This document is a summary of the *Probabilistic Artificial Intelligence* course at ETH Zürich. This summary was created during the autumn semester of 2023. Due to updates to the syllabus content, some material may no longer be relevant for future versions of the course. I do not guarantee correctness or completeness, nor is this document endorsed by the lecturers. This draft cheatsheet exceeds the allowed number of pages for the exam, so make sure to cut it down to size. The order of the chapters is not necessarily the order in which they were presented in the course. For the full LaTeX source code, visit `github.com/Jovvik/eth-cheatsheets`.

1 Introduction

Prior: $P(X)$, **Likelihood**: $P(Y|X)$

Bayes: posterior $P(X|Y) = \frac{P(X,Y)}{P(Y)} = \frac{P(Y|X)P(X)}{P(Y)}$

Marginalization (sum rule): $P(X_{1:i-1}, X_{i+1:n}) = \sum_{x_i} P(X_{1:i-1}, x_i, X_{i+1:n})$. **Chain (product) rule**: $P(X_{1:n}) = P(X_1)P(X_2 \mid X_1)\dots P(X_n \mid X_{n-1})$

Conditional ind. $X \perp Y \mid Z$: $\forall x, y, z$ $P(X = x, Y = y \mid Z = z) = P(X = x \mid Z = z)P(Y = y \mid Z = z)$. If $P(Y = y \mid Z = z) > 0$, then equiv. to $P(X = x \mid Y = y, Z = z) = P(X = x \mid Z = z)$.

Gaussians are independent iff they are uncorr.

Gauss: $\mathcal{N} = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma^2}(x-\mu)^2)$

$\mathcal{N} = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp(-\frac{1}{2}(x-\mu)^\top\Sigma^{-1}(x-\mu))$

CDF: $\Phi(v; \mu, \sigma^2) = \int_{-\infty}^{v} \mathcal{N}(y; \mu, \sigma^2)dy = \Phi(\frac{v-\mu}{\sqrt{\sigma^2}}; 0, 1)$

For GRV X, $MX \sim \mathcal{N}(M\mu_X, M\Sigma_{XX}M^\top)$.

$X + X' \sim \mathcal{N}(\mu_X + \mu_{X'}, \Sigma_{XX} + \Sigma'_{XX})$

For disjoint index sets A, B the conditional distr. $p(X_A \mid X_B = x_B)$ is $\mathcal{N}(\mu_{A|B}, \Sigma_{A|B})$, where $\mu_{A|B} = \mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(x_B - \mu_B)$, $\Sigma_{A|B} = \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}$.

If X, Y are jointly Gaussian, $P(X = x \mid Y = y) = \mathcal{N}(x; \mu_{X|Y=y}, \sigma^2_{X|Y})$, $\mu_{X|Y=y} = \mu_X + \sigma^2_{XY}\sigma_Y^{-2}(y - \mu_Y)$, $\sigma^2_{X|Y} = \sigma^2_X - \sigma^2_{XY}\sigma_Y^{-2} \Rightarrow X = aY + b + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma^2_{X|Y})$, $a = \sigma^2_{XY}\sigma^2_Y$, $b = \mu_X - \sigma^2_{XY}\sigma_Y^{-2}\mu_Y$.

Change of variables: $Y \coloneqq g(X)$, g diff. & invert., $f_Y(y) = f_X(g^{-1}(y))|\frac{d}{dy}(g^{-1}(y))|$. For $\mathbb{R}^d$: $f_Y(y) = f_X(g^{-1}(y))|\det D(g^{-1}(y))|$, D is the Jacobian.

Law of the unconscious statistician: $\mathbb{E}[g(X)] = \int_{-\infty}^{+\infty} g(x)f_X(x)dx$ for diff. and invertible g.

Law of total expectation: $\mathbb{E}[\mathbb{E}[X \mid Y]] = \mathbb{E}[X]$

Tot. var.: $\mathbb{V}[X] = \mathbb{E}[\mathbb{V}[X \mid Y]] + \mathbb{V}[\mathbb{E}[X \mid Y]]$

A vector of i.i.d. $\mathcal{N}(0, 1)$ is a standard GRV. Arbitrary GRV is $\Sigma^{1/2}X + \mu$.

Σ is symmetric; positive semidefinite, i.e.: $\exists L : \Sigma = LL^\top$; all eigenval. ≥ 0; $\forall x\ x^\top\Sigma x \geq 0$ & eq. iff $x = 0$. MAP $\xrightarrow[n\to\infty]{}$ MLE.

1.1 Homework ideas

To show that $f(X_1, X_2 \dots)$ is (not) Gaussian, find the PDF of $f(X_1 \mid X_2 = x_2)$ (probably Gaussian) and compute PDF of f using the Bayes rule.

2 Bayesian Linear Regression

Bayesian learning: given prior $p(\theta)$ on weights and likelihood $p(D \mid \theta)$, by Bayes rule posterior $p(\theta \mid D) = (1/Z)p(\theta)p(D \mid \theta)$.

