Optimizers and Metrics

What is an optimizer?

Optimizers are algorithms or methods in machine learning, used to change the attributes of a neural network, namely weights and biases, in order to reduce the loss. In other words, an optimizer contributes to a model's improvement by modifying the model's parameters to reduce the loss function value.

The optimizer's job is to determine which combination of the neural network's parameters will provide the- best chance to generate accurate predictions.

Stochastic Gradient Descent (SGD) and Stochastic Gradient Descent with momentum (SGD + Momentum) are two examples of optimizers which we will discuss further in the lecture.

Why do we need optimizers?

When we are building a machine learning model, it is impossible to know what the best values for the parameters are. However, with some trial and error, based on the loss function, we are likely to have an efficient model.
Optimization algorithms are responsible for reducing the losses to provide the most accurate results possible. Therefore, having a suitable optimizer is vital for training a neural-network, as it directly influences the model's ability to learn from the given data.

Terminology Breakdown

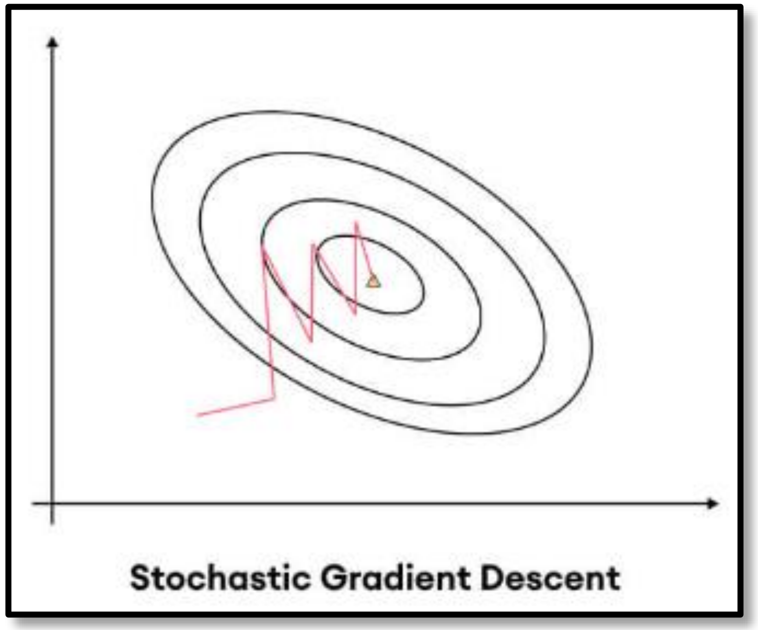
Before digging deeper into different types of optimizers, we will first get acquainted with a few terms:

- **Batch**: A subset of the training data used to update the model’s weights
- **Sample**: A single record of data in a data set
- **Learning Rate**: A parameter determining the scale of the model’s weight updates
- **Batch weight**: The number of samples used to update the model’s parameters

Types of optimizers

1. Stochastic Gradient Descent (SGD)

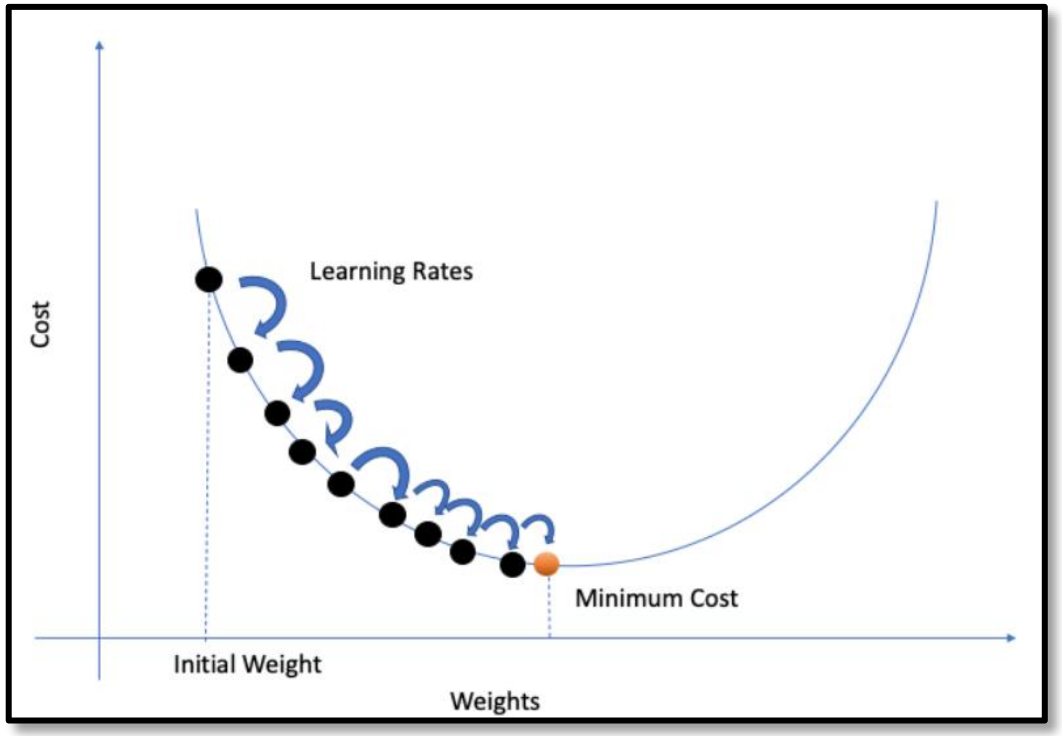
Stochastic Gradient Descent (SGD) is an iterative optimization algorithm used primarily in machine learning and deep learning.
SGD updates the model parameters using only a single training example or a small batch of examples at each iteration. This introduces randomness into the optimization process, hence the term “stochastic”.



