Variational Continual Learning

5.24 来嘉豪

1. **methodology**
   - theory
   - application in continual learning
2. **application in Predictive Domain Adaptation**

Bayesian inference formalizes *model inversion*, the process of passing from a prior to a posterior in light of data.

$$\underbrace{p(\theta|y)}_{\text{posterior}} = \frac{\overbrace{p(y|\theta)}^{\text{likelihood}}\ \overbrace{p(\theta)}^{\text{prior}}}{\underbrace{\int p(y,\theta)\mathrm{d}\theta}_{\text{marginal likelihood } p(y)\text{ (model evidence)}}}$$

Approximate Bayesian Inference

There are two approaches to approximate inference. They have complementary strengths and weaknesses.

Stochastic approximate inference

in particular sampling

1. design an algorithm that draws samples $\theta^{(1)}, \ldots, \theta^{(m)}$ from $p(\theta|y)$
2. inspect sample statistics (e.g., histogram, sample quantiles, …)

☑ asymptotically exact
☒ computationally expensive
☒ tricky engineering concerns

Structural approximate inference

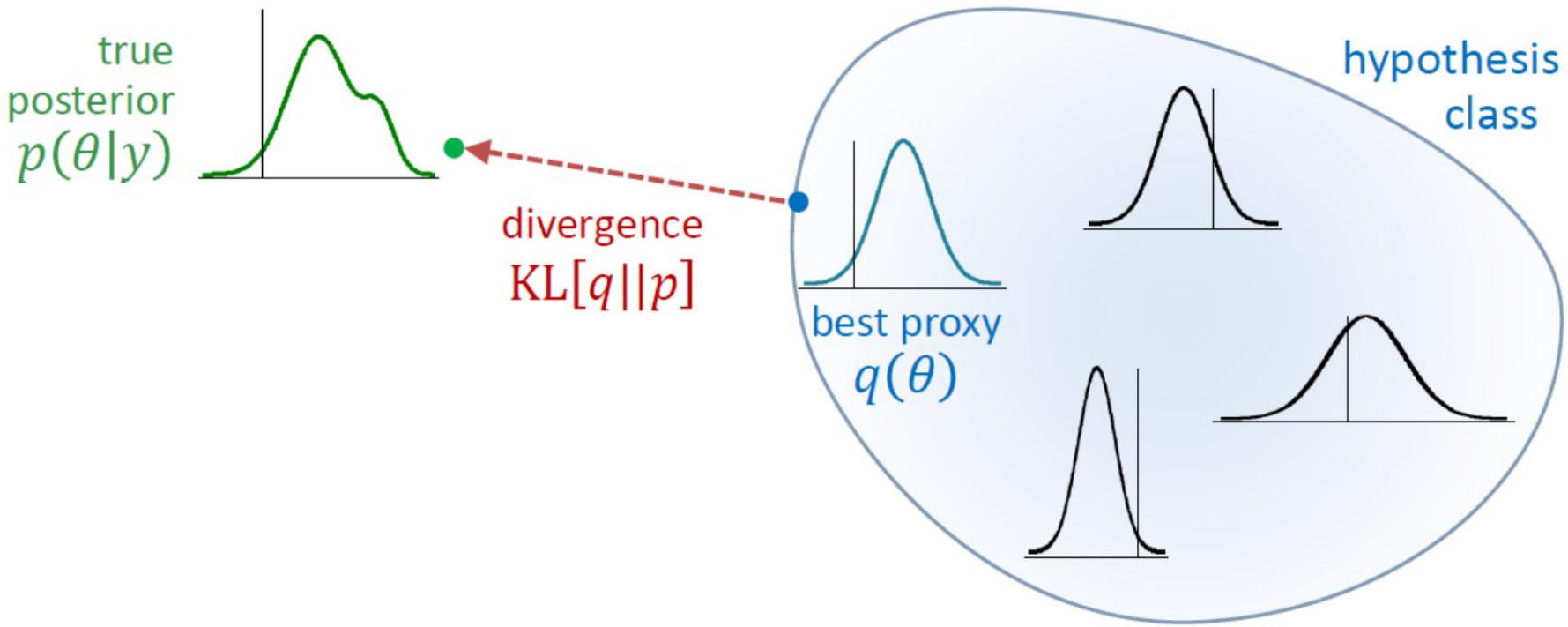
in particular variational Bayes

1. find an analytical proxy $q(\theta)$ that is maximally similar to $p(\theta|y)$
2. inspect distribution statistics of $q(\theta)$ (e.g., mean, quantiles, intervals, …)

☑ often insightful – and lightning-fast!
☒ often hard work to derive
☒ requires validation via sampling

Variational Bayesian Inference

Variational Bayesian (VB) inference generalizes the idea behind the Laplace approximation. In VB, we wish to find an approximate density that is maximally similar to the true posterior.