Assuming Gaussian prior $p(\theta)$ and i.i.d. Gaussian noise $p(y \mid x, \theta)$, the **maximum a posteriori** (MAP) estimate is equivalent to ridge regression. **BLR** considers not only the mode (MAP), but the whole posterior. For BLR with Gaussian prior $p(\theta) = \mathcal{N}(0, \sigma_p^2 I)$ and likelihood $p(y \mid x, \theta, \sigma_n) = \mathcal{N}(y; \theta^\top x, \sigma_n^2)$, the posterior is $\mathcal{N}$ with $\Sigma = (\sigma_n^{-2}X^\top X + \sigma_p^{-2}I)^{-1}$ and $\mu = \sigma_n^{-2}\Sigma X^\top y$.

Inference: for input x^*, let $f^* \coloneqq \theta^\top x^*$. Then $p(f^* \mid X, y, x^*) = \mathcal{N}(\mu^\top x^*, x^{*\top}\Sigma x^*)$ and $p(y^* \mid X, y, x^*) = \mathcal{N}(\mu^\top x^*, x^{*\top}\Sigma x^* + \sigma_n^2)$. We are averaging over all θ weighted by their posterior probability.

- **Aleatoric uncertainty**: irreducible: σ_n^2
- **Epistemic uncertainty**: $x^{*\top}\Sigma x^*$ — uncertainty about the model, removed with more data

Graphical model is a DAG of dependencies.

Recursive updates, use posterior as next prior: $p^{(j)}(\theta) = p(\theta \mid y_{1:j}) = \frac{1}{Z}p(\theta \mid y_{1:j-1})p_j(y_j \mid w) = \frac{1}{Z'}p(w)p_1(y_1 \mid w)\dots p_{j-1}(y_{j-1} \mid w)p_j(y_j \mid w)$

2.1 Homework ideas

Hierarchical Bayesian Modeling: make priors on hyperparams, **Empirical Bayes**: estimate hyperparams from domain knowledge / data.

Computing the μ and σ of a distribution is plain algebra. MLE minimizes the log-likelihood due to exponents. MAP uses the prior less, if it has more data. When computing f_X, ignore all multiplicative terms without x.

3 Gaussian Processes

For nonlinear functions, can use nonlinear feature space, but it is expensive. **Kernel trick**: express problem in terms of inner products and replace them by kernel $k(x_i, x_j)$.

Weight view: model distribution over weights $p(w)$ and use $f(x) = w^\top\phi(x)$. **Function view**: model distribution over predictions directly.

Gaussian process model a distr. over functions $f(x)$ using an ∞dim X. Params: $\mu : X \to \mathbb{R}$ (often zero) and $k : X \times X \to \mathbb{R}$, $\forall x \in X\ \exists$ RV f_x s.t. $\forall A \subseteq X$ $f_A \sim \mathcal{N}(\mu_A, K_{AA})$. k is symm. and pos. semi-def. $\forall A$. Posterior k doesn't depend on y.

- Linear with features $k(x, x') = \Phi(x)^\top\Phi(x')$
- RBF (Gaussian) $k(x, x') = \exp(-||x - x'||_2^2/h^2)$: locally correlated, far independent, a.s. ∞-diff. Higher h (**lengthscale**) $\Rightarrow$ smoother functions.
- Exponential: RBF with l_1 norm. Nowhere diff.
- Matern with param. ν: $\lceil\nu\rceil - 1$ times diff. With $\nu = 0.5$ Matern is expon., with $\nu \to \infty$ RBF.

k are closed under $+$, $\cdot$ and scaling. If $k = c \cdot k'$, c is the **output scale**. Higher $c \Rightarrow$ higher amplitude.

Stationary: $k(x - x')$, **isotropic**: $k(||x - x'||_2)$.

Learning: $p(f \mid x_{1:m}, y_{1:m}) = GP(f; \mu', k')$, $\mu'(x) = \mu(x) + k_{x,A}^{-1}(K_{AA} + \sigma^2 I)(y_A - \mu_A)$, $k'(x, x') = k(x, x') - k_{x,A}^{-1}(K_{AA} + \sigma^2 I)k_{x',A}^\top$

Sampling: decompose K as $LL^\top$, $\varepsilon \sim \mathcal{N}(0, I)$; $f \coloneqq \mu + L\varepsilon$. Forward sampling: $f_i \sim p(f_i \mid f_{1:i-1})$.

Minim. MSE to find hyperparams encourages making k low. Maxim. marginal likelihood balances epistemic and aleatoric uncertainty.

$$-\log p(y \mid X, \theta) = \underbrace{0.5y^\top K_y^{-1}y}_{\text{goodness of fit}} + \underbrace{0.5\log|K_y|}_{\text{regularization}}.$$

This can be optim. by GD: $\nabla = \frac{1}{2}\mathrm{tr}((\alpha\alpha^\top - K^{-1})\nabla K)$, where $\alpha \coloneqq K^{-1}y$. This is $O(m^3)$ and may have nonglobal optima, due to diff. interpret. of data.

3.1 Fast Gaussian Processes

Local methods: to predict for x, only use points where $|k(x, x')|$ is not small.

k approximation: use low-dim ϕ s.t. $k(x, x') \approx \phi(x)^\top\phi(x')$, apply BLR. $O(nm^2 + m^3)$ instead of n^3.

