1. Задача обучения c учителем. Регрессия, Классификация.

Задача обучения с учителем (supervised learning) представляет собой один из типов машинного обучения, при котором модель обучается на основе размеченных данных, где для каждого входного примера имеется правильный выходной ответ.

1. Регрессия в задаче обучения с учителем:

* Регрессия - это метод статистического анализа, который используется для изучения взаимосвязи между зависимой переменной (обычно непрерывной) и одной или несколькими независимыми переменными.
* В задаче регрессии с учителем, модель обучается предсказывать непрерывное числовое значение на основе входных данных. Например, предсказание цены дома на основе его характеристик.

1. Классификация в задаче обучения с учителем:

* Классификация - это метод машинного обучения, который используется для разделения данных на категории или классы на основе их признаков.
* В задаче классификации с учителем, модель обучается разделять данные на категории или классы на основе их признаков. Например, определение, является ли письмо спамом или не спамом на основе его содержания.

Задача обучения с учителем является одной из основных парадигм машинного обучения и находит широкое применение в различных областях, таких как обработка естественного языка, компьютерное зрение, медицинская диагностика и финансовый анализ.

2. Как измерить качество в классификации: точность, сбалансированная точность, прецизионность, полнота, f1-скор, ROC-AUC, расширения для многоклассовой классификации.

Для измерения качества в задачах классификации используются различные метрики, которые позволяют оценить производительность модели. Вот некоторые из наиболее распространенных метрик качества в задачах классификации:

1. Точность (Accuracy):

* Точность измеряет долю правильно классифицированных примеров относительно общего числа примеров.

2. Сбалансированная точность (Balanced Accuracy):

* Сбалансированная точность учитывает дисбаланс классов и является средним значением чувствительности (true positive rate) по всем классам.

3. Прецизионность (Precision):

* Прецизионность измеряет долю истинно положительных предсказаний относительно всех положительных предсказаний.

4. Полнота (Recall):

* Полнота измеряет долю истинно положительных предсказаний относительно всех истинно положительных примеров.

5. F1-скор (F1-Score):

* F1-скор объединяет в себе прецизионность и полноту, представляя собой гармоническое среднее этих двух метрик.

6. ROC-AUC (Receiver Operating Characteristic - Area Under the Curve):

* ROC-AUC измеряет площадь под кривой ROC, которая отображает отношение между чувствительностью и специфичностью модели.

7. Метрики для многоклассовой классификации:

* Для многоклассовой классификации также используются метрики, такие как макро-усредненная F1-мера, микро-усредненная F1-мера, макро-усредненная точность и др.

Использование соответствующих метрик зависит от конкретной задачи классификации, а также от особенностей данных. Выбор метрик качества помогает оценить производительность модели и принимать решения о ее улучшении.

3. Как измерить качество в регрессии: MSE, MAE, R2.

1. Среднеквадратичная ошибка (MSE - Mean Squared Error):

* Формула: MSE = (1/n) \* Σ(yi - ŷi)^2
* MSE измеряет среднее значение квадратов разностей между фактическими значениями (yi) и предсказанными значениями (ŷi) модели.
* Чем меньше значение MSE, тем лучше модель предсказывает целевую переменную.

1. Средняя абсолютная ошибка (MAE - Mean Absolute Error):

* Формула: MAE = (1/n) \* Σ|yi - ŷi|
* MAE измеряет среднее значение абсолютных разностей между фактическими значениями (yi) и предсказанными значениями (ŷi) модели.
* MAE также помогает оценить точность модели, причем в отличие от MSE, MAE не учитывает квадрат разности, что делает его более устойчивым к выбросам.

1. Коэффициент детерминации (R2 - R-squared):

* Формула: R2 = 1 - (Σ(yi - ŷi)^2) / (Σ(yi - ȳ)^2)
* R2 измеряет долю дисперсии зависимой переменной, которая объясняется моделью. Значение R2 близкое к 1 указывает на хорошее соответствие модели данным.
* R2 также можно интерпретировать как долю дисперсии целевой переменной, которая объясняется моделью, в процентах.

Эти метрики позволяют оценить качество регрессионной модели и помогают определить, насколько точно модель предсказывает целевую переменную. Когда вы оцениваете качество регрессионной модели, важно учитывать не только значения метрик, но и контекст задачи, чтобы принять информированное решение относительно производительности модели.

4. Оценка максимального правдоподобия, как она связана с регрессией и классификацией

Оценка максимального правдоподобия (Maximum Likelihood Estimation, MLE) является статистическим методом, который используется для оценки параметров модели на основе наблюдаемых данных. Он широко применяется как в задачах регрессии, так и в задачах классификации.

MLE позволяет найти такие значения параметров модели, при которых вероятность получить наблюдаемые данные максимальна. Это позволяет построить модель, которая наилучшим образом соответствует данным и может быть использована для прогнозирования или классификации новых данных.

1. Связь с регрессией:

* В задачах регрессии, MLE может использоваться для оценки параметров модели, таких как коэффициенты регрессии, которые наилучшим образом соответствуют наблюдаемым данным. Например, при построении линейной регрессии, MLE может быть использован для оценки коэффициентов наклона и сдвига.

2. Связь с классификацией:

* В задачах классификации, MLE может использоваться для оценки параметров модели, таких как вероятности принадлежности к классам. Например, в байесовских классификаторах, MLE может быть применен для оценки априорных вероятностей классов и условных вероятностей признаков.

Таким образом, MLE играет важную роль в статистическом моделировании и машинном обучении, обеспечивая метод оценки параметров моделей регрессии и классификации на основе имеющихся данных.

5. Наивный байесовский классификатор, как он работает

Наивный байесовский классификатор - это простой вероятностный классификатор, основанный на теореме Байеса с предположением о независимости между признаками. Он широко используется для задач классификации, таких как определение категории электронного письма (спам или не спам), анализ тональности текста и другие.

Принцип работы наивного байесовского классификатора:

1. Предварительная обработка данных:

* Для начала, данные подготавливаются и представляются в виде признаков (features), которые характеризуют объекты или документы.

2. Оценка вероятностей:

* Для каждого класса и каждого признака оцениваются вероятности появления признака в данном классе. Это может включать оценку априорных вероятностей классов и условных вероятностей признаков для каждого класса.

