1. Задача обучения с учителем.

Задача обучения с учителем (supervised learning) представляет собой процесс обучения модели на основе размеченных данных, где каждый обучающий пример представляет собой пару "входные данные - целевая метка". Цель состоит в том, чтобы модель могла научиться предсказывать целевую метку для новых, ранее не виденных данных.

Основные задачи обучения с учителем включают в себя классификацию и регрессию. В задаче классификации модель предсказывает категориальную метку или класс (например, "собака" или "кошка"), в то время как в задаче регрессии модель предсказывает непрерывное числовое значение (например, цену дома).

Примеры обучающих данных используются для обучения модели, а затем модель проверяется на тестовых данных, чтобы оценить ее способность обобщения на новые данные. Множество алгоритмов машинного обучения, таких как линейная регрессия, метод опорных векторов (SVM), случайный лес и нейронные сети, применяются для задач обучения с учителем.

Обучение с учителем широко применяется в прогнозировании, классификации, распознавании образов, обработке естественного языка и других областях, где требуется предсказание на основе имеющихся данных.

1. Задача обучения без учителя. Предоставьте по крайней мере две проблемы-примера.

Задача обучения без учителя (unsupervised learning) заключается в извлечении структуры из данных без наличия размеченных примеров или меток. В этом случае алгоритм обучения работает с неразмеченными данными и стремится найти скрытые закономерности, шаблоны или группировки в данных.

Основные задачи обучения без учителя включают в себя кластеризацию, понижение размерности, поиск ассоциативных правил, и поиск аномалий. Например, в кластеризации данные группируются в кластеры на основе их сходства, в понижении размерности данные проецируются на пространство меньшей размерности для упрощения анализа, в поиске ассоциативных правил находятся связи между различными признаками, а в задаче поиска аномалий выявляются необычные или выбивающиеся объекты.

Проблема 1: Группировка новостных статей

Описание: У вас есть набор новостных статей на различные темы, такие как спорт, политика, наука и т. д. Ваша задача - автоматически сгруппировать эти статьи на основе их содержания, чтобы выявить общие темы и тренды.

Проблема 2: Кластеризация покупателей

Описание: Имеется набор данных о покупках в розничном магазине, включающий информацию о покупках различных товаров. Задача состоит в том, чтобы сегментировать покупателей на группы на основе их покупательского поведения и предпочтений. Это позволит магазину создавать персонализированные маркетинговые стратегии для каждой группы покупателей.

3. Что такое i.i.d. данные?

I.I.D. означает "независимые и одинаково распределенные" (independent and identically distributed). Этот термин используется в статистике и машинном обучении для описания свойств набора данных.

Набор данных считается независимым и одинаково распределенным, если каждое наблюдение в наборе данных независимо от других наблюдений и имеет одно и то же вероятностное распределение. Это означает, что вероятность появления любого конкретного наблюдения не зависит от других наблюдений, и все наблюдения имеют одинаковое распределение вероятностей.

В контексте машинного обучения, предположение о независимости и одинаковом распределении данных важно для многих моделей, таких как линейная регрессия, метод опорных векторов и другие. Это предположение позволяет модели корректно обобщать на новые данные и делать статистически обоснованные выводы на основе обучающего набора данных.

В реальных сценариях данные не всегда идеально соответствуют этому предположению, но концепция i.i.d. является важной основой для понимания многих методов анализа данных и машинного обучения.

4. Как работает наивный байесовский классификатор? Почему он наивный?

Наивный байесовский классификатор - это вероятностный классификатор, основанный на применении теоремы Байеса с наивным предположением о независимости между признаками. Он считается "наивным" из-за этого предположения о независимости, которое, хотя и упрощает модель, часто не соответствует реальным данным.

Принцип работы наивного байесовского классификатора:

1. Подсчитывается вероятность каждого класса на основе обучающих данных.
2. Для данного наблюдения вычисляются условные вероятности принадлежности к каждому классу на основе значений его признаков.
3. Применяется теорема Байеса для вычисления апостериорной вероятности принадлежности наблюдения к каждому классу.
4. Наблюдение относится к классу с наибольшей апостериорной вероятностью.

Почему он "наивный":

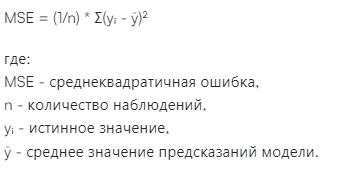
Наивное предположение о независимости между признаками означает, что каждый признак влияет на класс независимо от других признаков. Это упрощение, которое часто не соответствует реальным данным, поскольку признаки могут быть взаимосвязаны. Однако даже с этим упрощением, наивный байесовский классификатор может быть эффективным для многих задач классификации, особенно в случаях больших объемов данных.

Несмотря на свою "наивность", наивный байесовский классификатор широко используется в задачах классификации текста, фильтрации спама, анализе тональности и других областях из-за своей простоты и хороших практических результатов.

5. Модель линейной регрессии для минимизации проблемы MSE. Запишите формулу и производную функции потерь по весам.

Для модели линейной регрессии минимизация проблемы среднеквадратичной ошибки (MSE) осуществляется путем нахождения оптимальных значений весов (коэффициентов) модели. Формула для MSE и производная функции потерь по весам выглядят следующим образом:

Формула для MSE:



Производная функции потерь по весам:

∂MSE/∂wⱼ = -(2/n) \* Σxⱼ(yᵢ - (w₀ + w₁x₁ + w₂x₂ + ... + wₚxₚ))

Где:

∂MSE/∂wⱼ - производная MSE по весу wⱼ

n - количество наблюдений в обучающем наборе

xⱼ - значение признака (предиктора) j-го наблюдения

yᵢ - фактическое значение целевой переменной для i-го наблюдения

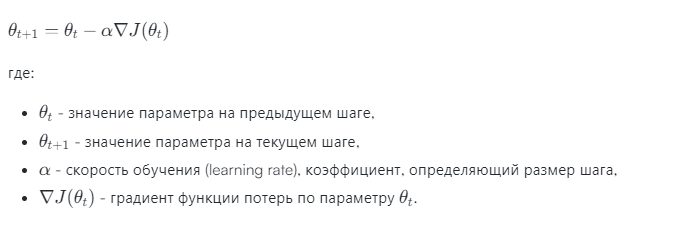
w₀, w₁, w₂, ..., wₚ - веса модели

x₁, x₂, ..., xₚ - значения признаков (предикторов) для i-го наблюдения

Эти формулы используются для обновления весов в процессе обучения модели линейной регрессии с целью минимизации среднеквадратичной ошибки.

