathbb{E}_{\varepsilon\sim\phi}[\nabla_\lambda f(g(\varepsilon;\lambda))]$

Makov Chain Monte Carlo (MCMC)

Approx. unnormalized distribution via sampling

Hoeffding's inequality prob. of error decreases exp. in N.

Problem is that normalizing factor Z is intractable. Use **ergodic** MC (i.e. every states can be reached from every states in a finite number of steps) that has stationary distribution $\pi(x)=P(X)=\frac{1}{Z}Q(X)$. Ergodic MC has a **unique** $\lim_{N\to\infty}P(X_N=x)=\pi(x)>0\ \forall x$ and it is **independent of the initial state**.

If MC satisfies the **detailed balance equation** (DBE) then MC has $\pi(x)=P(X)$

DBE: $\frac{1}{Z}Q(x)P(x'|x)=\frac{1}{Z}Q(x')P(x|x')$

Markov Chain $X_{t+1}\perp X_{1:t-1}|X_t$

Metropolis-Hasting MCMC old state: x

1. Proposal $x'\sim R(X'|X=x)$ proposed state: x'

2. $P(X'=x'|X=x)=\alpha$ $\quad P(X'=x|X=x)=1-\alpha$

Acceptance probability $\alpha=min\left(1,\frac{Q(x')R(x|x')}{Q(x)R(x'|x)}\right)$

Use **Gibbs sampling**, from X_i given *all* other, to specify the proposal distribution. Random order Gibbs satisfies DBE, practical variant doesn't but still has correct $\pi(x)$

Variational inference scales better than sampling techniques like Metropolis-Hasting.

Because joint sample at time t depends on sample at $t-1$, the **law of large numbers doesn't apply**. We use the following theorem to compute expectation with MCMC instead:

Ergodic Theorem with D a finite state space

$\lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^N f(x_i)=\sum_{x\in D}\pi(x)f(x)=\mathbb{E}_{x\sim\pi}f(x)$

With continuous RVs, use **MALA** for proposal distribution, which converges to $\pi(x)$ for log-concave distributions, e.g. Bayesian logistic regression $p(x)=\frac{1}{Z}\exp(-f(x))$ is log-concave if the energy function $f(x)$ is convex. For large data sets, use **SGLD**, which convergences to $\pi(x)$ if $\eta_t\in\mathcal{O}(t^{-1/3})$

MALA and SGLD can be improved by adding momentum, resulting in **HMC** (remembring what the gradient was before)

Bayesian Deep Learning

Consider **nonlinear dependencies** of the likelihood function parameters on the inputs.
Heteroscedastic noise depends on input

Bayesian Neural Network (BNN)

Models are the output of the network, MAP parameter estimate: $\hat{\theta} = \operatorname{argmin}_\theta \mathcal{L}(\theta)$, Gaussian pr. on the weight are equiv. to applying weight decay.

Approximate inference for BNNs

VI for BNNs aka Bayes-by-backpropagation. Prediction $p(y^*|x^*,x_{1:n},y_{1:n}) \approx \mathbb{E}_{\theta\sim q(\cdot|\lambda)}[p(y^*|x^*,\theta)] \approx \frac{1}{m}\sum_{j=1}^{m} p(y^*|x^*,\theta^{(j)})$ i.e. draw m sets of weights from posterior and average the neural network predictions. Catches both aleatoric and epistemic uncertrainties. Optimize ELBO via SGD.
MCMC for BNNs Only requires stochastic gradients of (unnormalized) joint probability, i.e. the same used for MAP estimation. Prediction $p(y^*|X,y,x^*) \approx \frac{1}{T}\sum_{j=1}^{T} p(y^*|x^*,\theta^{(\theta)})$. Problem is to store all T samples or models: Can use **Subsampling**, i.e. keep a subset of m snapshots, or keep track of **Gaussian approximation** of the parameters (e.g. SWAG method).
Specialized inference techniques for BNNs Monte-Carlo **Dropout** regularization: randomly ignore hidden units during each iteration of SGD with probability p. Both during training and prediction. Can be viewed as VI. **Probabilistic Ensemble** uses bootstrap sampling, train multiple models on random subsamples of the data

Active Learning

1. Choose and add the point with the **highest uncertainty** to the training set (often the farthest points from the current evaluation points) 2. Train on the new training set 3. Repeat **Exploration**
We want to pick the points to observe f which maximizes the **Information Gain** (aka Mutual Information) $F(S) = H(f) - H(f|y_S)$. Problem $F(S)$ is NP-Hard to optimize. We can use **Uncertainty Sampling** (greedy) to get near-optimal result because $F(S)$ is **monotone submodular**, i.e. satisfies **diminishing returns** property: $H(X|Y) \geq H(X|Y,Z)$. It **fails for herteroscedastic** cases because it can't distinguish aleatoric from epistemic uncertainty. Can use BALD.

