More about Training A Network

10/01/2024

Agenda

- Announcements:
  - HW3 extended to the end of this Friday
  - Midterm quiz next Tuesday (10/8)
- Topics:
  - Training, validation, and test
  - Tweaking neural networks

Midterm Quiz

- When and where: 10/8 (Tue.) in class.
- It covers all materials up to today’s lecture (inclusive)
- You will have 75 minutes to complete.
  - You won’t need to write a lot of things, but it does test your understanding.
- You can bring one page of note (max 8.5 inch by 11.75 inch, or standard writing pad paper), and you can write anything on both sides.
- No electronics (you won’t need a calculator).

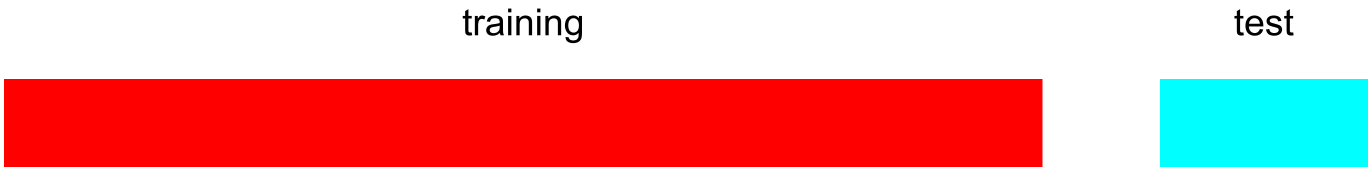
Parameters vs. Hyper-parameters

- (Model) parameters = weights + biases
    - The optimal values are selected by the model
- (Model) hyper-parameters = learning rate + layer counts + layer density + …
    - The optimal values are chosen by you

Steps of building a good model:

1. For each combination of hyper-parameter values, do
    a. train a model/models with the hyper-parameter values
    b. evaluate the performance
2. Find the best set of hyper-parameter values, and use them to train a model

Training, and Test

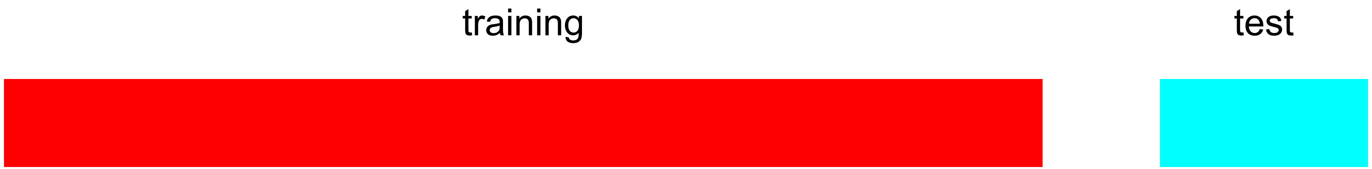


training

test

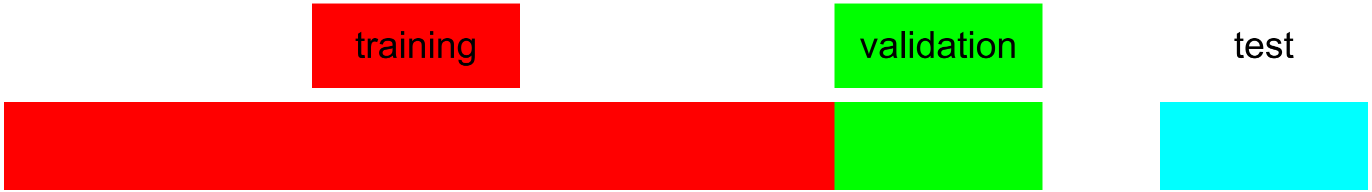
- Training set is for building your model, while test set is reserved for final evaluation.
  - For example, in a competition, competitors only receive the training set and the organizer keeps the test set.
  - If you are given a whole dataset, you can reserve a portion for test set yourself.
- If your training procedure involves a test sample, then any result is considered invalid.
- The test set can be large or small.

