Inductive Bias

Inductive bias is the set of assumptions that a machine learning algorithm makes about the relationship between input variables (features) and output variables (labels) based on the training data. In other words, it's the prior knowledge or beliefs that the algorithm uses to generalize from the training data to new, unseen data.

Inductive bias is necessary in machine learning because it allows the algorithm to make predictions on new data based on what it learned from the training data. Without any prior knowledge, the algorithm would have to start from scratch every time it encountered new data, making it much less efficient and accurate.

Types of Inductive Bias

There are two main types of inductive bias in machine learning: restrictive bias and preferential bias.

Restrictive Bias
Restrictive bias refers to the assumptions that limit the set of functions that the algorithm can learn. For example, a linear regression model assumes that the relationship between the input variables and the output variable is linear. This means that the model can only learn linear functions, and any non-linear relationships between the variables will not be captured.

Another example of restrictive bias is the decision tree algorithm, which assumes that the relationship between the input variables and the output variable can be represented by a tree-like structure. This means that the algorithm can only learn functions that can be represented by a decision tree.

Preferential Bias
Preferential bias refers to the assumptions that make some functions more likely to be learned than others. For example, a neural network with a large number of hidden layers and parameters has a preferential bias towards complex, non-linear functions. This means that the algorithm is more likely to learn complex functions than simple ones.

Another example of preferential bias is the k-nearest neighbors algorithm, which assumes that similar inputs have similar outputs. This means that the algorithm is more likely to predict the same output for inputs that are close together in feature space.

Why is Inductive Bias Important?

Inductive bias is important because it affects the generalization performance of the machine learning algorithm. A machine learning algorithm with a good inductive bias will be able to generalize well to new, unseen data, while an algorithm with a bad inductive bias may overfit to the training data and perform poorly on new data.

For example, if a linear regression model is used to predict housing prices, but the relationship between the input variables and the output variable is non-linear, the model may perform poorly on new data. On the other hand, if a decision tree algorithm is used to predict whether a customer will buy a product, but the relationship between the input variables and the output variable is linear, the model may also perform poorly.

Therefore, it's important to choose a machine learning algorithm with an inductive bias that matches the problem at hand. This will ensure that the algorithm is able to learn the underlying relationship between the input variables and the output variable, and generalize well to new, unseen data.

How to Choose the Right Inductive Bias?

Choosing the right inductive bias depends on the nature of the problem you're trying to solve. Here are some tips to help you choose the right inductive bias:

1. Start with a simple model: Start with a model that has a restrictive bias and can only learn a limited set of functions. This will help you understand the structure of the data and the relationship between the input variables and the output variable.
2. Evaluate the model performance: Evaluate the performance of the model on a validation set to see how well it generalizes to new, unseen data. If the performance is poor, try a different algorithm with a different inductive bias.
3. Consider the complexity of the problem: If the problem is complex and the relationship between the input variables and the output variable is non-linear, consider using a model with a preferential bias towards complex, non-linear functions.
4. Consider the amount of data: If you have a small amount of data, consider using a model with a restrictive bias that can generalize well with limited data.

Bias, Variance and Trade-off

The bias-variance trade-off is a fundamental concept in machine learning that relates to the performance of a model. It refers to the balance between the bias of the model and its variance, with the goal of minimizing the total error.

1. Bias: Bias is the error introduced by approximating a real-world problem with a simplified model. Models with high bias tend to oversimplify the underlying relationships between features and the target variable. They may consistently miss relevant patterns in the data, leading to systematic errors. High bias often results in underfitting, where the model performs poorly on both the training and unseen data.

2. Variance: Variance, on the other hand, refers to the model's sensitivity to small fluctuations in the training data. Models with high variance are overly complex and capture noise in the training data as if it were true signal. As a result, they tend to perform well on the training data but poorly on unseen data, a phenomenon known as overfitting. High variance models fail to generalize well to new data because they are too closely tailored to the idiosyncrasies of the training set.

Finding the right balance between bias and variance is essential for building models that generalize well to unseen data. Ideally, we want to minimize both bias and variance simultaneously, but there is often a trade-off between the two:

- **High Bias, Low Variance**: Simple models with high bias and low variance tend to **underfit** the data. They may not capture the complexity of the underlying relationships, resulting in poor performance on both training and test data.

