Probabilistic AI Tutorial

Bayesian Learning and RL

Vincent Fortuin (*fortuin@inf.ethz.ch*)

December 2019

Institute of Machine Learning, ETH Zürich

Table of contents

Bayesian learning

Learning to approximate functions

- Let us have a data set $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d}$ with labels $\mathbf{y} = [y_1, \ldots, y_n]^\top \in \mathbb{R}^n$.
- We assume there is an underlying function $f : \mathbb{R}^d \to \mathbb{R}$ with $y = f(\mathbf{x}) + \epsilon$ where ϵ is some noise.
- We are trying to approximate f with some parameterized function $\hat{f}_\theta$, such that $\hat{f}_\theta(\mathbf{x}) \approxeq f(\mathbf{x})$.

Optimization vs. inference

Optimization view:

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta; \mathbf{X}, \mathbf{y}) \quad \text{with} \quad \mathcal{L}(\theta; \mathbf{X}, \mathbf{y}) = \sum_{i=1}^{n} D(\hat{f}_\theta(\mathbf{x}_i), y_i) + \lambda\Omega(\theta)$$

for discrepancy $D(\cdot, \cdot)$ and regularizer $\Omega(\cdot)$

Inference view:

$$p(\theta^* \mid \mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{X}, \mathbf{y} \mid \hat{f}_{\theta^*})\; p(\theta^*)}{p(\mathbf{X}, \mathbf{y})} = \frac{p(\mathbf{y} \mid \mathbf{X}, \hat{f}_{\theta^*})\; p(\theta^*)}{\sum_{\tilde{\theta}} p(\mathbf{y} \mid \mathbf{X}, \hat{f}_{\tilde{\theta}})}$$

Refresher: Linear regression

- Assume $y = f(\mathbf{x}) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma_y^2)$.
- The log likelihood is

$$\log p(\mathbf{y} \mid \mathbf{X}, \hat{f}_\theta) = \log \mathcal{N}(\mathbf{y}; \hat{f}_\theta(\mathbf{X}), \sigma_y^2 \mathbf{I}) = -\frac{1}{2\sigma_y^2} \|\hat{f}_\theta(\mathbf{X}) - \mathbf{y}\|_2^2 + \text{const.}$$

- With a linear model $\hat{f}_\theta(\mathbf{x}) = \mathbf{x}^\top \theta$ this gives

$$\theta_{MLE} = \arg\max_\theta \, p(\mathbf{y} \mid \mathbf{X}, \hat{f}_\theta) = \arg\min_\theta \|\hat{f}_\theta(\mathbf{X}) - \mathbf{y}\|_2^2$$

- This can be solved in closed form[1] as

$$\theta_{MLE} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}$$

[1]Bishop 2006.

Bayesian linear regression

- Assume now we have some prior $p(\theta) = \mathcal{N}(\mu_\theta, \Sigma_\theta)$.
- The posterior over θ becomes

$$p(\theta^* \mid \mathbf{X}, \mathbf{y}) \propto p(\mathbf{y} \mid \mathbf{X}, \hat{f}_{\theta^*})\, p(\theta^*) = \mathcal{N}(\hat{f}_\theta(\mathbf{X}), \sigma_y^2 \mathbf{I})\, \mathcal{N}(\mu_\theta, \Sigma_\theta)$$

- Due to Gaussian conjugacy, we can solve this in closed form[2] as

$$\begin{aligned} p(\theta^* \mid \mathbf{X}, \mathbf{y}) &= \mathcal{N}(\theta^*; \mu_\theta^*, \Sigma_\theta^*) \\ \text{with} \quad \Sigma_\theta^* &= \left(\Sigma_\theta^{-1} + \frac{1}{\sigma_y^2} \mathbf{X}^\top \mathbf{X} \right)^{-1} \\ \text{and} \quad \mu_\theta^* &= \Sigma_\theta^* \left(\Sigma_\theta^{-1} \mu_\theta + \frac{1}{\sigma_y^2} \mathbf{X}^\top \mathbf{y} \right) \end{aligned}$$

[2]Bishop 2006.