Training, and Test



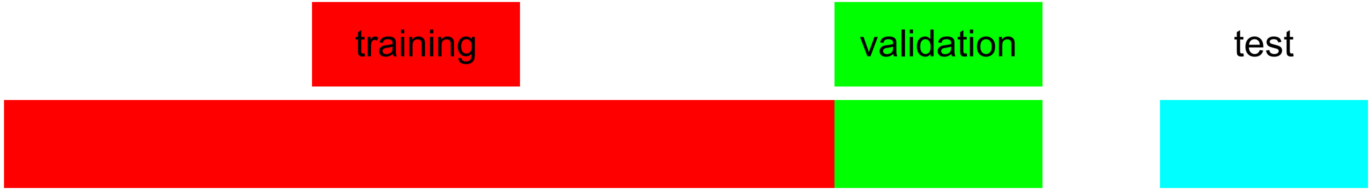
- If we feed all training samples to a model, we will never know how good our model is on unseen data (test data **are** unseen data).
- Therefore, we need to have some data to simulate the test set.

Training, Validation, and Test



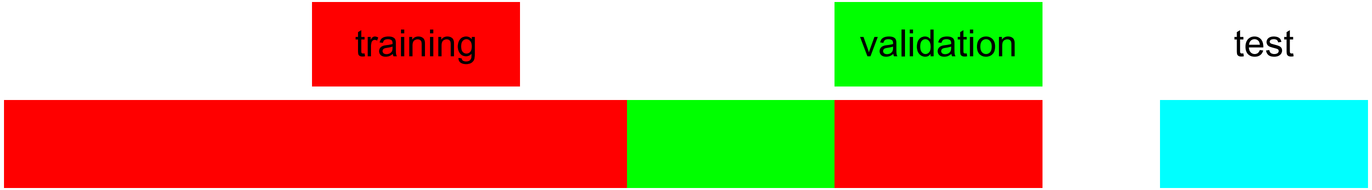
- If we feed all training samples to a model, we will never know how good our model is on unseen data (test data **are** unseen data).
- Therefore, we need to have some data to simulate the test set.
- We can (randomly) split the training data into two:
  - the (larger) subset is still called the training set,
  - the other is called the validation set.
- The hypothesis is that, if a model can do well on the validation set, then it should do well on the test set at the end.

Training, Validation, and Test



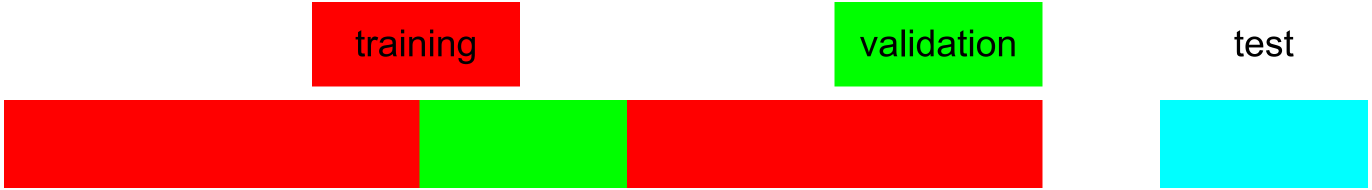
- But can we do more with this setting?
- In particular, can we run more experiments of this type to obtain a more convincing “best model”?
- What about repeating the training and validation trial by choosing a different subset for validation?
- This leads to the so called **cross validation (CV)**.

Training, Validation, and Test



- But can we do more with this setting?
- In particular, can we run more experiments of this type to obtain a more convincing “best model”?
- What about repeating the training and validation trial by choosing a different subset for validation?
- This leads to the so called **cross validation (CV)**.