- **Low Bias, High Variance**: Complex models with low bias and high variance tend to **overfit** the data. They may capture the noise in the training data as if it were signal, leading to excellent performance on the training data but poor performance on test data.

To strike the right balance, various techniques and algorithms are employed in machine learning, including:

- **Regularization**: Techniques like L1 and L2 regularization penalize complex models, helping to reduce overfitting and control variance.

- **Cross-validation**: Cross-validation techniques, such as k-fold cross-validation, can help estimate the model's performance on unseen data, enabling the selection of models with the optimal bias-variance trade-off.

- **Model selection**: Choosing the appropriate model complexity based on the dataset size and complexity can also help in managing the bias-variance trade-off.

- **Ensemble methods**: Ensemble methods, like random forests and boosting algorithms, combine multiple models to reduce variance and improve generalization performance.

In summary, understanding and managing the bias-variance trade-off is crucial for developing machine learning models that generalize well to unseen data while avoiding underfitting and overfitting.

Common Errors and Comprehensive Strategies for Resolution:

Error: Poor model performance on the validation set
Handling: When confronted with subpar performance on the validation set, it’s imperative to conduct a thorough reevaluation of the chosen inductive bias. Consider alternative algorithms that incorporate different biases, and delve into the specifics of their impact on the model’s learning process. Additionally, assess the model’s complexity and be prepared to make necessary adjustments. This might involve fine-tuning hyperparameters, altering the depth of neural networks, or exploring ensemble methods to improve the model’s generalization capabilities.

Error: Overfitting to training data
Handling: Overfitting, a common challenge in machine learning, necessitates a thoughtful approach to ensure model robustness. One effective strategy involves opting for a less complex model, which can mitigate the risk of capturing noise in the training data. Consider revisiting the chosen inductive bias and adjusting it to strike a balance between complexity and generalization. Regularization techniques, such as L1 or L2 regularization, can be employed to penalize overly complex models and prevent them from fitting noise in the data. Additionally, techniques like dropout in neural networks can help prevent overfitting by randomly dropping neurons during training.

Error: Underfitting, poor performance on both training and validation sets
Handling: Underfitting indicates that the model is not sufficiently capturing the underlying patterns in the data, leading to poor performance on both the training and validation sets. To address this, consider increasing the model’s complexity. This might involve adding more layers to a neural network, increasing the polynomial degree in a regression model, or adjusting parameters to allow for more intricate relationships between variables. Alternatively, revisiting the inductive bias and choosing one that aligns more closely with the underlying problem can provide a fresh perspective.

Conclusion

Inductive bias is an important concept in machine learning that refers to the set of assumptions that a machine learning algorithm makes about the relationship between input variables and output variables. Choosing the right inductive bias depends on the nature of the problem you’re trying to solve and the amount of data you have. By understanding inductive bias, you can choose the right machine learning algorithm and improve the generalization performance of your models.

Regression Model

Regression is a statistical method used in finance, investing, and other disciplines that attempts to determine the strength and character of the relationship between one dependent variable (usually denoted by Y) and a series of other variables (known as independent variables).

Regression captures the correlation between variables observed in a data set and quantifies whether those correlations are statistically significant or not.

The two basic types of regression are simple linear regression and multiple linear regression, although there are non-linear regression methods for more complicated data and analysis. Simple linear regression uses one independent variable to explain or predict the outcome of the dependent variable Y, while multiple linear regression uses two or more independent variables to predict the outcome (while holding all others constant).

Machine Learning Regression is a technique for investigating the relationship between independent variables or features and a dependent variable or outcome. It's used as a method for predictive modelling in machine learning, in which an algorithm is used to predict continuous outcomes.

Solving regression problems is one of the most common applications for machine learning models, especially in supervised machine learning. Algorithms are trained to understand the relationship between independent variables and an outcome or dependent variable. The model can then be leveraged to predict the outcome of new and unseen input data, or to fill a gap in missing data.

Regression analysis is an integral part of any forecasting or predictive model, so is a common method found in machine learning powered predictive analytics. Alongside classification, regression is a common use for supervised machine learning models. This approach to training models required labelled input and output training data. Machine learning regression models need to understand the relationship between features and outcome variables, so accurately labelled training data is vital.