Random Fourier features: for a stationary kernel $k(x - x') = \int_{\mathbb{R}^d} p(\omega)\exp(j\omega^\top(x - x'))d\omega$, then $p(\omega) = \frac{1}{2\pi}\int_{\mathbb{R}^d} k(\Delta)\exp(-j\omega^\top\Delta)d\Delta$, sample $\omega_{1:d}$ and $b_{1:d} \sim U(0, 2\pi)$. $z(x) \coloneqq \sqrt{2/D}(\cos(\omega_1^\top x + b_1)\dots\cos(\omega_D^\top x + b_D))$.

Inducing points: use less points to approx. posterior. To not compute K, set to 0 (SoR) or diag (FITC). $k_{\mathrm{SoR}}(x, x') = k_{x,U}^\top K_{UU}^{-1}k_{x',U}$.

Prior: $p(\theta)$, likelihood: $p(y_{1:n} \mid x_{1:n}, \theta) = \prod_i p(y_i \mid x_i, \theta)$, posterior: $p(\theta \mid y_{1:n}, x_{1:n}) \propto p(\theta)p(y_{1:n} \mid x_{1:n}, \theta)$, predictions: $p(y^* \mid x^*, y_{1:n}, x_{1:n}) = \int p(y^* \mid x^*, \theta)p(\theta \mid y_{1:n}, x_{1:n})d\theta$.

4 Variational Inference

As $p(\theta \mid y) = \frac{1}{Z}p(\theta, y)$, assume we can evaluate the joint distribution, but not Z. Let $p(\theta \mid y) \approx q(\theta \mid \lambda)$.

Laplace approximation: 2nd order Taylor around mode: $q(\theta) = \mathcal{N}(\hat\theta, \Lambda^{-1})$, $\hat\theta = \arg\max_\theta p(\theta \mid y)$, $\Lambda = -\nabla\nabla\log p(\hat\theta \mid y)$. ∇ is 0, as we are at the mode. The $\nabla\nabla$ may be expensive to compute and invert $\Rightarrow$ use diagonal Gaussians or sparse H approx.

For BLR, $\hat\omega = \arg\min_\omega \frac{1}{2\sigma_p^2}||\omega||_2^2 + \sum_i \log(1 + \exp(-y_i\omega^\top x_i))$, which is just regularized logistic regression and $\Lambda = X^\top\mathrm{diag}([\pi_i(1 - \pi_i)]_i)X$, where $\pi_i = \sigma(\hat\omega^\top x_i)$, which does not depend on the normalizer. Prediction: $p(y^* \mid x^*, x_{1:n}, y_{1:n}) \approx \int \sigma(y^* f)\mathcal{N}(f; \hat w^\top x^*, x^{*\top}\Lambda^{-1}x^*)df$, which is 1d $\Rightarrow$ easily approximated. Laplace is bad for multimodal or highly nonsymmetric distributions.

Variational inference: find a tractable distribution from a family Q that is close to the true posterior. $q^* \in \arg\min_{q\in Q} KL(q||p)$, where $KL(q||p) = \int q(\theta)\log\frac{q(\theta)}{p(\theta)}d\theta = \mathbb{E}_{\theta\sim q}\log\frac{q(\theta)}{p(\theta)}$. KL is non-negative and 0 iff $q = p$, not symmetric.

Forward $KL(p||q)$ vs reverse $KL(q||p)$: reverse is overconfident, but cheaper to compute.

$KL(\mathcal{N}(\mu_p, \Sigma_p)||\mathcal{N}(\mu_q, \Sigma_q)) = \frac{1}{2}\mathrm{tr}(\Sigma_q^{-1}\Sigma_p) + (\mu_p - \mu_q)^\top\Sigma_q^{-1}(\mu_p - \mu_q) - d + \log\frac{\det\Sigma_q}{\det\Sigma_p}$

$KL(\mathrm{Bern}(p)||\mathrm{Bern}(q)) = p\log\frac{p}{q} + (1 - p)\log\frac{1-p}{1-q}$

Entropy of a distr.: $H(q) = -\int q(\theta)\log q(\theta)d\theta = \mathbb{E}_{\theta\sim q}[-\log q(\theta)]$, where $-\log q(\theta)$ is **surprisal**. $H(\mathcal{N}(\mu, \Sigma)) = 0.5\ln|2\pi e\Sigma|$. $KL(q||p) = H(q||p) - H(q) \Rightarrow$ minimizing KL is the same as minimiz. CE $H(q||p) = \mathbb{E}_{x\sim q}[-\log p(x)]$, while maximiz. the uncertainty of q. It is also equivalent to maximizing $\mathbb{E}_{\theta\sim q}[\log p(y \mid \theta)] - KL(q||p_{\text{prior}})$ or maximizing **ELBO**: $\mathbb{E}_{\theta\sim q}[\log p(y_{1:n}, \theta \mid x_{1:n})] + H(q)$

If Q is the family of Gaussians, $q = \arg\min_Q KL(p||q)$ matches the first and second moments of p. As opposed to Laplace, which matches the mode and 2nd derivative. When minizing the reverse KL, q matches the average and 2nd derivative of p.

