Содержание

Введение в уменьшение размерности

В машинном обучении размерность (dimensionality) — это количество признаков (features) в наборе данных.

Уменьшение размерности данных - это процесс сокращения количества признаков, содержащихся в наборе данных, при котором сохраняется или максимизируется информация, содержащаяся в исходных данных. Признаки представляют собой характеристики или атрибуты данных, которые могут быть измерены или наблюдены.

Данные в сценариях машинного обучения и интеллектуального анализа данных обычно имеют очень высокую размерность. Высокая размерность приводит к обременительным вычислениям и проблемам «проклятия размерности». По мере увеличения числа измерений количество возможных комбинаций признаков увеличивается экспоненциально, что затрудняет получение репрезентативной выборки данных с вычислительной точки зрения, а выполнение таких задач, как кластеризация или классификация, становится дорогостоящим. Кроме того, некоторые алгоритмы машинного обучения могут быть чувствительны к количеству измерений, требуя больше данных для достижения того же уровня точности, что и для данных меньшей размерности.

Именно поэтому методы уменьшения размерности часто применяются в задачах машинного обучения для решения этих проблем. Традиционные методы уменьшения размерности, такие как PCA и LDA, широко изучались и использовались в последние десятилетия [1].

Цель уменьшения размерности данных заключается в том, чтобы сохранить максимальное количество информации, содержащейся в исходных данных, при снижении размерности. Это означает, что в результате преобразования данных мы стремимся минимизировать потери информации, которые могут возникнуть из-за сокращения количества признаков.

Преимущества методов уменьшения размерности [2]:

1. По мере уменьшения количества измерений можно уменьшить объем хранимых данных.
2. Они уменьшают времени вычислений последующих алгоритмов.
3. Можно удалить избыточные, ненужные и зашумленные данные.
4. Можно улучшить качество данных.
5. Некоторые алгоритмы неэффективны при большем количестве измерений. Таким образом, уменьшение этих размеров помогает алгоритму работать эффективно и повышает точность.
6. Они помогают визуализировать данные.
7. Они упрощают процесс классификации, а также повышают его эффективность.

Сохранение максимального количества информации важно для обеспечения того, чтобы новые данные после уменьшения размерности оставались достаточно информативными для анализа или использования в алгоритмах машинного обучения. Если мы потеряем слишком

много информации в результате уменьшения размерности, это может привести к ухудшению производительности моделей или неверному пониманию данных.

Уменьшение размерности данных играет важную роль в предобработке данных и подготовке их для дальнейшего анализа и применения алгоритмов машинного обучения. Оно помогает улучшить качество данных, сделать их более интерпретируемыми и подготовить для эффективного использования в различных задачах анализа данных.

Уменьшение размерности данных является важным инструментом в различных областях, где требуется анализ и обработка больших объемов информации. В обработке изображений, этот подход позволяет не только уменьшить размер файлов, но и выделить ключевые особенности и структуры изображений, что может быть полезно, например, в задачах распознавания образов или анализа медицинских изображений. В анализе текста, уменьшение размерности позволяет сократить пространство признаков, используемых для представления текстов, выделяя ключевые слова, темы или семантические структуры, что облегчает анализ и категоризацию текстовых данных. В биоинформатике, уменьшение размерности данных позволяет анализировать сложные геномные или белковые данные, выделяя важные генетические паттерны или структуры, что может помочь в исследованиях биологических процессов или в разработке лекарств. В финансовой аналитике, этот метод помогает выявить важные финансовые показатели или паттерны, влияющие на рыночные тренды или инвестиционные решения, что имеет значение для принятия обоснованных финансовых решений.

В каждой из этих областей уменьшение размерности данных позволяет сократить объем информации до более управляемого уровня, выделяя ключевые характеристики и структуры, что способствует более глубокому анализу и пониманию данных, а также повышает эффективность решения задач в этих областях.