Bayesian Optimization

1. Pick next point to evaluate (acc. to the acquisition function by optimizing it) 2. Update the model (Bayes' rule) 3. Repeat **Exploration-Exploitation**
Bandit task: multiple options with unknown probability of producing a reward, the goal is to chose the best option to maximize the overall reward. Idea: pick points that minimize the **Cumulative Regret** Problem is NP-Hard, and we don't know the best possible output. We can use **GP-UCB** acquisition function: $x_t = \operatorname{argmax}_{x\in D}\mu_{t-1}(x) + \beta_t\sigma_{t-1}(x)$
Picks points for which the *upper confidence bound* is currently the highest. The higher the estimate μ, the higher the bound. The higher the uncertainty σ, the higher the bound. **Maximum information gain** γ_T determines the regret. Linear: $\gamma_T = \mathcal{O}(d\log T)$, RBF: $\gamma_T = \mathcal{O}((\log T)^{d+1})$, Matérn ($\nu > 2$): $\gamma_T = \mathcal{O}(T^{\frac{d(d+1)}{2\nu+d(d+1)}}\log T)$ **Thompson Sampling**:draw $\tilde{f} \sim P(f|x_{1:t},y_{1:t}), x_{t+1} = \operatorname{argmax}_{x\in D}\tilde{f}(x)$, the randomness of $\tilde{f}$ is sufficient to trade exploraiton and exploitation.

Markov Decision Processes (MDP)

$V^\pi(x) = \sum_{x'} P(x'|x,\pi(x))[r(x,\pi(x),x') + \gamma V^\pi(x')]$
$\pi_g = \operatorname{argmax}_a \sum_{x'} P(x'|x,a)[r(x,a,x') + \gamma V_\pi(x')]$
Bellman's Theorem
Policy is optimal $\Leftrightarrow$ greedy w.r.t. its induced V_π
Policy iteration converges in $\mathcal{O}(\frac{n^2 m}{1-\gamma})$ iterations
Value iteration converges in $\mathcal{O}(\ln\frac{1}{\varepsilon})$ iterations of complexity $\mathcal{O}(nms)$ per iteration
Value function for optimal policy $V^*(x) = \max_a[r(x,a) + \gamma\sum_{x'} P(x'|x,a)V^*(x')]$
POMDP (aka controlled HMM) Generalization of MDP when partially observable state X_i via noisy observation Y_i. The agent has a certain **belief** over the states. Most belief states are never reached. We can use **Policy gradient** methods.

Reinforcement Learning (RL)

Use observed state transitions and rewards to learn an underlying (unkown) MDP. Data is not iid.
On-policy agent has control over which action to pick. Can chose to trade exploration-exploitation **Off-policy** agent only gets observational data. **Online:** after every step, **Offline:** with replay dataset/batches.
Discount γ: $\ll 1$ focused on instant rewards, ≈ 1 less greedy, more on future rewards.

Model-based RL

Learn the MDP and optimize
$P(X_{t+1}|X_t,A) \approx \frac{\text{Count}(X_{t+1},X_t,A)}{\text{Count}(X_t,A)}$ $\quad r(x,a) \approx \frac{1}{N_{x,a}}\sum_{t:X_t=x,A_t=a} R_t$
ε-greedy pick random action with proba. ε, pick best action with proba. $1-\varepsilon$. Converges to optimal policy if ε satisfies RM conditions. Problem: doesn't quickly eliminate suboptimal actions.
R$_{\text{max}}$ Algorithm (On-policy) *Initialize* 1. add fairy tale state x^* 2. set $r(x,a) = R_{max}$ and $P(x^*|x,a) = 1\ \forall\ x,a$ 3. chose optimal policy. *Repeat* 4. execute policy 5. update $r(x,a)$ for each visited state-action pair 6. estimate $P(x'|x,a)$. Until $n \in \mathcal{O}(\frac{R_{max}^2}{\varepsilon^2}\log\frac{1}{\delta})$ observations