Variational Calculus

Variational Bayesian inference is based on variational calculus.

Standard calculus

Newton, Leibniz, and others

- functions $f: x \mapsto f(x)$
- derivatives $\frac{\mathrm{d}f}{\mathrm{d}x}$

Example: maximize the likelihood expression $p(y|\theta)$ w.r.t. θ

Variational calculus

Euler, Lagrange, and others

- functionals $F: f \mapsto F(f)$
- derivatives $\frac{\mathrm{d}F}{\mathrm{d}f}$

Example: maximize the entropy $H[p]$ w.r.t. a probability distribution $p(x)$

Leonhard Euler
(1707 – 1783)

Swiss mathematician, 'Elementa Calculi Variationum'

Variational Calculus and Free Energy

Variational calculus lends itself nicely to approximate Bayesian inference.

$$\ln p(y) = \ln \frac{p(y,\theta)}{p(\theta|y)}$$

$$= \int q(\theta) \ln \frac{p(y,\theta)}{p(\theta|y)} \, d\theta$$

Multiply both sides with q(θ) and take derivative with respect to dθ

$$= \int q(\theta) \ln \frac{p(y,\theta)}{p(\theta|y)} \frac{q(\theta)}{q(\theta)} \, d\theta$$

Multiply and divide by q(θ)

$$= \int q(\theta) \left(\ln \frac{q(\theta)}{p(\theta|y)} + \ln \frac{p(y,\theta)}{q(\theta)} \right) d\theta$$

$$= \underbrace{\int q(\theta) \ln \frac{q(\theta)}{p(\theta|y)} \, d\theta}_{\mathrm{KL}[q||p]} + \underbrace{\int q(\theta) \ln \frac{p(y,\theta)}{q(\theta)} \, d\theta}_{F(q,y)}$$

$\mathrm{KL}[q||p]$
divergence between $q(\theta)$ and $p(\theta|y)$

$F(q,y)$
free energy

Computing the free energy

We can decompose the free energy $F(q, y)$ as follows:

$$
\begin{aligned}
F(q, y) &= \int q(\theta) \ln \frac{p(y,\theta)}{q(\theta)} \mathrm{d}\theta \\
&= \int q(\theta) \ln p(y, \theta) \; d\theta - \int q(\theta) \ln q(\theta) \; d\theta \\
&= \underbrace{\langle \ln p(y, \theta) \rangle_q}_{\text{expected log-joint}} + \underbrace{H[q]}_{\text{Shannon entropy}}
\end{aligned}
$$

Variational inference under the mean-field assumption

$$
\begin{aligned}
F(q,y) &= \int q(\theta) \ln \frac{p(y,\theta)}{q(\theta)} \mathrm{d}\theta \\
&= \int \prod_i q_i \times \left(\ln p(y,\theta) - \sum_i \ln q_i \right) \mathrm{d}\theta \qquad \text{mean-field assumption: } q(\theta) = \textstyle\prod_i q_i(\theta_i) \\
&= \int q_j \prod_{\setminus j} q_i \left(\ln p(y,\theta) - \ln q_j \right) \mathrm{d}\theta - \int q_j \prod_{\setminus j} q_i \sum_{\setminus j} \ln q_i \, \mathrm{d}\theta \\
&= \int q_j \left(\underbrace{\int \prod_{\setminus j} q_i \ln p(y,\theta) \, \mathrm{d}\theta_{\setminus j}}_{\langle \ln p(y,\theta) \rangle_{q_{\setminus j}}} - \ln q_j \right) \mathrm{d}\theta_j - \int q_j \int \prod_{\setminus j} q_i \ln \prod_{\setminus j} q_i \, \mathrm{d}\theta_{\setminus j} \, \mathrm{d}\theta_j \\
&= \int q_j \ln \frac{\exp\left(\langle \ln p(y,\theta) \rangle_{q_{\setminus j}} \right)}{q_j} \mathrm{d}\theta_j + c \\
&= -\mathrm{KL}\left[q_j \,||\, \exp\left(\langle \ln p(y,\theta) \rangle_{q_{\setminus j}} \right) \right] + c
\end{aligned}
$$

Variational inference under the mean-field assumption

In summary:

$$F(q,y) = -\mathrm{KL}\left[q_j \,||\, \exp\left(\langle \ln p(y,\theta)\rangle_{q_{\setminus j}}\right)\right] + c$$

Suppose the densities $q_{\setminus j} \equiv q(\theta_{\setminus j})$ are kept fixed. Then the approximate posterior $q(\theta_j)$ that maximizes $F(q,y)$ is given by:

$$\begin{aligned} q_j^* &= \arg\max_{q_j} F(q,y) \\ &= \frac{1}{Z}\exp\left(\langle \ln p(y,\theta)\rangle_{q_{\setminus j}}\right) \end{aligned}$$

Therefore:

$$\ln q_j^* = \underbrace{\langle \ln p(y,\theta)\rangle_{q_{\setminus j}}}_{=:I(\theta_j)} - \ln Z$$

This implies a straightforward algorithm for variational inference:

1. Initialize all approximate posteriors $q(\theta_i)$, e.g., by setting them to their priors.
2. Cycle over the parameters, revising each given the current estimates of the others.
3. Loop until convergence.

