11/15/19

8805 Learning from Data: Lecture 24

Today: Finish Bayesian Optimization and start Bayesian Neural Networks.

- Bayesian optimization notebooks:
  Bayesian_optimization.ipynb and mini-project_IIIa_bayesian_optimization.ipynb

- Step through Mini-project IIIa to see what is needed.
  - Code for function and standard SciPy optimization
  - Specification of statistical model and acquisition function, plotting 10 (or so) iteration and summary plots.

  GP → stat model

  - * Changing acquisition function to 'LCB'. What's that?
  - * Assessing exploration vs. exploitation
  - * Adding noise
  - * Changing the GP kernel and analyzing the results
  - Optional task: implementing BayesOpt algorithm (open black box)
  - * Bivariate example → for a plus
    ⇒ follow details.

- Now go back to Bayesian_optimization.ipynb and see where these come from.

  ① Ingredients:

  put on board

  a. Objective function to optimize $f(\vec{\theta}) = \chi^2(\vec{\theta}) = \sum_{i=1}^{N} \frac{[y_i^{exp} - y_i^{th}(\vec{\theta})]^2}{\sigma_i^2}$
     - assumed costly to evaluate. Find $\vec{\theta}_*$ that minimizes.

  b. Statistical model for $f(\vec{\theta})$: $p(f|\mathcal{D})$ where $\mathcal{D}$ is the current set of evaluations of f: $\mathcal{D} = \{(\vec{\theta}_i, f(\vec{\theta}_i))\}$. Start with $p(f)$ as a Gaussian process GP and update via Bayes theorem (recall GP procedure) with each additional $(\theta_i, f(\theta_i))$.

  c. Acquisition function $A(\vec{\theta}|\mathcal{D})$. Take maximum wrt $\vec{\theta}$ to determine θ_{i+1} given the current data $\mathcal{D}$ (full history). Balances between "exploration" and "exploitation"

11/15/19

② Step Through code

- univariate example: plot function and find minimum from Scipy optimize.minimize (requires starting point).
  - no noise at first.
- Create GPyOpt object with GPyOpt.methods.BayesianOptimization
  - specify objective function $f(\theta)$, domain, initial data, acquisition function, whether or not exact function.
- run_optimization(max_iter, max_time, eps)

  # of evaluations — time budget — min distance between θ_i, θ_{i+1}

  try running multiple times

  - plot with plot_acquisition
- Bivariate example
  - This uses a built-in example
  - notice the setup for the BayesianOptimization object
    - look at choices for 'model_type'
    - look at choices for 'acquisition_type'
  - separate plots for mean, sd, acquisition function. with red dots for evaluation points.

③ Look at options for starting samples

- LHS and Mersenne twister (standard rng) have "holes"
- Sobol sequence does not have holes. — but good LHS projections

④ Concluding remarks — Step Through

⑤ Step back to acquisition Function

f_{min} is the lowest result so far.

- Expected improvement ⇒ EI. At each θ point, calculate expectation of $f_{min} - f(\theta)$ ⇒ improvement, using 0 if no improvement ($f_{min} - f(\theta) \geq 0$).
- Analytic evaluation of expectation value, because at a given θ_* the distribution pdf for f is a Gaussian (because it is a GP) specified by $\mu(\theta_*)$ and $\sigma_v^2(\theta_*) = C(\theta_*, \theta_*)$.
- Two pieces to the integral with $z = (f_{min} - \mu)/\sigma$

explorative if $\phi(z)$ dominates ⇒ prior has large uncertainty (large σ)

exploitive if $z\Phi(z)$ dominates ⇒ prior has low μ.

LCB ⇒ change ratio

11/15/19

- Bayesian neural networks: What is the idea?
  - ⇒ Think of how you would combine (or modify NN)?

- We use BNNs when we care about uncertainty

- Standard NN training via optimization is equivalent to doing MLE for the weights.
  - So point estimates only.
  - Recall issues with MLEs from "Why Bayes is Better"
  - General problem: susceptible to overfitting
- Can address overfitting (in part) by regularization ⇒ don't let weights get too big
  - Bayesian equivalent: put priors on weights (an L2 regularization ⇒ Gaussian prior pdf)
  - So now finding MAP estimate (maximizing prior)
- Bayesian way: posterior inference ⇒ BNNs (start with model, update with data)
  - This is a challenge both to model and to compute
  - Approximations like Laplace's method inadequate and MCMC computationally infeasible (many parameters)
  - ⇒ alternative is variational inference ← approximate the posterior
  - important for decision-making systems, smaller data situations, ...

- Ordinary workflow of neural network (supervised learner)
  - randomly initialize weights
  - given inputs compute outputs of neurons by layers, propagate to prediction
  - "loss function" computes deviation of predicted output $\hat{y}$ and expected y.
  - loss value is "back propagated" through layers, adjusting weights
- Output of ordinary ML does not come with variability or credibility
  - just a point prediction — no model of the world is explicitly constructed
  - there are weights and network topology, but no direct correlation to statistical model
- BNN: Prior describes key parameters, utilized as input to neural net. Output used to compute likelihood with pdf. Get posterior distribution by variational inference.

11/15/19

- Notebooks from Christian Forssen's course at Chalmers
  - (A) Bayesian_neural_networks_tif285.ipynb
  - (B) demo-Bayesian_neural_networks_tif285.ipynb

(B) Just look through and see what a problem, its modeling, and the output looks like.

(A) Bayesian neural networks
- Basic neural network
  - Step through details on the board. Connect to diagram.

Bottom line: goal is $p(y|\vec{x}, D) = \int p(y|\vec{x}, \vec{w})\, p(\vec{w}|D)\, d\vec{w}$

new output ↗ ; new input ↑ ; training data ↑ ; ↑ by marginalization

[this is what the neural network gives ⇒ deterministic given $\vec{x}$ and $\vec{w}$.]

Here $D = \{\vec{x}^{(i)}, y^{(i)}\}$ is a given training dataset.

- We need $p(\vec{w}|D) \propto p(D|\vec{w})\, p(\vec{w})$ by Bayes. ← ordinary ML finds MAP
  Then $p(D|\vec{w}) = \prod_i p(y^{(i)}|\vec{x}^{(i)}, \vec{w})$ is the likelihood.

- So how do we calculate the marginalization integral with thousands of parameters?
  ⇒ Variational inference.

- Review basic idea of VI and KL divergence.
  - $p(\vec{w}|D)$ is intractable → approximate true posterior with proxy variational distribution $q(\vec{w}|\vec{\theta})$ where we need to find $\vec{\theta}$ optimal $\equiv \vec{\theta}^*$
  - Find $\vec{\theta}^*$ by using Kullback-Leibler divergence
  - ⇒ Find $\vec{\theta}^*$ that maximizes $J_{ELBO}(\vec{\theta})$ ← Evidence Lower Bound