BAYESIAN DEEP LEARNING

Clarotto Lucia, Franchini Alessandro, Lamperti Letizia

November 22, 2019

Bayesian Statistics, Politecnico di Milano

INTRODUCTION

In this project, we aim at studying, coding and analysing two different ways of introducing **Bayesian uncertainty** in a **Deep Learning** process.

- **Deep Learning** works very well in practice for many tasks, BUT is unable to reason about uncertainty over the features.
- **Bayesian models** capture how much the model is confident in its estimation.

We will consider two different methods:

- BAYESIAN NEURAL NETWORKS
- MONTE-CARLO DROPOUT

NEURAL NETWORKS

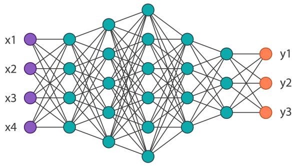
NEURAL NETWORKS

An Artificial Neural Network is a non-linear model characterized by

- number of neurons
- neurons' topology
- activation functions
- values of weights and biases

PROCESS:

- Neurons in the input layer receive the data to process;
- In the hidden layer data are processed by neurons at the same distance;
- Neurons in the output layer give the final result of the network.



NEURAL NETWORKS

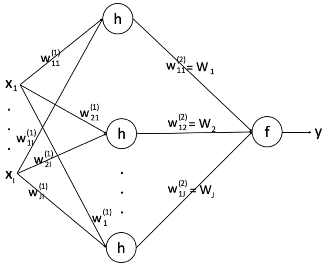
In an artificial neural network, information can flow along different paths, but always forward.

In multilayer perceptrons the output of the network is:

$$y = f(\sum_{j=0}^{J} W_j \cdot h(\sum_{i=0}^{I} w_{ij} \cdot x_i))$$

And it has to be close as possible to the target function $y_n \approx t_n$. Hence, we want to minimize the error function:

$$E = \sum_{n=0}^{N} (t_n - y_n)^2$$



GRADIENT DESCENT

To minimize the network error, a classical numerical approach is the gradient descent. It performs the following steps:

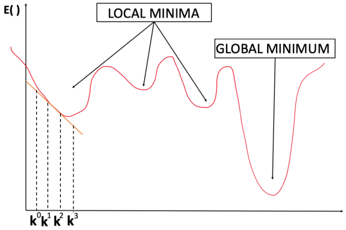
1. Pick up a possible solution $\mathbf{w_0}$ (at random)
2. Compute the function derivative $\left.\frac{\partial E}{\partial \mathbf{w}}\right|_k$
3. Update the solution

$$\mathbf{w^{k+1}} = \mathbf{w^k} - \eta \left.\frac{\partial E}{\partial \mathbf{w}}\right|_k$$

4. Repeat 2 and 3 until convergence.

Possible problems:

1. Local minima
2. Slow convergence
3. No convergence at all



BAYESIAN DEEP LEARNING

BAYESIAN APPROACH

First we assume a **Gaussian** conditional probability distribution for the output:

$$P(\mathbf{y}|\mathbf{x},\mathbf{w}) = (2\pi\sigma^2)^{-\frac{D}{2}} \exp\left(-\frac{|\mathbf{y} - f(\mathbf{x},\mathbf{w})|^2}{2\sigma^2}\right)$$

where $\mathbf{w}$ is the weight vector and σ assumed known.

The **predictive distribution** of the output is given by:

$$P(\mathbf{y}_{n+1}|\mathbf{x}_{n+1}(\mathbf{x}_1,\mathbf{y}_1),\ldots,(\mathbf{x}_n,\mathbf{y}_n)) = \int_{\mathbb{R}^N} P(\mathbf{y}_{n+1}|\mathbf{x}_{n+1},\mathbf{w}) \\ P(\mathbf{w}|(\mathbf{x}_1,\mathbf{y}_1),\ldots,(\mathbf{x}_n,\mathbf{y}_n))\, d\mathbf{w}$$

BAYESIAN APPROACH

The **posterior probability** for the weight vector is:

$$P(\mathbf{w}|(\mathbf{x}_1,\mathbf{y}_1),\dots,(\mathbf{x}_n,\mathbf{y}_n)) = \frac{P(\mathbf{w})\prod_{i=1}^{n} P(\mathbf{y}_i|\mathbf{x}_i,\mathbf{w})}{P(\mathbf{y}_1,\dots,\mathbf{y}_n|\mathbf{x}_1,\dots,\mathbf{x}_n)}$$

where the **prior distribution** $P(\mathbf{w})$ for the network weights is:

$$P(\mathbf{w}) = (2\pi\omega)^{-\frac{N}{2}} \exp\left(-\frac{|\mathbf{w}|^2}{2\omega^2}\right)$$

The expected scale of the weights is given by ω.