Zero-mean isotropic prior

- If we choose the prior to be zero-mean and isotropic, i.e. $p(\theta) = \mathcal{N}(\mathbf{0}, \sigma_\theta^2 \mathbf{I})$, the posterior simplifies to

$$
\begin{aligned}
p(\theta^* \mid \mathbf{X}, \mathbf{y}) &= \mathcal{N}(\theta^*; \mu_\theta^*, \Sigma_\theta^*) \\
\text{with} \quad \Sigma_\theta^* &= \left(\frac{1}{\sigma_\theta^2} \mathbf{I} + \frac{1}{\sigma_y^2} \mathbf{X}^\top \mathbf{X} \right)^{-1} \\
\text{and} \quad \mu_\theta^* &= \frac{1}{\sigma_y^2} \Sigma_\theta^* \mathbf{X}^\top \mathbf{y}
\end{aligned}
$$

- The posterior mode μ_θ^* as also called *maximum a posteriori (MAP) estimate*.
- Note that this is equivalent to ridge regression[3]

$$
\mu_\theta^* = \arg\min_\theta \|\hat{f}_\theta(\mathbf{X}) - \mathbf{y}\|_2^2 + \lambda \|\theta\|_2^2 \quad \text{with} \quad \lambda = \frac{\sigma_y^2}{\sigma_\theta^2}
$$

[3] Bishop 2006.

Predictive posterior

- Assume we want to predict the response y^* at a new test point $\mathbf{x}^*$. The MAP estimate would just be $\hat{y}^* = \mathbf{x}^{*\top}\mu_\theta^*$.
- In contrast, the full predictive posterior is

$$
\begin{aligned}
p(y^* \mid \mathbf{X}, \mathbf{y}, \mathbf{x}^*) &= \int p(y^* \mid \theta^*, \mathbf{x}^*)\, p(\theta^* \mid \mathbf{X}, \mathbf{y})\ \mathrm{d}\theta^* \\
&= \int \mathcal{N}(\mathbf{x}^{*\top}\theta^*, \sigma_y^2)\, \mathcal{N}(\mu_\theta^*, \Sigma_\theta^*)\ \mathrm{d}\theta^*
\end{aligned}
$$

- Due to Gaussian conjugacy this has a closed form[4] solution:

$$
\begin{aligned}
p(y^* \mid \mathbf{X}, \mathbf{y}, \mathbf{x}^*) &= \mathcal{N}(y^*; \mu_{y^*}, \sigma_{y^*}^2) \\
\text{with} \quad \mu_{y^*} &= \mathbf{x}^{*\top}\mu_\theta^* \\
\text{and} \quad \sigma_{y^*}^2 &= \sigma_y^2 + \mathbf{x}^{*\top}\Sigma_\theta^*\mathbf{x}^*
\end{aligned}
$$

[4]Murphy 2012.

Gaussian processes

- A Gaussian process is a prior over functions $f \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot))$, where $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ is called a *mean function* and $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}\left[(f(\mathbf{x}) - m(\mathbf{x}))^\top (f(\mathbf{x}') - m(\mathbf{x}'))\right]$ is called a *kernel function*.
- The predictive posterior[5] is

$$
\begin{aligned}
& p(\mathbf{y}^* \mid \mathbf{X}, \mathbf{y}, \mathbf{x}^*) = \mathcal{N}(\mu_y^*, \Sigma_y^*) \\
\text{with} \quad & \mu_y^* = \mathbf{k}(\mathbf{X}, \mathbf{x}^*)^\top \left[\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_y^2 \mathbf{I}\right]^{-1} \mathbf{y} \\
\text{and} \quad & {\Sigma_y}^* = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{X}, \mathbf{x}^*)^\top \left[\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_y^2 \mathbf{I}\right]^{-1} \mathbf{k}(\mathbf{X}, \mathbf{x}^*) + \sigma_y^2 \mathbf{I}
\end{aligned}
$$