Training, Validation, and Test



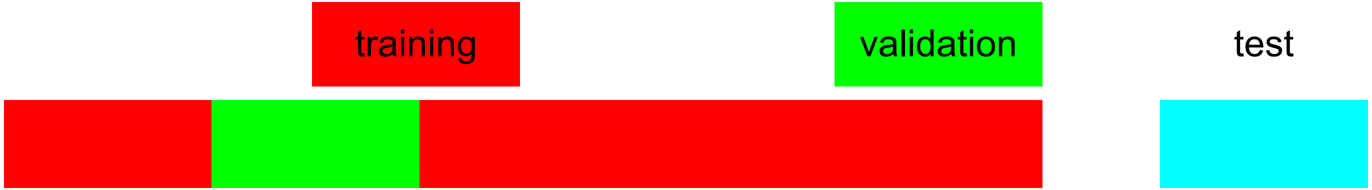
training

validation

test

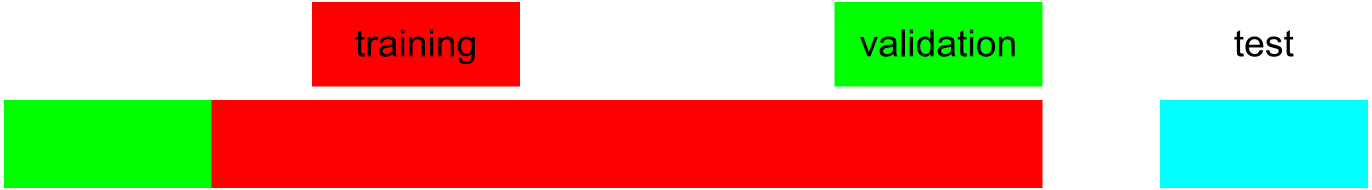
- But can we do more with this setting?
- In particular, can we run more experiments of this type to obtain a more convincing “best model”?
- What about repeating the training and validation trial by choosing a different subset for validation?
- This leads to the so called **cross validation (CV)**.

Training, Validation, and Test



- But can we do more with this setting?
- In particular, can we run more experiments of this type to obtain a more convincing “best model”?
- What about repeating the training and validation trial by choosing a different subset for validation?
- This leads to the so called **cross validation (CV)**.

Training, Validation, and Test



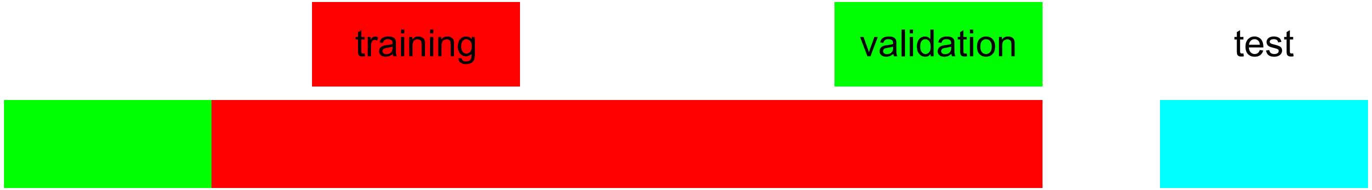
training

validation

test

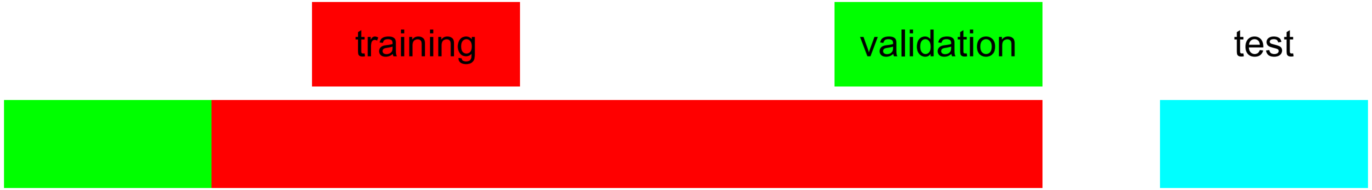
- But can we do more with this setting?
- In particular, can we run more experiments of this type to obtain a more convincing “best model”?
- What about repeating the training and validation trial by choosing a different subset for validation?
- This leads to the so called **cross validation (CV)**.

Training, Validation, and Test



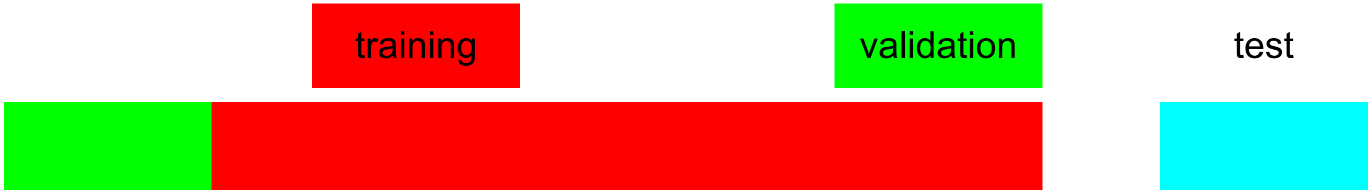
- During CV, each training sample is used for validation exactly once.
- Note that samples are still randomly chosen.
  - This can be achieved by first shuffling the training set.
- Each trial is called one **split** or one **fold.**
  - The example is 5-fold CV.

