Scientific Machine Learning

Recap Lecture

Table of Content

Linear Models

- Supervised Learning approaches
  - Requires labeled data
- Examples of it are
  - Linear Regression
  - Logistic Regression

$$\left\{x^{(i)}, y^{(i)}\right\}_{i=1,...,m'}$$

Modus Operandi of Linear Regression

1. Formulate a hypothesis for the relation between input x , and output y
2. Look for a strategy to fit the hypothesis parameters, such as gradient descent

$$h(x) = \theta^\top x$$
$$\theta = (X^\top X)^{-1} X^\top Y$$

Probabilistic Interpretation

- Modeling all data points as independent, identically distributed (iid) random variables

$$y^{(i)} = \theta^\top x^{(i)} + \varepsilon^{(i)}$$

- This then allows us to use maximum likelihood estimation.
  1. Construct the log-likelihood $l(\theta)$
  2. Maximize the log-likelihood $\theta = \operatorname{argmax} l(\theta)$

Classification: Logistic Regression

- Apply the sigmoid function on top of the linear regression, and use a different cost function

$$h(x) = \varphi(\theta^\top x) = \frac{1}{1 + e^{-\theta^\top x}}$$

- To apply the maximum likelihood approach, one then needs to construct discrete probability events

$$p(y|x;\theta) = h^y(x)(1 - h(x))^{1-y}$$

Exponential Family

- Consists of Bernoulli, Gaussian, Dirichlet, Gamma, Poisson, and Beta-distribution

$$p(x|\eta) = h(x) \exp\left(\eta^\top t(x) - a(\eta)\right)$$

  - Natural parameter η
  - Sufficient statistics $t(x)$
  - Probability support $h(x)$
  - Log normalizer $a(\eta)$ s.t. probability mass sums to 1

Bayesian Inference: Core Principles

- Uncertainty over parameters
- Probability over the parameter expresses a degree of belief in the result
- Inference over parameters
- Prior knowledge and newly observed data are being combined according to **Bayes Theorem**

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}$$

Bayesian Inference

- Seek to arrive at the **posterior distribution**
- Computationally very expensive
  - Unscaled posterior is easy to find, scaled posterior very expensive to find
  - Requires expensive inference approaches such as Markov-Chain Monte Carlo

Bayesian Inference: Monte Carlo Sampling

- Draws from the posterior to find new samples
- Guaranteed to converged if given infinite computation (Monte Carlo theorem)
- In its simpplest form

$$\mathbb{E}(h(x)) = \int h(x)p(x)dx$$

Bayesian Inference: Advantages

- Gives us an approximation which can be made as precise as desired through the regulation of the computational budget
- Have probability/certainty estimates over our parameters
  - Always need to consider whether our distributions are converged yet
- Can analyze uncertainties/sensitivities of our model

Sampling Approaches

- Acceptance-Rejection sampling: $M \times g_0(\theta) \leq g(\theta)f(y|\theta)$
- Sampling-importance resampling then extends upon this by introducing a further step in which one resamples from the space of parameters
- Adaptive rejection sampling as the last alternative in case the other two approaches failed

Bayesian Inference: Summary

- Unscaled posterior $g(\theta|y) \propto g(\theta)f(y|\theta)$ contains shape information of posterior
- True posterior obtained by normalizing the unscaled posterior which requires the evaluation of an expensive high-dimensional integral
- Integral usually evaluated numerically using Monte-Carlo approaches

Bayesian Machine Learning

1. Introduces probability distribution over the parameters
2. Perform inference to obtain the probability distribution over the parameters

- Loss of computational efficiency
- Gain of uncertainties, especially useful in the small data limit

Optimization

- Optimization seek to solve the minimization problem of the cost function

$$\mathrm{argmin}_{\theta} J(\theta)$$

- If our cost function is convex, then it is easier to optimize
  - Least mean squares loss for linear regression
  - Log-likelihood for logistic regression

