Bayesian Neural Networks for GW parameter Estimation

Hongyu Shen
Dept. of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

Spotlights

- Bayesian Neural Network on GW Parameter Estimation

- TensorFlow Application

https://github.com/skyve2012/BayesianNetTutorial

Backgrounds

- Assume Likelihood function on noisy observations x:

$$p(x \mid m_1, m_2)$$

- Assume flat prior on parameters (e.g. masses m_1, m_2)

$$p(m_1, m_2)$$

- Apply Bayesian rules to achieve the the posterior of the parameters

$$p(m_1, m_2 \mid x) \propto p(x \mid m_1, m_2)p(m_1, m_2)$$

Backgrounds

- Posterior function is not easy to evaluate due to difficulties in exploring the whole space of the parameters (red and blue)-> use MCMC sampling instead -> time-consuming

$$p(m_1, m_2 | x) = \frac{p(x | m_1, m_2)p(m_1, m_2)}{p(x)}$$

$$p(x | m_1, m_2) = \frac{1}{Z} \exp(-\frac{1}{2}(x - f(m_1, m_2))^{\mathsf{T}} \Sigma^{-1}(x - f(m_1, m_2)))$$

- We can make improvements with Bayesian Neural Networks!

https://github.com/skyve2012/BayesianNetTutorial

Content

- Hands-on Linear Regression Model
- Apply ideas from Linear Regression on GW parameter Estimation
- How to write DIY Bayesian Neural Networks using TensorFlow

https://github.com/skyve2012/BayesianNetTutorial

Resource

https://github.com/skyve2012/BayesianNetTutorial

Linear Regression

- Consider the following problem:

$$y = ax + b + \epsilon$$

- y and x are data; a and b are parameters to be learned. ϵ is universal noise across all values of x (0 mean, fixed variance)

- Best solution is least square under the linear assumption in the true model.

$$\min_{a,b} ||ax + b - y||^2 = max_{a,b} p(y|x; a, b)$$

$$p(y|x; a, b) \sim N(ax + b, \sigma_\epsilon^2)$$

Linear Regression

- What if the problem is slightly changed:

$$y = ax + b + \epsilon(x)$$

- ϵ is dependent on x in this case (0 mean).

- One way is to use weighted least square

- The other way is to treat $\epsilon(x) = \epsilon_\theta(x)$

$$max_{a,b,\theta} p(y \mid x; a, b, \theta)$$

$$p(y \mid x; a, b, \theta) \sim N(ax + b, \sigma^2_{\epsilon_\theta(x)})$$

Linear Regression

- What if we want to also consider the variance on the estimation $\hat{a}, \hat{b}$ given the distribution of the data: $p(x, y)$. Or equivalently, $p(a, b \mid x)$.

- Consider the previous problem: $max_{a,b,\theta} p(y \mid x; a, b, \theta)$

- We re-write the problem as: $max_{\theta,w} p(y \mid x; \theta, w)$

$$p(y \mid x; \theta, w) = \int_{a,b} p(y \mid a, b, x; \theta) p_w(a, b \mid x) dadb$$

- The blue part is can be learned via Variational Inference

https://github.com/skyve2012/BayesianNetTutorial

Variational Inference

- Learn a parameterized distribution to approximate the true posterior via minimizing the KL-divergence, making it as close to the true posterior as possible.
- $$KL(q_\theta(y)\,||\,p(y\,|\,x)) = E_q[\log \frac{q_\theta(y)}{p(y\,|\,x)}]$$

$$E_q[\log \frac{q_\theta(y)}{p(y\,|\,x)}] = E_q[\log q_\theta(y)] - E_q[\log \frac{p(x\,|\,y)p(y)}{p(x)}]$$

$$\log p(x) - KL(q_\theta(y)\,||\,p(y\,|\,x)) = E_q[\log p(x\,|\,y)] + E_q[\log \frac{p(y)}{q_\theta(y)}]$$

