**MCSE 666: Assignment 08**

**Roll: MCE 079 05536 Name: Shyed Shahriar Housaini**

**Backpropagation Variants:**

Backpropagation is an algorithm that back-propagates the errors from the output nodes to the input nodes. Therefore, it is simply referred to as the backward propagation of errors. It uses in the vast applications of neural networks in data mining like Character recognition, Signature verification, etc.

Some important variants of backpropagation used in pattern recognition tasks:

**1. Vanilla Backpropagation:**

Definition: Vanilla backpropagation is the foundational algorithm for training neural networks. It calculates gradients of the loss with respect to network parameters and updates them using gradient descent.

Description: This method involves iteratively adjusting weights to minimize the loss function. It can be slow and may struggle with vanishing gradients for deep networks but serves as the basis for more advanced variants.

Math Equation: Weight Update: W(t+1)=W(t)−α⋅∇J(W(t))

Pros: Conceptually simple and widely applicable.

Cons: Slow convergence, issues with vanishing gradients for deep networks.

Use Cases: Used as a starting point for training neural networks.

**2. Stochastic Gradient Descent (SGD):**

Definition: SGD is a variation of backpropagation that accelerates training by updating weights after processing a mini-batch, introducing stochasticity.

Description: It converges faster and can be more robust, escaping local minima.

Math Equation: Weight Update: W(t+1)=W(t)−α⋅∇J(W(t))

Pros: Faster convergence, robust against local minima.

Cons: Stochasticity might introduce noise.

Use Cases: Suitable for large datasets and deep learning tasks.

**3. Mini-batch Gradient Descent:**

Definition: Mini-batch gradient descent is a compromise between vanilla backpropagation and SGD.

Description: It updates weights using random mini-batches, reducing noise in the gradient estimates and speeding up training.

Math Equation: Weight Update: *W*(*t*+1)=*W*(*t*)−*α*⋅∇*J*(*W*(*t*))

Pros: Balance between efficiency and stability.

Cons: Requires tuning of mini-batch size.

Use Cases: Common choice for training deep neural networks.

**4. Adam (Adaptive Moment Estimation):**

Definition: Adam is an advanced optimization algorithm that combines elements of momentum and RMSprop.

Description: It adapts learning rates for each parameter individually, well-suited for non-stationary or noisy problems.

Math Equation: Weight Update: *m*(*t*+1)=*β*1​⋅*m*(*t*)+(1−*β*1​)⋅∇*J*(*W*(*t*))

Pros: Fast convergence, adaptability, handles different scales.

Cons: Requires tuning of hyperparameters.

Use Cases: Versatile optimizer used in various deep learning tasks.

**5. RMSprop (Root Mean Square Propagation):**

Definition: RMSprop adapts learning rates using moving averages of squared gradients.

Description: Effective in handling different scales and non-uniform landscapes in optimization.

Math Equation: Weight Update: *s*(*t*+1)=*β*2​⋅*s*(*t*)+(1−*β*2​)⋅(∇*J*(*W*(*t*)))2

Pros: Effective in complex loss landscapes.

Cons: Hyperparameter sensitivity.

Use Cases: Suitable for scenarios with varying gradient scales.

**6. Adagrad (Adaptive Gradient Algorithm):**

Definition: Adagrad adapts learning rates based on the historical sum of squares of gradients.

Description: Effective when dealing with sparse data or varying learning rate requirements.

Math Equation: Weight Update: *G*(*t*)=*G*(*t*−1)+(∇*J*(*W*(*t*)))2

Pros: Individual parameter learning rates.

Cons: Aggressive learning rate decay.

Use Cases: Applicable when specific features require distinct learning rates.

**7. Adadelta:**

Definition: Adadelta is an extension of Adagrad that reduces the aggressive learning rate decay.

Description: Tends to converge faster than Adagrad in many cases.

Math Equation: Weight Update:   
*G*(*t*)=*β*⋅*G*(*t*−1)+(1−*β*)⋅(∇*J*(*W*(*t*)))2

Pros: Faster convergence, no need for a fixed initial learning rate.

Cons: More complex to implement.

