Dropout as Bayesian Approximation
Uncertainty in Deep Learning Models

Brandon Walker

Charleston Data Analytics Meetup

June 13, 2018
Introduction
Deep Learning Models
Deep Learning Models

\[ \mathbf{x} \rightarrow z^{(k)} \rightarrow z^{(k-1)} \rightarrow \cdots \rightarrow z^{(0)} \rightarrow y \]

Where at each step we say

\[ z^{(k)} = \sigma(W\mathbf{x} + b) \]

This is also known as *Adaptive Basis Regression* and the goal is to approximate some function \( f^* \).
Overfitting

Toy regression data set

- Degree 300 polynomial model
- Training data
- Testing data
Overfitting

Toy regression data set

- Degree 300 polynomial model
- Training data
- Testing data
**Definition**

*Regularization* is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.[2]

- $L_1/L_2$ Norm Penalties
- Early Stopping
- Data Augmentation
Dropout

Base network

Ensemble of subnetworks

(Goodfellow [1])
# Create NN
model = Sequential()

# Input Layer
model.add(Dense(20,
    input_dim=x_train.shape[1],
    kernel_initializer='normal',
    activation='relu'))

model.add(Dropout(0.5))

# Hidden Layer
model.add(Dense(20,
    input_dim=x_train.shape[1],
    kernel_initializer='normal',
    activation='relu'))

# Output Layer
model.add(Dense(1))

# Compile the network
model.compile(loss='mean_squared_error',
    optimizer='adam')
Prediction

```python
y = y_test[30:31][0]
y_hat = np.squeeze(model.predict(x_test[30:31]))
print('Predicted Value: \t%.3f' % y_hat)
```

Predicted Value: 16.352

How certain are we of this prediction?
Uncertainty

This model tells us nothing about how certain we are about our prediction!
Bayesian Methods address the problem of uncertainty, but have drawbacks...
Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning

Yarin Gal
Zoubin Ghahramani
University of Cambridge

Abstract

Deep learning tools have gained tremendous attention in applied machine learning. However, such tools for regression and classification do not capture model uncertainty. In comparison, Bayesian models offer a mathematically grounded framework to reason about model uncertainty, but usually come with a prohibitive computational cost. In this paper we develop a new theoretical framework casting dropout training in deep neural networks (NNs) as approximate Bayesian inference in deep Gaussian processes. A direct result of this theory gives us tools to model uncertainty with dropout NNs — extracting information from existing models that has been thrown away so far. This mitigates the problem of representing uncertainty in deep learning without sacrificing either computational complexity or test accuracy. We perform an extensive study of the properties of dropout’s uncertainty. Various network architectures and nonlinearities are assessed on tasks of regression and classification, using MNIST as an example. We show a considerable improvement in predictive log-likelihood and RMSE compared to existing state-of-the-art methods, and finish by using dropout’s uncertainty in deep reinforcement learning.

1. Introduction

With the recent shift in many of these fields towards the use of Bayesian uncertainty (Herzog & Ostwald, 2013; Trafton & Marks, 2015; Nuzzo, 2014), new needs arise from deep learning tools.

Standard deep learning tools for regression and classification do not capture model uncertainty. In classification, predictive probabilities obtained at the end of the pipeline (the softmax output) are often erroneously interpreted as model confidence. A model can be uncertain in its predictions even with a high softmax output (fig. 1). Passing a point estimate of a function (solid line 1a) through a softmax (solid line 1b) results in extrapolations with unjustified high confidence for points far from the training data. z* for example would be classified as class 1 with probability 1. However, passing the distribution (shaded area 1a) through a softmax (shaded area 1b) better reflects classification uncertainty far from the training data.

Model uncertainty is indispensable for the deep learning practitioner as well. With model confidence at hand we can treat uncertain inputs and special cases explicitly. For example, in the case of classification, a model might return a result with high uncertainty. In this case we might decide to pass the input to a human for classification. This can happen in a post office, sorting letters according to their zip code, or in a nuclear power plant with a system responsible for critical infrastructure (Linda et al., 2009). Uncertainty is important in reinforcement learning (RL) as well (Szepesvári, 2010). With uncertainty information an agent can decide when to exploit and when to explore its environment. Recent advances in RL have made use of NNs for
```python
f = K.function([model.layers[0].input, K.learning_phase()],[model.layers[-1].output])

def predict_with_uncertainty(f, x, no_classes, n_iter=100):
    result = np.zeros((n_iter,) + (x.shape[0], no_classes))

    for i in range(n_iter):
        result[i, :, :] = f((x, 1))[0]

    prediction = result.mean(axis=0)
    uncertainty = result.std(axis=0)

    return prediction, uncertainty, result
```
Dropout as Bayesian Approximation

```python
y_hat_do, se, results = predict_with_uncertainty(f, x_test[30:31], 1)
results = np.squeeze(results)
print('Predicted Value: \t%.3f' % np.squeeze(y_hat_do))
print('Standard Error: \t%.3f' % np.squeeze(se))
```

Predicted Value: 20.324
Standard Error: 1.575
Dropout as Bayesian Approximation

Bayesian Approximation

Predicted Housing Cost

Count

0.0  25  50  75  100  125  150  175
16  17  18  19  20  21  22  23
Dropout as Bayesian Approximation
Thank you

**Twitter**
@bsethwalker

**Email**
bsw462@g.harvard.edu
References