Continual learning: data continuously arrive in non-i.i.d way or new tasks may emerge. Continual learning models adapt to perform well on entire tasks in an incremental way.

Variational Continual learning: merge online variational inference(VI) , Monte Carlo VI and coreset data summarization method to yield VCL. This framework is applicable to discriminative and generative models.

sequentially arriving datasets $\{\boldsymbol{x}_t^{(n)}, y_t^{(n)}\}_{n=1}^{N_t}$

$$p(\boldsymbol{\theta}|\mathcal{D}_{1:T}) \propto p(\boldsymbol{\theta}) \prod_{t=1}^{T} \prod_{n_t=1}^{N_t} p(y_t^{(n_t)}|\boldsymbol{\theta}, \boldsymbol{x}_t^{(n_t)}) = p(\boldsymbol{\theta}) \prod_{t=1}^{T} p(\mathcal{D}_t|\boldsymbol{\theta}) \propto p(\boldsymbol{\theta}|\mathcal{D}_{1:T-1}) p(\mathcal{D}_T|\boldsymbol{\theta}).$$

$$q_t(\boldsymbol{\theta}) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\Big(q(\boldsymbol{\theta}) \parallel \frac{1}{Z_t} q_{t-1}(\boldsymbol{\theta})\, p(\mathcal{D}_t|\boldsymbol{\theta})\Big), \text{ for } t = 1, 2, \ldots, T.$$

parametric modeling under mean-field assumption

$$q_t(\boldsymbol{\theta}) = \prod_{d=1}^{D} \mathcal{N}(\theta_{t,d}; \mu_{t,d}, \sigma_{t,d}^2)$$

$$\mathcal{L}_{\mathrm{VCL}}^t(q_t(\boldsymbol{\theta})) = \sum_{n=1}^{N_t} \mathbb{E}_{\boldsymbol{\theta} \sim q_t(\boldsymbol{\theta})} \Big[\log p(y_t^{(n)}|\boldsymbol{\theta}, \mathbf{x}_t^{(n)})\Big] - \mathrm{KL}(q_t(\boldsymbol{\theta})||q_{t-1}(\boldsymbol{\theta}))$$

Coreset: retains important training data from previous dataset

Algorithm 1 Coreset VCL

Input: Prior $p(\boldsymbol{\theta})$.

Output: Variational and predictive distributions at each step $\{q_t(\boldsymbol{\theta}), p(y^*|\boldsymbol{x}^*, \mathcal{D}_{1:t})\}_{t=1}^T$.

Initialize the coreset and variational approximation: $C_0 \leftarrow \emptyset$, $\tilde{q}_0 \leftarrow p$.

for $t = 1 \dots T$ **do**

Observe the next dataset $\mathcal{D}_t$.

$C_t \leftarrow$ update the coreset using C_{t-1} and $\mathcal{D}_t$.

Update the variational distribution for non-coreset data points:

$$\tilde{q}_t(\boldsymbol{\theta}) \leftarrow \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\big(q(\boldsymbol{\theta}) \parallel \tfrac{1}{\tilde{Z}} \tilde{q}_{t-1}(\boldsymbol{\theta})\, p(\mathcal{D}_t \cup C_{t-1} \setminus C_t|\boldsymbol{\theta})\big). \tag{2}$$

Compute the final variational distribution (only used for prediction, and not propagation):

$$q_t(\boldsymbol{\theta}) \leftarrow \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\big(q(\boldsymbol{\theta}) \parallel \tfrac{1}{Z} \tilde{q}_t(\boldsymbol{\theta}) p(C_t|\boldsymbol{\theta})\big). \tag{3}$$

Perform prediction at test input $\boldsymbol{x}^*$: $p(y^*|\boldsymbol{x}^*, \mathcal{D}_{1:t}) = \int q_t(\boldsymbol{\theta}) p(y^*|\boldsymbol{\theta}, \boldsymbol{x}^*) \mathrm{d}\boldsymbol{\theta}$.