Training, Validation, and Test



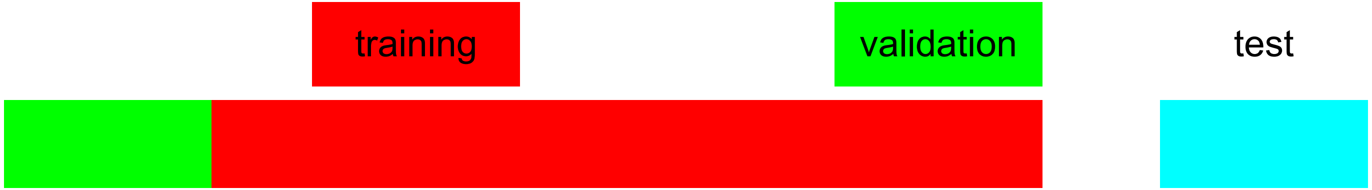
- In addition, we can make CV better by ensuring that for each split, the ratio of different classes are the same as in the whole training set.
- This is extremely important for imbalanced training sets.
- Such CV is called **stratified K-fold CV.**
  - Sidenote: the StratifiedKFold module from sklearn does **not** shuffle the data by default, but you should make it do.

Training, Validation, and Test



- After a stratified K-fold CV, there are K performance scores.
- The average score is used to judge the hyper-parameter values.
- The same process is repeated for all combinations of hyper-parameter values.
  - The randomization should be seeded for all repetitions for fair comparison.
- Once the best combination is found, train a fresh model with it on the **entire** training set.

Training, Validation, and Test



training

validation

test

- This setting is not always the gold standard.
  - For example, when outputting the best model is not the goal.
- Sometimes, a CV is sufficient.
  - There is no “test” set in this scenario, but people often refer to the validation set as test set.
- Sometimes, people perform a nested CV.
- Which setting to choose depends on the task and the dataset.

Tweaking Neural Networks

- Our ultimate goal is to have our model perform well on all data.
  - Especially the unseen ones.
- If our model performs bad on the training set (loss remains high or does not decrease low enough), we know it is unlikely to perform well on the test set.
  - This is called **underfitting**: the model hasn't learned enough.
- On the other hand, if our model performs too well on the training set (loss = 0), we also need to be cautious, as it is likely to perform terrible on the test set.
  - This is called **overfitting**: the model has learned too much from the training set.

Tweaking Neural Networks - Bias and Variance

To keep it simple:

- Bias is the error between ground truth and prediction (on the training set).
    - A high bias indicates underfitting.
- Variance is the difference between the performance on the training set and the performance on the test set.
    - A high variance means overfitting.

These are two types of error and we want to reduce both.

Usually there is a tradeoff between them, so we try to balance it by looking for the optimal point.

Tweaking Neural Networks - Bias (Underfitting)

This error is often not a real problem.

Because if a model is underfitting, we can simply train for more epochs, or make the structure more complex, or adjusting other hyper-parameters.

In short, we can resolve it by trying harder.

Tweaking Neural Networks - Variance (Overfitting)

Overfitting means the model fits too well on the training set and cannot generalize to the unseen data.

There are different ways to address this error:

1. Adding more (and more diverse) training data
2. Introducing bias to the network
3. Introducing regularization term to loss
4. Introducing dropout to the network
5. Early stopping

Tweaking Neural Networks - Variance (Overfitting)

Overfitting means the model fits too well on the training set and cannot generalize to the unseen data.

There are different ways to address this error:

1. Adding more (and more diverse) training data
2. Introducing bias to the network
3. Introducing regularization term to loss
4. Introducing dropout to the network
5. Early stopping

Tweaking Neural Networks - Regularization

An overfit model is said to have too complex weights.

Basically, regularization means adding a term that is related to the model complexity to the loss calculation.

In this way, the model is penalized for its complexity.

When computing the gradients, the extra term is taken into account.

