INF721 - Deep Learning
L8: Regularization

Prof. Lucas N. Ferreira
Universidade Federal de Viçosa

2024/2

1 Introduction

This lecture covers regularization techniques in deep learning, which are essential methods for reducing overfitting and improving model generalization. We will discuss several regularization approaches, including L1 and L2 regularization, dropout, data augmentation, and early stopping.

2 Experimenting with Neural Networks

When working with neural networks, it's crucial to experiment with different configurations to find the best-performing model. The key hyperparameters to consider include:

- Number of hidden layers
- Number of hidden units per layer
- Activation functions
- Learning rate

The goal is to find a configuration that performs well on the validation set. This process involves dealing with underfitting and overfitting scenarios.

2.1 Underfitting vs. Overfitting

When experimenting with neural networks, you may encounter different scenarios:

	Underfit	Overfit	Good Fit
Training error	High	Low	Low
Validation error	High	High	Low

- **Underfitting (High Bias):** The model performs poorly on both training and validation sets, indicating that it's too simple to capture the underlying patterns in the data.
- **Overfitting (High Variance):** The model performs well on the training set but poorly on the validation set, suggesting that it has memorized the training data rather than learning generalizable patterns.
- **Good Fit:** The model performs well on both training and validation sets, indicating that it has learned the underlying patterns without overfitting.

2.2 Example: Image Classification

Consider an image classification task for cats vs. dogs, with a balanced dataset and a human baseline accuracy of approximately 100%:

	Underfit	Overfit	Good Fit
Training Accuracy	45%	99%	95%
Validation Accuracy	42%	67%	94%

This example illustrates how different scenarios manifest in terms of training and validation accuracies.

2.3 Experimental Recipe

A common approach to defining a good neural network for a particular problem is to follow an experimental recipe that involves iteratively adjusting hyperparameters based on the model's performance on that problem:

Algorithm 1 Neural Network Experimentation

```
Start with an initial set of hyperparameters
while not done do
    Train the model on the training set
    if high bias (underfitting) then
        Try: bigger network, train longer, different architecture
    else if high variance (overfitting) then
        Try: more data, regularization, different architecture
    else
        Done
    end if
end while
Report test performance
```

This iterative process helps in systematically improving the model's performance.

3 Regularization

Regularization techniques aim to simplify models and reduce overfitting. The main regularization methods we'll discuss are:

- L1 regularization
- L2 regularization
- Dropout
- Data augmentation
- Early stopping

4 L1 Regularization

L1 regularization adds the L1 norm of the weight matrices to the loss function, penalizing neural networks with high weight values:

$$L(h) = -\frac{1}{m}\sum_{i=1}^{m} L(y^{(i)}, \hat{y}^{(i)}) + \frac{\lambda}{2m}\sum_{l} \|W^{[l]}\|_1$$

where λ is a hyperparameter controlling the strength of regularization.
For a weight matrix $W^{[l]}$, the L1 norm is defined as:

$$\|W^{[l]}\|_1 = \sum_{i=1}^{n^{[l-1]}} \sum_{j=1}^{n^{[l]}} |w_{ij}|$$

L1 regularization tends to make the weight matrix sparse, pushing many weights to exactly zero.

5 L2 Regularization

L2 regularization adds the square of the L2 norm of the weight matrices to the loss function:

$$L(h) = -\frac{1}{m}\sum_{i=1}^{m} L(y^{(i)}, \hat{y}^{(i)}) + \frac{\lambda}{2m}\sum_{l} \|W^{[l]}\|_2^2$$

For a weight matrix $W^{[l]}$, the squared L2 norm is defined as:

$$\|W^{[l]}\|_2^2 = \sum_{i=1}^{n^{[l-1]}} \sum_{j=1}^{n^{[l]}} |w_{ij}|^2$$

L2 regularization decays weights more uniformly over time, rarely setting them to exactly zero.

6 Gradient Descent Update Rules with Regularization

When we modify the loss function with regularization terms, the gradient descent update rule changes accordingly. Let's examine how the gradients of the loss function change for both L1 and L2 regularization.

6.1 L1 Regularization

The L1 regularized loss function $L_r(h)$ of a model h with l weight matrices is:

$$L_r(h) = L(h) + \frac{\lambda}{2m} \sum_l \|W^{[l]}\|_1$$

where $L(h)$ is the original loss function (e.g., BCE or MSE) and $\|W^{[l]}\|_1$ is the L1 norm of the weight matrix $W^{[l]}$.