3. Применение теоремы Байеса:

* После оценки вероятностей, применяется теорема Байеса для вычисления вероятности принадлежности объекта к каждому классу на основе его признаков.

4. Классификация:

* Наивный байесовский классификатор выбирает класс, для которого оценка вероятности принадлежности объекта этому классу максимальна.

Особенности наивного байесовского классификатора:

* Он основан на предположении о независимости между признаками, что делает его "наивным".
* Хотя это предположение редко выполняется в реальных данных, наивный байесовский классификатор часто демонстрирует хорошую производительность на практике.
* Он быстро обучается и эффективно работает с большими объемами данных.

Таким образом, наивный байесовский классификатор представляет собой простой и эффективный метод классификации, который находит широкое применение в различных областях анализа данных и машинного обучения.

6. Классификатор ближайших соседей, как он работает

Классификатор ближайших соседей (K-Nearest Neighbors, KNN) - это метод машинного обучения, который используется для классификации объектов на основе их сходства с ближайшими соседями в пространстве признаков. Он также может применяться для задач регрессии.

Принцип работы классификатора ближайших соседей:

1. Хранение обучающих данных:

* В начале, KNN хранит обучающий набор данных, состоящий из объектов и соответствующих им классов (в случае классификации) или значений целевой переменной (в случае регрессии).

2. Выбор числа соседей (K):

* Затем выбирается количество ближайших соседей (K), которые будут использоваться для классификации или регрессии новых объектов.

3. Расчет расстояний:

* При поступлении нового объекта, KNN вычисляет расстояние от этого объекта до каждого объекта обучающего набора. Расстояние может быть измерено, например, с использованием евклидова расстояния или других метрик расстояния.

4. Определение класса или значения целевой переменной:

* KNN определяет K ближайших соседей нового объекта и определяет его класс (в случае классификации) или значение целевой переменной (в случае регрессии) на основе классов или значений целевой переменной ближайших соседей.

5. Прогноз:

* Новый объект относится к классу, который является наиболее представленным среди его K ближайших соседей.

Особенности классификатора ближайших соседей:

* KNN является ленивым алгоритмом (lazy learning), что означает, что он не требует обучения во время фазы обучения, а просто запоминает обучающий набор данных.
* Он основан на принципе "подобное соседство". Это означает, что объекты, находящиеся близко друг к другу в пространстве признаков, склонны иметь схожие характеристики и, следовательно, принадлежать к одному классу (в случае классификации) или иметь похожие значения целевой переменной (в случае регрессии).
* Классификатор ближайших соседей не создает явной модели, а вместо этого осуществляет классификацию или регрессию на основе сходства между объектами в пространстве признаков. Этот метод является примером ленивого (lazy) обучения, так как он откладывает процесс обучения до момента поступления новых данных для классификации или регрессии.

7. Линейная регрессия. Формулировка задачи для случая функции потерь MSE. Аналитическое решение. Теорема Гаусса-Маркова. Градиентный подход в линейной регрессии.

Линейная регрессия - это метод машинного обучения, который используется для построения модели, предсказывающей зависимую переменную на основе одной или нескольких независимых переменных. Давайте рассмотрим формулировку задачи для случая функции потерь MSE (среднеквадратичная ошибка), аналитическое решение, теорему Гаусса-Маркова и градиентный подход в линейной регрессии.

1. Формулировка задачи для случая функции потерь MSE:

* Пусть у нас есть набор обучающих данных, состоящий из пар (xi, yi), где xi - вектор признаков, а yi - соответствующее значение зависимой переменной. Мы хотим найти параметры модели (веса) w, чтобы минимизировать среднеквадратичную ошибку (MSE) между предсказанными значениями и фактическими значениями: MSE = (1/n) \* Σ(yi - ŷi)^2, где ŷi = w^T xi.

2. Аналитическое решение:

* Для случая функции потерь MSE, аналитическое решение сводится к нахождению оптимальных параметров модели w. Это может быть сделано путем минимизации функции потерь MSE по параметрам w с использованием методов оптимизации, таких как метод наименьших квадратов (МНК).

3. Теорема Гаусса-Маркова:

* Теорема Гаусса-Маркова утверждает, что в линейной регрессии оценки параметров модели, полученные методом наименьших квадратов, являются лучшими линейными несмещенными оценками (BLUE) с минимальной дисперсией, если выполняются определенные предположения о модели и ошибках.

4. Градиентный подход в линейной регрессии:

* Градиентный подход в линейной регрессии используется для нахождения оптимальных параметров модели путем итеративной оптимизации функции потерь с использованием градиентного спуска. Градиентный спуск позволяет найти минимум функции потерь путем изменения параметров модели в направлении, противоположном градиенту функции потерь.

Эти концепции и методы являются основой линейной регрессии и используются для построения моделей и нахождения оптимальных параметров в задачах регрессии.

8. Регуляризация в линейных моделях: L1 (\Leftrightarrow) L2, их свойства. Вероятностная интерпретация.

Регуляризация в линейных моделях - это метод добавления дополнительной информации в процесс обучения с целью предотвращения переобучения и улучшения обобщающей способности модели. Два наиболее распространенных метода регуляризации в линейных моделях - это L1-регуляризация (лассо) и L2-регуляризация (гребневая регрессия).

1. L1-регуляризация (лассо):

* L1-регуляризация добавляет штраф к функции потерь, который пропорционален сумме абсолютных значений весов модели.
* Свойства: L1-регуляризация обеспечивает разреженность весов, что позволяет отбирать наиболее важные признаки и упрощает интерпретацию модели.

2. L2-регуляризация (гребневая регрессия):

* L2-регуляризация добавляет штраф к функции потерь, который пропорционален сумме квадратов весов модели.
* Свойства: L2-регуляризация способствует сглаживанию весов и предотвращает переобучение путем уменьшения их значений.

Вероятностная интерпретация:

* Вероятностная интерпретация регуляризации заключается в том, что она может быть рассмотрена как априорное распределение параметров модели в байесовском подходе. L1-регуляризация соответствует априорному распределению Лапласа, а L2-регуляризация соответствует априорному распределению Гаусса.

Оба метода регуляризации имеют свои преимущества и могут быть эффективно использованы для улучшения производительности линейных моделей в различных задачах.

9. Логистическая регрессия. Эквивалентность подходов MLE и минимизации логистических потерь.