Reparametrization trick: if $\theta = g(\varepsilon,\lambda)$ and $\varepsilon \sim p(\varepsilon)$, then $q(\theta \mid \lambda) = \phi(\varepsilon)|\nabla_\varepsilon g(\varepsilon,\lambda)|^{-1}$ and $\mathbb{E}_{\theta\sim q}[f(\theta)] = \mathbb{E}_{\varepsilon\sim p(\varepsilon)}[f(g(\varepsilon,\lambda))]$, so we can use MC sampling because the ε does not depend on λ. This allows to compute ∇ELBO.

4.1 Homework ideas

If X has invertible CDF F, then $X = F^{-1}(\mathrm{Unif}(0,1))$. For fixed mean and variance, $\mathcal{N}$ is the maximum entropy distribution (with support $\mathbb{R}$). $H(f||g) = H(g)$ for gauss. f. Probit likelihood: $\Phi(0,\sigma_n^2)$.

5 Markov Chain Monte Carlo

Approximate p by sampling from it.

Hoeffding: $P(|\frac{1}{N}\sum_i f(x_i) - \mathbb{E}_P[f(X)| > \epsilon) \le 2\exp(-2n\epsilon^2/C^2)$, i.e. error probability decreases exponentially with N. if f is bounded in $[0,C]$.

Markov chain is a sequence of RVs $X_{1:n}$ with a prior $P(X_1)$ and transition probabilities $P(X_{i+1} \mid X_i)$, independent of i.

Ergodic MC: $\exists t$: any state can be reached from any state in exactly t steps. Such an MC has a unique **stationary distribution** $\pi(X) > 0$ s.t. $\forall x\ \lim_{N\to\infty} P(X_N = x) = \pi(x)$, independent of $P(X_1)$. Equivalently, $\pi(x') = \sum_x p(x' \mid x)\pi(x)$. When sampling long enough, the distribution of the samples converges to $\pi(x)$.

MC satisfies the **detailed balance** equation for an unnormalized distr. $Q(x)$ if $\forall x, x'$: $Q(x)P(x' \mid x) = Q(x')P(x \mid x')$. Equivalently, MC is the same when running it backwards. Then $Q(x)/Z$ is $\pi(x)$.

Metropolis-Hastings algorithm: let $X_t = x$. Sample proposal $x' \sim R(X' \mid X = x)$. With prob. $\alpha \coloneqq \min(1, \frac{Q(x')R(x|x')}{Q(x)R(x'|x)})$, set $X_{t+1} = x'$, else $X_{t+1} = x$. By detailed balance $\pi(x) = Q(x)/Z$.

Gibbs sampling: MH with $\alpha = 1$ and proposal $r(x' \mid x) = p(x_i' \mid x_{-i}')$ if x' differs from x only in entry i, 0 else. Equivalently, we pick an index and sample from the cond. distr. of that index given the rest. $p(x_i' \mid x_{-i}')$ requires a normalizer, but over discrete states, which is fast.

Ergodic theorem: if MC is over a finite state space D with SD π and $f : D \to \mathbb{R}$, almost surely $\lim\limits_{N\to\infty} N^{-1}\sum_{i=1}^N f(x_i) = \sum_{x\in D}\pi(x)f(x)$.

MCMC: sample T samples, drop first t_0 as burn-in, then $\mathbb{E}[f(X)] \approx$ average of $f(X_t)$.

We focus on positive distr., which can be written as $p(x) = Z^{-1}\exp(-f(x))$, where $f(x)$ is an **energy function**. If f is convex, p is called log-concave. Energy is just unnormalized surprisal.

If the proposal is $\mathcal{N}(x,\tau I)$, then the ratio is 1 and $\alpha = \min(1, \exp(f(x) - f(x')))$ and the direction of movement is uninformed. **Metropolis adjusted Langevin algorithm**: $\mathcal{N}(x - \tau\nabla f(x), 2\tau I)$. This converges to the true distr. fast(er) for log-concave distrs. This can be expensive for big data $\Rightarrow$ approximate ∇ via batches. Also, never reject and use decreasing step sizes $\Rightarrow$ **Stochastic gradient Langevin dynamics**. This is just SGD with $\mathcal{N}$ noise, converges with assumptions if $\eta_t \in \Theta(t^{-1/3})$.

VI: optimization is efficient, but bias from choice of Q $\Rightarrow$ no convergence to true posterior.

MCMC: asymptotic convergence with rates with assumptions (p log-concave), but samples are dependent and can't tell when to stop

6 Bayesian deep learning

BNN: prior on weights, likelihood parametrized by NN, e.g. $p(y \mid x,\theta) = \mathcal{N}(f(x,\theta),\sigma^2)$.

Heteroschedastic noise: depends on input. Can be modelled as $p(y \mid x,\theta) = \mathcal{N}(f_\mu(x,\theta), \exp(f_{\sigma^2}(x,\theta)))$. Hard to support in GP. Then MAP estimate is $\hat\theta = \arg\min_\theta \lambda||\theta||_2^2 - \frac{1}{2}\sum_i \frac{1}{\sigma(x_i,\theta)^2}||y_i - \mu(x_i,\theta)||^2 + \log\sigma(x_i,\theta)^2$. NN can lower the loss for a data point by being uncertain (high σ). Gradients are computed with autodiff, gaussian prior is weight decay.

Approx. inference can be done with Laplace, black-box stochastic VI or SGLD.