Types of optimizers

1. Stochastic Gradient Descent (SGD)

In each iteration, SGD randomly selects one data point, or a mini-batch, from the training data set, computes the gradient of the loss function with respect to the model parameters for that data point, and updates the parameters in the direction that reduces the loss. This process is repeated until the model converges to a minimum of the loss function.



Types of optimizers

1. Stochastic Gradient Descent (SGD)

Here is the update rule for SGD:

$$\theta = \theta - \alpha . \nabla_{\theta} . L(\theta)$$

Where:

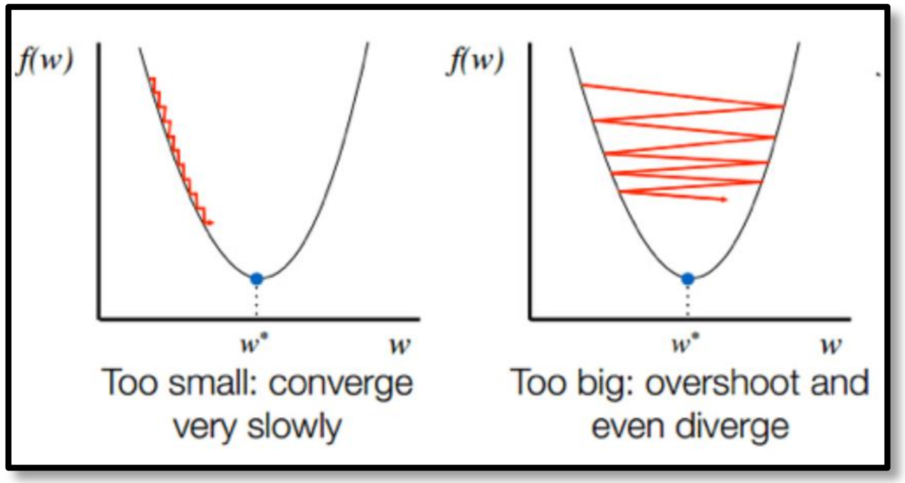
θ: parameters of the model, namely weights and biases
α: learning rate
L: loss function

The first step in performing stochastic gradient descent is to calculate the gradient of the loss function with respect to our parameters. Once the gradient is calculated, we multiply it by our learning rate. Then, we simply update the parameters in this direction.
We repeat this process iteratively until convergence is achieved.