Use Cases: Suitable for scenarios where aggressive learning rate decay hinders convergence.

**8. L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno):**

Definition: L-BFGS is an optimization algorithm suitable for training neural networks.

Description: It combines the efficiency of gradient descent with the global convergence properties of Newton's method.

Math Equation: Weight Update: Uses a limited-memory approximation of the Hessian matrix.

Pros: Efficient navigation of high-dimensional parameter spaces.

Cons: Limited memory requirement.

Use Cases: Effective for complex, high-dimensional optimization problems.

**9. Recurrent backpropagation:**

Recurrent backpropagation is a popular technique for training artificial neural networks, which involves computing the gradients of the loss function with respect to the network's parameters and then changing the parameters to minimize the loss. Recurrent neural networks have an internal feedback loop that enables data to be transferred from one phase to the next. Because the parameters of the network are shared across time steps, it is necessary to properly combine the gradients from each time step in order to update the parameters.

Equation:

Recurrent backpropagation uses a similar equation to the conventional backpropagation algorithm, but it calculates weight updates by adding the gradients from the current time step and all prior time steps. It is shown below:

Δwij(n) = -η ∂E(n) / ∂wij(n) - λ ∂E(n-1) / ∂wij(n) - λ^2 ∂E(n-2) / ∂wij(n) - ... - λ^(n-1) ∂E(1) / ∂wij(n)

where Δwij(n) is the weight update for the j-th neuron in the i-th layer at time step n, η is the learning rate, λ is the forgetting factor, and ∂E(t) / ∂wij(n) is the partial derivative of the error with respect to the weight wij at time step t.

Advantages:

* RBP is a good choice for resolving sequence issues because it can handle sequences of different lengths.
* The feedback links in recurrent backpropagation allows the network to recall earlier inputs.
* Recurrent backpropagation allows sequential data can be handled in real time.

Disadvantages:

* Due to the feedback connections and the requirement to handle sequences of various lengths, training recurrent backpropagation can be computationally expensive.
* When the sequence length is large, recurrent backpropagation can experience gradients that disappear or explode.

Applications:

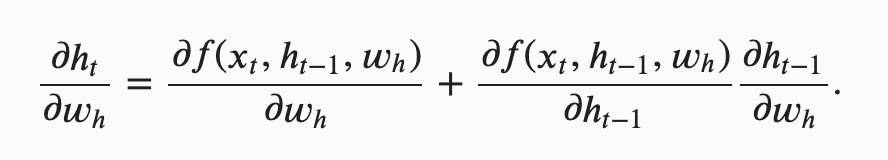
* Recurrent backpropagation is utilized for speech recognition when audio signals are present.
* Recurrent backpropagation is used to process text input that is written in natural language.

**10. Backpropagation through time:**

Backpropagation through time is used in recurrent neural networks (RNNs) to compute gradients for each weight in the network. Like any other neural network, it may be used to train sequences of input/output patterns. However, unlike other neural networks, it also preserves the "memory" of earlier inputs in the sequence, which makes it especially useful for time series data. The feedforward neural networks are trained using the backpropagation technique, which is extended to the RNNs via Backpropagation through time.

Equation:

The Backpropagation through time algorithm involves the use of chain rule of differentiation to compute the gradient of the error function with respect to the weights of the recurrent neural network. The algorithm is based on the following equation:



Advantages:

* For time series data, such as speech recognition, stock price forecasting, and natural language processing, Backpropagation through time is very successful. It has the ability to recall information from the past and can handle input data of any length.
* It is a commonly used and well-known algorithm that has been proven to function effectively in real-world settings.
* Many well-liked deep learning frameworks, like TensorFlow and PyTorch, utilise Backpropagation through time.

Disadvantages:

* Backpropagation through time can be computationally expensive and demands a lot of CPU power.
* The backpropagation of errors over long sequences makes the BPTT training process unstable when the sequence length is large, which causes the vanishing or expanding gradient problem.
* The approach favors short-term memory since it only propagates errors backwards as far as the length of the sequence.