VI: same as regular VI, but the likelihood has an NN, optimize θ via reparam. trick. Inference: sample m sets of weights and average the predictions, variance is by law of total variance: $\mathbb{E}[\mathbb{V}[y^* \mid x^*,\theta]] + \mathbb{V}[\mathbb{E}[y^* \mid x^*,\theta]] = \frac{1}{m}\sum_j\sigma^2(x^*,\theta^{(j)}) + \frac{1}{m}\sum_j(\mu(x^*,\theta^{(j)}) - \overline{\mu}(x^*))^2 =$ aleatoric + epistemic.

SGLD: don't want to store T weights. Subsample or approx. with $\mathcal{N}$ with same μ and σ with running averages. SWAG: same, but no noise.

Dropout is VI with family where each weight is 0 with prob. p, else λ_j. Dropout at inference.

Probabilistic Ensembles: train m NNs with different behaviour, e.g. on diff. data, diff. init.

Aleatoric uncertainty can be modelled by adding learneable noise before softmax.

Calibration is evaluated by comparing the true probability vs the predicted probability. Split the predicted probs. into buckets and compare the mean predicted prob. (confidence) to the true prob. (accuracy) in each bucket. Improve calibration: assign accuracy to each bin (const. or learn f).

7 Active learning

Find $S \subseteq D$ maximizing inf. gain: $F(S) \coloneqq H(f) - H(f \mid y_S) = I(f;y_S) = 0.5\log|I + \sigma^{-2}K_S|$, where I is mutual information. Greedy: given S_t, find optimal x_{t+1} to add: $x_{t+1} = \arg\max_x \sigma_f^2(x)/\sigma_n^2(x)$.

F is **monotone submodular** (diminishing returns): $\forall x \in D, A \subseteq B \subseteq D : F(A \cup \{x\}) - F(A) \ge F(B \cup \{x\}) - F(B)$. Constant-factor approx.: $F(S_{\text{greedy}}) \ge (1 - e^{-1})\max_S F(S) \approx 0.63\max F$.

If noise is heteroschedastic, more uncertain points are not necessarily more informative.

Different ways to scalarize the posterior cov.: D-optimal: entropy or log-determinant — uncertainty sampling, A-opt.: trace, E-opt.: max eigenvalue.

BALD: Informative Sampling for Classification.

8 Bayesian optimization

Select measurements to find the maximum of a function. Trade-off between exploration and exploitation. **Regret**: $R(T) = \max_x f(x) - f(x_t)$. Sublinear regret is good.

GP-UCB: $x_{t+1} = \arg\max_x \mu_t(x) + \beta_t\sigma_t(x)$, where β_t is a trade-off parameter. If β is "good", $\frac{1}{T}\sum_t[f(x^*) - f(x_t)] = O^*(\sqrt{\gamma_T T^{-1}})$, where $\gamma_T = \max_{|S|\le T} I(f;y_S)$. For linear kernel, $\gamma_T = O(d\log T)$, RBF: $\gamma_T = O((\log T)^{d+1})$, Matern with $\nu > \frac{1}{2}$: $O(T^{\frac{d}{2\nu+d}}(\log T)^{\frac{2\nu}{2\nu+d}})$. Regret is sublinear.

If $f \in$ hilbert space for k, the average regret is $O^*(\sqrt{\beta_T\gamma_T T^{-1}})$, if $\beta_T = O(||f||_k^2 + \gamma_T\log^3 T)$.

How to find the argmax for x_{t+1}? If low dim, use Lipschitz optimizataion, else use GD.

Thompson sampling: sample a function, find its argmax, sample a measurement there.

8.1 Homework ideas

VI's stationary conditions are $\mathbb{E}$ of the Laplace approximation, which means it's more global.

9 Markov Decision Processes

MDP is: states X, actions A, initial state distr. $P(x_0)$, transition probabilities $P(x' \mid x,a)$ (if do action a in x, prob. of ending up in x'), reward function $r(x,a)$ — can be stochastic.

Assuming r, P are known.

It is enough for a policy to only use the current state if there is no horizon. Deterministic is enough, if you have full knowledge of the environment.

A policy induces a markov chain. Expected value of a policy is $J(\pi) = \mathbb{E}[\sum\gamma^i r(X_i,\pi(X_i))]$, where γ is the **discount factor**. A **value function** is $V^\pi(x) = J(\pi \mid X_0 = x)$. $V^\pi(x) = r(x,\pi(x)) + \gamma\sum_{x'}p(x' \mid x,\pi(x))V^\pi(x')$. $V^\pi = r^\pi + \gamma T^\pi V^\pi \Rightarrow r^\pi = (I - \gamma T^\pi)V^\pi$. If $\gamma < 1$, $V^\pi = (I - \gamma T^\pi)^{-1}r^\pi$. Inversion is slow, use fixed point iteration: random initially, T times do $V_t^\pi = r^\pi + \gamma T^\pi V_{t-1}^\pi =: B^\pi V_{t-1}^\pi$. l_∞ error $\le \gamma^t||V_0^\pi - V^\pi||_\infty$.

Every policy induces a V^π, a V^π induces a greedy policy. **Bellman** theorem: a policy π is optimal iff it is greedy w.r.t. V^π.

Policy iteration: start with random policy, N times: compute V^π, select greedy π' w.r.t. V^π. Converges to optimal in $O^*(n^2m/(1-\gamma))$.