Types of optimizers

1. Stochastic Gradient Descent (SGD)

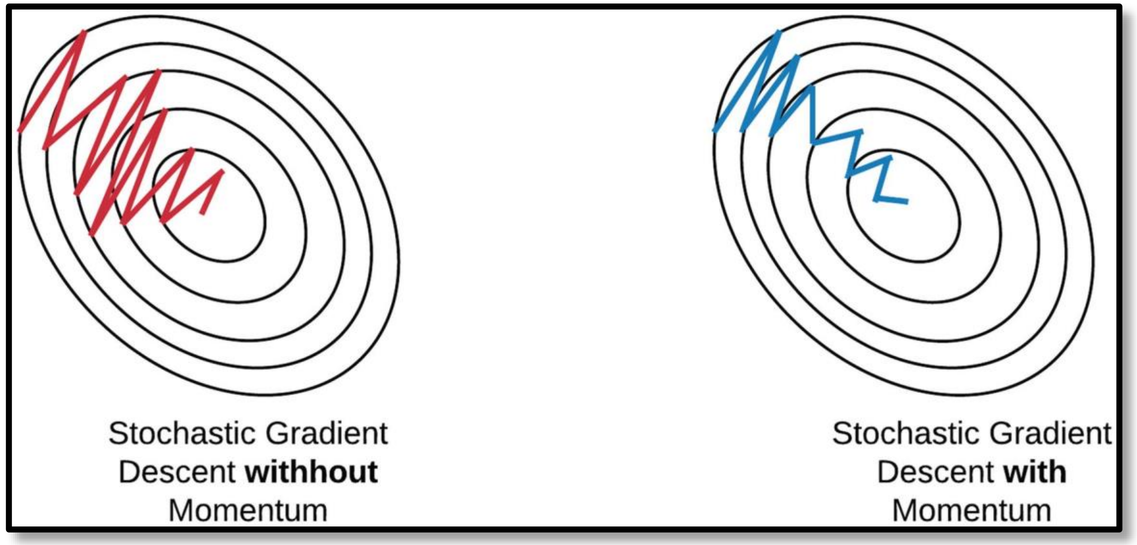
SGD is computationally efficient, especially for large data sets, as it updates parameters frequently and does not require a lot of memory. The frequent updates can also lead to fast convergence to a minimum, facilitating a more rapid learning process for the model.
However, this optimizer has some limitations. The updates can be noisy due to the use of a single data point,- which may lead to fluctuations in the optimization path and potentially prevent convergence to the optimal solution.
Choosing a suitable learning rate in this algorithm is also crucial; if the learning rate is too high, the algorithm may overshoot the minimum, while if it too low, convergence may be slow.


$f(w)$
w^*
w
Too small: converge very slowly
$f(w)$
w^*
w
Too big: overshoot and even diverge

Types of optimizers

2. Stochastic Gradient Descent + Momentum (SGD + Momentum)

Stochastic Gradient Descent with Momentum is a variant of SGD that incorporates a momentum term. Momentum simulates the inertia of a moving object; specifically, it retains the direction of the previous update while using the current gradient to fine-tune the final update direction. This mechanism enhances stability to a certain extent, allowing the model to learn more rapidly and providing the capability to escape local optima.


Stochastic Gradient Descent **withhout** Momentum
Stochastic Gradient Descent **with** Momentum

Types of optimizers

2. Stochastic Gradient Descent + Momentum (SGD + Momentum)

Here is the update rule for SGD with Momentum:

$$v = \beta . v + (1 - \beta). \nabla_{\theta} . L(\theta)$$
$$\theta = \theta - \alpha . v$$

Where:
v: momentum vector
β : momentum hyperparameter (typically set to a value between 0 and 1)

This expression closely resembles the exponentially weighted moving average. At each step, we first calculate the direction in which we need to move by taking the moving average of the previous gradients and the new one. The higher we set β, the more weight we give to the previous gradients.
After calculating the new direction, we multiply it by the learning rate and then adjust the parameters accordingly.