Loss functions

- Regression loss
  - L1 loss: $\frac{1}{m}\sum_{i=1}^{m}|y - h_\theta(x)|$
  - MSE loss: $\frac{1}{m}\sum_{i=1}^{m}(y - h_\theta(x))^2$
- Classification loss
  - Cross-entropy loss

$$-\sum_{i=1}^{m}\sum_{k=1}^{K}(y_{i_k} \cdot \log h_{\theta,k}(x_i))$$

Gradient Methods

- Gradient descent in its purest form
- Stochastic gradient descent more commonly used
    - Adam
    - Stochastic Gradient Descent

Gradient Methods: Adam

- Adam utilizes the first, and second momentum of the gradient

$$v_t \leftarrow \beta_1 v_{t-1} + (1 - \beta_1) g_t$$
$$s_t \leftarrow \beta_2 s_{t-1} + (1 - \beta_2) g_t^2$$

- typical initialization: $\beta_1 = 0.9$, $\beta_2 = 0.999$

Gradient Methods: SGD

- Consider the gradient as an expectation, we construct an unbiased estimator of the gradient

$$x \leftarrow x - \eta \nabla f_i(x)$$

- Estimator then constructred as follows

$$\mathbb{E}_i \nabla f_i(x) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x)$$

- This can then be further enhanced by minibatching

Second-Order Methods

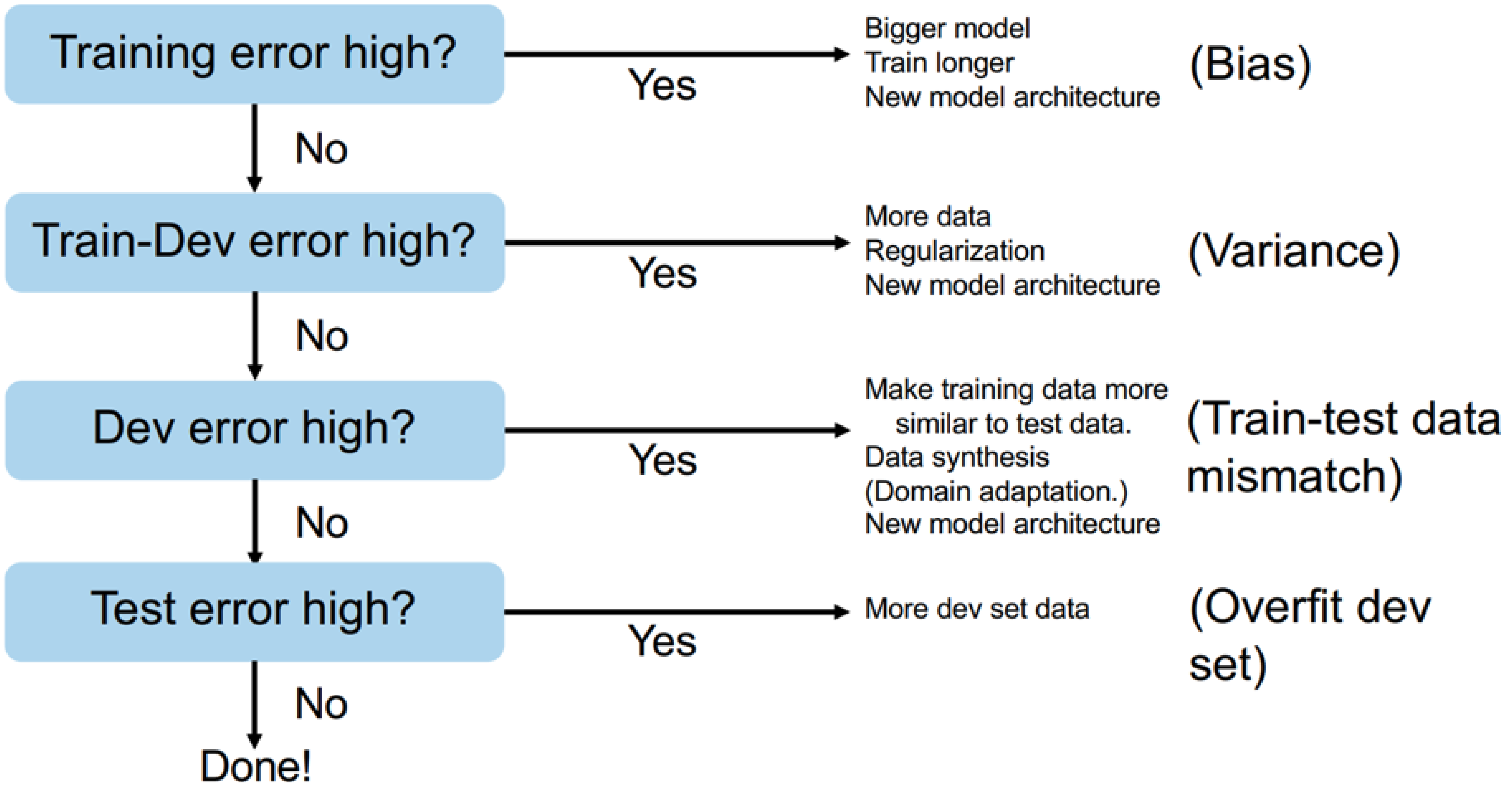
- Don't only take the gradient into account, but also the gradient curvature
- Examples of second order methods include, but are not limited to:
    - Newton's method
    - Quasi-Newton Approach
    - BFGS
    - L-BFGS