Tweaking Neural Networks - Regularization

Two most common regularization:

L1 regularization

$$L_{\text{reg}} = L + \tfrac{\lambda}{2} \sum_k ||w^{(k)}|| = L + \tfrac{\lambda}{2} \sum_k \sum_{i,j} |w_{ij}^{(k)}|$$

L2 regularization

$$L_{\text{reg}} = L + \tfrac{\lambda}{2} \sum_k ||w^{(k)}||^2 = L + \tfrac{\lambda}{2} \sum_k \sum_{i,j} {w_{ij}^{(k)}}^2$$

Use L2 by default

Tweaking Neural Networks - Regularization

$$L_{\text{reg}} = L + \frac{\lambda}{2} \sum_k ||w^{(k)}||^2 = L + \frac{\lambda}{2} \sum_k \sum_{i,j} {w_{ij}^{(k)}}^2$$

The gradient of the L2 norm is computed as

$$\frac{\partial}{\partial w^{(k)}} ||w^{(k)}||^2 = \frac{\partial}{\partial w^{(k)}} \sum_{i,j} {w_{ij}^{(k)}}^2 = \{\frac{\partial}{\partial w_{ij}^{(k)}} \sum_{i,j} {w_{ij}^{(k)}}^2\} = \{2w_{ij}^{(k)}\} = 2w^{(k)}$$

Then, the gradient of the regularized loss is

$$\frac{\partial L_{\text{reg}}}{\partial w^{(k)}} = \frac{\partial L}{\partial w^{(k)}} + \lambda w^{(k)}$$

Lastly, to complete the story

$$w_{\text{new}}^{(k)} = w^{(k)} - \frac{\gamma}{m} \cdot \frac{\partial L_{\text{reg}}}{\partial w^{(k)}} = w^{(k)} - \frac{\gamma}{m} \cdot (\frac{\partial L}{\partial w^{(k)}} + \lambda w^{(k)})$$

Tweaking Neural Networks - Regularization

- How to choose the value for λ?
  - Unlike the learning rate, the regularization parameter can go beyond 1.
  - But you should try small ones first (0.0001, 0.001, etc.).
- How does the value of λ impact the final performance?
  - Why do we introduce regularization term again?
  - What will happens if we set it to a small value (like 0)?
  - And what may happen if we set it to a large value?
- Is there regularization for bias?
  - No. We use bias to penalize high variance. So why penalizing it when we are penalizing high variance?

Tweaking Neural Networks - Dropout

If we set the dropout rate to be ***p*** for a certain layer, then each node in that layer has a ***p*** chance to be removed during a training batch.

This is like training a slightly smaller network per batch.

During test time, all nodes are present.

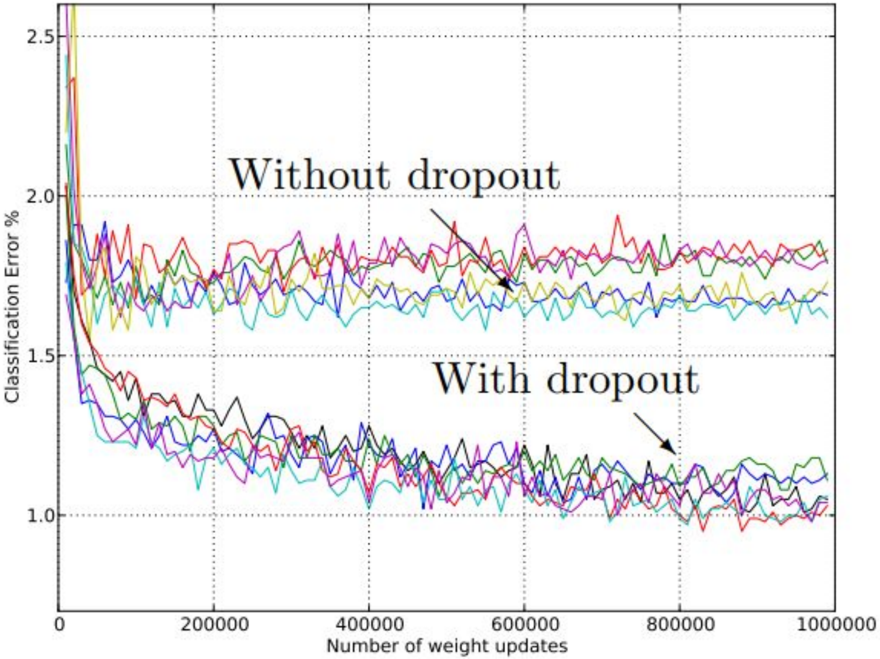
One intuition behind this technique is that it forces the network to learn from less training information.

Tweaking Neural Networks - Dropout

The mechanism is simple but powerful.

The figure is taken from the paper (Srivastava et al., 2014) which proposed the mechanism.

Check out the original paper here online if interested.


Without dropout
With dropout
Classification Error %
Number of weight updates