Questions and Answers - Machine Learning and Deep Learning

Created by Aldo Zaimi

November 29, 2019

1. **What kind of performance metrics should be used when facing class imbalance issues in a classification problem?**

   Standard metrics such as accuracy can be biased when working with imbalanced datasets. For instance, in a test dataset containing 95 positive samples and 5 negative samples, the accuracy will still be 95% even if the classifier identifies all samples as positive.

   The following metrics can be used instead (depending on the task/dataset):

   - True positive rate (also called recall or sensitivity): proportion of actual positive samples that are correctly identified as such.
   - True negative rate (specificity): proportion of actual negative samples that are correctly identified as such.
   - Precision (positive predictive value): proportion of true positives over all positive calls.
   - False positive rate. proportion of samples that are incorrectly classified as positives.
   - F1 score: the harmonic mean between precision and sensitivity.

2. **Consider a dataset of** 10000 **samples split in training/dev using a ratio of** 70/30%**. How many gradient descent iterations will be necessary to train a model for** 10 **epochs, using a mini-batch size of** 64**?**

   - The training dataset consists of 70%: 7000 samples.
   - Each iteration will go through 64 samples from the training dataset.
   - One epoch (to go through all the training dataset) will require 110 iterations (109 iterations of 64 samples each and a last iteration of only 24 samples).
   - After 10 epochs, the algorithm will have done 1100 iterations.

3. **You are doing semantic segmentation of cells on medical images (2 classes, loss function is standard cross entropy). You are facing two issues: (i) the samples present an imbalanced pixel distribution (i.e. underrepresented cell pixels compared to the background), (ii) difficulty to correctly segment cell contours (borders). Propose two ways to modify the standard cross entropy loss in order to resolve these issues.**

- (1) Class imbalance issue: use a weighted cross entropy loss instead. The classification of cell pixels should be more weighted in the loss function. To determine an adequate weight factor, you can analyze the ratio of cell pixels to background pixels in the training dataset.
- (2) Cell borders segmentation issue: use a distance based cross entropy loss that will put more importance on incorrectly classified border pixels (for instance, use a Gaussian distance function that gives more weight to border pixels).

4. **Name the common hyperparameters that may need tuning/optimization in a neural network.**

   - Standard neural networks: learning rate, number of hidden layers, number of hidden units, weight initialization scheme used, activation functions used in each hidden layer, number of epochs, mini-batch size.
   - Convolutional neural networks: kernel (filter) size, number of filters/features per convolutional layer (i.e. convolutional layer depth), pooling layer type, convolution stride.
   - Regularization: L2 and/or L1 regularization parameters, learning rate decay, momentum, dropout value, batch normalization.

5. **You want to design a standard fully connected network to classify weather data points of 5 input features into 3 classes (sunny, cloudy, rainy). How many trainable parameters will your architecture have if you decide to use 2 hidden layers of 8 and 4 units respectively?**

   Number of input layer units: 5.
   Number of output layer units: 3 (3-class classification task).

   - Number of weights between input layer and first hidden layer: $5 \times 8 = 40$.
   - Number of biases in the first hidden layer: 8.
   - Number of weights between first hidden layer and second hidden layer: $8 \times 4 = 32$.
   - Number of biases in the second hidden layer: 4.
   - Number of weights between second hidden layer and output layer: $4 \times 3 = 12$.
   - Number of biases in the output layer: 3.

   Total: 99 trainable parameters.

6. **Determine the number of trainable parameters of the following network architecture: (i) input layer that takes grayscale digit images of size 16×16, (ii) conv. layer 1 with 3×3 filters and 4 feature maps, (iii) max pooling layer 1 of size 2×2, (iv) conv. layer 2 with 3×3 filters and 8 feature maps, (v) flattening before the fully connected layer, (vi) fully connected layer of 8 units, (vii) output layer of 2 units (binary classification problem), (viii) no padding used and all convolutions use a stride of 1.**

   The total number of trainable parameters in the network is the sum of the trainable parameters of all the layers. The input layers, the pooling layers and the dropout layers have no trainable

parameters. For the convolutional layers, the number of trainable parameters can be computed by $(n \times m \times l + 1) \times k$, where $(n \times m)$ is the filter size, l is the number of input feature maps and k is the number of output feature maps. The *+1* term in the equation takes into account the bias terms. For the fully connected layers, the number of trainable parameters can be computed by $(n + 1) \times m$, where n is the number of input units and m is the number of output units. The *+1* term in the equation takes into account the bias terms.