6. Запишите шаг градиентного спуска. Как его адаптировать для больших наборов данных?

Шаг градиентного спуска включает обновление весов модели в направлении, противоположном градиенту функции потерь. Этот процесс повторяется до сходимости к оптимальным весам. Формула для обновления весов на каждом шаге выглядит следующим образом:



Адаптация для больших наборов данных включает использование методов ускорения и оптимизации градиентного спуска. Некоторые из них включают:

1. Стохастический градиентный спуск (SGD): Вместо вычисления градиента на всем наборе данных, SGD использует только одно случайное наблюдение для вычисления градиента на каждом шаге. Это позволяет обработать большие наборы данных более эффективно.
2. Мини-пакетный градиентный спуск (Mini-batch GD): Этот метод использует случайно выбранные подмножества данных (мини-пакеты) для вычисления градиента. Он является компромиссом между SGD и обычным градиентным спуском.
3. Адаптивные методы скорости обучения: Методы, такие как Adam, RMSprop и Adagrad, адаптируют скорость обучения на основе истории градиентов, что делает их более эффективными для различных типов данных и функций потерь.

Эти методы помогают справиться с проблемой больших наборов данных, ускоряя сходимость и повышая эффективность градиентного спуска.

7. Что такое правдоподобие? Где обычно используется Максимальное Правдоподобие (MLE)?

Правдоподобие - это статистическая мера, используемая для оценки того, насколько хорошо параметры модели соответствуют наблюдаемым данным. В более простых терминах, правдоподобие измеряет вероятность получить наблюдаемые данные при заданных параметрах модели.

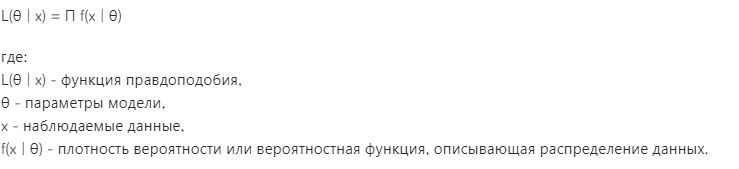
Максимальное правдоподобие (MLE) - это метод оценки параметров модели, который использует максимизацию функции правдоподобия. Суть MLE заключается в том, что мы стараемся найти такие значения параметров модели, при которых вероятность получить наблюдаемые данные максимальна.

MLE обычно используется в статистике, эконометрике и машинном обучении для:

Оценки параметров распределений вероятностей: Например, MLE может использоваться для оценки параметров нормального распределения (среднего и дисперсии) на основе наблюдаемых данных.

Оценки параметров моделей: В машинном обучении MLE часто используется для оценки параметров моделей, таких как линейная регрессия, логистическая регрессия, нейронные сети и другие.

Применение MLE позволяет получить оценки параметров модели, которые наилучшим образом соответствуют наблюдаемым данным и позволяют сделать наиболее вероятные предсказания.



Оценка максимального правдоподобия заключается в выборе таких значений параметров θ, которые максимизируют функцию правдоподобия L(θ | x) для заданных наблюдаемых данных x.

8. Что такое кросс-валидация? Как количество фолдов влияет на валидацию?

Кросс-валидация (cross-validation) - это статистический метод оценки производительности модели и ее способности обобщения на новые данные. Он используется для оценки того, насколько хорошо модель будет работать на независимых данных.

Основная идея кросс-валидации заключается в разделении доступных данных на несколько частей (фолдов), обучении модели на одной части данных и проверке ее производительности на оставшихся частях. Этот процесс повторяется несколько раз, и результаты усредняются для получения общей оценки производительности модели.

Количество фолдов влияет на валидацию следующим образом:

1. Меньшее количество фолдов: Меньшее количество фолдов может привести к более высокому разбросу (вариативности) оценок производительности модели, так как каждая оценка будет более чувствительна к конкретному разделению данных.
2. Большее количество фолдов: Большее количество фолдов обычно приводит к более стабильным оценкам производительности модели, так как каждая оценка будет усреднена из большего количества тестовых наборов данных.

На практике обычно используются 5 или 10 фолдов для кросс-валидации, но выбор количества фолдов зависит от размера доступных данных, вычислительных ресурсов и желаемой точности оценки производительности модели.

\*\*Написать на типы фолдов

9. Что такое переобучение и недообучение? Как их обнаружить?

Переобучение (overfitting) и недообучение (underfitting) - это две проблемы, связанные с обучением моделей машинного обучения.

1. Переобучение (overfitting):

Переобучение происходит, когда модель слишком точно подстраивается под обучающие данные и теряет способность обобщения на новые, ранее не виденные данные. Это может произойти, если модель слишком сложная или если она обучается на избыточном количестве признаков. Признаки шума или случайной вариации могут быть интерпретированы моделью как существенные закономерности.

1. Недообучение (underfitting):

Недообучение происходит, когда модель слишком проста или недостаточно адаптирована к обучающим данным, что приводит к плохим результатам на как обучающих, так и тестовых данных. Модель не способна извлечь закономерности из данных и делает слишком простые предсказания.