Types of optimizers

2. Stochastic Gradient Descent + Momentum (SGD + Momentum)

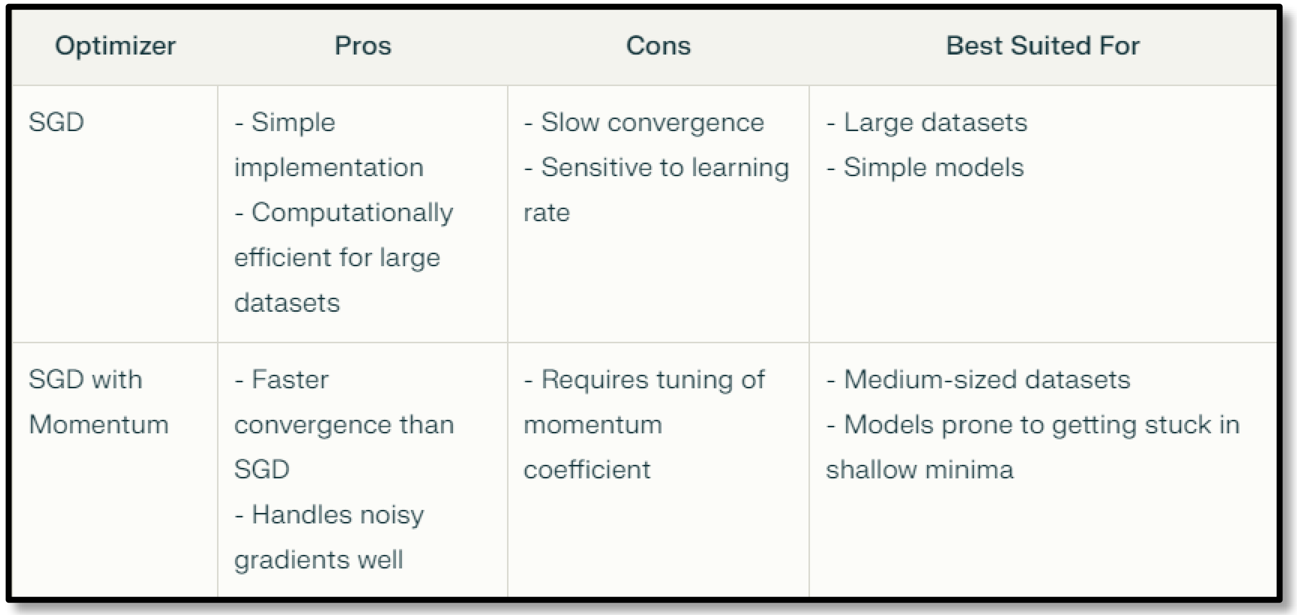
Overall, by incorporating momentum, this algorithm reduces oscillations that can occur when navigating steep or noisy gradients. This leads to a more stable convergence towards the minimum of the loss function.

Additionally, the combination of the current gradient and the accumulated momentum results in larger updates in the correct direction, which can accelerate the convergence process- compared to standard SGD.

Lastly, the momentum term helps the optimization process to overcome local minima and saddle points, as the accumulated velocity can carry the model past these regions.

Choosing the right optimizer

There is no single optimizer that can be universally applied across different machine learning problems. The choice of optimizer depends on various factors, including the size of the data set, the model's architecture, computational resources, and more.
The following table provides a clearer overview of the features of the discussed optimizers.



Optimizer	Pros	Cons	Best Suited For
SGD	- Simple implementation - Computationally efficient for large datasets	- Slow convergence - Sensitive to learning rate	- Large datasets - Simple models
SGD with Momentum	- Faster convergence than SGD - Handles noisy gradients well	- Requires tuning of momentum coefficient	- Medium-sized datasets - Models prone to getting stuck in shallow minima

What are Metrics in neural networks?

Metrics in neural networks are essential tools used to evaluate the performance of machine learning models. They provide quantitative measures that help assess how well a model is performing, both during training and testing.

Metrics are functions that quantify the performance of a model. Unlike loss functions, which guide the training process by indicating how far off a model's predictions are from the-actual outcomes, metrics are primarily used for evaluation purposes and do not directly influence the training process.

Different metrics can indicate different aspects of model performance, helping to choose the best model for a specific task.

Accuracy, Precision, Recall, F1 Score and R2 Score are a few we will further discuss.

Types of Metrics

1. Accuracy

Accuracy is defined as the ratio of correct predictions to total predictions. In other words, accuracy measures how often a machine learning model correctly predicts the outcome.

$$Accuracy = \frac{Correct\ predictions}{All\ predictions}$$

2. Precision

Precision is defined as the ratio of true positive predictions to the sum of the true positives and false positives. In other words, precision measures how often a model correctly predicts the positive class.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Types of Metrics

3. Recall (Sensitivity)

Recall is defined as the ratio of true positive predictions to the sum of true positive and false negatives. In other words, accuracy measures how often a model (true positives) correctly identifies positive instances from all the actual positive samples in the data set.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

4. F1 Score

F1 Score is the harmonic mean of precision and recall. Providing a balance between the two.

$$F1\ Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Types of Metrics

5. R2 Score

R2 Score, also known as the coefficient of determination, is a statistical measure used in machine learning to evaluate the quality of a regression model. It measures how well the model fits the data by assessing the proportion of variance in the dependent variable explained by the independent variables.