Applications:

* Natural Language Processing: Because Backpropagation through time can produce text sequences based on prior inputs, it is frequently used for language modeling, text categorization, and machine translation.
* Speech Recognition: In order to recognize spoken words and translate them into text, Backpropagation through time is employed in speech recognition tasks.
* Stock Price Prediction: Using historical data trends and Backpropagation through time, one may forecast stock prices.
* Using prior notes and musical patterns, Backpropagation through time can be utilized to create musical sequences.

**11. Nesterov Accelerated Gradient (NAG):**

- A modification of the momentum algorithm.

- Provides better convergence properties by considering future gradients.

**12. Quickprop:**

- A non-iterative method that estimates the direction of the steepest descent.

- Can be faster than standard backpropagation.

**13. Resilient Propagation (Rprop):**

- Optimizes training by adjusting weight updates based on the sign of the gradient.

- Converges quickly, especially in high-dimensional parameter spaces.

**14. Hessian-Free Optimization:**

- Uses second-order information (Hessian matrix) for guiding optimization.

- Efficient and effective in some cases, particularly for large-scale networks.

**15. Natural Gradient Descent:**

- Adapts learning rates in directions that are more relevant for optimization.

- Useful for training deep networks and in non-Euclidean spaces.

**16. K-FAC (Kronecker-factored Approximate Curvature):**

- Approximates the curvature matrix of the loss function.

- Useful for training large-scale neural networks.

**17. LBFGS-NN (LBFGS for Neural Networks):**

- An adaptation of the L-BFGS method specifically for training neural networks.

- Efficiently navigates high-dimensional parameter spaces.

**18. Conjugate Gradient Descent:**

An iterative optimization algorithm used for training neural networks.

Efficiently navigates high-dimensional parameter spaces.

**19. Momentum:**

- Introduces a velocity term to accumulate gradients over time.

- Speeds up convergence and helps the algorithm escape local minima.

Momentum does indeed help the optimization algorithm escape local minima and navigate flat regions more efficiently, but it's not a guaranteed method to escape all local minima. It provides a smoothing effect, which can be particularly beneficial in regions with oscillating gradients. However, in very high-dimensional spaces, there may be multiple types of saddle points, and momentum can sometimes hinder convergence.

20. **Conjugate Gradient Backpropagation (CGP):** A method that uses conjugate gradient optimization to train neural networks.

21. **Quick Propagation (QP):** A method that uses a combination of conjugate gradient optimization and Newton’s method to train neural networks.

**Speed Up with Momentum and Dropout:**

Speeding up the training of deep neural networks is a crucial challenge in the field of deep learning. Two popular techniques used to accelerate the training process are momentum and dropout.

**Momentum:** Momentum is a technique that helps accelerate gradient descent by adding a fraction of the previous weight update to the current update. This helps the optimizer to move more quickly in the right direction and dampens oscillations in the optimization process. Momentum can help speed up convergence and improve generalization performance ¹.

**Dropout:** Dropout is a regularization technique that helps prevent overfitting by randomly dropping out some neurons during training. This forces the network to learn more robust features and reduces the risk of overfitting. Dropout can help speed up convergence and improve generalization performance ².

Momentum and Dropout are two popular techniques used to speed up training in deep neural networks. They work by improving convergence and reducing overfitting, respectively. By using these techniques together, practitioners can achieve even better results. However, they work in different ways and can be used together to achieve even better results.

**Advantages:**

Momentum has several advantages, including faster convergence, improved generalization performance, and reduced oscillations in the optimization process. However, it may require careful tuning of hyper-parameters to achieve optimal results.

Dropout has several advantages, including reduced overfitting, improved generalization performance, and increased robustness to noise. However, it may increase training time and may require careful tuning of hyper-parameters.

Together, momentum and dropout can be used to achieve even better results than using either technique alone. For example, a study on image classification tasks showed that using both momentum and dropout together resulted in faster convergence and better generalization performance than using either technique alone ³.