Обнаружение переобучения и недообучения:

1. Кривые обучения: График зависимости функции потерь от количества эпох обучения. Переобучение обычно характеризуется увеличением ошибки на тестовых данных после определенной точки, в то время как недообучение проявляется высокой ошибкой как на обучающих, так и на тестовых данных.
2. Кривые валидации: График зависимости оценки производительности на валидационном наборе от сложности модели. Переобучение может быть обнаружено, если оценка производительности на валидационном наборе начинает ухудшаться после достижения определенного уровня сложности модели.
3. Анализ ошибок: Изучение типов ошибок, которые совершает модель на обучающих и тестовых данных, может помочь выявить признаки переобучения или недообучения.
4. Перекрестная проверка: При использовании кросс-валидации можно оценить, как модель обобщается на различных подмножествах данных.

10. В чем разница между параметрами и гиперпараметрами? Приведите пример для линейных моделей/деревьев решений.

Разница между параметрами и гиперпараметрами заключается в их ролях в модели машинного обучения:

1.Параметры модели:

* Параметры модели - это переменные, которые модель настраивает в процессе обучения на основе обучающих данных.
* Значения параметров модели определяются в ходе обучения и используются для делания предсказаний.
* Пример для линейной модели: Веса (коэффициенты) в линейной модели, такие как коэффициенты наклона и сдвиг (bias).

2.Гиперпараметры модели:

* Гиперпараметры модели - это переменные, которые не могут быть настроены в процессе обучения и должны быть установлены до начала обучения.
* Гиперпараметры влияют на поведение самого алгоритма обучения и настройку параметров модели.
* Пример для деревьев решений: Глубина дерева, критерий разделения, минимальное количество объектов в листе и т.д.

Пример использования гиперпараметров:

При обучении дерева решений, мы можем выбирать гиперпараметры, такие как максимальная глубина дерева, минимальное количество объектов в листе, критерий разделения и т.д. Эти гиперпараметры влияют на структуру и поведение самого дерева решений, но не являются частью данных, на которых происходит обучение.

Итак, гиперпараметры определяют "как" модель должна быть обучена, в то время как параметры определяют "что" модель учится из обучающих данных.

11. Что такое регуляризация? В чем разница между L1 и L2 регуляризацией в линейных моделях? Это единственный способ ограничить решение?

Регуляризация - это метод добавления дополнительной информации к модели с целью предотвращения переобучения и улучшения обобщающей способности модели. Она часто используется в контексте линейных моделей для управления сложностью модели и снижения переобучения.

Разница между L1 и L2 регуляризацией в линейных моделях:

1. L1 Регуляризация (Lasso):

* L1 регуляризация добавляет штраф к функции потерь, пропорциональный сумме абсолютных значений весов модели.
* L1 регуляризация может приводить к разреженным весам, поскольку она способствует обнулению весов для некоторых признаков.
* L1 регуляризация полезна для отбора признаков и уменьшения размерности модели.

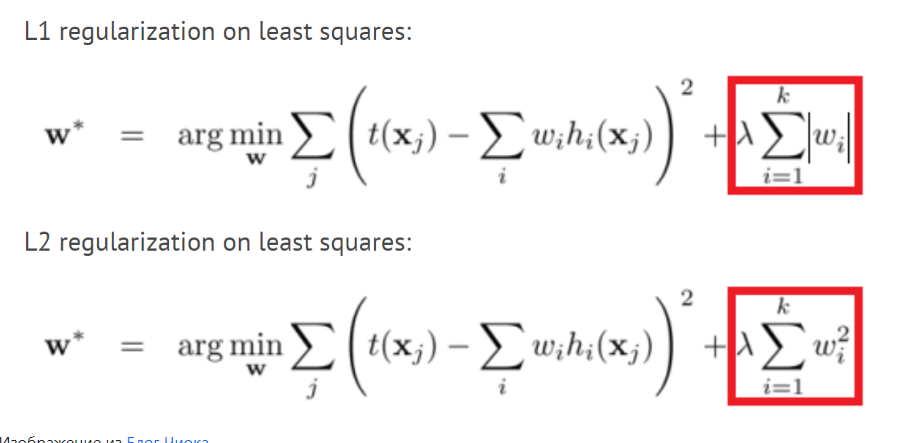
2. L2 Регуляризация (Ridge):

* L2 регуляризация добавляет штраф к функции потерь, пропорциональный сумме квадратов весов модели.
* L2 регуляризация обычно не обнуляет веса, но уменьшает их значения, способствуя стабильности и уменьшению чувствительности модели к выбросам.

Это единственный способ ограничить решение?

Нет, регуляризация не единственный способ ограничить решение. Другие методы ограничения решения включают добавление ограничений на значения весов (например, ограничение norm(w) <= C), использование методов отбора признаков, а также использование алгоритмов оптимизации с встроенными механизмами контроля сложности модели.

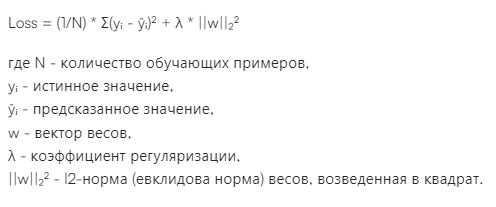
В целом, регуляризация - это мощный метод управления сложностью модели и предотвращения переобучения, особенно в контексте линейных моделей.



Красный прямоугольник сверху вниз представляет регуляризацию L1 и регуляризацию L2.

12. Регуляризует ли L2 регуляризация член смещения (w\_0 или b)? Почему?

Да, L2 регуляризация (Ridge) также регуляризует член смещения (w₀ или b) в линейных моделях. Это происходит потому, что в формуле для L2 регуляризации включается сумма квадратов всех весов, включая вес смещения.



13.Почему хорошая идея нормализовать данные перед применением линейной модели?

Таким образом, нормализация данных перед применением линейной модели помогает улучшить процесс обучения, повысить качество модели и сделать её результаты более интерпретируемыми.

Нормализация данных в контексте применения линейной модели означает приведение значений признаков к одному масштабу или диапазону. Это важный шаг предобработки данных, который позволяет улучшить процесс обучения линейной модели и повысить ее производительность.

Одним из распространенных методов нормализации данных является стандартизация, при которой каждый признак центрируется и масштабируется так, чтобы иметь среднее значение 0 и стандартное отклонение 1. Это делается путем вычитания среднего значения признака из каждого значения признака и деления на стандартное отклонение.