Regression is a key element of predictive modelling, so can be found within many different applications of machine learning. Whether powering financial forecasting or predicting healthcare trends, regression analysis can bring organisations key insight for decision-making. It's already used in different sectors to forecast house prices, stock or share prices, or map salary changes.

This guide explores regression in machine learning, including what it is, how it's used, and the different types of regression in machine learning.

What is machine learning regression?

Regression is a method for understanding the relationship between independent variables or features and a dependent variable or outcome. Outcomes can then be predicted once the relationship between independent and dependent variables has been estimated. Regression is a field of study in statistics which forms a key part of forecast models in machine learning. It's used as an approach to predict continuous outcomes in predictive modelling, so has utility in forecasting and predicting outcomes from data. Machine learning regression generally involves plotting a line of best fit through the data points. The distance between each point and the line is minimised to achieve the best fit line.

Alongside classification, regression is one of the main applications of the supervised type of machine learning. Classification is the categorisation of objects based on learned features, whereas regression is the forecasting of continuous outcomes. Both are predictive modelling problems. Supervised machine learning is integral as an approach in both cases, because classification and regression models rely on labelled input and output training data. The features and output of the training data must be labelled so the model can understand the relationship.

Regression analysis is used to understand the relationship between different independent variables and a dependent variable or outcome. Models that are trained to forecast or predict trends and outcomes will be trained using regression techniques. These models will learn the relationship between input and output data from labelled training data. It can then forecast future trends or predict outcomes from unseen input data, or be used to understand gaps in historic data.

As with all supervised machine learning, special care should be taken to ensure the labelled training data is representative of the overall population. If the training data is not representative, the predictive model will be overfit to data that doesn't represent new and unseen data. This will result in inaccurate predictions once the model is deployed. Because regression analysis involves the relationships of features and outcomes, care should be taken to include the right selection of features too.

What are regression models used for?

Machine learning regression models are mainly used in predictive analytics to forecast trends and predict outcomes. Regression models will be trained to understand the relationship between different independent variables and an outcome. The model can therefore understand the many different factors which may lead to a desired outcome. The resulting models can be used in a range of ways and in a variety of settings. Outcomes can be predicted from new and unseen data, market fluctuations can be predicted and accounted for, and campaigns can be tested by tweaking different independent variables.

In practice, a model will be trained on labelled data to understand the relationship between data features and the dependent variable. By estimating this relationship, the model can predict the outcome of new and unseen data. This could be used to predict missing historic data, and estimate future outcomes too. In a sales environment, an organisation could use regression machine learning to predict the next month's sales from a number of factors. In a medical environment, an organisation could forecast health trends in the general population over a period of time.

Supervised machine learning models are generally used for either classification or regression problems. Classification is when a model is trained to categorise an object based on its features. This could include facial recognition software, or to identify a spam email in a firewall. A model will be trained on labelled input and output data to understand the specific features which classify a labelled object. On the other hand, a regression problem is when a model is used to predict continuous outcomes or values. This could be a model that forecasts salary changes, house prices, or retail sales. The model is trained on labelled input and output data to understand the strength of relationships between data features and output.

Regression is used to identify patterns and relationships within a dataset, which can then be applied to new and unseen data. This makes regression a key element of machine learning in finance, and is often leveraged to help forecast portfolio performance or stock costs and trends. Models can be trained to understand the relationship between a variety of diverse features and a desired outcome. In most cases, machine learning regression provides organisations with insight into particular outcomes. But because this approach can influence an organisation's decision-making process, the explainability of machine learning is an important consideration.

Common use for machine learning regression models include:

- Forecasting continuous outcomes like house prices, stock prices, or sales.
- Predicting the success of future retail sales or marketing campaigns to ensure resources are used effectively.
- Predicting customer or user trends, such as on streaming services or e-commerce websites.
- Analysing datasets to establish the relationships between variables and an output.

- Predicting interest rates or stock prices from a variety of factors.
- Creating time series visualisations.

What are the types of regression?