Step 1: Expand the L1 norm

$$L_r(h) = L(h) + \frac{\lambda}{2m} \sum_l \sum_{i,j} |w_{ij}^{[l]}|$$

Step 2: Take the partial derivative with respect to a single weight $w_{ab}^{[k]}$

$$\frac{\partial L_r}{\partial w_{ab}^{[k]}} = \frac{\partial L}{\partial w_{ab}^{[k]}} + \frac{\lambda}{2m} \frac{\partial}{\partial w_{ab}^{[k]}} \sum_l \sum_{i,j} |w_{ij}^{[l]}|$$

Step 3: Simplify the regularization term

$$\frac{\partial L_r}{\partial w_{ab}^{[k]}} = \frac{\partial L}{\partial w_{ab}^{[k]}} + \frac{\lambda}{2m} \frac{\partial}{\partial w_{ab}^{[k]}} |w_{ab}^{[k]}|$$

Step 4: Apply the derivative of the absolute value function

$$\frac{\partial L_r}{\partial w_{ab}^{[k]}} = \frac{\partial L}{\partial w_{ab}^{[k]}} + \frac{\lambda}{2m} \text{sign}(w_{ab}^{[k]})$$

where $\text{sign}(x)$ is defined as:

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Step 5: Generalize to the entire weight matrix $W^{[l]}$

$$\frac{\partial L_r}{\partial W^{[l]}} = \frac{\partial L}{\partial W^{[l]}} + \frac{\lambda}{2m} \text{sign}(W^{[l]})$$

6.2 L2 Regularization

The L2 regularized loss function is:

$$L_r(W) = L(W) + \frac{\lambda}{2m} \sum_l \|W^{[l]}\|_2^2$$

where $\|W^{[l]}\|_2^2$ is the squared L2 norm of the weight matrix $W^{[l]}$.

Step 1: Expand the squared L2 norm

$$L_r(W) = L(W) + \frac{\lambda}{2m} \sum_l \sum_{i,j} (w_{ij}^{[l]})^2$$

Step 2: Take the partial derivative with respect to a single weight $w_{ab}^{[k]}$

$$\frac{\partial L_r}{\partial w_{ab}^{[k]}} = \frac{\partial L}{\partial w_{ab}^{[k]}} + \frac{\lambda}{2m} \frac{\partial}{\partial w_{ab}^{[k]}} \sum_l \sum_{i,j} (w_{ij}^{[l]})^2$$

Step 3: Simplify the regularization term

$$\frac{\partial L_r}{\partial w_{ab}^{[k]}} = \frac{\partial L}{\partial w_{ab}^{[k]}} + \frac{\lambda}{2m} \frac{\partial}{\partial w_{ab}^{[k]}} (w_{ab}^{[k]})^2$$

Step 4: Apply the power rule of differentiation

$$\frac{\partial L_r}{\partial w_{ab}^{[k]}} = \frac{\partial L}{\partial w_{ab}^{[k]}} + \frac{\lambda}{2m} \cdot 2w_{ab}^{[k]}$$

Step 5: Simplify

$$\frac{\partial L_r}{\partial w_{ab}^{[k]}} = \frac{\partial L}{\partial w_{ab}^{[k]}} + \frac{\lambda}{m} w_{ab}^{[k]}$$

Step 6: Generalize to the entire weight matrix $W^{[l]}$

$$\frac{\partial L_r}{\partial W^{[l]}} = \frac{\partial L}{\partial W^{[l]}} + \frac{\lambda}{m} W^{[l]}$$

6.3 Gradient Descent Update Rules

Based on these derivations, we can now write the gradient descent update rules for both regularization techniques:

For L1 regularization:

$$W^{[l]} = W^{[l]} - \alpha \left(\frac{\partial L}{\partial W^{[l]}} + \frac{\lambda}{2m} \cdot \text{sign}(W^{[l]}) \right)$$

For L2 regularization:

$$W^{[l]} = W^{[l]} - \alpha \left(\frac{\partial L}{\partial W^{[l]}} + \frac{\lambda}{m} W^{[l]} \right)$$