Логистическая регрессия - это метод машинного обучения, который используется для моделирования вероятности принадлежности к классу в задачах бинарной классификации. Этот метод часто используется для оценки вероятности принадлежности объекта к одному из двух классов.

Эквивалентность подходов MLE и минимизации логистических потерь:

* Подход MLE (максимального правдоподобия) и минимизация логистических потерь эквивалентны в контексте логистической регрессии.

1. MLE:

* В подходе MLE модель логистической регрессии оценивает параметры таким образом, чтобы максимизировать правдоподобие данных. Это означает, что мы ищем параметры, которые делают наши данные наиболее вероятными.

2. Минимизация логистических потерь:

* Минимизация логистических потерь также является методом обучения модели логистической регрессии. Здесь мы находим параметры модели, которые минимизируют ошибку классификации и уменьшают логистические потери.

Эти два подхода эквивалентны в том смысле, что минимизация логистических потерь приводит к оценке параметров, которая максимизирует правдоподобие данных, и наоборот. Таким образом, обучение модели логистической регрессии с использованием MLE и минимизации логистических потерь приводит к одним и тем же результатам.

10. Многоклассовая классификация. Один-против-одного, один-против-всех, их свойства.

В задаче многоклассовой классификации существует несколько подходов к обучению моделей, включая подход "один-против-одного" (one-vs-one) и подход "один-против-всех" (one-vs-all).

1. Подход "Один-против-одного" (One-vs-One):

* В этом подходе для каждой пары классов создается бинарный классификатор. Например, если у нас есть K классов, то мы создаем K(K-1)/2 классификаторов. Каждый классификатор обучается на данных, где один класс считается положительным, а другой - отрицательным.
* Свойства: Этот метод может быть вычислительно дорогостоящим из-за большого количества классификаторов, но он может быть эффективен в случае, если бинарные классификаторы легко обучаются на небольших наборах данных.

2. Подход "Один-против-всех" (One-vs-All):

* В этом подходе для каждого класса создается бинарный классификатор, который разделяет данный класс от всех остальных. Например, если у нас есть K классов, то мы создаем K бинарных классификаторов. Каждый классификатор обучается на данных, где один класс считается положительным, а все остальные - отрицательными.
* Свойства: Этот метод более вычислительно эффективен, чем "один-против-одного", потому что требуется обучить только K бинарных классификаторов. Однако он может быть менее точным в случае, если классы сложно разделимы или несбалансированы.

Каждый из этих подходов имеет свои преимущества и недостатки, и выбор между ними зависит от конкретной задачи и особенностей данных.

11. Метод опорных векторов. Задача оптимизации для SVM. Трюк с ядром. Свойства ядра.

Метод опорных векторов (SVM) - это метод машинного обучения, который используется для задач классификации и регрессии. Он работает путем поиска оптимальной гиперплоскости, которая разделяет данные на классы с максимальным зазором между ними.

Задача оптимизации для SVM:

* Для задачи классификации, основной задачей оптимизации SVM является поиск оптимальной разделяющей гиперплоскости, которая максимизирует зазор между классами и минимизирует ошибки классификации. Это формулируется как задача квадратичного программирования.

Трюк с ядром:

* Трюк с ядром (kernel trick) позволяет обобщить SVM для работы с нелинейными разделяющими поверхностями. Он заключается в том, что вместо явного преобразования признакового пространства, мы можем использовать ядровую функцию, которая вычисляет скалярное произведение между двумя точками в пространстве более высокой размерности.

Свойства ядра:

* Ядро в методе опорных векторов должно удовлетворять условиям положительной полуопределенности и симметричности.
* Хорошее ядро должно быть способно выражать сложные нелинейные зависимости между данными, что позволяет SVM строить сложные разделяющие поверхности.
* Популярными ядрами являются полиномиальное ядро, гауссовское (RBF) ядро и сигмоидное ядро.

Использование ядер позволяет SVM эффективно работать с нелинейными данными и строить сложные разделяющие поверхности, что делает его мощным инструментом для решения различных задач машинного обучения.

12. Анализ главных компонент. Связь с SVD. Теорема Эккарта-Янга. Как применять PCA на практике.

Анализ главных компонент (PCA) - это метод уменьшения размерности данных, который используется для проекции многомерных данных на пространство меньшей размерности, сохраняя при этом как можно больше информации. Вот некоторые ключевые аспекты анализа главных компонент:

1. Связь с SVD:

* Метод PCA тесно связан с сингулярным разложением (SVD) матрицы данных. При применении PCA к матрице данных происходит вычисление SVD, и главные компоненты выбираются на основе собственных векторов и собственных значений, полученных из SVD.

2. Теорема Эккарта-Янга:

* Теорема Эккарта-Янга утверждает, что главные компоненты, полученные методом PCA, являются ортогональными друг другу. Это означает, что они представляют независимые направления в пространстве признаков.

3. Применение PCA на практике:

* Определение количества главных компонент: Для начала определяется количество главных компонент, которые необходимо оставить для сохранения определенного процента дисперсии в данных.
* Центрирование данных: Данные центрируются путем вычитания среднего значения каждого признака.
* Вычисление главных компонент: Главные компоненты вычисляются путем вычисления собственных векторов и собственных значений ковариационной матрицы данных или путем применения SVD.
* Проекция данных: Данные проецируются на пространство главных компонент для уменьшения размерности.

PCA является мощным инструментом для обработки и визуализации многомерных данных, а также для уменьшения размерности перед применением других методов машинного обучения.

13. Этапы обучения, валидации и тестирования модели. Проблема переобучения, способы её обнаружения.

Этапы обучения, валидации и тестирования модели - это важные шаги в процессе построения и оценки моделей машинного обучения. Вот как обычно проходят эти этапы:

1. Обучение модели:

* На этом этапе модель обучается на обучающем наборе данных. Это включает в себя подгонку параметров модели таким образом, чтобы минимизировать функцию потерь.

2. Валидация модели:

* После обучения модели часто проводится валидация на валидационном наборе данных. Это позволяет оценить обобщающую способность модели и настроить гиперпараметры.

3. Тестирование модели:

* Наконец, модель оценивается на тестовом наборе данных, который не использовался ни в обучении, ни в валидации. Это позволяет оценить производительность модели на новых данных.

Проблема переобучения:

* Переобучение возникает, когда модель слишком точно подстраивается под обучающие данные, что приводит к плохой обобщающей способности на новых данных.