Уменьшение размерности данных является мощным инструментом, однако существуют определенные ограничения и проблемы, с которыми может столкнуться этот процесс.

Одним из основных ограничений уменьшения размерности данных является потеря информации. При снижении размерности мы отбрасываем часть исходных признаков, что может привести к потере важной информации, необходимой для правильного анализа или моделирования данных. Поэтому важно балансировать между уменьшением размерности и сохранением максимального количества информации.

Еще одной проблемой является возможность переобучения моделей машинного обучения после уменьшения размерности данных. Если процесс уменьшения размерности не проводится осторожно, это может привести к потере способности модели обобщать данные, что может привести к переобучению. Поэтому важно внимательно контролировать процесс уменьшения размерности и выбирать подходящие методы.

Выбор подходящего метода уменьшения размерности может быть нетривиальной задачей, особенно когда имеется много методов и разнообразные характеристики данных. Разные методы могут иметь разные предположения о структуре данных и могут давать разные результаты в зависимости от типа данных. Поэтому важно иметь хорошее понимание методов уменьшения размерности и их применимости к конкретным данным перед принятием решения.

Проекционная перспектива в машинном обучении

Проекционная перспектива в машинном обучении представляет собой важный аспект анализа данных, направленный на преобразование многомерных данных в пространство меньшей размерности с сохранением существенных структурных характеристик. Этот подход нашел широкое применение в различных областях, включая классификацию, кластеризацию, а также визуализацию данных.

Методы с использованием проекционной перспективы, такие как метод главных компонент (PCA), t-распределенное стохастическое внедрение соседей (t-SNE), предоставляют инструменты для анализа данных, основанные на их геометрической структуре. Они позволяют выделить основные направления изменчивости в данных и представить их в пространстве меньшей размерности.

Одним из основных применений проекционной перспективы является снижение размерности данных. При работе с данными высокой размерности, такими как медицинские изображения или текстовые данные, проекционные методы позволяют сохранить основные характеристики данных, уменьшив их размерность и упростив последующий анализ.

Визуализация данных также является важным аспектом проекционной перспективы. После преобразования данных в пространство меньшей размерности, можно визуализировать их с помощью двумерных или трехмерных графиков, что делает понимание структуры данных более наглядным.

Кроме того, проекционная перспектива может быть использована для обнаружения скрытых структур в данных. Путем анализа проекций данных на новые признаки можно выявить неочевидные зависимости и отношения между объектами.

Вычисление собственных векторов и низкоранговые аппроксимации

Традиционные методы поиска информации и машинного обучения работают с данными в векторном представлении. Затем набор данных сохраняется в одной матрице A ∈ RN×n, где каждый столбец A соответствует вектору в N-мерном пространстве. Основным преимуществом этой модели векторного пространства является то, что можно использовать алгебраическую структуру векторного пространства.

Для многомерных данных хотелось бы упростить данные, чтобы можно было применить традиционные методы машинного обучения и статистические методы. Однако при таком упрощении не следует удалять важную информацию, содержащуюся в данных. Широко используемый метод для этой цели — аппроксимировать единственную матрицу данных A матрицей более низкого ранга [3].

Собственные векторы представляют собой важные направления изменчивости данных, а низкоранговые аппроксимации позволяют представить матрицы с высокой размерностью в более компактной форме, сохраняя при этом их существенные свойства.

Вычисление собственных векторов

Собственные векторы матрицы A определяются как ненулевые векторы v, удовлетворяющие условию $Av=\lambda v$, где λ - собственное значение, соответствующее собственному вектору v. Процесс нахождения собственных векторов включает в себя решение характеристического уравнения $\det(A-\lambda I)=0$, где I - единичная матрица.

Существует несколько алгоритмов для вычисления собственных векторов, включая методы степенной итерации, метод QR-разложения, а также методы Якоби и Ланцоша. Эти методы различаются по точности, скорости и применимости к различным типам матриц.