GOAL: calculate numerically the **predictive distribution**.

HAMILTONIAN MONTE CARLO

We generate the Markov chain in terms of an Hamiltonian function:

$$H(\mathbf{w},\mathbf{p}) = E(\mathbf{w}) + \frac{1}{2}|\mathbf{p}|^2$$

where:

$$\begin{aligned} E(\mathbf{w}) &= -\log(P(\mathbf{w}|(\mathbf{x}_1,\mathbf{y}_1),\cdots,(\mathbf{x}_n,\mathbf{y}_n))) - \log(Z_E) = \\ &= \frac{|\mathbf{w}|^2}{2\omega^2} + \sum_{i=1}^{n} \frac{|\mathbf{y}_i - f(\mathbf{x}_i,\mathbf{w})|^2}{2\sigma^2} \end{aligned}$$

A candidate state $(\widetilde{\mathbf{w}}_{t+1},\widetilde{\mathbf{p}}_{t+1})$ is accepted if, after the dynamic moves:

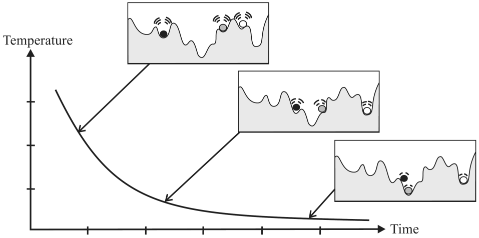
$$(\mathbf{w}_{t+1},\mathbf{p}_{t+1}) = \begin{cases} (\widetilde{\mathbf{w}}_{t+1},\widetilde{\mathbf{p}}_{t+1}) & \text{if } U < \exp(-\Delta H) \\ (\mathbf{w}_t,\mathbf{p}_t) & \text{otherwise} \end{cases}$$

where U is the **Uniform Distribution** in $(0,1)$.

SIMULATED ANNEALING

The posterior has lot of local minima $\longrightarrow$ **Simulated annealing**

- **Temperature** parameter;
- Acceptance probability: $\exp(-\Delta H/T)$;
- **Cooling process**.



DROP-OUT

DROPOUT

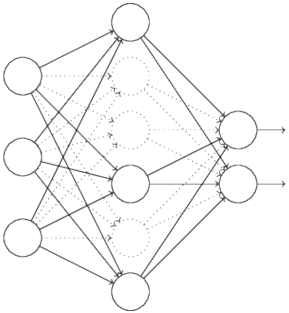
Goal: show that
Neural Network with dropout
$$\Updownarrow$$
approximation of a Bayesian model (Deep Gaussian Process)

Dropout is an approach to reduce interdependent learning amongst the neurons, which can causes overfitting.

$\mathbf{W_i}$ matrix of layer i weights

$$\begin{cases} \mathbf{W_i} = \mathbf{M_i} \cdot \text{diag}([\mathbf{z_{i,j}}])_{j=1}^{K_i} \\ [\mathbf{z_{i,j}}] \sim Be(p_i), \text{ for } i = 1, ..., L, \\ \qquad\qquad\qquad\qquad j = 1, ..., K_{i-1} \end{cases}$$

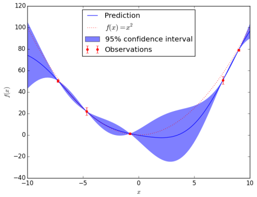
$[\mathbf{z_{i,j}}] = 0$ if unit j in layer $i-1$ is dropped out



GAUSSIAN PROCESS

Gaussian Process (GP) is a probability distribution over functions. It offers:

- **Uncertainty estimates** over the function values;
- **Robustness to over-fitting**.