Другим методом нормализации данных является мин-макс нормализация, при которой значения признаков масштабируются так, чтобы они находились в определенном диапазоне, обычно от 0 до 1 или от -1 до 1.

Нормализация данных перед применением линейной модели имеет несколько преимуществ:

1. Повышение стабильности и скорости обучения: Нормализация данных помогает сделать значения признаков сопоставимыми по их масштабу, что может ускорить сходимость алгоритма обучения и сделать его более стабильным.
2. Предотвращение проблемы масштабирования: Без нормализации признаки с разными масштабами могут оказать неравномерное влияние на модель, что может привести к нежелательным результатам.
3. Улучшение интерпретируемости модели: Нормализация данных позволяет лучше интерпретировать веса признаков в линейной модели, так как они будут отражать важность признаков в одинаковом масштабе.
4. Снижение вероятности переобучения: Нормализация данных может помочь уменьшить вероятность переобучения модели, особенно если признаки имеют большой разброс в значениях.

14. Сформулируйте задачу линейной классификации. Что такое маржа?

Задача линейной классификации заключается в том, чтобы построить модель, которая может разделить данные на различные классы с помощью линейной гиперплоскости. Для этого используется линейный классификатор, который стремится найти оптимальное разделение между классами на основе обучающих данных.

Формулировка задачи линейной классификации:

Дано множество обучающих примеров, каждый из которых представляет собой пару (x, y), где x - вектор признакового описания объекта, а y - метка класса (0 или 1, -1 или 1 и т.д.). Задача состоит в том, чтобы найти параметры модели (веса и смещение), которые определяют разделяющую гиперплоскость таким образом, чтобы минимизировать ошибку классификации на обучающих данных.

Маржа (margin) в контексте линейной классификации - это расстояние от разделяющей гиперплоскости до ближайшего обучающего примера. Маржа является важным показателем качества разделения классов, поскольку большая маржа указывает на более уверенное и надежное разделение классов, тогда как меньшая маржа может указывать на близость к переобучению. В методе опорных векторов (SVM) маржа играет ключевую роль, поскольку целью SVM является максимизация маржи при поиске оптимальной разделяющей гиперплоскости.

15. Что такое точность и полнота? Как их использовать для оценки качества модели?

Точность (precision) и полнота (recall) - это две метрики, используемые для оценки качества модели классификации.

1. Точность (Precision): Точность представляет собой долю истинно положительных предсказаний среди всех положительных предсказаний модели. Формально, точность вычисляется как отношение числа истинно положительных предсказаний к сумме числа истинно положительных и ложноположительных предсказаний.

Формула для точности:



1. Полнота (Recall): Полнота представляет собой долю истинно положительных предсказаний среди всех реальных положительных примеров. Формально, полнота вычисляется как отношение числа истинно положительных предсказаний к сумме числа истинно положительных и ложноотрицательных предсказаний.

Формула для полноты:



Как использовать точность и полноту для оценки качества модели:

* Точность и полнота являются взаимообратными метриками: увеличение точности обычно сопровождается уменьшением полноты и наоборот.
* Для балансированной оценки качества модели можно использовать F1-меру, которая является гармоническим средним между точностью и полнотой:
* В зависимости от конкретной задачи, можно выбирать метрику в зависимости от важности минимизации ложноположительных или ложноотрицательных предсказаний.
* В некоторых случаях, например, при работе с дисбалансированными классами, может быть полезно использовать ROC-кривую и AUC-ROC метрику для оценки качества модели.
* Использование точности, полноты и связанных с ними метрик позволяет получить более глубокое понимание производительности модели классификации и её способности правильно классифицировать объекты.

16. Предположим, что набор данных для бинарной классификации несбалансирован, так что 95% данных относятся к первому классу. Как скорректировать меры качества классификации для работы с такими данными? Почему?

Когда набор данных для бинарной классификации несбалансирован и один из классов преобладает (например, 95% данных относятся к первому классу), меры качества классификации, такие как точность, полнота и F1-мера, могут дать искаженное представление о качестве модели. В таких случаях можно использовать скорректированные меры качества, такие как следующие:

1. Матрица ошибок (Confusion Matrix): Использование матрицы ошибок позволяет явно увидеть количество и тип ошибок, совершаемых моделью, и может быть полезным при анализе несбалансированных данных.
2. ROC-кривая и AUC-ROC: ROC-кривая (кривая работы приемника) и площадь под ROC-кривой (AUC-ROC) являются мерами, которые учитывают полноту и специфичность модели, и могут быть более информативными при работе с несбалансированными данными.
3. PR-кривая и AUC-PR: PR-кривая (кривая точности-полноты) и площадь под PR-кривой (AUC-PR) также учитывают точность и полноту модели, что делает их более подходящими для оценки моделей на несбалансированных данных.
4. Взвешенные меры: Вместо использования обычных мер точности, полноты и F1-меры, можно использовать их взвешенные версии, которые учитывают дисбаланс классов. Например, взвешенная F1-мера учитывает дисбаланс классов при вычислении среднего гармонического.

Использование скорректированных мер качества в случае несбалансированных данных важно, поскольку обычные меры могут дать искаженное представление о качестве модели, особенно в контексте редкого класса. Эти скорректированные меры позволяют более точно оценить способность модели правильно классифицировать редкий класс и обнаружить проблемы, связанные с дисбалансом классов.

17. Что такое ROC AUC? Как построить ROC-кривую?

ROC AUC (Receiver Operating Characteristic Area Under the Curve) - это метрика, используемая для оценки качества бинарной классификации. Она представляет собой площадь под кривой ROC (Receiver Operating Characteristic), которая отображает отношение между долей истинно положительных результатов (True Positive Rate, TPR) и долей ложно положительных результатов (False Positive Rate, FPR) при изменении порога классификации.