Способы обнаружения переобучения:

* Использование валидационного набора данных для оценки производительности модели.
* Использование регуляризации для уменьшения переобучения.
* Анализ кривых обучения и валидации для оценки разрыва между производительностью на обучении и валидации.
* Применение методов ансамблей, таких как бутстраппинг и скользящий контроль, для улучшения оценки производительности модели.

14. Стратегии валидации. Кросс-валидация. Утечки данных.

Стратегии валидации используются для оценки производительности модели на данных, которые не участвовали в обучении. Некоторые из наиболее распространенных стратегий валидации включают в себя:

1. K-кратная кросс-валидация (K-fold Cross-Validation):

* Данные разбиваются на K частей. Модель обучается K раз, каждый раз используя K-1 частей для обучения и оставшуюся часть для валидации. Затем производится усреднение результатов.

2. Стратегическая разбивка (Stratified Fold):

* Этот метод разбивает данные на K частей таким образом, чтобы каждая часть содержала пропорциональное количество примеров из каждого класса. Это особенно полезно в случае несбалансированных классов.

3. Групповая разбивка (Group Fold):

* В случае, когда данные содержат группировку (например, данные по пользователям или местоположению), групповая разбивка используется для того, чтобы убедиться, что объекты из одной группы не попадают одновременно в обучающий и тестовый наборы данных.

4. Разбивка по времени (Time Fold):

* Этот метод используется в случаях, когда данные упорядочены по времени. Он обеспечивает то, что обучающий набор содержит данные, предшествующие данным в тестовом наборе, чтобы модель могла быть оценена на последующих данных.

Утечки данных (Data Leakage) возникают, когда информация из тестового набора данных непреднамеренно попадает в процесс обучения модели. Это может привести к завышенной оценке производительности модели. Некоторые примеры утечек данных включают использование информации из будущего, утечку целевой переменной и др. Для предотвращения утечек данных необходимо тщательно обрабатывать данные и использовать подходящие стратегии валидации.

15. Компромисс смещения-дисперсии.

Компромисс смещения-дисперсии - это концепция в машинном обучении, которая описывает баланс между смещением и дисперсией модели.

* Смещение (bias) отражает ошибку, которая возникает из-за упрощенных предположений в модели. Модель с высоким смещением склонна к недостаточной подгонке данных и может проявлять систематическую ошибку при прогнозировании.
* Дисперсия (variance) отражает чувствительность модели к изменениям в обучающих данных. Модель с высокой дисперсией склонна к переобучению и может проявлять излишнюю чувствительность к шуму в данных.

Компромисс смещения-дисперсии заключается в том, что при улучшении смещения (увеличении сложности модели) дисперсия может увеличиться, и наоборот. Цель состоит в том, чтобы найти такой баланс между смещением и дисперсией, который обеспечит наилучшую обобщающую способность модели на новых данных.

Это значит, что модель должна быть достаточно сложной, чтобы улавливать сложные зависимости в данных, но при этом не слишком сложной, чтобы избежать переобучения. На практике это может достигаться путем выбора подходящей сложности модели, использования регуляризации, ансамблевых методов или тщательной настройки гиперпараметров.

16. Процедура построения дерева решений.

Процедура построения дерева решений включает следующие шаги:

1. Выбор корневого признака:

* На этом этапе выбирается признак, который лучше всего разделяет данные на подгруппы. Это может быть сделано с использованием различных критериев, таких как прирост информации (information gain) или коэффициент Джини (Gini impurity).

2. Разделение данных:

* Данные разделяются на подгруппы на основе значения выбранного признака.

3. Рекурсивное построение дерева:

* Процедура разделения повторяется для каждой подгруппы, пока не будет выполнен критерий останова, такой как достижение максимальной глубины дерева или минимального количества объектов в узле.

4. Присвоение меток:

* В листовых узлах дерева присваиваются метки классов или значения целевой переменной.

5. Обрезка дерева (опционально):

* В некоторых случаях может потребоваться обрезка дерева для предотвращения переобучения.

6. Оценка производительности:

* После построения дерева решений оценивается его производительность на тестовом наборе данных.

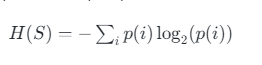
Дерево решений строится таким образом, чтобы оптимально разделять данные на подгруппы и прогнозировать целевую переменную на основе значений признаков.

17. Критерии информации. Критерии энтропии, неопределенности Джини.

Критерии информации, такие как энтропийный критерий и критерий неопределенности Джини, используются для выбора оптимального разделения в узлах дерева решений. Вот как они работают:

1. Энтропийный критерий:

* Энтропия измеряет степень хаоса или неопределенности в системе. В контексте деревьев решений, энтропийный критерий используется для измерения того, насколько хорошо разделение в узле уменьшает неопределенность в данных. Чем ниже энтропия, тем более однородными являются данные после разделения.
* 



2. Критерий неопределенности Джини:

* Критерий неопределенности Джини измеряет вероятность неправильной классификации случайно выбранного элемента, если он был случайно помечен в соответствии с распределением меток классов в узле. Цель состоит в том, чтобы минимизировать неопределенность Джини путем выбора оптимального разделения, которое уменьшает эту вероятность.
* 



Оба критерия используются для выбора оптимального разделения в узлах дерева решений, и выбор между ними зависит от конкретной задачи и типа данных.

18. Ансамблевые методы. Бутстрап. Бэггинг.

Ансамблевые методы в машинном обучении - это методы, которые объединяют прогнозы нескольких моделей, чтобы получить более точный и устойчивый результат, чем у каждой отдельной модели. Два популярных ансамблевых метода - это бутстрап и бэггинг.

1. Бутстрап:

* Бутстрап - это метод генерации случайных подвыборок из обучающего набора данных путем выбора объектов с возвращением. Это позволяет создавать несколько выборок, которые могут перекрываться. Бутстрап используется в бэггинге для создания набора подвыборок, на которых будут обучаться базовые модели.

2. Бэггинг (Bootstrap Aggregating):

* Бэггинг - это метод ансамблирования, в котором несколько моделей обучаются на различных подмножествах данных, созданных с помощью бутстрапа, а затем их прогнозы усредняются для получения окончательного прогноза. Это позволяет уменьшить дисперсию модели и предотвратить переобучение.