- Here, $\mathbf{k}(\mathbf{X}, \mathbf{x}^*)$ is a column vector with elements $\mathbf{k}(\mathbf{X}, \mathbf{x}^*)_i = k(\mathbf{x}_i, \mathbf{x}^*)$ and $\mathbf{K}(\mathbf{X}, \mathbf{X})$ is a symmetric square matrix with elements $\mathbf{K}(\mathbf{X}, \mathbf{X})_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

[5] Rasmussen & Williams 2006.

Choice of kernel

- The kernel function $k(\cdot, \cdot)$ has to be positive-semidefinite, i.e. the matrix $\mathbf{K}(\mathbf{X}, \mathbf{X})$ has to be positive-semidefinite for every $\mathbf{X}$.
- Every positive-semidefinite kernel function can be written as an inner product in a certain feature (Hilbert) space[6] $\mathcal{H}$, i.e. $k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\Phi}(\mathbf{x})^\top \boldsymbol{\Phi}(\mathbf{x}')$ for some $\boldsymbol{\Phi} : \mathbb{R}^d \to \mathcal{H}$.
- A popular kernel function is the *radial basis function* (RBF) kernel
$$k_{RBF}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{x}'\|^2\right)$$
- It turns out that $\dim(\mathcal{H}_{RBF}) = \infty$. So the GP allows to perform a regression in an infinite-dimensional feature space, while the computational complexity only depends on n. It is thus a form of *nonparametric* regression.

[6] Mercer 1909.

Bayesian deep learning

- The Bayesian learning paradigm can be extended to complex parametric models, such as deep neural networks. This is called *Bayesian deep learning*.
- The basic assumptions are still the same, i.e.

$$p(\theta^* \mid \mathbf{X}, \mathbf{y}) \propto p(\mathbf{y} \mid \mathbf{X}, \hat{f}_{\theta^*})\, p(\theta^*)$$

$$p(y^* \mid \mathbf{X}, \mathbf{y}, \mathbf{x}^*) = \int p(y^* \mid \theta^*, \mathbf{x}^*)\, p(\theta^* \mid \mathbf{X}, \mathbf{y})\ \mathrm{d}\theta^*$$

- However, without conjugacy, exact inference on the network weights θ is usually not tractable anymore.
- One therefore needs to resort to approximate inference methods (variational inference, MCMC, etc.).

Reinforcement learning

Types of RL

- There are generally two types of RL: *model-based* and *model-free*.
- Model-based RL involves learning a transition model $P(s_{t+1} \mid s_t, a_t)$ of the environment, as well as a state value function $V(s)$.
- Model-free RL does not model the transitions, but directly learns a policy $\pi(a_t \mid s_t)$ or a state-action value function (a.k.a. Q-function) $Q(s_t, a_t)$.
- Learning just the policy or Q-function can often be cheaper than learning the whole transition model.
- A good intuition for different types of value learning is provided at `https://distill.pub/2019/paths-perspective-on-value-learning/`.

Q-learning

- In Q-learning, we learn a state-action value function $Q(s_t, a_t)$.
- The policy can then for instance be chosen to be $\pi(a_t \mid s_t) = \delta(\arg\max_{a_t} Q(a_t, s_t))$, where $\delta(\cdot)$ is the Dirac measure.
- During learning, for every transition tuple (s_t, a_t, r_t, s_{t+1}) we observe, we update the Q-function as

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

- Here, α_t is the step size and γ is the discount factor.

Questions?

References

- Bishop, C. M. (2006). Pattern recognition and machine learning. Springer Science+ Business Media.
- Murphy, K. P. (2012). Machine learning: a probabilistic perspective. MIT press.
- Rasmussen, C. E., & Williams, C. K. (2006). Gaussian Processes for Machine Learning. MIT press.
- Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character, 83(559), 69-70.