Which can be rewritten as:

$$W^{[l]} = (1 - \frac{\alpha\lambda}{m}) W^{[l]} - \alpha \frac{\partial L}{\partial W^{[l]}}$$

These step-by-step derivations show how the regularization terms affect the gradients and, consequently, the weight updates during training. The key difference lies in how the regularization term is added to the original loss gradient:

- L1 regularization adds a constant term ($\frac{\lambda}{2m}$) multiplied by the sign of each weight. This pushes weights towards zero, potentially making them exactly zero, which leads to sparse weight matrices.
- L2 regularization adds a term proportional to the weight matrix itself ($\frac{\lambda}{m} W^{[l]}$). This causes the weights to decay towards zero, but they are less likely to become exactly zero.

6.4 Implementation Considerations

In practice, when implementing these regularization techniques:

1. For L1 regularization, you need to compute the sign of the weight matrix, which can be computationally expensive for large networks.
2. For L2 regularization, the weight decay term can be easily incorporated into the update rule, often leading to more efficient implementations.
3. Some deep learning frameworks provide built-in support for these regularization techniques, allowing you to simply specify the regularization type and strength (λ) when defining your model or optimizer.

7 Dropout

Dropout is a regularization technique used to reduce overfitting in neural networks. It works by randomly "dropping out" or deactivating a subset of neurons during each training iteration.

7.1 How Dropout Works

The process of dropout can be described as follows:

1. For each layer, assign a probability (keep probability) of retaining each node.

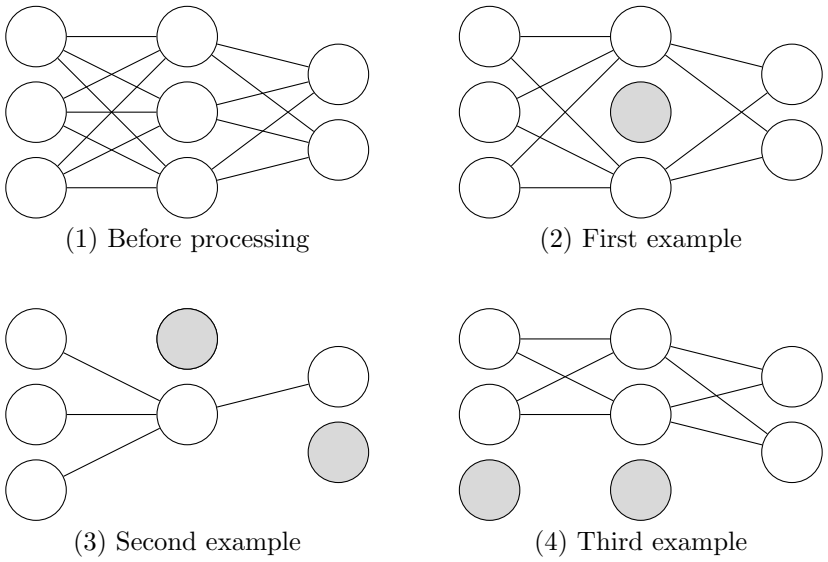

Figure 1: Illustration of dropout dynamics in a neural network across different training examples. Gray neuraons have been dropped out.

2. During each training iteration:

   (a) Randomly select nodes to keep based on the assigned probability.

   (b) Remove the selected nodes and their corresponding input and output connections.

   (c) Train the network on this reduced architecture for the current example.

3. Repeat this process for each training example, creating a different reduced network each time.

This technique forces the network to learn more robust features that can work with many different random subsets of neurons.

7.2 Implementing Dropout

The most common implementation of dropout is called "inverted dropout." Here's a basic implementation in Python:

```
def inverted_dropout(a, keep_prob):
    d = np.random.rand(*a.shape) < keep_prob
    a *= d
    a /= keep_prob
    return a, d
```

In this implementation:

- `a` is the activation of the current layer
- `keep_prob` is the probability of keeping a neuron active
- `d` is a mask of 1s and 0s, where 1 indicates a kept neuron
- We multiply `a` by `d` to zero out dropped neurons
- We divide by `keep_prob` to scale the activations, ensuring the expected value remains unchanged

7.3 Dropout at Test Time

During testing or inference, dropout is typically not used. Instead:

1. Use the full network with all neurons.
2. No random dropping of neurons occurs.
3. No need for additional scaling if inverted dropout was used during training.