Таким образом, бутстрап используется для создания подмножеств данных, а бэггинг использует эти подмножества для обучения нескольких моделей и комбинирования их прогнозов для улучшения обобщающей способности модели.

19. Случайный лес, метод случайных подпространств.

Метода случайного леса (Random Forest). Этот метод является ансамблевым методом машинного обучения, что означает, что он объединяет прогнозы нескольких моделей, чтобы получить более точный и устойчивый результат.

1. Построение деревьев решений: Сначала для каждого дерева в случайном лесу случайным образом выбирается подмножество данных из обучающего набора. Это означает, что каждое дерево строится на основе разных данных.

2. Случайный выбор признаков: Кроме того, при построении каждого дерева происходит случайный выбор подмножества признаков из всего набора признаков. Это помогает уменьшить корреляцию между деревьями.

3. Голосование: После построения всех деревьев в случайном лесу, они используются для прогнозирования. Затем результаты голосуются, и принимается решение на основе большинства голосов.

Методу случайных подпространств (Random Subspace). Этот метод является вариацией метода случайного леса и фокусируется на использовании случайного подмножества признаков из исходного набора данных.

1. Случайный выбор признаков: В отличие от обычного случайного леса, где каждое дерево использует случайное подмножество признаков из всего набора данных, метод случайных подпространств выбирает случайное подмножество признаков для каждого дерева из всего набора признаков.

2. Уменьшение корреляции: Это позволяет уменьшить корреляцию между деревьями, поскольку каждое дерево использует разные признаки для построения, что может привести к улучшению обобщающей способности модели.

Оба этих метода используют случайность при построении деревьев, что помогает им избежать переобучения и повысить точность прогнозирования.

20. Бустинг и градиентный бустинг. Основная идея, производная градиента.

Бустинг (Boosting) - это метод машинного обучения, который объединяет слабые модели (например, деревья решений) в сильную модель. Основная идея бустинга заключается в последовательном обучении моделей, где каждая новая модель исправляет ошибки предыдущей модели.

Градиентный бустинг (Gradient Boosting) - это конкретный метод бустинга, который использует градиентный спуск для минимизации ошибки модели. Основная идея градиентного бустинга заключается в том, что каждая новая модель обучается на остатках (разнице между предсказанными значениями и реальными значениями) предыдущих моделей.

Производная градиента (Gradient Descent) - это метод оптимизации, который используется для нахождения минимума функции. В контексте градиентного бустинга, производная градиента используется для нахождения направления, в котором нужно изменить модель, чтобы уменьшить ошибку предсказания.

Теперь давайте рассмотрим процесс градиентного бустинга более подробно:

1. Инициализация: Начинаем с построения базовой модели (например, дерева решений) и вычисления начальных предсказаний.

2. Расчет остатков: Вычисляем остатки (разницу между предсказанными значениями и реальными значениями) для каждого образца в обучающем наборе.

3. Обучение новой модели: Строим новую модель, которая будет предсказывать остатки предыдущей модели. Это делается путем минимизации ошибки с использованием градиентного спуска.

4. Обновление предсказаний: Обновляем предсказания путем добавления предсказаний новой модели к предыдущим предсказаниям.

5. Итерации: Повторяем шаги 2-4 до тех пор, пока не достигнем определенного критерия останова (например, достижения определенного числа итераций или улучшения качества модели).

Таким образом, градиентный бустинг использует производные градиента для обучения последовательной серии моделей, каждая из которых исправляет ошибки предыдущей модели, что позволяет создать сильную ансамблевую модель.

Производная градиента функции f(x) вычисляется как:

∇f(x) = (∂f/∂x₁, ∂f/∂x₂, ..., ∂f/∂xn)

где ∂f/∂xi представляет собой частную производную функции f по переменной xi. Это вектор, который указывает направление наибольшего возрастания функции. Для минимизации функции мы двигаемся в направлении, противоположном градиенту, с использованием шага градиентного спуска, чтобы постепенно приблизиться к минимуму функции.

Формула для обновления параметров в градиентном спуске выглядит следующим образом:

x\_new = x\_old - α \* ∇f(x\_old)

где x\_old - это текущие параметры, α (альфа) - это скорость обучения (learning rate), и ∇f(x\_old) - это градиент функции f в точке x\_old. Мы вычитаем градиент, умноженный на скорость обучения, из текущих параметров, чтобы обновить их в направлении минимума функции.

21. Матричное исчисление и производные матриц. Как получить производную матричного/скалярного произведения, напр.: $ \mathbf{a}^T \mathbf{x}$,

Если у нас есть матрица-строка размерности 1 x n и столбцовый **x** размерности n x 1, то матричное произведение представляет собой скаляр.

Теперь, чтобы получить производную матричного произведения  по вектору **x**, мы можем использовать правило дифференцирования произведения для скалярной функции f(**x**) и вектора **x**, которое выглядит следующим образом:



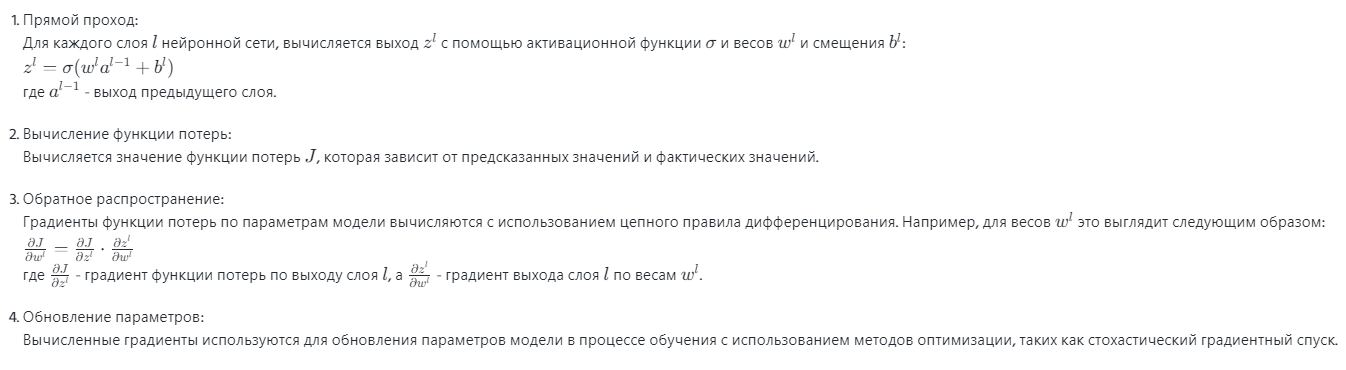
Таким образом, производная матричного произведения по вектору **x** равна просто вектору **a**.