Построение ROC-кривой и вычисление ROC AUC включает в себя следующие шаги:

1. Вычисление вероятностей принадлежности к положительному классу: Сначала необходимо обучить модель классификации и получить вероятности принадлежности к положительному классу для каждого примера в тестовом наборе данных.
2. Вычисление TPR и FPR: Затем, используя эти вероятности, можно построить ROC-кривую путем изменения порога классификации и вычисления TPR и FPR для каждого порога. TPR и FPR вычисляются следующим образом:

TPR = True Positives / (True Positives + False Negatives)

FPR = False Positives / (False Positives + True Negatives)

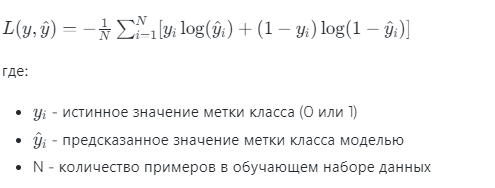
1. Построение кривой: После вычисления TPR и FPR для различных порогов классификации, строится ROC-кривая, где ось X представляет FPR, а ось Y представляет TPR.
2. Вычисление ROC AUC: ROC AUC вычисляется как площадь под ROC-кривой. Чем больше площадь под кривой, тем лучше модель способна различать между классами.

ROC-кривая и ROC AUC являются важными инструментами для оценки производительности модели в задачах бинарной классификации, особенно при работе с несбалансированными данными.

18. Функция потерь логистической регрессии. Как она связана с оценкой максимального правдоподобия?

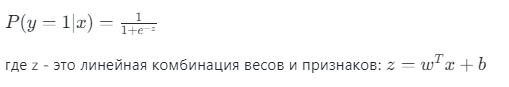
Функция потерь логистической регрессии называется логистической функцией потерь (logistic loss) или кросс-энтропией (cross-entropy loss). Она используется для оценки качества модели логистической регрессии и оптимизации параметров модели во время обучения.

Логистическая функция потерь выражается следующим образом:



Теперь, касательно связи с оценкой максимального правдоподобия (MLE), логистическая функция потерь связана с MLE следующим образом:

При использовании логистической регрессии для бинарной классификации, мы можем интерпретировать предсказания модели как вероятности принадлежности к положительному классу. При этом, мы можем выразить вероятность положительного класса через логистическую функцию:



Оценка максимального правдоподобия (MLE) заключается в максимизации вероятности получения наблюдаемых данных при заданных параметрах модели. В случае логистической регрессии, MLE связана с логистической функцией потерь, так как минимизация этой функции эквивалентна максимизации правдоподобия.

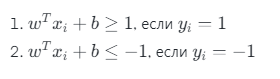
Таким образом, логистическая функция потерь в логистической регрессии связана с MLE через максимизацию вероятности правильной классификации и минимизацию кросс-энтропии между предсказанными вероятностями и истинными метками классов.

19. Основная идея машины опорных векторов. Функционал оптимизации для линейно разделимого случая.

Основная идея машины опорных векторов (SVM) заключается в поиске оптимальной разделяющей гиперплоскости, которая максимально разделяет два класса данных. SVM стремится найти гиперплоскость, которая максимизирует "маржу" (расстояние от гиперплоскости до ближайших точек обучающих данных), что делает классификацию более устойчивой и обобщаемой.

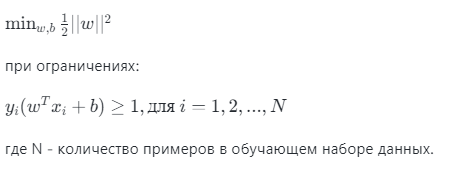
Функционал оптимизации для линейно разделимого случая (Hard Margin SVM) выглядит следующим образом:

Для каждого обучающего примера (x\_i, y\_i), где x\_i - вектор признаков, y\_i - метка класса (-1 или 1), SVM стремится найти веса w и смещение b такие, что выполнены следующие условия:



Эти условия гарантируют, что все точки обучающего набора данных находятся на правильной стороне разделяющей гиперплоскости с соответствующими отступами.

Таким образом, функционал оптимизации для линейно разделимого случая (Hard Margin SVM) можно выразить в виде задачи квадратичного программирования:



Этот функционал оптимизации стремится минимизировать норму вектора весов w при условии, что все точки находятся на правильной стороне разделяющей гиперплоскости.

20. Опишите жадный оптимизационный алгоритм для дерева решений.

Жадный оптимизационный алгоритм для дерева решений используется для построения дерева решений путем жадного выбора лучших признаков и пороговых значений на каждом узле дерева. Он стремится максимизировать информационный выигрыш (или минимизировать критерий неопределенности) на каждом шаге, чтобы построить оптимальное дерево решений.

Жадный алгоритм для дерева решений может быть описан следующим образом:

1. Начало: В начале все обучающие данные представлены в корне дерева.
2. Выбор признака: Для каждого узла дерева выбирается признак, который дает наибольший информационный выигрыш (или минимизирует критерий неопределенности), когда данные разделяются по этому признаку.
3. Выбор порогового значения: Для выбранного признака выбирается пороговое значение, которое максимизирует информационный выигрыш (или минимизирует критерий неопределенности) при разделении данных на две подгруппы.
4. Разделение узла: Узел разделяется на два новых узла, соответствующих двум подгруппам данных, основываясь на выбранном признаке и пороговом значении.
5. Рекурсивное повторение: Процесс выбора признака, выбора порогового значения и разделения узла повторяется для каждого нового узла, пока не будет выполнен какой-либо критерий останова (например, достигнута максимальная глубина дерева или минимальное количество образцов в узле)
6. Останов: Процесс завершается, когда выполнен какой-либо критерий останова, и все листовые узлы содержат прогнозы для соответствующих классов.

Жадный алгоритм для дерева решений стремится построить оптимальное дерево решений, основываясь на жадном выборе признаков и пороговых значений на каждом шаге.

21. Почему неограниченное дерево решений может достичь нулевой ошибки на обучающем наборе с уникальными объектами?