There are a range of different approaches used in machine learning to perform regression. Different popular algorithms are used to achieve machine learning regression. The different techniques may include different numbers of independent variables or process different types of data. Distinct types of machine learning regression models may also assume a different relationship between the independent and dependent variables. For example, linear regression techniques assume that the relationship is linear, so wouldn't be effective with datasets with nonlinear relationships.

Some of the most common regression techniques in machine learning can be grouped into the following types of regression analysis:

- Simple Linear Regression
- Multiple linear regression
- Logistic regression

What is simple linear regression?

Simple Linear regression is a linear regression technique which plots a straight line within data points to minimise error between the line and the data points. It is one of the most simple and basic types of machine learning regression. The relationship between the independent and dependent variables is assumed to be linear in this case. This approach is simple because it is used to explore the relationship between the dependent variable and one independent variable. Outliers may be a common occurrence in simple linear regression because of the straight line of best fit.

What is multiple linear regression?

Multiple linear regression is a technique used when more than one independent variable is used. Polynomial regression is an example of a multiple linear regression technique. It is a type of multiple linear regression, used when there is more than one independent variable. It achieves a better fit in the comparison to simple linear regression when multiple independent variables are involved. The result when plotted on two dimensions would be a curved line fitted to the data points.

What is logistic regression?

Logistic regression is used when the dependent variable can have one of two values, such as true or false, or success or failure. Logistic regression models can be used to predict the probability of a dependent variable occurring. Generally, the output values must be binary. A sigmoid curve can be used to map the relationship between the dependent variable and independent variables.

Example of a regression model:

We can say that age and height can be described using a linear regression model. Since a person's height increases as age increases, they have a linear relationship. Regression models are commonly used as statistical proof of claims regarding everyday facts.

Goals of Regression Analysis

All models define the outcome (Y) as a function of one or more parameters and an independent variable (X) [or several independent variables].

The goal of is to adjust the values of the model's parameters to find the line or curve that comes closest to your data. For example, with linear regression, the goal is to find the best-fit values of the slope and intercept that makes the line come close to the data.

More precisely, the goal of regression is to find the values of the parameters that are most likely to be correct. To do this requires making an assumption about the scatter of data around the curve.

The goals of regression

Scientists use regression with one of three distinct goals:

- To fit a model to your data in order to obtain best-fit values of the parameters, or to compare the fits of alternative models. If this is your goal, you must pick a model (or two alternative models) carefully, and pay attention all the results. The whole point is to obtain best-fit values for the parameters, so you need to understand what those parameters mean scientifically.
- To fit a smooth curve in order to interpolate values from the curve, or perhaps to draw a graph with a smooth curve. If this is your goal, you can assess it purely by looking at the graph of data and curve. There is no need to learn much theory.
- To make predictions.

Overfitting

Overfitting is an undesirable machine learning behavior that occurs when the machine learning model gives accurate predictions for training data but not for new data. When data scientists use machine learning models for making predictions, they first train the model on a known data set. Then, based on this information, the model tries to predict outcomes for new data sets. An overfit model can give inaccurate predictions and cannot perform well for all types of new data.

Why does overfitting occur?

You only get accurate predictions if the machine learning model generalizes to all types of data within its domain. Overfitting occurs when the model cannot generalize and fits too closely to the training dataset instead. Overfitting happens due to several reasons, such as:

- The training data size is too small and does not contain enough data samples to accurately represent all possible input data values.
- The training data contains large amounts of irrelevant information, called noisy data.
- The model trains for too long on a single sample set of data.
- The model complexity is high, so it learns the noise within the training data.

Overfitting examples

Consider a use case where a machine learning model has to analyze photos and identify the ones that contain dogs in them. If the machine learning model was trained on a data set that contained majority photos showing dogs outside in parks, it may may learn to use grass as a feature for classification, and may not recognize a dog inside a room.