22. Обратное распространение ошибки.

Обратное распространение ошибки (backpropagation) - это метод обучения нейронных сетей, который используется для вычисления градиентов функции потерь по параметрам модели. Он является ключевым компонентом обучения нейронных сетей с использованием градиентного спуска.

Процесс обратного распространения ошибки начинается с прямого прохода, во время которого входные данные проходят через нейронную сеть, и каждый нейрон вычисляет свой выход. Затем вычисляется значение функции потерь, которое показывает, насколько предсказания модели отличаются от фактических значений.

После этого происходит обратное распространение, во время которого градиент функции потерь вычисляется с помощью цепного правила дифференцирования. Этот градиент затем передается назад через сеть, и каждый нейрон вычисляет свой вклад в общую ошибку.

Таким образом, обратное распространение ошибки позволяет вычислить градиенты всех параметров модели относительно функции потерь. Эти градиенты используются для обновления параметров модели в процессе обучения с использованием методов оптимизации, таких как стохастический градиентный спуск.

23. Концепция нейронной сети. Полносвязный слой (FC).

Нейронная сеть - это вычислительная модель, инспирированная работой человеческого мозга, которая используется для решения задач машинного обучения. Она состоит из нейронов, объединенных в слои, и использует методы обучения, такие как обратное распространение ошибки, для настройки параметров модели.

Полносвязный слой (Fully Connected Layer, FC) - это тип слоя в нейронной сети, в котором каждый нейрон связан с каждым нейроном предыдущего и последующего слоя. В полносвязном слое каждый входной признак (или активация предыдущего слоя) соединен с каждым нейроном текущего слоя.

Математически, операции в полносвязном слое можно описать следующим образом:

Пусть **x** - это вектор входных признаков размерности **n**, а **W** - это матрица весов размерности **m** x **n**, где **m** - количество нейронов в текущем слое, а **b** - это вектор смещений размерности **m**. Тогда выход **z** полносвязного слоя можно выразить следующим образом:

**z = Wx + b**

где **z** - это выходной вектор активаций полносвязного слоя.

После этого активации могут быть применены к выходу полносвязного слоя, например, с помощью функции активации, такой как ReLU (Rectified Linear Unit) или сигмоидальная функция.

Полносвязные слои широко используются в глубоком обучении и являются основным строительным блоком многих типов нейронных сетей, таких как многослойные персептроны и глубокие нейронные сети.

24. Логистическая регрессия как простая нейронная сеть.

Логистическая регрессия может быть рассмотрена как простая нейронная сеть с одним слоем, который состоит из одного нейрона. Этот нейрон принимает входные признаки, вычисляет их взвешенную сумму, добавляет смещение и применяет функцию активации (в данном случае - сигмоидальную функцию) к результату.

Математически, операции в логистической регрессии можно описать следующим образом:

Пусть **x** - это вектор входных признаков размерности **n**, **W** - это вектор весов размерности **n**, а **b** - это смещение (bias). Тогда выход **z** логистической регрессии можно выразить следующим образом:

**z = Wx + b**

где **z** - это выходная взвешенная сумма.

Затем к выходу **z** применяется сигмоидальная функция, которая выражается следующим образом:



Это значение представляет вероятность принадлежности к классу 1. В задачах бинарной классификации, логистическая регрессия использует эту вероятность для принятия решения о принадлежности объекта к классу.

Таким образом, логистическая регрессия может быть рассмотрена как простая нейронная сеть с одним нейроном, который принимает входные признаки, вычисляет их взвешенную сумму, добавляет смещение и применяет функцию активации для получения предсказания.

25. Функции потерь для НС в задаче классификации.

Для задачи классификации в нейронных сетях используются различные функции потерь (loss functions), которые помогают оценивать расхождение между предсказанными значениями и фактическими метками классов. Вот некоторые из наиболее распространенных функций потерь для задач классификации:

1. Кросс-энтропия (Cross-Entropy Loss):

* Используется для бинарной и многоклассовой классификации.
* Функция потерь выражается как отрицательная логарифмическая вероятность правильного класса.
* Применяется в сочетании с функцией активации Softmax.

2. Логистическая функция потерь (Logistic Loss):

* Также известная как бинарная кросс-энтропия.
* Используется в задачах бинарной классификации.
* Оценивает расхождение между предсказанной вероятностью и фактической меткой класса.

3. Hinge Loss:

* Обычно используется в методе опорных векторов (SVM) и некоторых архитектурах нейронных сетей.
* Применяется в задачах классификации с опорными векторами (SVM) и в некоторых архитектурах нейронных сетей.
* Подходит для многоклассовой классификации.

4. Категориальная кросс-энтропия (Categorical Cross-Entropy Loss):

* Используется в многоклассовой классификации с несколькими классами.
* Оценивает расхождение между предсказанными вероятностями классов и фактическими метками классов.

Выбор функции потерь зависит от конкретной задачи классификации и особенностей данных. Кроме того, для различных архитектур нейронных сетей могут использоваться разные функции потерь.

26. Функции активации, их влияние на сеть, вычислительная сложность. Функции Softmax и LogSoftmax, численная стабильность.

Функции активации в нейронных сетях играют важную роль, поскольку они определяют, как нейроны реагируют на входные данные и как передают свои выходы следующим слоям. Вот некоторые из наиболее распространенных функций активации и их влияние на сеть:

1. Сигмоидальная функция (Sigmoid):

* Преобразует входные значения в диапазон от 0 до 1.
* Часто используется в выходном слое для бинарной классификации.
* Может приводить к проблеме затухания градиента (vanishing gradient problem) при обратном распространении ошибки.

2. Гиперболический тангенс (Tanh):

* Преобразует входные значения в диапазон от -1 до 1.
* Обычно используется в скрытых слоях нейронных сетей.
* Помогает справиться с проблемой центрирования данных и может ускорить сходимость обучения.