This approach ensures deterministic output and takes advantage of the full network's capacity.

7.4 Why Dropout Works

Dropout is effective for several reasons:

1. It prevents co-adaptation of neurons, where features only work well in the context of other specific features.
2. It approximates training an ensemble of many sub-networks, as each training iteration uses a different subset of neurons.
3. It encourages each neuron to learn more robust features that are useful in many contexts, rather than relying on specific co-occurrences of other neurons.

7.5 Practical Considerations

When implementing dropout:

- Typically, lower dropout rates (higher keep probabilities) are used for input layers, and higher dropout rates for hidden layers.
- Dropout is usually not applied to the output layer.
- The keep probability is a hyperparameter that may require tuning for optimal performance.

- Dropout often requires larger networks and more training iterations to achieve the best results.

Dropout has become a standard technique in many neural network architectures due to its simplicity and effectiveness in reducing overfitting.

8 Data Augmentation

Data augmentation consists of artificially increase the size of your training set and reduce overfitting in neural networks. It involves creating new training examples by applying transformations to existing ones. For example, for image classification tasks, common data augmentation techniques include:

- **Flipping**: Horizontally flipping images. For example, flipping a cat image horizontally still results in a valid cat image.
- **Random cropping**: Taking random crops or zooms of the original image.
- **Rotation**: Applying small random rotations to the images.
- **Color jittering**: Slightly altering the color balance or intensity.

It's important to note that the transformations should preserve the image's class. For instance, vertical flips might not be suitable for cat classification, as upside-down cats are not typically encountered.

8.1 Benefits of Data Augmentation

- Increases the effective size of the training set without the cost of collecting new data
- Helps the model learn invariance to certain transformations
- Reduces overfitting by exposing the model to more varied examples

8.2 Limitations

While data augmentation is beneficial, it's not as effective as collecting genuinely new, independent examples. The augmented data is not truly independent of the original data.

9 Early Stopping

Early stopping is a regularization technique that involves halting the training process before the model begins to overfit on the training data.

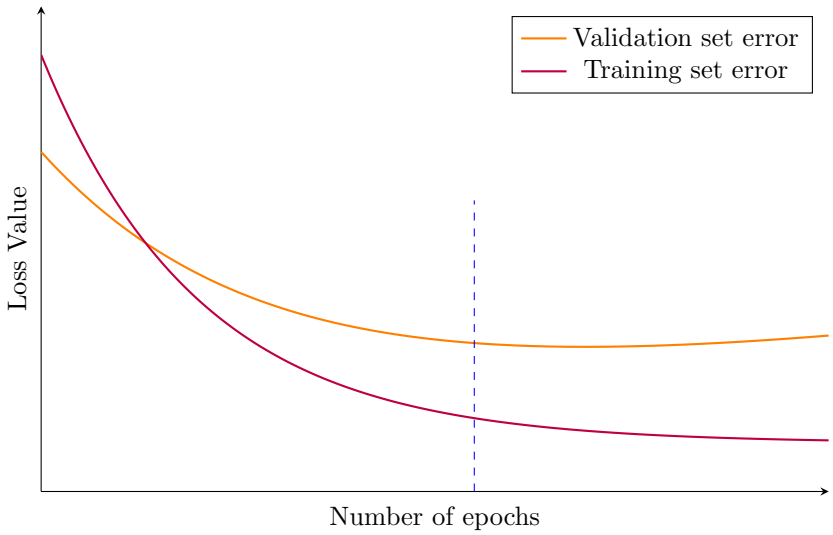

Figure 2: Training and Testing Error Curves. The vertical dashed line indicates the point where early stopping would be applied.

9.1 How Early Stopping Works

1. During training, monitor both the training error and the validation (dev set) error.
2. The training error typically decreases monotonically as training progresses.
3. The validation error often decreases initially but starts to increase as the model begins to overfit.
4. Stop training when the validation error starts to increase or plateau.
5. Use the model parameters from the point where validation error was lowest.

