Deep Learning Lecture 1 - Model inference in Keras

MA8701 General Statistical Methods

Thiago G. Martins, Department of Mathematical Sciences, NTNU

Spring 2019

Data pre-processing

Data normalization

- Features should be homogeneous and standardized.

```
# MNIST dataset - image data
train_images <- array_reshape(train_images, c(60000, 28 * 28))
train_images <- train_images / 255
```

```
# House price dataset - numerical features
mean <- apply(train_data, 2, mean)
std <- apply(train_data, 2, sd)
train_data <- scale(train_data, center = mean, scale = std)
```

Vectorize labels

To vectorize the labels, there are two possibilities:

- We can use one-hot encoding:

```r
to_one_hot <- function(labels, dimension) {
  results <- matrix(0, nrow = length(labels), ncol = dimension)
  for (i in 1:length(labels))
    results[i, labels[[i]] + 1] <- 1
  results
}

one_hot_train_labels <- to_one_hot(train_labels)
one_hot_test_labels <- to_one_hot(test_labels)
```

Note that there is a built-in way to do this in Keras:

```r
train_labels <- to_categorical(train_labels)
```

- We can cast the label list as an integer tensor:

The only change is that we would have to use `sparse_categorical_crossentropy` loss function instead of the `categorical_crossentropy` loss function used for one-hot vector labels.

Handling missing values

Input missing values as number zero.

- In general safe with neural networks, assuming 0 is not already a meaningful value.

Artificially generating training samples with missing entries

- (Chollet and Allaire 2018) recommends to artificially generate training samples with missing entries
    - In case you expect missing values on the test set without having them on the training set
    - **This is not statistically sound. Duplicating data would artificially reduce uncertainty.**

Feature engeneering

- Neural networks are capable of automatically extracting useful features from raw data.

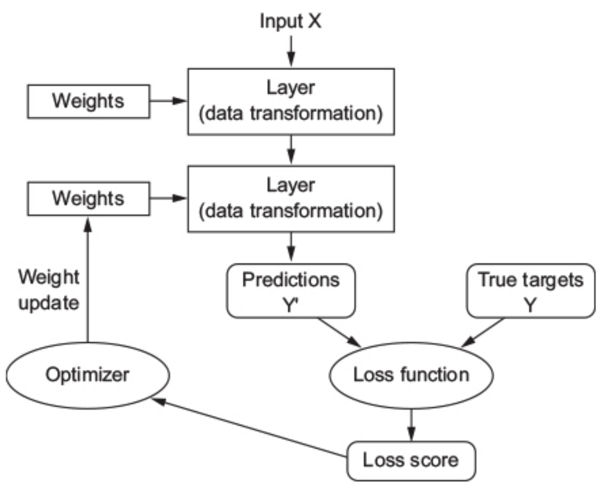
- Does this mean we do not have to worry about feature engineering as long as you're using deep neural networks?

  No, for two reasons:

    - Good features still allow you to solve problems more elegantly while using fewer resources.
    - Good features let you solve a problem with far less data.

Training

We have now defined our model, configured the training scheme by specifying a loss function, an optimizer and metrics for evaluation. We can now train and validate our model.



Fit and evaluate

Training the model

```
model %>% fit(train_data, train_labels, epochs = 5, batch_size = 128)
```

- `train_data` : Input tensor with the training features.
- `train_labels` : Input tensor with the training labels.
- `epochs` : Number of passes through the data.
- `batch_size` : Number of samples to use for each iteration of mini-batch SGD.

Training and validating a model

```
# We can later plot and manipulate data in 'history'
history <- model %>%
  fit(partial_train_data, partial_label_data,
      epochs = 20, batch_size = 512,
      validation_data = list(validation_train_data, validation_label_data))
```

- `validation_data` : Data on which to evaluate the loss and any model metrics at the end of each epoch.

Different way to train and validate a model

```
history <- model %>%
  fit(partial_train_data, partial_train_labels,
      epochs = num_epochs, batch_size = 1) %>%
  evaluate(validation_train_data, validation_label_data)
```

Information leak

- Be mindful of the following: every time you use feedback from your validation process to tune your model, you leak information about the validation process into the model.

- This makes the evaluation process less reliable, if done systematically over many iterations.

Predicting

```
model %>% predict(test_data)
```

- `test_data` : Tensor with features to use for prediction.

Reference material

This lecture note is based on (Chollet and Allaire 2018).

References

Chollet, F., and J. Allaire. 2018. *Deep Learning with R*. Manning Publications. https://books.google.no/books?id=xnIRtAEACAAJ.