Неограниченное дерево решений может достичь нулевой ошибки на обучающем наборе с уникальными объектами из-за своей способности к переобучению. Переобучение возникает, когда дерево решений слишком глубокое и сложное, что позволяет ему "запомнить" обучающие данные вместо того, чтобы обобщать закономерности.

Когда дерево решений строится без ограничений на глубину или минимальное количество объектов в листе, оно может легко "запомнить" каждый объект обучающего набора. Это может привести к построению дерева, которое идеально подстраивается под каждый объект обучающего набора, включая выбросы и шум.

Когда дерево решений "запоминает" каждый объект обучающего набора, оно может достичь нулевой ошибки на обучающем наборе, поскольку каждый объект будет правильно классифицирован или предсказан. Однако это не гарантирует хорошего обобщения на новых, ранее не виденных данных.

Таким образом, неограниченное дерево решений может достичь нулевой ошибки на обучающем наборе с уникальными объектами из-за переобучения и слишком сложной модели, которая не учитывает обобщение.

22. Как присвоить метку класса объекту в листе дерева в классификации?

В классификации с использованием дерева решений, присвоение метки класса объекту в листе дерева происходит следующим образом:

1. Объект проходит через узлы дерева: Объект классифицируется путем прохождения через узлы дерева, начиная с корневого узла и двигаясь вниз по дереву в соответствии с условиями разделения на каждом узле.
2. Достижение листа: Когда объект достигает листового узла (узла без потомков), это означает, что для этого объекта было выполнено условие разделения, приведшее к попаданию его в этот лист.
3. Присвоение метки класса: В листе дерева хранится метка класса, которая присваивается объекту как предсказанная метка класса. Это может быть метка класса большинства объектов в листе или другая метка, определенная в процессе обучения дерева.
4. Предсказание: Полученная метка класса в листе становится предсказанием модели для данного объекта.

Таким образом, присвоение метки класса объекту в листе дерева в классификации происходит на основе метки класса, хранящейся в этом листе.

23. Как присвоить метку класса объекту в листе дерева в регрессии? Зависит ли это от информационного критерия?

В регрессии с использованием дерева решений, присвоение метки объекту в листе дерева происходит следующим образом:

1. Объект проходит через узлы дерева: Объект проходит через узлы дерева, начиная с корневого узла и двигаясь вниз по дереву в соответствии с условиями разделения на каждом узле.
2. Достижение листа: Когда объект достигает листового узла (узла без потомков), это означает, что для этого объекта было выполнено условие разделения, приведшее к попаданию его в этот лист.
3. Прогнозирование значения: В листе дерева хранится значение, которое становится предсказанием модели для данного объекта. Это значение может быть, например, средним значением целевой переменной для всех объектов, попавших в этот лист.
4. Предсказание: Полученное значение в листе становится предсказанием модели для данного объекта.

Относительно информационного критерия, на который зависит присвоение метки объекту в листе дерева в регрессии, можно сказать следующее:

* В регрессии с использованием дерева решений, информационный критерий (например, критерий Джини или энтропийный критерий) используется для выбора оптимального разделения на каждом узле дерева. Однако, после достижения листового узла, прогнозируемое значение в листе не зависит от информационного критерия. Вместо этого, оно обычно вычисляется на основе статистических характеристик целевой переменной (например, среднее значение) для всех объектов в этом листе.

Таким образом, присвоение метки объекту в листе дерева в регрессии зависит от значения, хранящегося в этом листе, и не зависит от информационного критерия.

24. Что такое бэггинг?

Бэггинг (bagging) - это метод ансамблирования, который используется для улучшения качества моделей машинного обучения путем комбинирования предсказаний нескольких моделей. Он основан на идее создания нескольких независимых моделей, обученных на подмножествах обучающих данных, и последующем усреднении их предсказаний.

Основные принципы бэггинга:

1. Bootstrap выборка: Для каждой модели создается подвыборка обучающих данных путем случайной выборки с возвращением. Это означает, что некоторые обучающие примеры могут быть выбраны несколько раз, в то время как другие могут быть пропущены.
2. Обучение независимых моделей: На каждой bootstrap выборке обучается отдельная модель машинного обучения.
3. Усреднение предсказаний: Предсказания, полученные от каждой модели, усредняются для получения окончательного предсказания ансамбля.

Преимущества бэггинга:

* Сокращение дисперсии: Благодаря использованию независимых моделей и усреднению их предсказаний, бэггинг может снизить дисперсию предсказаний, что делает его более устойчивым к переобучению.
* Улучшение обобщающей способности: Бэггинг может улучшить обобщающую способность модели путем комбинирования нескольких моделей, обученных на различных подмножествах данных.
* Применимость к различным типам моделей: Бэггинг может быть применен к различным видам моделей машинного обучения, таким как деревья решений, случайные леса, и т.д.

Примером бэггинга является метод случайного леса (random forest), который использует бэггинг для построения нескольких деревьев решений и комбинирования их предсказаний.

25. Что такое Случайный лес? Чем он отличается от Бэггинга над деревьями решений?

Случайный лес (Random Forest) - это ансамбль моделей машинного обучения, основанный на идее бэггинга (bagging), который используется для построения нескольких деревьев решений и комбинирования их предсказаний. Однако, в случайном лесе также применяются дополнительные методы для увеличения разнообразия и случайности моделей.

Отличия случайного леса от бэггинга над деревьями решений:

1. Выбор подмножества признаков: В случайном лесе при построении каждого дерева решений из обучающего набора данных случайным образом выбирается подмножество признаков. Это позволяет увеличить разнообразие моделей и снизить корреляцию между деревьями.
2. Случайное разделение узлов: При построении каждого узла дерева решений в случайном лесе происходит случайный выбор признаков для разделения. Это способствует увеличению разнообразия моделей и уменьшению коррелированности.
3. Усреднение предсказаний: Предсказания, полученные от каждого дерева решений, усредняются для получения окончательного предсказания случайного леса.