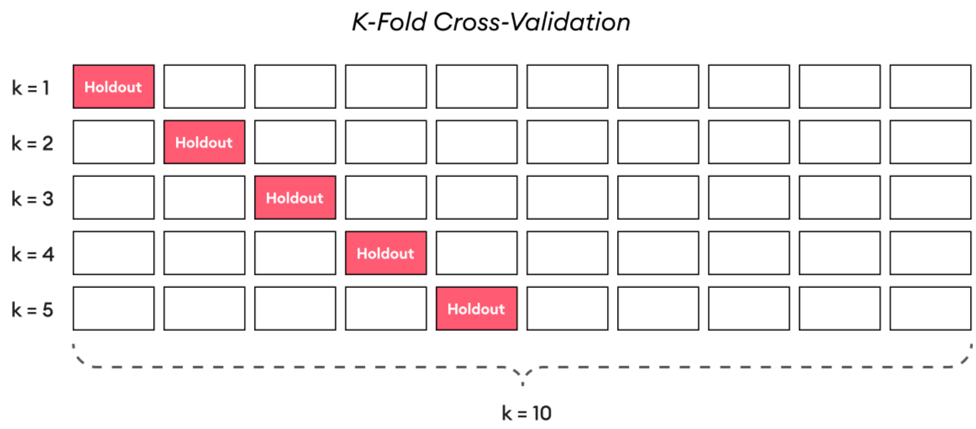
Another overfitting example is a machine learning algorithm that predicts a university student's academic performance and graduation outcome by analyzing several factors like family income, past academic performance, and academic qualifications of parents. However, the test data only includes candidates from a specific gender or ethnic group. In this case, overfitting causes the algorithm's prediction accuracy to drop for candidates with gender or ethnicity outside of the test dataset.

How can you detect overfitting?

The best method to detect overfit models is by testing the machine learning models on more data with with comprehensive representation of possible input data values and types. Typically, part of the training data is used as test data to check for overfitting. A high error rate in the testing data indicates overfitting. One method of testing for overfitting is given below.

In machine learning and AI, overfitting is one of the key problems an engineer may face. Some of the techniques you can use to detect overfitting are as follows:

1) Use a resampling technique to estimate model accuracy. The most popular resampling technique is k-fold cross-validation. It allows you to train and test your model k-times on different subsets of training data and build up an estimate of the performance of a machine learning model on unseen data. The drawback here is that it is time-consuming and cannot be applied to complex models, such as deep neural networks.



K-fold cross-validation

2) Hold back a validation set. Once a model is trained on the training set, you can evaluate it on the validation dataset, then compare the accuracy of the model in the training dataset and the validation dataset. A significant variance in these two results allows assuming that you have an overfitted model.

3) Another way to detect overfitting is by starting with a simplistic model that will serve as a benchmark. With this approach, if you try more complex algorithms, you will have a general understanding of whether the additional complexity for the model is worthwhile, if at all. This principle is known as *Occam's razor* test. This principle suggests that with all else being equal, simpler solutions to problems are preferred over more complex ones (if your model is not getting significantly better after using a much more complex model, it is preferable to use a simpler model).

K-fold cross-validation

Cross-validation is one of the testing methods used in practice. In this method, data scientists divide the training set into K equally sized subsets or sample sets called folds. The training process consists of a series of iterations. During each iteration, the steps are:

1. Keep one subset as the validation data and train the machine learning model on the remaining K-1 subsets.
2. Observe how the model performs on the validation sample.
3. Score model performance based on output data quality.

Iterations repeat until you test the model on every sample set. You then average the scores across all iterations to get the final assessment of the predictive model.

How can you prevent overfitting?

You can prevent overfitting by diversifying and scaling your training data set or using some other data science strategies, like those given below.

Early stopping

Early stopping pauses the training phase before the machine learning model learns the noise in the data. However, getting the timing right is important; else the model will still not give accurate results.

In iterative algorithms, it is possible to measure how the model iteration performance. Up until a certain number of iterations, new iterations improve the model. After that point, however, the model's ability to generalize can deteriorate as it begins to overfit the training data. Early stopping refers to stopping the training process before the learner passes that point.

Pruning
You might identify several features or parameters that impact the final prediction when you build a model. Feature selection—or pruning—identifies the most important features within the training set and eliminates irrelevant ones. For example, to predict if an image is an animal or human, you can look at various input parameters like face shape, ear position, body structure, etc. You may prioritize face shape and ignore the shape of the eyes.

Regularization
Regularization is a collection of training/optimization techniques that seek to reduce overfitting. These methods try to eliminate those factors that do not impact the prediction outcomes by grading features based on importance. For example, mathematical calculations apply a penalty value to features with minimal impact. Consider a statistical model attempting to predict the housing prices of a city in 20 years. Regularization would give a lower penalty value to features like population growth and average annual income but a higher penalty value to the average annual temperature of the city.

