Bayesian Deep Learning and its application for survival analysis models

Elnura Zhalieva

Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI)

4.08.2022

Introduction

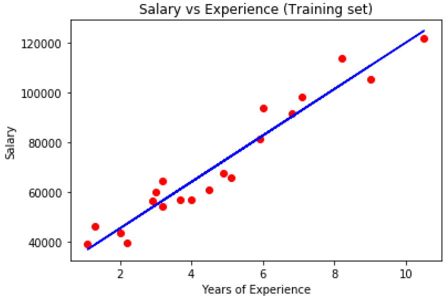


Figure 1: Basic Regression Problem

- There is a set of N target variables $y = [y(x_1), \dots, y(x_N)]$
- Set of N observations $X = [x_1, \dots, x_N]$
- The goal is to predict the value of $y(x_*)$ at a test point x_*

In other words, given the salary values y corresponding to years of experience X, what will be the salary of a person with years of experience $x_* = 15.5$?

Possible solution

1. Try to guess a parametric form of the function that fits the data
   - $f(x, w) = w^T x$
   - $f(x, w) = w^T \phi(x)$
   - $f(x, w) = \sigma(w^T \phi(x))$

   where $\phi(x)$ is a feature extractor.
   If $\phi(x) = [1, x]$, then $f(x, w) = w_0 + w_1 x$
   elif $\phi(x) = [1, x, x^2]$ then $f(x, w) = w_0 + w_1 x + w_2 x^2$.
2. Choose an error function and minimize w.r.t w

$$L(w) = \sum_{i=1}^{n} (f(x_i, w) - y(x_i))^2$$

More probabilistic approach

1. We could add noise term to our model $y(x) = f(x, w) + \epsilon(x)$ where $\epsilon(x) \sim \mathcal{N}(0, \sigma^2)$ and is i.i.d. From this it follows that

$$p(y(x)|x, w, \sigma^2) \sim \mathcal{N}(f(x, w), \sigma^2) \tag{1}$$

$$p(y|x, w, \sigma^2) = \prod_{i=1}^{n} \mathcal{N}(y(x_i); f(x, w), \sigma^2) \quad \text{Likelihood} \tag{2}$$

2. Then maximize the likelihood in (2) w.r.t w, σ^2

Note that with Gaussian additive noise, maximizing likelihood in (2) is the same as minimizing the squared error:

$$\log p(y|x, w, \sigma^2) \propto -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (f(x_i, w) - y(x_i))^2$$

Adding regularization

Since both approaches are prone to over-fitting, we could use the penalized log-likelihood

$$L(w) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(f(x_i, w) - y(x_i))^2 - \lambda w^T w \tag{3}$$

where the last term is complexity penalizer.

Equation (3) can be interpreted as maximizing a log posterior $\log p(w|y, X) = \log(y|w, X) + \log p(w)$ with a Gaussian prior $p(w)$

But this is not Bayesian yet

Bayesian Inference

Bayes Rule

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}, \quad \text{posterior} \propto \text{likelihood} \times \text{prior}$$

Sum Rule

$$p(x) = \sum_y p(x, y) = \int p(x, y)\, dy$$

Product Rule

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x)$$

Approximate Inference

In Bayesian inference we are interested to compute the Bayesian model average (BMA) below

$$p(y|x_*, \mathcal{D}) = \int p(y|x_*, w) p(w|\mathcal{D}) \, dw \tag{4}$$

Consider each setting of w as a different model. BMA is an average of infinitely many models weighted by their posterior prob.

Since integral in (4) is intractable to compute, it's often approximated by Monte Carlo sampling technique

$$p(y|x_*, \mathcal{D}) \approx \frac{1}{J} \sum_{j=1}^{J} p(y|x_*, w_j), \; w_j \sim q(w|\mathcal{D}) \tag{5}$$

where w_j are samples from approximate posterior $q(w|\mathcal{D})$

Approximate posterior

There are two well known methods to approximate the true posterior $p(w|\mathcal{D})$

- Deterministic methods: $p(w|\mathcal{D})$ is approximated by defining an auxiliary function $q(w|\mathcal{D}, \theta)$ with a simple distribution (Gaussian). Then the parameters θ of function $q(w|\mathcal{D}, \theta)$ are tweaked to make q as close as possible to p.

  E.g. Variational Inference, Laplace, etc.

- MCMC: Form a Markov chain of approximate samples from $p(w|\mathcal{D})$

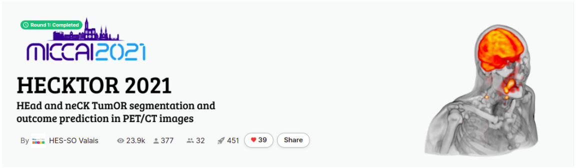
  E.g. Hamiltonian Monte Carlo (HMC), etc.

Motivation

Inspired by many exceptional works in Bayesian Deep Learning (BDL) such as

- *Averaging Weights Leads to Wider Optima and Better Generalization* Izmailov et. al
- *A Simple Baseline for Bayesian U* Maddox et. al, NeurIPS 2019
- *Bayesian Deep Learning and a Probabilistic Perspective of Generalization* Wilson et. al, NeurIPS 2020
- *Weight Uncertainty in Neural Networks* Blundell et. al, JMLR 2015

I was looking for a project to apply BDL.