Value iteration: $V_t(x) \coloneqq$ max. expected reward when starting in x and taking t steps. Use DP: $V_0(x) = \max_a r(x,a)$, $V_{t+1} = \max_a Q_{t+1}(x,a) \coloneqq \max_a r(x,a) + \gamma\sum_{x'}p(x' \mid x,a)V_t(x')$. Break if $||V_t - V_{t-1}||_\infty \le \varepsilon$. Then choose greedy policy w.r.t. V_t. Converges to an ε-optimal policy, because B^* is a contraction.

POMDP: don't know X_t, only noisy observation $O(x',y) = P(Y_{t+1} = y \mid X_{t+1} = x')$. $\lessdot$ MDP with belief states $P(X_t \mid y_{1:t})$ and reward function $\sum_x b(x)r(x,a)$. $b_{t+1}(x) = Z^{-1}O(y_{t+1},x)\sum_{x'}b_t(x')P(x \mid x',a)$.

Stochastic Bellman expectation equations: $V^\pi(x) = \sum_a \pi(a \mid x)(R(x,a) + \gamma\sum_{x'}P(x' \mid x,a)V^\pi(x'))$, $Q^\pi(x,a) = R(x,a) + \gamma\sum_{x'}P(x' \mid x,a)\sum_{a'}\pi(a' \mid x')Q^\pi(x',a')$.

Policy trees: choose sequence of actions via observations made. Each observ. is a split. Each tree $\leftrightarrow$ value vector α for each state. $V(b) = \alpha b$.

Approx. solutions: discretize the belief space, use point based VI or PI, reduce dim.

9.1 Homework ideas

If PI does not change the policy, it is optimal. Multiplying all rewards by $c \in \mathbb{R}^+$ does not change π^*. Adding c to all rewards changes π^* (negative rewards). Adding $\gamma\phi(x') - \phi(x)$ does not change π^*. $\forall x \in X\ V_{t+1}^\pi(x) \ge V_t^\pi(x) \quad V^\pi(x) = Q^\pi(x,\pi(x))$

10 Reinforcement Learning

In RL data is trajectories of states, actions and rewards. **Episodic** setting: agent learns over episodes, after which the environment is reset. **Online** setting: one trajectory, no reset.

On-policy methods require the agent to choose actions to learn, **off-policy** can be observational.

On-policy: REINFORCE, Rmax, TD, SARSA. Off-policy: Q-learning, DQN.

Model-based RL: find approximate MDP and solve it. **Model-free** RL: direct V estimation, policy gradient, actor-critic.

Transition estimates: $\hat{P}(X_{t+1} \mid X_t, A) = \frac{N(X_{t+1}, X_t, A)}{N(X_t, A)}$. Reward estimates: $\hat{r}(x, a) = \frac{1}{N(x,a)} \sum_{t: X_t = x, A_t = a} R_t$.

ε-greedy: prob. ε choose random a, else greedy.

$\mathbf{R_{max}}$ **algorithm**: initialize $\hat{r}(x,a)$ to the max value $R_{\max}$, if $P(x' \mid x, a)$ is not known, set $P(x^* \mid x, a) = 1$, where x^* is a fairy tale state, which is terminal with reward $R_{\max}$. On each transition, update r and P. When observed enough transitions, recompute π. Enough is per Hoeffding bound (see MCMC). Every T timesteps, either observe at least one new (x,a)-pair or obtain near-optimal reward. Converges with prob. $1-\delta$ to an ε-optimal policy after $\mathrm{poly}(|X|, |A|, \varepsilon^{-1}, \log(\delta^{-1}))$ steps.

Memory: for (x, x', a) need to store $\hat{p}$ and $\hat{r}$, $O(n^2 m)$ for dense MDPs. CPU: many MDP solves.

Temporal difference learning: use policy π to get a transition, update with bootstrapping: $\hat{V}^\pi(x) \leftarrow (1-\alpha_t)\hat{V}^\pi(x) + \alpha_t(r + \gamma \hat{V}^\pi(x'))$. If LR α_t is s.t. $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$ and all states are chosen infinitely often, $\hat{V}^\pi$ converges to V^π with prob. 1. This is on-policy. Same can be done for $\hat{Q}$, but off-policy.

Q-learning: estimate Q^* from samples. Observe transition x, a, x', r; $\hat{Q}^*(x,a) \leftarrow (1-\alpha_t)\hat{Q}^*(x,a) + \alpha_t(r + \gamma \max_{a'} \hat{Q}^*(x', a'))$. Same convergence (need to visit all state-action pairs infinitely often).

Optimistic Q-learning: initialize $\hat{Q}^*(x,a)$ to a high value, e.g. $\frac{R_{max}}{1-\gamma} \prod_{t=1}^{T_{init}} (1-\alpha_t)^{-1}$. At time t, pick $a_t \in \arg\max_a \hat{Q}^*(x_t, a)$ and observe the transition. Memory: $O(nm)$ (store $\hat{Q}^*$), CPU: $O(nm)$ per step.