Input shape: 16×16.
Shape after first convolutional layer: $14 \times 14 \times 4$.
Shape after pooling layer: $7 \times 7 \times 4$.
Shape after second convolutional layer: $5 \times 5 \times 8$.
Dense layer input (from the output of the second conv. layer): $5 \times 5 \times 8 = 200$ units.

- Conv. layer 1: $(3 \times 3 \times 1 + 1) \times 4 = 40$ trainable parameters.
- Conv. layer 2: $(3 \times 3 \times 4 + 1) \times 8 = 296$ trainable parameters.
- Dense layer: $(200 + 1) \times 8 = 1608$ trainable parameters.
- Output layer: $(8 + 1) \times 1 = 9$ trainable parameters.

The network has a total of $40 + 296 + 1608 + 9 = 1953$ trainable parameters.

7. **In what context would input data normalization be necessary when doing multivariate linear regression?**

   Input data normalization consists of subtracting the mean and dividing by the standard deviation for each input feature. It is especially useful when the input features have very different ranges. It helps speed up the learning process (i.e. faster convergence).

8. **You design a CNN for multiclass image classification that has 5 convolutional layers (and the corresponding max pooling layers). (a) How can you detect if your model is overfitting during training? (b) What kind of activation function would be appropriate for the output layer? (c) How can you modify your architecture to avoid or reduce overfitting?**

   - (a) You need to monitor the training error/accuracy and the dev error/accuracy during the training procedure. Overfitting typically happens when the training accuracy improves while the dev accuracy deteriorates.
   - (b) Softmax activation function.
   - (c) You can (i) reduce model complexity (by reducing the number of layers and/or reducing the size of the filters), (ii) use data augmentation, (iii) add more data, (iv) add regularization techniques such as dropout and L1/L2 regularization, (v) use early stopping.

9. **What are the advantages of using mini-batch gradient descent instead of stochastic gradient descent or batch gradient descent?**

Standard batch gradient descent takes too much time per iteration (going through all the training dataset before updating the weights), while stochastic gradient descent is very noisy, has high chances of getting stuck in local minima, is very slow to converge and does not make use of vectorized implementation of CPUs/GPUs. Mini-batch gradient descent is a tradeoff between the two.

10. **What are some advantages of starting with a baseline design/model before going for more complex architectures?**

- Figuring out what is the right type of architecture for your task/dataset.
- Making sure that a simple model works before experimenting more complex architectures or adding/tuning more hyperparameters (helpful for debugging).
- Using it as reference baseline model to determine if future enhanced models implemented perform better.

11. **How can transfer learning help during the training procedure?**

Transfer learning can help obtain better performance and faster convergence. Concretely, transfer learning has been shown to result in (i) higher start (the learning curve has a better starting performance), (ii) higher slope (the learning curve has a larger improvement rate) and (iii) higher asymptote (the performance after convergence is higher).

12. **What are the main differences between L1 and L2 regularization?**

Both try to regularize the loss function by adding penalty terms. For L1 regularization (lasso regression), the penalty term is based on the absolute value of magnitude of the weights. On the other side, L2 regularization (ridge regression) uses the squared magnitude of the weights as penalty term. L1 regularization can be very useful when trying to compress a model (i.e. having too many features) as it can reduce some of the weights to 0. Otherwise, L2 regularization is the preferred method, especially in deep learning.

13. **Describe the bias vs variance tradeoff in machine learning.**

- High bias is linked to underfitting (both training and validation errors are similar and high during training).
- High variance is linked to overfitting (the validation error is much higher than the training error during training).
- It is typically hard to find a balance between having low bias and low variance. A more complex model (more learnable parameters) will reduce bias, but also increase the risk of overfitting (higher variance). On the other hand, a simpler model (less parameters) will reduce variance, but also increase the risk of underfitting.

