[Paper review 20]

Uncertainty in Deep Learning - Chapter 1

(Yarin Gal, 2016)

[Contents]

1. Introduction

The importance of Knowing What We Don't Know

Probabilistic view : offers confidence bounds

Knowing Uncertainty is the fundamental concern in Bayesian ML

but most DL models are deterministic functions

$\rightarrow$ we can get uncertainty information from existing DL models for free!

1.1 Deep Learning

x $\&$ y need not be linear. (non-linear function $f(x)$)

linear basis function regression

- non-linear transformations $\phi_k(x)$: our basis function
- compose a feature vector with $\phi_k(x)$

  $\Phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_K(\mathbf{x})]$

Parametrised basis function

- relax constraint
- define our basis function to be $\phi_k^{w_k, b_k}$
- $\langle \mathbf{w}_k, \mathbf{x} \rangle + b_k$

- example) $\phi_k(\cdot) = \sin(\cdot)$ then, $\phi_k^{\mathrm{w}_k, b_k}(\mathrm{x}) = \sin(\langle \mathrm{w}_k, \mathrm{x} \rangle + b_k)$

  $\mathrm{f}(\mathrm{x}) = \Phi^{\mathrm{W}_1,\, \mathrm{b}_1}(\mathrm{x})\mathrm{W}_2 + \mathrm{b}_2$, where $\Phi^{\mathbf{W}_1, \mathbf{b}_1}(\mathbf{x}) = \phi\left(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1\right), \mathbf{W}_1$

Feed-forward NN

(in case of single hidden layer)

$\widehat{\mathbf{y}} = \sigma\left(\mathbf{x}\mathbf{W}_1 + \mathbf{b}\right)\mathbf{W}_2$

Loss function

- Regression : Euclidean Loss

  $E^{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}}(\mathbf{X}, \mathbf{Y}) = \frac{1}{2N}\sum_{i=1}^{N} \left\| \mathbf{y}_i - \widehat{\mathbf{y}}_i \right\|^2$
- Classification : softmax loss

  $\hat{p}_d = \exp(\hat{y}_d) / \left(\sum_{d'} \exp(\hat{y}_{d'})\right)$

  $E^{\mathrm{W}_1,\, \mathrm{W}_2,\, \mathrm{b}}(\mathrm{X}, \mathrm{Y}) = -\frac{1}{N}\sum_{i=1}^{N} \log\left(\hat{p}_{i,d_i}\right)$

  where $d_i \in \{1, 2, \ldots, D\}$ is the observed class for input i.

Weight decay (λ_i)

- to prevent overfit
- $\mathcal{L}\left(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}\right) := E^{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}}(\mathbf{X}, \mathbf{Y}) + \lambda_1 \left\| \mathbf{W}_1 \right\|^2 + \lambda_2 \left\| \mathbf{W}_2 \right\|^2 + \lambda_3 \left\| \mathbf{b} \right\|^2$

CNN & RNN

CNN : 생략

RNN

- input sequence $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$ of length T
- hidden state $\mathbf{h}_t$ for time step t
- $\hat{\mathbf{y}} = \mathbf{f}_{\mathbf{y}}\left(\mathbf{h}_T\right) = \mathbf{h}_T\mathbf{W}_{\mathbf{y}} + \mathbf{b}_{\mathbf{y}}$

  where the hidden state is $\mathbf{h}_t = \mathbf{f}_{\mathbf{h}}\left(\mathbf{x}_t, \mathbf{h}_{t-1}\right) = \sigma\left(\mathbf{x}_t\mathbf{W}_{\mathbf{h}} + \mathbf{h}_{t-1}\mathbf{U}_{\mathbf{h}} + \mathbf{b}_{\mathbf{h}}\right)$

1.2 Model uncertainty

1) Out of Distribution

- the point lies outside of the data distribution

2) Aleatoric uncertainty

- noisy data

3) Model Uncertainty (=Epistemic uncertainty)

- uncertainty in "model parameters"
- structure uncertainty

1.3 Model uncertainty and AI safety

Need uncertainty for safety!

1.4 Applications of model uncertainty

Beside AI safety....

ex) to learn from "small amount" of data

- choosing what data to learn from $\rightarrow$ in Active Learning
- exploring an agent's environment efficiently $\rightarrow$ in RL

1.5 Model uncertainty in Deep Learning

probability vector in softmax IS NOT model confidence

Modern DL models : do not capture confidence

- (probability vector in softmax IS NOT model confidence)
- but still closely related to probabilistic models like GP

Bayesian Neural Network

- GP can be recovered in limit of infinitely many weights (Neal, 1995)
- For a finite number of weights,

  model uncertainty can still be obtained by placing "distribution over weights"
- Have been resurrected under different names with "variational techniques"

Models which gives us uncertainty..

- usually do not scale well to complex model & big data
- thus, require us to develop new models for which we already have well performing tools

  (we need practical techniques! such as SRT)

Stochastic Regularization Techniques (SRTs)

- for model regularization
- successful within DL

  (we can take almost any network trained with an SRT)
- adapt the model output "stochastically" as a way of model regularization
- ex) Dropout, Multiplicative Gaussian Noise, Drop Connect, ...

How does SRT work?

- predictive mean & predictive variance
- simulate a network with input x^*

  $\rightarrow$ random output
- repeat this many times

$$\mathbb{E}\left[\mathbf{y}^*\right] \approx \frac{1}{T}\sum_{t=1}^{T}\widehat{\mathbf{y}}_t^*\left(\mathbf{x}^*\right)$$

$$\operatorname{Var}[\mathbf{y}^*] \approx \tau^{-1}\mathbf{I}_D + \frac{1}{T}\sum_{t=1}^{T}\widehat{\mathbf{y}}_t^*(\mathbf{x}^*)^T\widehat{\mathbf{y}}_t^*\left(\mathbf{x}^*\right) - \mathbb{E}[\mathbf{y}^*]^T\mathbb{E}\left[\mathbf{y}^*\right]$$

(this is practical with large models and big data!)

1.6 Thesis Structure

The code for the experiments presented in this work is available at

https://github.com/yaringal