Низкоранговые аппроксимации

Низкоранговые аппроксимации позволяют приближенно представить матрицу A в виде произведения двух матриц меньшей размерности: $A\approx UVT$, где U и V - матрицы, состоящие из собственных векторов, соответствующих наибольшим собственным значениям матриц AAT и ATA соответственно.

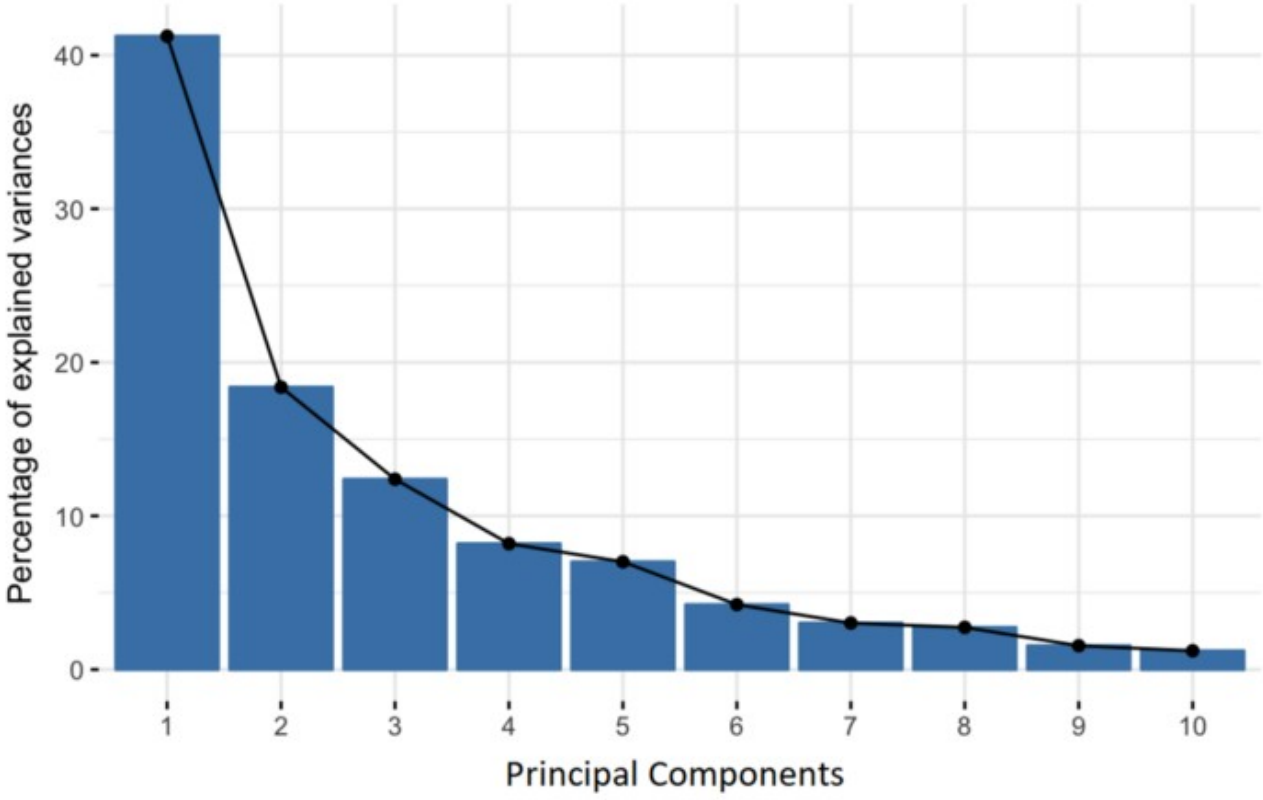
Одним из наиболее широко используемых методов низкоранговых аппроксимаций является сингулярное разложение (SVD). SVD разлагает матрицу A на произведение трех матриц $A=U\Sigma VT$, где U и V - ортогональные матрицы, а Σ - диагональная матрица, содержащая сингулярные значения. Путем отбрасывания малых сингулярных значений и соответствующих им столбцов матриц U и V, можно получить низкоранговую аппроксимацию матрицы A.