Regularization refers to a variety of techniques to push your model to be simpler. The approach you choose will be determined by the model you are training. For example, you can add a penalty parameter for a regression (L1 and L2 regularization), prune a decision tree or use dropout on a neural network.

Ensembling
Ensembling combines predictions from several separate machine learning algorithms. Some models are called weak learners because their results are often inaccurate. Ensemble methods combine all the weak learners to get more accurate results. They use multiple models to analyze sample data and pick the most accurate outcomes. The two main ensemble methods are bagging and boosting. Boosting trains' different machine learning models one after another to get the final result, while bagging trains them in parallel.

Data augmentation

Data augmentation is a machine learning technique that changes the sample data slightly every time the model processes it. You can do this by changing the input data in small ways. When done in moderation, data augmentation makes the training sets appear unique to the model and prevents the model from learning their characteristics. For example, applying transformations such as translation, flipping, and rotation to input images.

In machine learning, data augmentation techniques increase the amount of data by slightly changing previously existing data and adding new data points or by producing synthetic data from a previously existing dataset.

Adding more data

Most of the time, adding more data can help machine learning models detect the "true" pattern of the model, generalize better, and prevent overfitting. However, this is not always the case, as adding more data that is inaccurate or has many missing values can lead to even worse results.

Remove features

You can remove irrelevant aspects from data to improve the model. Many characteristics in a dataset may not contribute much to prediction. Removing non-essential characteristics can enhance accuracy and decrease overfitting.

Ensembling

Ensembling methods merge predictions from numerous different models. These methods not only deal with overfitting but also assist in solving complex machine learning problems (like combining pictures taken from different angles into the overall view of the surroundings). The most popular ensembling methods are boosting and bagging.

- Boosting – In boosting method, you train a large number of weak learners (constrained models) in sequence, and each sequence learns from the mistakes of the previous sequence. Then you combine all weak learners into a single strong learner.
- Bagging is another technique to reduce overfitting. It trains a large number of strong learners (unconstrained models) and then combines them all in order to optimize their predictions.

Underfitting

Underfitting is another type of error that occurs when the model cannot determine a meaningful relationship between the input and output data. You get underfit models if they have not trained for the appropriate length of time on a large number of data points.

How to detect underfitting:

1) **Training and test loss:** If the model is underfitting, the loss for both training and validation will be considerably high. In other words, for an underfitting dataset, the training and the validation error will be high.
2) **Over simplistic prediction graph:** If a graph with the data points and the fitted curve is plotted, and the classifier curve is over simplistic, then, most probably, your model is underfitting. In those cases, a more complex model should be tried out.

How to avoid underfitting

There are several things you can do to prevent underfitting in AI and machine learning models:

1) **Train a more complex model** – Lack of model complexity in terms of data characteristics is the main reason behind underfitting models. For example, you may have data with upwards of 100000 rows and more than 30 parameters. If you train data with the Random Forest model and set max depth (max depth determines the maximum depth of the tree) to a small number (for example, 2), your model will definitely be underfitting. Training a more complex model (in this respect, a model with a higher value of max depth) will help us solve the problem of underfitting.
2) **More time for training** - Early training termination may cause underfitting. As a machine learning engineer, you can increase the number of epochs or increase the duration of training to get better results.

3) **Eliminate noise from data** – Another cause of underfitting is the existence of outliers and incorrect values in the dataset. Data cleaning techniques can help deal with this problem.
4) **Adjust regularization parameters** - the regularization coefficient can cause both overfitting and underfitting models.
5) **Try a different model** – if none of the above-mentioned principles work, you can try a different model (usually, the new model must be more complex by its nature). For example, you can try to replace the linear model with a higher-order polynomial model.

Underfitting vs. Overfitting

Underfit models experience high bias—they give inaccurate results for both the training data and test set. On the other hand, overfit models experience high variance—they give accurate results for the training set but not for the test set. More model training results in less bias but variance can increase. Data scientists aim to find the sweet spot between underfitting and overfitting when fitting a model. A well-fitted model can quickly establish the dominant trend for seen and unseen data sets.