end for

$$p(\boldsymbol{\theta}|\mathcal{D}_{1:t} \backslash C_t) = \underbrace{p(\boldsymbol{\theta}|\mathcal{D}_{1:t-1} \backslash C_{t-1})}_{\substack{\text{previous posterior from} \\ \text{data not in old coreset}}} \underbrace{p(C_{t-1} \backslash C_t|\boldsymbol{\theta})}_{\substack{\text{likelihood from} \\ \text{data leaving coreset}}} \underbrace{p(D_t \backslash C_t|\boldsymbol{\theta})}_{\substack{\text{likelihood from new} \\ \text{data not in new coreset}}} \approx \tilde{q}_{t-1}(\boldsymbol{\theta}) p(\mathcal{D}_t \cup C_{t-1} \backslash C_t|\boldsymbol{\theta}).$$

$$p(\boldsymbol{\theta}|\mathcal{D}_{1:t}) \propto \underbrace{p(\boldsymbol{\theta}|\mathcal{D}_{1:t} \setminus C_t)}_{\substack{\text{posterior from data} \\ \text{not in new coreset}}} \underbrace{p(C_t|\boldsymbol{\theta})}_{\substack{\text{likelihood from data} \\ \text{in new coreset}}} \approx \tilde{q}_t(\boldsymbol{\theta}) p(C_t|\boldsymbol{\theta}).$$

Permuted MNIST: This is a popular continual learning benchmark (Goodfellow et al., 2014a; Kirkpatrick et al., 2017; Zenke et al., 2017). The dataset received at each time step $\mathcal{D}_t$ consists of labeled MNIST images whose pixels have undergone a fixed random permutation. We compare VCL to EWC, SI, and diagonal LP. For all algorithms, we use fully connected single-head networks with two hidden layers, where each layer contains 100 hidden units with ReLU activations. We evaluate three versions of VCL: VCL with no coreset, VCL with a random coreset, and VCL with a coreset selected by the K-center method. For the coresets, we select 200 data points from each task.

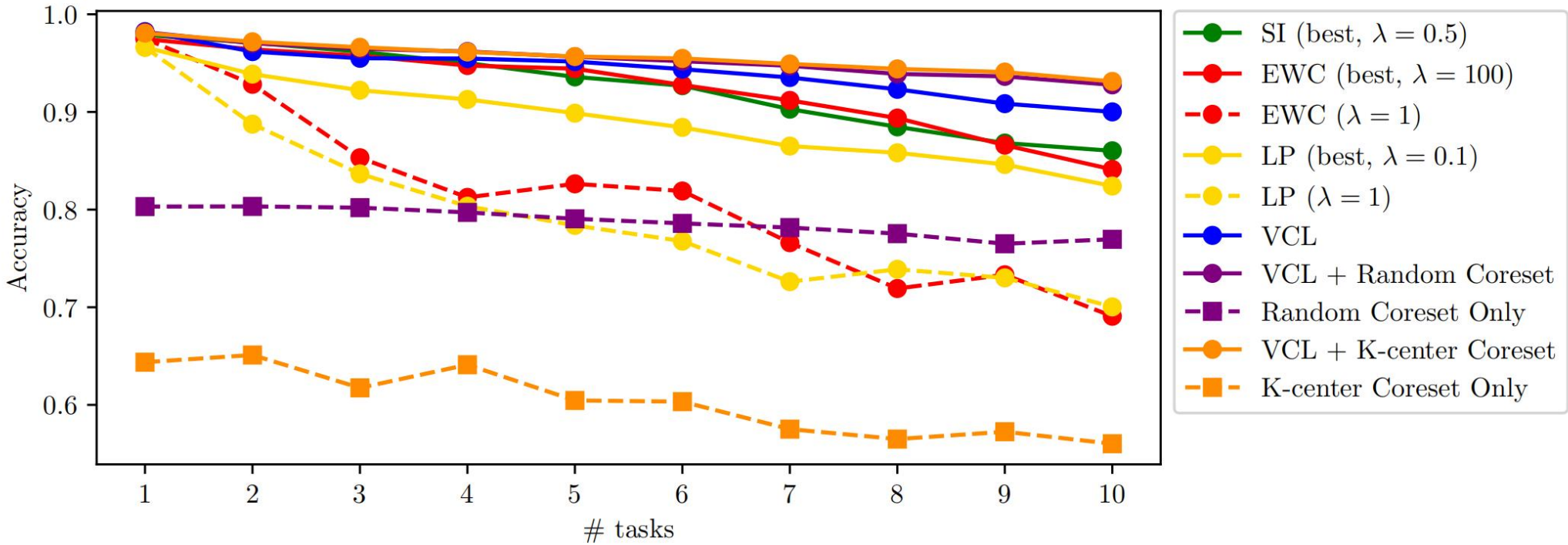


Figure 2: Average test set accuracy on all observed tasks in the Permuted MNIST experiment.

- whether useful for domain adaptation
- architecture of neural network
- unsurperised version of the VCL
- one model or multiple models
- execute it in a graph-based manner rather than a sequential order
- or choose to use optimal transport or other topological-critical methods to find a transfer path and perform VCL algorithm directly