3. ReLU (Rectified Linear Unit):

* Преобразует отрицательные значения в 0, оставляя положительные значения без изменения.
* Позволяет избежать проблемы затухания градиента и ускоряет обучение нейронной сети.
* Одна из самых популярных функций активации в глубоком обучении.

4. Softmax:

* Преобразует выходные значения в вектор вероятностей, сумма которых равна 1.
* Часто используется в выходном слое для многоклассовой классификации.
* Позволяет интерпретировать выходы как вероятности принадлежности к различным классам.

Функция Softmax вычислительно неустойчива при работе с большими числами, поскольку экспоненциальная функция может привести к переполнению (overflow) или потере точности при работе с числами большого порядка. Для решения этой проблемы используют функцию LogSoftmax, которая вычисляет логарифмы вероятностей перед применением экспоненты, что повышает численную стабильность.

27. Методы оптимизации в глубоком обучении. Градиентный спуск, SGD, его улучшения: Momentum, RMSProp, Adam.

Методы оптимизации играют важную роль в глубоком обучении, поскольку они позволяют эффективно настраивать параметры нейронных сетей. Вот некоторые из наиболее распространенных методов оптимизации и их особенности:

1. Градиентный спуск (Gradient Descent):

* Один из основных методов оптимизации для обучения нейронных сетей.
* Использует градиент функции потерь по параметрам модели для обновления параметров в направлении уменьшения функции потерь.

2. Стохастический градиентный спуск (Stochastic Gradient Descent, SGD):

* Вместо вычисления градиента по всему набору данных, SGD вычисляет градиент и обновляет параметры для каждого отдельного примера из обучающего набора.
* Это позволяет быстрее сходиться и уменьшает вычислительную сложность.

3. Метод Momentum:

* Добавляет инерцию к обновлению параметров, учитывая предыдущие изменения.
* Помогает ускорить сходимость и уменьшить осцилляции в процессе обучения.

4. RMSProp (Root Mean Square Propagation):

* Адаптивный метод оптимизации, который адаптирует скорость обучения для каждого параметра на основе истории градиентов.
* Помогает справиться с проблемой адаптивной скорости обучения в SGD.

5. Adam (Adaptive Moment Estimation):

* Комбинация метода Momentum и RMSProp.
* Использует оценки первого и второго моментов градиента для адаптивного вычисления скорости обучения для каждого параметра.

Каждый из этих методов имеет свои преимущества и недостатки, и выбор конкретного метода оптимизации зависит от конкретной задачи и архитектуры нейронной сети.

28. Регуляризация в глубоком обучении: Dropout, Batch Normalization. Различия в стадиях обучения и оценки.

Регуляризация в глубоком обучении - это методы, которые помогают предотвратить переобучение и улучшить обобщающую способность модели. Два распространенных метода регуляризации в глубоком обучении - это Dropout и Batch Normalization.

1. Dropout:

* Dropout - это метод регуляризации, который случайным образом "выключает" (обнуляет) нейроны во время обучения с определенной вероятностью.
* Это помогает предотвратить коадаптацию нейронов и улучшить обобщающую способность модели.
* Во время оценки (тестирования), все нейроны остаются активными, но их выходы масштабируются на коэффициент (1 - вероятность отключения) для компенсации.

2. Batch Normalization:

* Batch Normalization - это метод, который нормализует активации каждого слоя путем вычитания среднего и деления на стандартное отклонение по мини-пакету обучающих данных.
* Это помогает ускорить обучение и улучшить стабильность градиентного спуска.
* Во время оценки (тестирования), параметры нормализации, вычисленные во время обучения, используются для нормализации активаций.

Различия в стадиях обучения и оценки заключаются в том, что во время обучения эти методы могут применяться с некоторыми модификациями (например, для учета случайности и изменения статистики по мини-пакетам), тогда как во время оценки (тестирования) они используются без изменений, используя параметры, вычисленные во время обучения.

29. Классическая рекурсивная НС. Обратное распространение через RNN. Проблема затухающего градиента.

Классическая рекуррентная нейронная сеть (RNN) - это тип нейронной сети, который может обрабатывать последовательности входных данных переменной длины. Она состоит из повторяющихся блоков (ячеек), которые позволяют сети сохранять информацию о предыдущих состояниях и использовать ее для обработки последующих входов.

Обратное распространение через RNN (RNN Backpropagation) - это метод обучения RNN с использованием обратного распространения ошибки. Он позволяет вычислять градиенты функции потерь по параметрам модели и обновлять эти параметры в процессе обучения.

Проблема затухающего градиента (Vanishing Gradient Problem) возникает при обратном распространении через RNN из-за длительных зависимостей во времени. При обновлении градиентов через множество временных шагов градиенты могут становиться очень маленькими, что затрудняет обучение модели.

Эта проблема особенно заметна в традиционных RNN, таких как простая RNN, из-за их способности сохранять информацию только на несколько временных шагов. В результате, градиенты могут затухать или взрываться, что затрудняет обучение модели на длинных последовательностях.

Для решения проблемы затухающего градиента были разработаны более продвинутые архитектуры RNN, такие как Long Short-Term Memory (LSTM) и Gated Recurrent Unit (GRU), которые способны эффективно передавать и сохранять информацию на длительные временные интервалы, облегчая обучение на длинных последовательностях.

30. LSTM/GRU, концепция памяти, идеи вентилей (gates).

LSTM (Long Short-Term Memory) и GRU (Gated Recurrent Unit) - это две распространенные архитектуры рекуррентных нейронных сетей, которые были разработаны для решения проблемы затухающего градиента и обработки длительных зависимостей во времени. Обе архитектуры используют идею вентилей (gates), чтобы контролировать поток информации внутри сети.

Концепция памяти в LSTM и GRU представляет собой способность модели сохранять информацию в течение длительного периода времени. Это достигается благодаря внутренним механизмам, которые позволяют модели сохранять, обновлять или забывать информацию в зависимости от входных данных.

Идея вентилей (gates) в LSTM и GRU позволяет модели контролировать поток информации, проходящий через нейронную сеть. В каждом блоке LSTM и GRU присутствуют вентили, которые решают, какую информацию следует передавать дальше, а какую - забывать или обновлять.

В LSTM присутствуют три типа вентилей:

1. Вентиль забывания (Forget Gate): Решает, какую информацию следует забыть из предыдущего состояния.

2. Входной вентиль (Input Gate): Решает, какую информацию следует добавить в память.