TD-learning is SGD, if $\hat{V}^\pi$ is parametrized by θ s.t. $\hat{V}^\pi(x, \theta) = \theta(x)$. Loss is $0.5(r + \gamma\theta_{\text{old}}(x') - \theta(x))^2$, after computing ∇ the update is: $\theta(x) \leftarrow (1-\alpha_t)V^\pi(x;\theta) + \alpha_t(r + \gamma V^\pi(x'; \theta_{\text{old}}))$.

Parametric value function approximation: learn an approximation of V or Q.

Q**-learning with FA**: until converged, pick action a, observe (x', r), update $\theta \leftarrow \theta - \alpha_t \delta \nabla_\theta Q(x, a; \theta)$, where $\delta \coloneqq Q(x,a;\theta) - r - \gamma \max_{a'} Q(x', a'; \theta)$ (**Bellman error**).

DQN (neural fitted Q-iteration): maintain replay buffer, only train every N steps. More stable. Estimates are too optimistic, as the max of noisy Q leads to maximization bias. $l = 0.5 \sum_{(x,a,r,x') \in D} (r + \gamma \max_{a' \in A} Q^*(x', a'; \theta_{\text{old}}) - Q^*(x, a; \theta))^2$

Double DQN: use current network to select action to avoid maximization bias. $l = 0.5 \sum_{(x,a,r,x') \in D} (r + \gamma Q^*(x', a^*(x', \theta); \theta_{\text{old}}) - Q^*(x, a; \theta))^2$

Q-learning is expensive for large action spaces.

Policy search methods: learn parametrized policy $\pi(x) = \pi(x;\theta)$. If task is episodic, approx. $J(\theta)$ by MC. Expectation depends on θ. By a theorem under some regularity ass. $\nabla J(\theta) = \nabla \mathbb{E}_{\tau \sim \pi_\theta} r(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[r(\tau) \nabla \log \pi_\theta(\tau)]$. $\nabla \log \pi_\theta(\tau) = \sum_{t=0}^{T-1} \nabla \log \pi_\theta(a_t \mid x_t)$.

This gradient estimate is unbiased, but with high variance. Changing all rewards by **baseline** b does not change the gradient. Baseline can depend on previous states: $b(\tau_{0:t-1})$. Can use this to e.g. leave only the "reward to go": $\nabla J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \sum_{t=0}^{T} \gamma^t G_t \nabla \log \pi(a_t \mid x_t; \theta)$, where $G_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_t$. This is **REINFORCE**.

Policy gradient theorem: $\nabla J(\theta) = \mathbb{E}_{(x,a) \sim \pi_\theta} Q(x,a) \nabla \log \pi(a \mid x; \theta)$.

Advantage: $A^\pi(x,a) \coloneqq Q^\pi(x,a) - V^\pi(x)$.

Actor says what to do (π), **critic** says how good the decision is (Q).

Online AC allows to use policy gradient methods in online setting. TD-learn the critic.

$\theta_\pi \leftarrow \theta_\pi + \eta_t Q(x, a; \theta_Q) \nabla \log \pi(a \mid x; \theta_\pi)$

$\theta_Q \leftarrow \theta_Q - \eta_t (Q(x,a;\theta_Q) - r - \gamma Q(x', \pi(x', \theta_\pi); \theta_Q)) \nabla Q(x, a; \theta_Q)$.

A2C: for actor update, use baseline: subtract $V(x;\theta_V)$ from $Q(x,a;\theta_Q)$, resulting in A. This is more stable. **A3C**: parallel A2C.

Trust-region policy optimization (TRPO): not offline, but can reuse data from rollouts in the same iteration. On each iteration a fixed critic optimizes the policy. Select $\theta_{k+1} = \arg\max_\theta \hat{J}(\theta_k, \theta)$ s.t. $\mathbb{E}_{x \sim p_{\theta_k}} KL(\pi(\cdot \mid x) \| \pi(\cdot \mid x)) \le \delta$, where $\hat{J}(\theta_k, \theta) = \mathbb{E}_{(x,a) \sim \pi_{\theta_k}} \frac{\pi(a|x;\theta)}{\pi(a|x;\theta_k)} A^{\pi_{\theta_k}}(x,a)$. Don't go far from θ_k.

PPO: lagrangian relaxation of TRPO.

DDPG: off-policy actor-critic, uses a replay buffer. Replace $\max_a$ with policy actions and add some noise to policy for exploration.

SVG (stoch. value gradients): DDPG with support for stochastic policies. If $a \sim \pi(x;\theta_\pi)$ is s.t. $a = \psi(x;\theta_\pi,\varepsilon)$, then $\nabla_{\theta_\pi} \mathbb{E}_{a \sim \pi_{\theta_\pi}} Q(x,a;\theta_Q) = \mathbb{E}_\varepsilon \nabla_{\theta_\pi} Q(x, \psi(x;\theta_\pi,\varepsilon);\theta_Q)$.

SAC: add $\lambda H(\pi_\theta)$ to $J(\theta)$ to incentivize stochasticity. **RLHF**: unknown reward, can expensively (human) compare two states.

11 Model-based Deep RL

Learning a model can help reduce the sample complexity.

Receding-horizon/model-predictive control: plan over finite horizon, execute first action, replan. If model is deterministic, extrapolating is cheap. Planning requires optimization, which has vanishing/exploding gradients. **Random shooting**: pick random sequences of actions (Gaussian). Can mix with MPC. If we have V, can add it to the J estimate. **Stochastic average approximation**: for stochastic policies, average the trajectories. Approx. avg. by MC sampling.