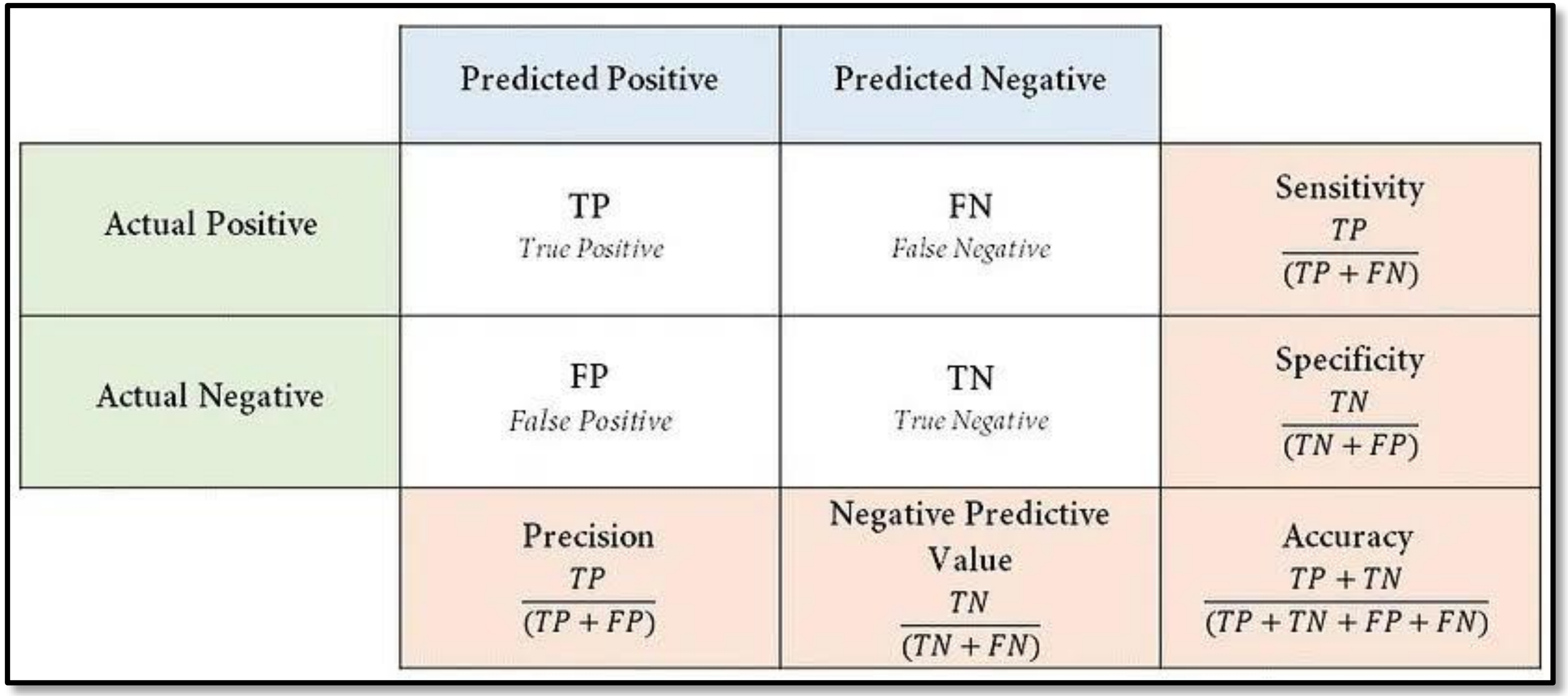
$$R2\ Score = \frac{SSE}{SST}$$

Where:

SSE: the sum of the squared differences between the actual dependent variable values and the predicted values from the regression model

SST: the sum of squared differences between each actual dependent variable value and the mean of all dependent variable values

Types of Metrics



	Predicted Positive	Predicted Negative	
Actual Positive	TP *True Positive*	FN *False Negative*	Sensitivity $\frac{TP}{(TP+FN)}$
Actual Negative	FP *False Positive*	TN *True Negative*	Specificity $\frac{TN}{(TN+FP)}$
	Precision $\frac{TP}{(TP+FP)}$	Negative Predictive Value $\frac{TN}{(TN+FN)}$	Accuracy $\frac{TP+TN}{(TP+TN+FP+FN)}$

Choosing the best metric

The choice of metric depends on the specific problem. For example, in a medical diagnosis problem, a high recall may be more important to avoid missing true positives, while in a spam detection problem, a high precision may be prioritized to avoid flagging too many false positives.