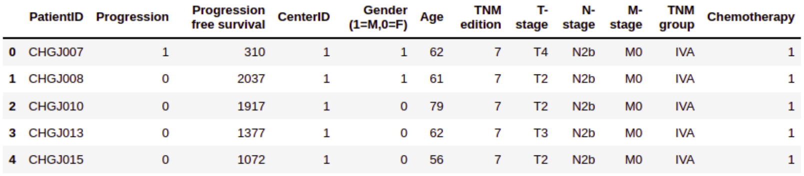
HECKTOR 2021

HEad and neCK TumOR segmentation and outcome prediction in PET/CT images

Figure 2: Hecktor challenge

Problem Description

- Task: predict progression-free survival, time from the start of the treatment till the progression of cancer
- Given: PET/CT images and electronic health records of 222 patients, 15 non-outcome features (5 were dropped)



	PatientID	Progression	Progression free survival	CenterID	Gender (1=M,0=F)	Age	TNM edition	T-stage	N-stage	M-stage	TNM group	Chemotherapy
0	CHGJ007	1	310	1	1	62	7	T4	N2b	M0	IVA	1
1	CHGJ008	0	2037	1	1	61	7	T2	N2b	M0	IVA	1
2	CHGJ010	0	1917	1	0	79	7	T2	N2b	M0	IVA	1
3	CHGJ013	0	1377	1	0	62	7	T3	N2b	M0	IVA	1
4	CHGJ015	0	1072	1	0	56	7	T2	N2b	M0	IVA	1

Figure 3: Hecktor dataset

Issues: data scarcity

Multi-task logistic regression

1. We start by dividing time axis into K intervals as in the figure

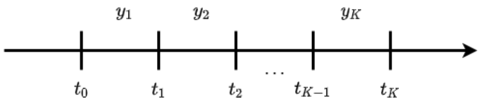


Figure 4: Divided time axis

2. Then we build a binary classification problem to predict the probability that the event occurs within each interval $[t_{k-1}, t_k)$, defined as

$$P_{\theta_k}(y_k = 1|x) = \frac{1}{1 + \exp(-(\theta_k^T x + b_k))}, \quad 1 \leq k \leq K$$

   where $y_k = 1$ if $T < t_k$. Also, each interval has a separate set of params (θ_k, b_k)

3. Then, taking into account only legal sequences of $y_1, y_2, \ldots, y_K$ and censored observations, Yu et. al proposed definitions for density, survival and likelihood functions (see the links below)

Learning Patient-Specific Cancer Survival Distributions as a Sequence of Dependent Regressors

Blog post explaining technical stuff

Being Bayesian

1. We treat traditional neural network as a probabilistic model, the goal of which is to learn posterior probs of weights given training data $p(w|D)$.
2. True posterior is approximated by variational posterior $q(w|\theta)$ chosen to follow Gaussian distr. parameters $\theta = (\mu, \sigma)$ are updated to minimize the KL divergence between $p(w|D)$ and $q(w|\theta)$

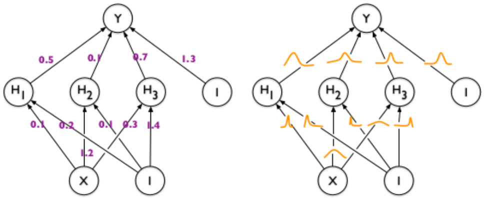


Figure 5: Left: each weight has a fixed value. Right: each weight is assigned a distribution. Link to the source

Expanding KL divergence

$$\mathrm{KL}[q(w|\theta)||p(w|D)] = \int q(w|\theta)\log\frac{q(w|\theta)}{p(w)p(D|w)}, \quad p(w|D) \propto p(w)p(D|w)$$
$$= \mathrm{KL}[q(w|\theta)||p(w)] - \mathbb{E}_{q(w|\theta)}[\log p(D|w)]$$

where $p(w)$ is the prior on weights.

The prior on weights is chosen to be a mixture of two Gaussian distr. with zero means but different variances

$$p(w) = \prod_j \pi\mathcal{N}(w_j|0,\sigma_1^2) + (1-\pi)\mathcal{N}(w_j|0,\sigma_2^2) \tag{6}$$

So the overall cost function to minimize is

$$L(\theta) = \mathrm{KL}[q(w|\theta)||p(w)] - \text{likelihood of neural MTLR} \tag{7}$$

Lab: implementation

We will implement Bayesian version of Neural MTLR. And we will see that uncertainty decreases if we increase the number of estimators, and vice versa

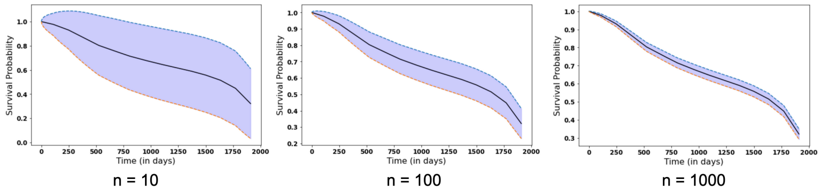


Figure 6: Survival probability in solid curve with 95% confidence interval