Instead of optimizing over $a_{t:t+H-1}$, optimize over **parametrized policies** $\pi(x;\theta)$.

12 Appendix

Jensen: g convex: $g(E[X]) \le E[g(X)]$

g concave (e.g. log): $g(E[X]) \ge E[g(X)]$

Inv: $A^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

$||x||_p = \left(\sum_{i=1}^d |x_i|^p\right)^{\frac{1}{p}}$ $\quad ||x||_\infty = \max_{i \in \{1,..,d\}} |x_i|$

Softmax $\sigma(z)_i = e^{z_i} / \sum_{j=1}^D e^{z_j}$

Laplace: $\mathcal{L} = \frac{1}{2b} exp(-\frac{|x-\mu|}{b})$

Variance & Covariance $\quad \mathbb{E}(X) \coloneqq \int_{-\infty}^{+\infty} x f(x) dx$

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

$\mathbb{V}(X+Y) = \mathbb{V}(X) + \mathbb{V}(Y) + 2\mathrm{Cov}(X,Y)$

$\mathbb{V}(AX) = A\mathbb{V}(X)A^T, \mathbb{V}[\alpha X] = \alpha^2 \mathbb{V}[X]$

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

Distributions $\quad \mathrm{Exp}(x|\lambda) = \lambda e^{-\lambda x}$, $\mathrm{Ber}(x|\theta) = \theta^x (1-\theta)^{(1-x)}$

Sigmoid: $\sigma(x) = 1/(1 + e^{-x})$

$a\mathcal{N}(\mu_1, \sigma_1^2) + \mathcal{N}(\mu_2, \sigma_2^2) = \mathcal{N}(a\mu_1 + \mu_2, a^2\sigma_1^2 + \sigma_2^2)$

Normal CDF $\quad x \sim \mathcal{N}(0,1) \Rightarrow \mathbb{P}(x \le z) = \Phi(z), \mathbb{P}(x \le \Phi^{-1}(z)) = z$. $x \sim \mathcal{N}(\mu, \sigma^2) \Rightarrow \mathbb{P}(x \le z) = \Phi(\frac{z-\mu}{\sigma}), \mathbb{P}(x \le \mu + \sigma\Phi^{-1}(z)) = z$

Derivatives $\quad (fg)' = f'g + fg'; (f/g)' = (f'g - fg')/g^2$

$f(g(x))' = f'(g(x))g'(x); \log(x)' = 1/x$

$\partial_x \mathbf{b}^\top \mathbf{x} = \partial_x \mathbf{x}^\top \mathbf{b} = \mathbf{b}, \partial_x \mathbf{x}^\top \mathbf{x} = \partial_x ||\mathbf{x}||_2^2 = 2\mathbf{x}$,

$\partial_x \mathbf{x}^\top \mathbf{A} \mathbf{x} = (\mathbf{A}^\top + \mathbf{A})\mathbf{x}, \partial_x(\mathbf{b}^\top \mathbf{A} \mathbf{x}) = \mathbf{A}^\top \mathbf{b}$,

$\partial_X(\mathbf{c}^\top \mathbf{X} \mathbf{b}) = \mathbf{c}\mathbf{b}^\top, \partial_X(\mathbf{c}^\top \mathbf{X}^\top \mathbf{b}) = \mathbf{b}\mathbf{c}^\top$,

$\partial_x(||\mathbf{x} - \mathbf{b}||_2) = \frac{\mathbf{x}-\mathbf{b}}{||\mathbf{x}-\mathbf{b}||_2}, \partial_X(||\mathbf{X}||_F^2) = 2\mathbf{X}$,

$\partial_x ||\mathbf{x}||_1 = \frac{\mathbf{x}}{|\mathbf{x}|}, \partial_x ||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2 = \mathbf{2}(\mathbf{A}^\top \mathbf{A} \mathbf{x} - \mathbf{A}^\top \mathbf{b})$,

Matrix Identities $\quad (AB)^\top = B^\top A^\top, U(VU+I)^{-1} = (UV+I)^{-1}U$

$(A+B)^{-1} = A^{-1} - A^{-1}(A^{-1} + B^{-1})^{-1}A^{-1}$

$(\sigma^2 I + QPQ^\top)^{-1} = \sigma^{-2}I - \sigma^{-2}Q(\sigma^2 P^{-1} + Q^\top Q)^{-1}Q^\top$

$\frac{1}{2}z^\top Az + b^\top z = \frac{1}{2}(z + A^{-1}b)^\top A(z + A^{-1}b) - b^\top A^{-1}b$

$(A + xx^\top)^{-1} = A^{-1} - A^{-1}x(1 + x^\top A^{-1}x)^{-1}x^\top A^{-1} = A^{-1} - \frac{(A^{-1}x)(A^{-1}x)^\top}{1 + x^\top A^{-1}x}$

The characteristic function of a RV X is $\phi_X(t) = \mathbb{E}[\exp(it^\top X)]$; if $X \sim \mathcal{N}: \phi_X = \exp(it^\top\mu - \frac{1}{2}t^\top\Sigma t)$.