14. **Consider the following activation functions: (a) sigmoid, (b) tanh, (c) ReLU.**
(i) Which activation is the least computationally expensive?
(ii) Which activation is more suited for the output of a binary logistic regression?
(iii) Which activation is robust against the vanishing gradient problem?
(iv) Which activation allows negative output values?
(v) Which activation can be easily modified to output negative values by adding a parameter?

- (i) ReLU.
- (ii) Sigmoid (output is between 0 and 1, which can easily be converted to a probability).
- (iii) ReLU.
- (iv) Tanh (output is between -1 and 1).
- (v) ReLU (see leaky ReLU).

15. **What is the difference between a dev (validation) dataset and a test dataset?**

- Dev/validation set: dataset used during the training process to tune the hyperparameters and/or select the best model (based on the validation error).
- Test set: dataset used after the training process to evaluate the final performance of the model (i.e. unseen data).

16. **You are training a deep learning model to perform binary classification and you have 125 training samples. (a) Describe the training methodology if you decide to use k-fold cross validation with k=5. (b) Describe the training methodology if you decide to use the standard deep learning train/dev/test split and considering you have an extra 25 unseen samples for the testing. (c) Which one of the two methodologies would be more suited in this specific case?**

(a) K-fold cross validation with k=5:

- Split training set of 125 samples into 5 folds of 25 samples each.
- Train a model on 4 folds and validate on the remaining fold.
- Repeat previous step 4 other times while using a different fold every time for the validation.
- Average the validation metric(s) (e.g. classification accuracy) of the 5 models to get the average accuracy.
- If the results are satisfying, retrain a final model using all 125 training samples.

(b) Standard train/dev/test split with 25 test samples:

- Split the training set of 125 samples into two sets: (1) a training set (e.g. 100 samples) and (2) a dev/validation set (e.g. 25 samples).
- Train models with different hyperparameters/architectures on the 100 training samples.
- Validate models on the dev/validation set by computing a metric such as the classification accuracy.

- Select the best model based on the dev/validation accuracy.
- Retrain a new model on all the 125 training samples (100 training + 25 validation samples).
- Evaluate final performance of the model on the test set (the 25 new samples).

(c) In this case, cross validation would be the best choice because the amount of data available for the training is small. Cross validation is not recommended when dealing with very large datasets as it becomes computationally expensive to train multiple models. Thus, the standard train/dev/test split is a good approach for large datasets.

17. **What data augmentation strategies could be appropriate when doing digit image recognition (multiclass image classification)?**

- Rescaling (i.e. zooming).
- Rotation with small range (e.g. +/- 10 degrees).
- Shifting with small range (vertical and/or horizontal).
- Flipping (vertical and/or horizontal).
- Elastic deformation with small range.
- Small noise addition.
- Small intensity transformations.

18. **You are doing transfer learning by fine-tuning an existing model. How do the amount of training data available and the similarity between your task and existing trained models can affect your transfer learning strategy?**

If the tasks are similar, you may need to fine tune less layers (freeze more of the early generic layers while updating the few layers on top that learn more specific features related to your task). If you have more data available, you may want to fine tune more layers (less chance of overfitting in that case), while less data available may require you to use more layers of the original model (to avoid overfitting).

19. **What is the advantage of dilated (atrous) convolutions over standard convolutions in a convolutional neural network?**

Dilated convolutions exponentially increase the receptive field without changing the number of learnable parameters. For example, a standard convolutional layer of size 3×3 will have a receptive field of 3×3, while a dilated convolutional layer of the same number of parameters and a dilation factor of 2 will have a receptive field of 7×7.

20. **What would be the uses of a linear (i.e. identity) activation function?**

Linear activation functions are rarely used in deep learning because we typically need to add nonlinearities in order to learn more complex functions. However, linear activation functions are used in linear regression models.