**Uses of Momentum and Dropout:**

Momentum is particularly useful when dealing with noisy or non-stationary data because it adapts to changes in the data distribution over time. [It also helps prevent getting stuck in local minima by allowing the optimizer to move more quickly in the right direction](https://www.physicsclassroom.com/Concept-Builders/Momentum-and-Collisions/Impulse-and-Force/conceptBuilderHelp/help8) [1](https://www.physicsclassroom.com/Concept-Builders/Momentum-and-Collisions/Impulse-and-Force/conceptBuilderHelp/help8).

[Dropout is particularly useful when dealing with large datasets because it reduces overfitting by forcing the network to learn more robust features](https://www.physicsclassroom.com/Concept-Builders/Momentum-and-Collisions/Impulse-and-Force/conceptBuilderHelp/help8) [2](https://datascience.stackexchange.com/questions/84167/what-is-momentum-in-neural-network). It can also be used with other regularization techniques such as weight decay to further improve generalization performance .

Together, momentum and dropout can be used to speed up training while improving generalization performance. They are widely used in deep learning applications such as image classification, speech recognition, natural language processing, and more.

References: All references were used/visited on 11/10/2023

[1](https://www.physicsclassroom.com/Concept-Builders/Momentum-and-Collisions/Impulse-and-Force/conceptBuilderHelp/help8): Qian, N. (1999). On the momentum term in gradient descent learning algorithms. [Neural Networks: The Official Journal of the International Neural Network Society, 12(1), 145-151.](https://datascience.stackexchange.com/questions/84167/what-is-momentum-in-neural-network)

[2](https://datascience.stackexchange.com/questions/84167/what-is-momentum-in-neural-network): Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929-1958. : Zhang, Y., & LeCun, Y. (2015). Regularization of neural networks using dropconnect. International Conference on Machine Learning (ICML). : Goodfellow I., Bengio Y., Courville A. (2016) Deep Learning. MIT Press.

[3](https://link.springer.com/article/10.1007/s11042-019-08453-9): Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. [ICML (3), 28(1139-1147), 5.](https://www.physicsclassroom.com/Concept-Builders/Momentum-and-Collisions/Impulse-and-Force/conceptBuilderHelp/help8)

[4](https://www.sparknotes.com/physics/linearmomentum/conservationofmomentum/problems_2/): Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. [Journal of Machine Learning Research, 15(1), 1929-1958.](https://web.njit.edu/~gary/111/assets/HW8_sol.pdf)

[5](https://web.njit.edu/~gary/111/assets/HW8_sol.pdf): Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. [Journal of Machine Learning Research, 15(1), 1929-1958.](https://www.physicsclassroom.com/Concept-Builders/Momentum-and-Collisions/Impulse-and-Force/conceptBuilderHelp/help8)

6: Murphy, *Machine Learning: A Probabilistic Perspective*(2012)

7: Goodfellow et al, *Deep Learning* (2016)

8: Cauchy, *Méthode générale pour la résolution des systèmes d’équations simultanées* (1847)

9: Lecun, *Backpropagation Applied to Handwritten Zip Code Recognition* (1989)

10: Parkhi et al, *Deep*[*Face Recognition*](https://deepai.org/machine-learning-model/facial-recognition), (2015)

11: Tsunoo et al (Sony Corporation, Japan), *End-to-end Adaptation with Backpropagation through WFST for On-device Speech Recognition System* (2019)

12: <https://www.physicsclassroom.com/Concept-Builders/Momentum-and-Collisions/Impulse-and-Force/conceptBuilderHelp/help8>

13: [Backpropagation Definition | DeepAI](https://deepai.org/machine-learning-glossary-and-terms/backpropagation), <https://deepai.org/machine-learning-glossary-and-terms/backpropagation>

14.

1. [deepai.org](https://deepai.org/machine-learning-glossary-and-terms/backpropagation)
2. [iq.opengenus.org](https://iq.opengenus.org/types-of-backpropagation/)
3. [globalspec.com](https://www.globalspec.com/reference/71811/203279/4-4-back-propagation-algorithm-and-its-variants)
4. [machinelearningmastery.com](https://machinelearningmastery.com/difference-between-backpropagation-and-stochastic-gradient-descent/)
5. [en.wikipedia.org](https://en.wikipedia.org/wiki/Backpropagation)