3. Выходной вентиль (Output Gate): Решает, какую информацию следует использовать для выхода из блока LSTM.

В GRU также присутствует два типа вентилей:

1. Вентиль обновления (Update Gate): Решает, какую часть предыдущего состояния следует обновить.

2. Вентиль перезагрузки (Reset Gate): Решает, какую часть предыдущего состояния следует забыть.

Использование вентилей позволяет LSTM и GRU контролировать поток информации и управлять памятью модели, что делает их более эффективными в обучении на длинных последовательностях и решении задач, связанных с длительными зависимостями во времени.

31. Операция свертки. Свёрточный слой, обратное распространение через него. Гиперпараметры свёрток. Сравнение 1x1 свёрток и полносвязных слоёв. Пулинг: max/average.

Операция свертки (convolution) - это основной элемент сверточных нейронных сетей, который позволяет извлекать признаки из входных данных. Свертка применяется к входным данным с использованием фильтра (ядра), которое скользит по входным данным и вычисляет скалярное произведение между фильтром и соответствующей областью входных данных.

Сверточный слой (convolutional layer) - это слой нейронной сети, который выполняет операцию свертки над входными данными. В результате применения сверточного слоя извлекаются пространственные признаки из входных данных.

Обратное распространение через сверточный слой использует цепное правило дифференцирования для вычисления градиентов функции потерь по параметрам сверточного слоя. Эти градиенты затем передаются назад через сеть для обновления параметров в процессе обучения.

Гиперпараметры сверток включают в себя размер ядра (фильтра), шаг (шаг скольжения фильтра), количество фильтров и тип дополнения (padding).

1x1 свертки (1x1 convolutions) часто используются для уменьшения размерности и увеличения глубины признакового пространства. Они могут выполнять операцию линейного преобразования на каждом пикселе входных данных, что делает их аналогичными полносвязным слоям. Однако 1x1 свертки имеют преимущество в том, что они могут сохранять пространственную структуру данных.

Сравнение 1x1 сверток и полносвязных слоев:

* 1x1 свертки имеют меньшее количество параметров и могут сохранять пространственную структуру данных, в отличие от полносвязных слоев.
* Полносвязные слои имеют большее количество параметров и требуют плоского (flatten) входа, теряя пространственную информацию.

Пулинг (pooling) - это операция, которая используется для уменьшения размерности признакового пространства путем агрегации информации. Существуют два основных типа пулинга: max pooling и average pooling. Max pooling выбирает максимальное значение из заданной области, тогда как average pooling вычисляет среднее значение.

Пулинг помогает уменьшить количество параметров и вычислений, улучшает инвариантность к масштабу и поворотам, а также уменьшает чувствительность к небольшим сдвигам входных данных.

32. Основные идеи AlexNet, VGG, Inception (GoogLeNet), архитектуры ResNet.

AlexNet, VGG, Inception (GoogLeNet) и ResNet - это известные архитектуры глубоких сверточных нейронных сетей, которые внесли значительный вклад в развитие области глубокого обучения. Вот краткий обзор основных идей каждой из этих архитектур:

1. AlexNet:

* AlexNet была одной из первых глубоких сверточных нейронных сетей, которая добилась большого успеха в соревнованиях по классификации изображений ImageNet.
* Использует сверточные слои с функцией активации ReLU, слои пулинга и методы регуляризации, такие как Dropout.
* Основная идея - параллельное использование нескольких графических процессоров для ускорения обучения.

2. VGG (Visual Geometry Group):

* VGG представляет собой семейство моделей сверточных нейронных сетей с глубокой архитектурой, состоящей из множества сверточных слоев и слоев пулинга.
* Основная идея - использование множества последовательных сверточных слоев с небольшим размером фильтра (3x3) и пулинга для извлечения более сложных признаков.

3. Inception (GoogLeNet):

* Inception (GoogLeNet) представляет собой нейронную сеть, которая использует модуль "Inception" - блок, состоящий из нескольких параллельных операций свертки и пулинга.
* Основная идея - эффективное использование ресурсов путем параллельного извлечения признаков различных масштабов и размеров ядер.

4. ResNet (Residual Network):

* ResNet представляет собой архитектуру нейронной сети, которая использует блоки "residual" для обучения более глубоких сетей.
* Основная идея - использование skip-connections (пропускающих соединений), которые позволяют градиентам легче распространяться через сеть, устраняя проблему затухания градиента при обучении очень глубоких сетей.

33. Опционально: Геометрические методы в машинном обучении. Задача кластеризации. IsoMap, LLE, DBSCAN, k-средние, t-SNE.

Геометрические методы в машинном обучении - это класс методов, которые используют геометрические свойства данных для анализа, извлечения признаков и построения моделей. Вот краткий обзор некоторых геометрических методов в машинном обучении:

1. Задача кластеризации:

* Кластеризация - это задача разделения множества объектов на группы (кластеры) таким образом, чтобы объекты внутри одного кластера были более похожи друг на друга, чем на объекты из других кластеров.

2. IsoMap (Isometric Mapping):

* IsoMap - это метод нелинейного снижения размерности, который сохраняет геодезические расстояния между всеми парами точек в исходном пространстве.
* Этот метод особенно полезен для сохранения глобальной структуры данных.

3. LLE (Locally Linear Embedding):

* LLE - это метод снижения размерности, который пытается сохранить локальные линейные связи между соседними точками в исходном пространстве.
* Этот метод хорошо работает для сохранения локальной структуры данных.

4. DBSCAN (Density-Based Spatial Clustering of Applications with Noise):

* DBSCAN - это алгоритм кластеризации, который основан на плотности данных.
* Он способен выделять кластеры произвольной формы и обнаруживать выбросы в данных.

5. k-средние (k-means):

* k-средние - это простой и популярный алгоритм кластеризации, который разделяет данные на k кластеров, минимизируя сумму квадратичных расстояний между объектами и их центроидами.

6. t-SNE (t-distributed Stochastic Neighbor Embedding):

* t-SNE - это метод снижения размерности, который позволяет визуализировать высокоразмерные данные в двумерном или трехмерном пространстве таким образом, чтобы близкие объекты в исходном пространстве оставались близкими, а далекие объекты - далекими.

Эти методы представляют различные подходы к анализу и обработке данных с использованием геометрических свойств и структуры данных.