Linear Regression

Demo

GW Parameter Estimation

- This is no difference to the linear regression
- Linear Regression: $y = ax + b + \epsilon(x)$
- GW Parameter Estimation: $y = f(x) + \epsilon(x)$, $y = f(x, \epsilon(x))$
- Assume we have a parameterized model $\hat{f}_\tau(x)$:

$$max_{\theta,w} p(y \mid x; \theta, w)$$

$$p(y \mid x; \theta, w) = \int_\tau {\color{red} p(y \mid \tau, x; \theta)} {\color{blue} p_w(\tau \mid x)} d\tau$$

https://github.com/skyve2012/BayesianNetTutorial

GW Parameter Estimation

Demo

How to Write TF Code

- A well-constructed deep learning model: Inputs, outputs, loss, and optimizer.
- Dataset Feeder: pass data from a dataset object to model during training
- Training, Testing, Prediction Wrappers: Handle the training, testing and prediction process, respectively

How to Write TF Code

- Model:

```
def gen_model():
    '''
    initialze input and output variabes,
    return the model wrapper that takes inputs and outputs as its input parameters

    '''
    inputs = tf.keras.layers.Input(shape=(BATCH_SIZE, 8192), dtype=tf.float32)
    x = inputs

    # get std
    x_branch = tf.keras.layers.Dense(units=512, activation=tf.nn.relu)(x)
    std_x = tf.keras.layers.Dense(units=1, activation=tf.nn.relu)(x_branch)

    # get mean estimation
    x = tfp.layers.DenseFlipout(512,  activation=None)(x)
    x = tfp.layers.DenseFlipout(1,  activation=tf.nn.relu)(x)

    model = tf.keras.models.Model(inputs=inputs, outputs=[x, std_x])

    return model
```

https://github.com/skyve2012/BayesianNetTutorial

How to Write TF Code

- Dataset:

```python
def generator(filename, shuffle=True, batch_size = 32, ...):

    '''
    filename: dataset file name
    shuffle: if shuffle the dataset
    batch_size: batch size for training with stochastic approaches
    ...


    return: an iterable object
    '''


    ...

    return iterable object
```

https://github.com/skyve2012/BayesianNetTutorial

How to Write TF Code

- Training, Testing, Prediction Wrappers

```
def ModelFunc(features, labels, mode):
    '''
    features: [batch size, 8192]
    labels: [batch size, label dim] # label dim = 1 for just single parameter
    mode: train, test, predictions, controlled by the Estimator object in TensorFlow

    return proper wrappers (training, testing and prediciton)
    '''
    x, y = features, labels
    model = gen_model()
    out_mean, out_std = model(x)

    final_distribution = tfd.Normal(loc=out_mean, scale=out_std + 1e-3)
    final_outputs = final_distribution.sample(SAMPLE_PER_PASS)
    ############
    #predictions
    ############
    predictions = {'predictions': tf.transpose(final_outputs, [1, 0, 2])}
    if mode == tf.estimator.ModeKeys.PREDICT:
        return tf.estimator.EstimatorSpec(mode=mode, predictions=predictions)
    ############
    #training
    ############
    if mode == tf.estimator.ModeKeys.TRAIN:
        optimizer = tf.compat.v1.train.AdamOptimizer(learning_rate = 0.0005, beta1=0.9, beta2=0.999,epsilon=1e-08)
        train_op = optimizer.minimize(loss=loss)
        global_step = tf.compat.v1.train.get_global_step()
        update_global_step = tf.compat.v1.assign(global_step, global_step + 1, name = 'update_global_step')

        return tf.estimator.EstimatorSpec(mode=mode,
                                          loss=loss,
                                          train_op=tf.group(train_op, update_global_step))
    ############
    #evaluation
    ############
    eval_metric_ops = {
            'relative_error_test': tf.compat.v1.metrics.mean_relative_error(y, final_outputs, y),
            'mse_loss_test': tf.compat.v1.metrics.mean_squared_error(y, final_outputs)}
    return tf.estimator.EstimatorSpec(mode=mode, loss=loss, eval_metric_ops=eval_metric_ops)
```

How to Write TF Code

Demo