Model-free RL

Value func. $V^\pi(x)$ and Action-Value func. $Q^\pi(x,a)$
Advantage func. $A^\pi(x,a) = Q^\pi(x,a) - V^\pi(x)$
TD-learning use Bootstraping for updating
$V(x) \leftarrow (1-\alpha)V(x) + \alpha(r + \gamma V(x'))$
$V^*(x) = \max_a Q^*(x,a)$, estimate Q^* from samples
Q-learning pick (greedy) $a = \operatorname{argmax}_a Q(x,a)$
$Q(x,a) \leftarrow (1-\alpha)Q(x,a) + \alpha(r + \gamma\cdot\max_{a'} Q(x',a'))$
Converges to optimal Q^* if α satisfies RM conditions.
Q^* depends on γ used during training, larger γ leads to larger Q, larger noise leads to smaller Q value.
TD-learning is On-policy, whereas Q-learning can be Off-policy.
Large state space: use **parametric function approximation** of (action) value function to scale up, e.g. using Neural Network leads to DQN. (less biased than Q-learning). Linear function approximation must be linear in the weights (not in the states).
Large action space: use **Policy gradient methods**
$\nabla J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[(r(\tau) - b)\nabla_\theta \log\pi_\theta(\tau)]$
Note $\nabla_\theta \log\pi_\theta(\tau) = \frac{\nabla_\theta\pi_\theta(\tau)}{\pi_\theta(\tau)}$ For MDP: $\pi_\theta(\tau) = \pi(a|x;\theta)$
Gradient $\nabla J(\theta)$ is unbiased but can have large variance, using a **baseline** $b > 0$ reduces variance.
REINFORCE reduces variance by using **reward to go** G instead of $r(\tau)$.
Actor-critic methods use value function estimate and policy gradient methods
$\nabla J(\theta) = \mathbb{E}_{(x,a)\sim\pi_\theta}[Q(x,a;\theta_Q)\nabla\log\pi(a|x;\theta)]$
derived by using the **discounted state occupancy** measure $\rho(x) = \sum_{t=0}^{\infty}\gamma^t P(x_t = x)$
Allows application in **online** setting
$\theta_\pi \leftarrow \theta_\pi + \eta_t Q(x,a;\theta_Q)\nabla\log\pi(a|x;\theta_\pi)$
$\theta_Q \leftarrow \theta_Q - \eta_t(Q(x,a;\theta_Q) - r - \gamma Q(x',\pi(x',\theta_\pi);\theta_Q))\nabla\log\pi(a|x;\theta_\pi)$
Updates improvement guaranteed under **compatibility conditions**
Can reduce variance via baselines (using advantage function) $A(x,a) = Q(x,a;\theta_Q) - V(x;\theta_V)$
$\theta_\pi \leftarrow \theta_\pi + \eta_t A(x,a)\nabla\log\pi(a|x;\theta_\pi)$
Actor-critic can be parallelized
TRPO and **PPO** are variant of policy gradient/actor-critic methods. PPO is a heuristic variant of TRPO widely used in practice.
Advantage function properties
$\max_a A^\pi(x,a) \geq 0\ \forall\pi,x$
π^* is optimal $\Leftrightarrow A^*(x,a) \leq \sigma\ \forall x,a$
$\pi_g(x) = \operatorname{argmax}_a Q^\pi(x,a) = \operatorname{argmax}_a A^\pi(x,a)$
Off-policy policy gradient methods If we use differentiable approximation of the maximum Q and differentiable **deterministic** policy, then we can use chain rule (i.e. backprop.) to obtain stochastic gradient. Problem, policy gradient methods rely on randomized policies for exploration, we have to **inject additional action noise** to encourage exploration in the off-policy variant, e.g. **DDPG**. Can encourage exploration via **entropy regularized** MDPs, e.g. **SAC**

Homeworks

If $y = Xw + \varepsilon \Rightarrow w_{MLE} = (X^T X)^{-1}X^T y$
Kalman filters is like HMM but for continuous variables. High gain means follow the most recent measurement. Low gain means follow the model predictions.
The support of q must be a subset of the support of p, otherwise $KL(q\,\|\,p) \to \infty$
Expected improvement
$EI(X) = \mathbb{E}[(t-y)_+] = \int_{-\infty}^{\infty}(t-y)_+ P(Y|X)dy$ where $(t-y)_+ = max(0, t-y)$
REINFORCE update:
$\theta \leftarrow \theta + \alpha G \cdot E$ where $E = \frac{\nabla_\theta\pi(a|x,\theta)}{\pi(a|x,\theta)}$ is the eligibility vector
Linear MDP with feature map $\phi(x,a)$ means (1) $\exists\theta$ s.t. $r(x,a) = \langle\phi(x,a),\theta\rangle$ (2) $P(x'|x,a) = \langle\phi(x,a), P(x')dx'\rangle$ Note: $P(x')$ doesn't have to be a distribution, but $P(x'|x,a)$ should.
Sum rule $P(X) + P(\bar{X}) = 1$ Product rule $P(X,Y,Z) = P(X|Y,Z)\cdot P(Y,Z) = P(Y|X,Z)\cdot P(X,Z) = P(Z|X,Y)\cdot P(X,Y)$
Policy gradient: $\mathcal{A} = \{0,1\}$, $\pi(1|s_t) = \theta$ and $\pi(0|s_t) = 1-\theta$, trajectory $\tau = (s_1,a_1,s_2,a_2,s_3,a_3)$, future reward $R_t^{future} = \sum_{t=t'}^{H} r_t'$ i.e. $R_1^{future} = r_1 + r_2 + r_3$ but $R_3^{future} = r_3$, we can calculate $\nabla_\theta\pi(1|s_t) = 1$ and $\nabla_\theta\pi(0|s_t) = -1$ then $\nabla_\theta\log\pi(1|s_t) = \frac{\nabla_\theta\pi(1|s_t)}{\pi(1|s_t)}$ and put together $Gradient = \sum_t R_t^{future}\nabla_\theta\log\pi_\theta(a_t|s_t)$