21. **Vanishing and exploding gradients. (a) Define the vanishing and exploding gradients problem in deep learning. (b) Propose solutions to deal with the vanishing gradient problem. (c) Propose solutions to deal with the exploding gradient problem.**

(a)

Vanishing gradients:

- Gradients of the loss function become very small (close to 0) in deeper networks during backpropagation (multiplication of multiple small derivatives).
- Causes: (i) using inadequate weight initialization schemes (e.g. weights too small or too close to 0), (ii) using inadequate activation functions that output small derivatives (e.g. sigmoid or tanh).
- Consequences: (i) weights and biases of the first layers will not get updated correctly (no features learned in the first layers, which will also affect the following layers), (ii) may take longer to train, (iii) may lead to slower convergence (or no convergence at all).

Exploding gradients:

- Gradients of the loss function become too large during training in deeper networks during backpropagation (multiplication of multiple large derivatives).
- Causes: (i) a learning rate that is too large (leading to large weight updates), (ii) poor choice of loss function (e.g. loss function that computes large error values), (iii) poor weight initialization.
- Consequences: (i) large gradients can cause huge leaps in the loss function, (ii) fail to converge during training, (iii) instability added to the training process.

(b) Some solutions for the vanishing gradient problem:

- (1) Use activation functions that do not output small derivatives, such as ReLU or its variants (e.g. leaky ReLU).
- (2) Use residual connections (skip connections). They improve the flow of gradients in very deep networks.
- (3) Use batch normalization layers. They control the range of the input so it does not get too small or too large, avoiding small derivatives.
- (4) Use more computational power (i.e. GPUs). Sometimes, training the deep network longer (i.e. slower convergence) can be acceptable if you have access to more computational power.
- (5) User better weight initialization schemes.

(c) Some solutions for the exploding gradient problem:

- (1) Use gradient scaling: rescale the gradients in order to have a certain vector norm (e.g. the tensor of gradients cannot exceed a norm of 1.0).
- (2) Use gradient clipping: clip gradients values that are too large (e.g. all gradient values smaller than -0.5 are set to -0.5 and all gradient values larger than 0.5 are set to 0.5).
- (3) Use ReLU activation functions.
- (4) Use better weight initialization schemes.

22. **You have the choice between a multivariate logistic regression model (standard machine learning approach) and a simple multilayer perceptron (deep learning approach) for classification of data points into 2 classes. What factors would influence your choice?**

- Complexity of the task: a deep learning model will allow you to learn more complex features, but is also more prone to overfitting (usually more learnable parameters).
- Amount of data available: a standard machine learning approach would be better for small training datasets (less risk of overfitting the data when training simpler models).

23. **You are doing MNIST digit classification with a simple neural network and you want to add dropout to your architecture to improve generalization. How would you select the best dropout value for your model (assuming you plan to use the same dropout value on all hidden layers)?**

- Design a baseline model with no dropout.
- Design different models by varying the dropout value (all the other hyperparameters such as learning rate, number of layers, number of units, mini-batch size, number of epochs should be kept the same). For instance, you can implement 6 different models with dropout values ranging from 0 (baseline) to 0.5, with leaps of 0.1.
- Train all designed models for the same number of epochs and save error values of the training and validation curves after each iteration.
- Plot all training and validation curves of the models on the same graph.
- Pick the model (i.e. dropout value) that has the better overall performance (i.e. small training error on both the training and validation curves and no overfitting trends).
- Retrain a final model on all training data (training + validation) using the optimal dropout value.

24. **Name one machine learning and one deep learning method that you can use to perform dimensionality reduction on your data.**

- Machine learning method: Principal Component Analysis (PCA).
- Deep learning method: auto-encoders.

25. **What are some variants of the rectified linear unit (ReLU) activation function and what are their advantages over the standard ReLU?**

- Standard ReLU: fast to compute, avoids the vanishing gradient problem, but can cause the dying ReLU problem.
- ELU (exponential linear unit): more computationally expensive than ReLU, but avoids both the vanishing gradient and the dying ReLU problem.
- Leaky ReLU: faster than ELU, avoids both vanishing and dying ReLU problems.