Blackbox or Derivative Free Optimization

- Mostly when one cannot compute the gradient in an acceptable timeframe, and can be broken down into the following decision tree
  - Expensive Function
    - Bayesian Optimization
  - Cheap function
    - Stochastic local search
    - Evolutionary search

Tricks of Optimization

- Tricks of Optimization we saw in the lecture are:
  - Data Splitting
  - Regularization
  - Input Normalization, and Parameter Initialization
  - Hyperparameter Search
  - Learning Rate Scheduling


Training error high?
Yes
Bigger model
Train longer
New model architecture
(Bias)
No
Train-Dev error high?
Yes
More data
Regularization
New model architecture
(Variance)
No
Dev error high?
Yes
Make training data more similar to test data.
Data synthesis
(Domain adaptation.)
New model architecture
(Train-test data mismatch)
No
Test error high?
Yes
More dev set data
(Overfit dev set)
No
Done!

Classical Machine Learning

- Support Vector Machines
- Gaussian Processes

“ jump to lecture script for a quick walk-through ”

Deep Learning

- Gradients (in the presentation)
- Multilayer Perceptron
- Convolutional Neural Networks
- Recurrent Models
- Encoder-Decoder Models

Gradients: The Why

- Writing manual derivatives is tedious, and error-prone
    - Intractable for large code-bases
- Seek to automatically generate the derivatives

Gradients: Forward-Mode Differentiation

- Differentiation performed in the direction of the computational flow with the gradients being accumulated as we go along.
- Does not require elaborate caching, and much less memory

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial x}\left(\frac{\partial c}{\partial b}\left(\frac{\partial b}{\partial a}\frac{\partial a}{\partial x}\right)\right)$$

Gradients: Reverse-Mode Differentiation

- Gradients accumulated in the opposite direction of the computational flow
    - Requires elaborate caching
    - Requires much more memory
    - Results in a much more difficult algorithm

$$\frac{\partial y}{\partial x} = \left(\left(\frac{\partial y}{\partial c} \frac{\partial c}{\partial b} \right) \frac{\partial b}{\partial a} \right) \frac{\partial a}{\partial x}$$

Gradient Forward vs Reverse

- For any function
  $$f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$$
- Forward-mode more efficient for gradients of scalar-to-vector functions $m >> n$
- Reverse-mode more efficient for gradients of vector-to-scalar function $m << n$ (most of machine learning)

Multilayer Perceptron

Jump to lecture notes