Таким образом, случайный лес отличается от бэггинга над деревьями решений добавлением случайности в процесс построения каждого дерева и выбора признаков для разделения узлов. Это позволяет случайному лесу создавать более разнообразные и устойчивые модели, что часто приводит к лучшему качеству предсказаний по сравнению с простым бэггингом над деревьями решений.

26. Как обучаются базовые алгоритмы в градиентном бустинге?

В градиентном бустинге базовые алгоритмы обучаются последовательно, с учетом ошибок предыдущих алгоритмов, с целью улучшения предсказательной способности модели. Процесс обучения базовых алгоритмов в градиентном бустинге можно описать следующим образом:

1. Инициализация: Начинается с построения начального базового алгоритма, который может быть простой моделью, например, деревом решений с небольшой глубиной.
2. Вычисление остатков: После построения первого базового алгоритма вычисляются остатки между предсказанными значениями и фактическими значениями целевой переменной.
3. Обучение следующего алгоритма: Следующий базовый алгоритм обучается на остатках, которые были вычислены на предыдущем шаге. Цель состоит в том, чтобы новый алгоритм учел ошибки предыдущих алгоритмов и попытался исправить их.
4. Обновление модели: Новый базовый алгоритм добавляется к существующей модели с учетом коэффициента (learning rate), который контролирует вклад каждого базового алгоритма в окончательное предсказание.
5. Итерации: Процесс обучения продолжается с итеративным добавлением новых базовых алгоритмов и коррекцией ошибок предыдущих алгоритмов до тех пор, пока не будет достигнуто определенное количество итераций или пока не будет достигнут критерий останова.

Таким образом, базовые алгоритмы в градиентном бустинге обучаются последовательно и учитывают ошибки предыдущих алгоритмов при построении модели.

27. Как работает обратное распространение ошибок в нейронных сетях? Что будет вектор-векторная производная?

Обратное распространение ошибок (backpropagation) - это метод обучения нейронных сетей, который используется для вычисления градиентов функции потерь по параметрам сети с целью их корректировки. Процесс обратного распространения ошибок состоит из двух основных этапов: прямого прохода (forward pass) и обратного прохода (backward pass).

Процесс обратного распространения ошибок в нейронных сетях можно описать следующим образом:

1. Прямой проход (forward pass): Во время прямого прохода входные данные проходят через нейронную сеть, и каждый нейрон вычисляет свой выход на основе взвешенной суммы входов и функции активации. Выход сети сравнивается с фактическими метками для вычисления функции потерь.
2. Обратный проход (backward pass): Во время обратного прохода градиенты функции потерь по параметрам сети вычисляются с помощью метода автоматического дифференцирования. Градиенты передаются от выходного слоя к входному слою сети, используя правило цепочки (chain rule) для вычисления градиентов по всем параметрам сети.

Вектор-векторная производная - это матрица, которая содержит производные всех элементов одного вектора по всем элементам другого вектора. В контексте нейронных сетей, вектор-векторная производная может использоваться для вычисления градиента функции потерь по вектору параметров модели. Это позволяет эффективно вычислять градиенты для коррекции параметров модели во время обучения.

28. Как работает свёрточный слой? Что такое свёрточная операция?

Свёрточный слой (Convolutional Layer) - это основной компонент свёрточной нейронной сети (CNN), который применяется для извлечения признаков из входных данных, таких как изображения. Свёрточные слои используются для обнаружения локальных шаблонов и структур во входных данных, что делает их особенно эффективными для обработки изображений.

Свёрточная операция - это основной шаг, выполняемый свёрточным слоем, и она заключается в применении фильтра (ядра) к различным областям входных данных с целью извлечения признаков. Процесс свёртки включает следующие шаги:

1. Фильтрация: Фильтр (ядро) перемещается по входным данным, и для каждой позиции производится операция свёртки. Это позволяет выделять локальные шаблоны и структуры в данных.
2. Свертка: На каждом шаге свёртки фильтр умножается на соответствующие элементы входных данных, и результаты умножения суммируются для создания выходного значения. Это позволяет выделять пространственные зависимости и признаки в данных.
3. Рецептивное поле: Размер области, на которую применяется фильтр, называется рецептивным полем, и он определяет, какие пространственные зависимости и признаки могут быть обнаружены.

Свёрточный слой может иметь несколько фильтров, каждый из которых выделяет различные признаки во входных данных. После процедуры свёртки может быть добавлен слой активации (например, ReLU), который добавляет нелинейность к выходу свёрточного слоя.

29. Почему полносвязные (плотные) сети не лучший выбор для работы с изображениями? Почему лучше работают свёрточные нейронные сети (CNN)?

Полносвязные (плотные) сети не являются лучшим выбором для работы с изображениями по нескольким причинам:

1. Игнорирование пространственной структуры: Полносвязные сети не учитывают пространственную структуру изображений, такую как пиксельная связность и локальные зависимости между пикселями. Они рассматривают каждый пиксель как отдельный признак, игнорируя пространственные шаблоны и структуры в изображениях.
2. Чувствительность к размеру входных данных: Полносвязные слои имеют фиксированный размер входа, поэтому они не масштабируются хорошо для обработки изображений различных размеров без предварительной обработки.

С другой стороны, свёрточные нейронные сети (CNN) лучше работают с изображениями по следующим причинам:

1. Учитывание пространственной структуры: Свёрточные слои в CNN учитывают пространственную структуру изображений, позволяя модели извлекать локальные шаблоны и признаки, такие как грани, углы и текстуры.
2. Общие веса: Свёрточные слои используют общие веса (ядра) для обработки различных областей изображения, что позволяет им обнаруживать одни и те же признаки в разных частях изображения. Это позволяет модели обучаться на меньшем количестве параметров и делает их более эффективными.
3. Инвариантность к трансляциям: Свёрточные слои обладают инвариантностью к трансляциям, что означает, что они могут распознавать те же признаки в различных частях изображения.

В целом, свёрточные нейронные сети (CNN) лучше подходят для обработки изображений благодаря своей способности учитывать пространственную структуру и общие признаки в изображениях.

30. Как работает базовая RNN (Vanilla RNN)?