Bayesian approach to find the regression function $y = f(x)$ of a NN:

- Put some prior distribution over the space of functions $p(\mathbf{f})$;
- Look for the posterior distribution over the space of functions given the dataset $(\mathbf{X}, \mathbf{Y})$;

$$p(\mathbf{f}|\mathbf{X}, \mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X}, \mathbf{f})p(\mathbf{f})$$

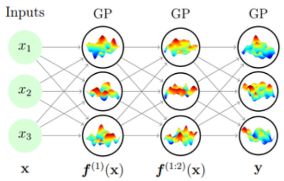
- Place a joint Gaussian distribution over all function values;

$$\mathbf{F}|\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{X}, \mathbf{X}))$$

$$\mathbf{Y}|\mathbf{F} \sim \mathcal{N}(\mathbf{F}, \tau^{-1}\mathbf{I}_N)$$

DEEP GAUSSIAN PROCESS

A **Deep Gaussian Process** is a model where an initial **input** variable is mapped to an **output** variable through a cascade of **hidden layers**.


Inputs
GP
GP
GP
x_1
x_2
x_3
$\mathbf{x}$
$\boldsymbol{f}^{(1)}(\mathbf{x})$
$\boldsymbol{f}^{(1:2)}(\mathbf{x})$
$\mathbf{y}$

The transformation between layers is probabilistic and modelled with GPs.

The **predictive probability** of the deep GP model is

$$p(\mathbf{y}|\mathbf{x}, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}|\mathbf{x}, \omega) p(\omega|\mathbf{X}, \mathbf{Y})\, d\omega$$

with $\omega = \{\mathbf{W}_i\}_{i=1}^{L}$.

Intractable **posterior** $p(\omega|\mathbf{X}, \mathbf{Y}) \rightarrow$ **Variational posterior** $q(\omega)$

DEEP GAUSSIAN PROCESS APPROXIMATION

The **approximate posterior** $q(\omega)$ is proven to be equal to the **Dropout** objective under the KL divergence minimization.

Approximate Predictive distribution: $q(\mathbf{y}^*|\mathbf{x}^*) = \int p(\mathbf{y}^*|\mathbf{x}^*, \omega)\, q(\omega)\, d\omega$

Approximation by Monte-Carlo integration $\rightarrow$ **MC DROPOUT**

$$p(\mathbf{y}|\mathbf{x}, \mathbf{X}, \mathbf{Y}) \approx \int p(\mathbf{y}^*|\mathbf{x}^*, \omega)\, q(\omega)\, d\omega \approx \sum_{t=1}^{T} p(y^*|\mathbf{x}^*, \omega_{\mathbf{t}}),\ \omega_{\mathbf{t}} \sim q(\omega)$$

<u>CONCLUSION</u>:

- A NN with **dropout** with arbitrary non-linearities becomes a Bayesian approximation of a **Deep Gaussian Process**
- We can model the **uncertainty** of the NN through MC estimates of mean and variance from the approximate predictive distribution

CODING AND ANALYSIS OF RESULTS

CODING AND ANALYSIS OF RESULTS

- **Coding** of a **Bayesian NN** (Pytorch)
  - Training with Hamiltonian Monte Carlo (Pystan) w/ or w/o simulated annealing
  - Testing
- **Coding** of a trained **NN with dropout** (Pytorch)
  - Standard dropout
  - Gaussian Process with SE covariance function
  - MC Dropout with ReLU or TanH non-linearities
- **Comparison** between the two Bayesian methods
  - Analysis of the degree of fitness of the models
  - Analysis of the **uncertainty** of predictions
  - Analysis of the computational cost

Bibliography

- Radford M. Neal, **Bayesian Training of Backpropagation Networks by the Hybrid Monte Carlo Method**, 10 April 1992
- Yarin Ga, Zoubin Ghahramani, **Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning**, University of Cambridge, 4 October 2016
- Andreas C. Damianou, Neil D. Lawrence, **Deep Gaussian Processes**, Dept. of Computer Science & Sheffield Institute for Translational Neuroscience - University of Sheffield, UK
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, **Dropout: A Simple Way to Prevent Neural Networks from Overfitting**, Dept. of Computer Science - University of Toronto, 14 June 2014