9.2 Advantages of Early Stopping

- Prevents overfitting by stopping training before the model becomes too complex
- Allows you to implicitly try different model complexities without explicitly trying many values of regularization parameters
- Can save computational resources by reducing training time

9.3 Disadvantages and Considerations

- Couples the optimization process with the regularization process, which can make the machine learning pipeline less modular
- May make it harder to separately analyze and improve the optimization and regularization aspects of your model
- Can be seen as interrupting the optimization process before it fully converges

10 Conclusion

Regularization techniques are crucial for improving the generalization of deep learning models. By applying these methods, we can reduce overfitting and create more robust models that perform well on unseen data. The choice of regularization technique depends on the specific problem, dataset, and model architecture. Often, a combination of these techniques yields the best results.

In practice, it's important to experiment with different regularization methods and hyperparameters to find the optimal configuration for your specific task. Remember to use the validation set for tuning these hyperparameters and reserve the test set for final evaluation.

Exercises

1. Consider a neural network model with the following performance:

   - Training set accuracy: 99.5%
   - Test set accuracy: 87%

   Which of the following techniques would be most appropriate to improve the model's generalization?

   (a) Increase the number of neurons in each layer
   (b) Add L2 regularization to the loss function
   (c) Reduce the learning rate
   (d) Remove all activation functions except in the output layer

2. In L2 regularization, what typically happens when you increase the regularization hyperparameter λ (lambda)?

   (a) The model's capacity to fit complex functions increases
   (b) The model's weights tend to become larger in magnitude
   (c) Weights are pushed toward becoming smaller (closer to 0)
   (d) The model becomes more prone to overfitting

3. Which regularization technique is more likely to produce sparse weight matrices?

   (a) L1 regularization
   (b) L2 regularization
   (c) Dropout
   (d) Data augmentation

4. What typically happens when you increase the keep_prob parameter in dropout?

   (a) The regularization effect becomes stronger
   (b) The network learns more slowly during training
   (c) The regularization effect becomes weaker
   (d) The computational cost of training increases significantly

5. Consider a weight matrix $W = \begin{bmatrix} 0.2 & -0.1 \\ 0.3 & 0.4 \end{bmatrix}$. What is its L1 norm? (Round to 2 decimal places)

   (a) 0.50
   (b) 0.72
   (c) 1.00
   (d) 1.28

A Vector and Matrix Norms

A.1 Vector Norms

A norm is a function that maps a vector to a non-negative real number, satisfying certain properties:

For any vectors $x, y \in X$ and scalar $\alpha \in \mathbb{R}$:

1. $\|x\| \geq 0$ and $\|x\| = 0$ if and only if $x = 0$
2. $\|x + y\| \leq \|x\| + \|y\|$ (triangle inequality)
3. $\|\alpha x\| = |\alpha| \|x\|$ (scalar multiplication)

A.1.1 Lp Norms

Lp norms are a special type of norm defined as:

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

Two commonly used norms are:

- L1 Norm: $\|x\|_1 = \sum_{i=1}^{n} |x_i|$
- L2 Norm (Euclidean norm): $\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$

A.1.2 Example: Vector Norms

Let's compute the L1 and L2 norms for the weight vector $w = [-1, 2]$:

L1 norm:

$$\|w\|_1 = |-1| + |2| = 1 + 2 = 3$$

L2 norm:

$$\|w\|_2 = \sqrt{(-1)^2 + 2^2} = \sqrt{1 + 4} = \sqrt{5} \approx 2.236$$

A.1.3 Geometric Representation of Vector Norms

The unit circle for L1 and L2 norms in 2D space looks like this:

This geometric representation helps visualize the differences between L1 and L2 norms.

A.2 Matrix Norms

Matrix norms are similar to vector norms but applied to matrices. They treat a matrix as a vector with mn dimensions, where m and n are the number of rows and columns, respectively.

Two popular matrix norms are:

- L1 Norm: $\|A\|_1 = \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|$
- L2 Norm (Frobenius norm): $\|A\|_2 = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$

A.2.1 Example: Matrix Norms

Let's calculate the L1 and L2 norms for the following weight matrix:

$$W = \begin{bmatrix} 0.1 & -0.05 \\ 0.02 & 0.15 \end{bmatrix}$$

L1 norm:

$$\|W\|_1 = |0.1| + |-0.05| + |0.02| + |0.15| = 0.1 + 0.05 + 0.02 + 0.15 = 0.32$$

L2 norm:

$$\|W\|_2 = \sqrt{0.1^2 + (-0.05)^2 + 0.02^2 + 0.15^2} = \sqrt{0.0354} \approx 0.188$$