Базовая рекуррентная нейронная сеть (RNN) или "Vanilla RNN" представляет собой простую форму рекуррентной нейронной сети, которая используется для моделирования последовательных данных с учетом контекста и последовательной зависимости. Вот как она работает:

1. Рекуррентная связь: Основное отличие RNN от других типов нейронных сетей заключается в наличии рекуррентной связи, которая позволяет модели учитывать предыдущие состояния и входы при обработке текущего входа.
2. Вычисление скрытого состояния: На каждом временном шаге RNN вычисляет скрытое состояние, используя текущий вход и предыдущее скрытое состояние. Это позволяет модели учитывать контекст и последовательную зависимость во временных данных.
3. Обновление скрытого состояния: Скрытое состояние обновляется на каждом временном шаге с учетом текущего входа и предыдущего скрытого состояния, используя активационную функцию (например, гиперболический тангенс или сигмоид).
4. Предсказание: На выходном временном шаге RNN может предсказывать метку класса (в задачах классификации) или значение (в задачах регрессии) на основе финального скрытого состояния или выхода каждого временного шага.

Однако, базовая RNN страдает от проблемы затухающего или взрывного градиента, что ограничивает ее способность обучения на длинных последовательностях. В связи с этим были разработаны более сложные архитектуры рекуррентных нейронных сетей, такие как LSTM (Long Short-Term Memory) и GRU (Gated Recurrent Unit), которые способны более эффективно учитывать длинные зависимости в последовательных данных.

31. Как работает Dropout?

Dropout - это метод регуляризации, который применяется в глубоком обучении для предотвращения переобучения модели. Он работает путем случайного "выключения" (отключения) нейронов во время обучения с определенной вероятностью, что позволяет модели обучаться более устойчиво и обобщать лучше на новые данные.

Вот как работает Dropout:

1. Случайное выключение нейронов: Во время обучения каждый нейрон в слое "выключается" с определенной вероятностью (обычно от 0.2 до 0.5), что означает, что его выход игнорируется на текущем примере обучающего набора.
2. Обучение на усредненных предсказаниях: Поскольку каждый нейрон может быть выключен в каждой итерации обучения, модель обучается на усредненных предсказаниях, что позволяет предотвратить переобучение и улучшить обобщающую способность.
3. Использование во время предсказания: Во время предсказания все нейроны остаются активными, но их выходы масштабируются на этапе предсказания в соответствии с вероятностью Dropout.

Преимущества Dropout:

* Предотвращение переобучения: Dropout помогает предотвратить переобучение модели, уменьшая зависимость между нейронами и снижая чувствительность модели к шуму в данных.
* Улучшение обобщающей способности: Dropout позволяет модели лучше обобщать на новые данные, так как она обучается на более усредненных предсказаниях.
* Эффективность: Dropout является эффективным методом регуляризации, который легко реализуется в различных типах нейронных сетей.

Таким образом, Dropout является мощным инструментом для улучшения обобщающей способности и предотвращения переобучения в глубоком обучении.

32. Как Dropout и пакетная нормализация меняют свое поведение на этапе вывода?

Во время вывода (предсказания) поведение Dropout и пакетной нормализации (batch normalization) изменяется по-разному.

Для Dropout:

Во время вывода все нейроны остаются активными, и их выходы масштабируются на этапе предсказания в соответствии с вероятностью Dropout, используемой во время обучения. Это означает, что при использовании Dropout на этапе вывода каждый нейрон сохраняет свой вклад в предсказания модели, но его выход масштабируется в соответствии с вероятностью Dropout, чтобы компенсировать усреднение, происходившее во время обучения.

Пакетная нормализация (Batch Normalization) - это метод нормализации активаций каждого слоя в нейронной сети по батчам (пакетам) данных во время обучения. Этот метод был предложен для улучшения стабильности и скорости обучения глубоких нейронных сетей.

В процессе обучения нейронной сети пакетная нормализация вычисляет среднее значение и дисперсию активаций каждого слоя на основе текущего пакета обучающих данных. Затем происходит центрирование и масштабирование активаций с использованием этих статистических данных. Это позволяет уравнять распределение активаций и облегчить обучение модели.

Для пакетной нормализации:

При использовании пакетной нормализации на этапе вывода параметры нормализации, такие как среднее и стандартное отклонение, используются для нормализации входных данных в соответствии с параметрами, полученными во время обучения. Это позволяет модели сохранять стабильность на этапе предсказания и улучшать скорость сходимости.

Таким образом, во время вывода Dropout и пакетная нормализация сохраняют свои основные принципы, но применяются с учетом параметров, полученных во время обучения, чтобы обеспечить стабильность и правильное функционирование модели на этапе предсказания.

33. В чем состоит формулировка задачи для Анализа Главных Компонентов?

Формулировка задачи для Анализа Главных Компонентов (PCA) заключается в поиске линейного преобразования, которое позволяет сжать данные путем проекции их на меньшее количество измерений (главные компоненты) с минимальной потерей информации. Основная цель PCA - найти новые оси (главные компоненты), по которым данные имеют наибольшую дисперсию.

Формулировка задачи PCA включает в себя следующие шаги:

1. Центрирование данных: Сначала данные центрируются путем вычитания среднего значения каждого признака из всех значений этого признака. Это позволяет установить среднее значение каждого признака в нуль.
2. Вычисление ковариационной матрицы: Затем вычисляется ковариационная матрица, которая отображает степень вариации между парами признаков в данных.
3. Вычисление главных компонент: Далее происходит вычисление собственных векторов и собственных значений ковариационной матрицы. Главные компоненты представляют собой собственные векторы, соответствующие наибольшим собственным значениям.
4. Проекция данных на главные компоненты: Наконец, данные проецируются на главные компоненты, что позволяет сжать данные и оставить только те измерения, которые содержат наибольшую дисперсию.

Таким образом, формулировка задачи PCA заключается в поиске преобразования данных, которое позволяет представить их в пространстве с меньшей размерностью, сохраняя при этом наибольшую часть информации.