Одним из примеров низкоранговой аппроксимации является метод анализа главных компонентов (PCA - Principal Component Analysis). PCA аппроксимирует исходные данные большой размерности с использованием собственных векторов ковариационной матрицы.

Математическая интуиция, лежащая в основе PCA

Метод анализа главных компонентов (PCA - Principal Component Analysis) был представлен математиком Карлом Пирсоном в 1901 году. В анализе главных компонентов предполагается, что информация передается в дисперсии признаков, то есть чем выше вариация признака, тем больше информации несут признаки. Поэтому при сопоставлении данных в пространстве более высокого измерения с данными в пространстве более низкого измерения, дисперсия признаков в пространстве более низкого измерения должна быть максимальной.

Главные компоненты — это новые переменные, которые строятся как линейные комбинации исходных переменных. Эти комбинации выполняются таким образом, что новые переменные (т. е. основные компоненты) не коррелируют, а большая часть информации в исходных переменных сжимается в первые компоненты. Идея заключается в том, что 10-мерные данные дают вам 10 основных компонентов, но PCA пытается поместить максимально возможную информацию в первый компонент, затем максимально оставшуюся информацию во второй и так далее.



Подобная организация информации в главных компонентах позволяет уменьшить размерность без большой потери информации, за счет отбрасывания компонентов с низкой информацией и рассмотрения оставшихся компонентов в качестве новых переменных. Важно понимать, что главные компоненты менее интерпретируемы и не имеют никакого реального значения, поскольку они построены как линейные комбинации исходных переменных.

С геометрической точки зрения главные компоненты представляют собой направления данных, которые объясняют максимальную величину дисперсии, то есть линии, которые

содержат большую часть информации о данных. Связь между дисперсией и информацией такова: чем больше дисперсия, переносимая линией, тем больше дисперсия точек данных вдоль нее, и чем больше дисперсия вдоль линии, тем больше информации она содержит [4].

Шаг 1. Стандартизация

Целью этого шага является стандартизация диапазона непрерывных исходных переменных, чтобы каждая из них вносила равный вклад в анализ.

Причина, по которой так важно выполнять стандартизацию перед PCA, заключается в том, что метод весьма чувствителен к дисперсиям исходных переменных. То есть, если между диапазонами исходных переменных существуют большие различия, переменные с более широкими диапазонами будут доминировать над переменными с небольшими диапазонами (например, переменная с диапазоном от 0 до 100 будет доминировать над переменной с диапазоном от 0 до 1), что приведет к необъективным результатам. Таким образом, преобразование данных в сопоставимые масштабы может предотвратить эту проблему.

Математически это можно сделать, вычитая среднее значение и разделив его на стандартное отклонение для каждого значения каждой переменной.

$$z = \frac{value - mean}{standard\ deviation}$$

После завершения стандартизации все переменные будут преобразованы в один и тот же масштаб.

Шаг 2. Вычисление ковариационной матрицы

Цель этого шага — понять, как переменные входного набора данных отличаются от среднего значения по отношению друг к другу, или, другими словами, увидеть, есть ли между ними какая-либо связь. Потому что иногда переменные сильно коррелируют и содержат избыточную информацию. Чтобы выявить эти корреляции, вычислим ковариационную матрицу.

Ковариационная матрица представляет собой симметричную матрицу размера p × p (где p — количество измерений), в которой в качестве элементов имеются ковариации, связанные со всеми возможными парами исходных переменных. Например, для трехмерного набора данных с тремя переменными x, y и z ковариационная матрица представляет собой матрицу данных 3×3 следующего вида:

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

Covariance Matrix for 3-Dimensional Data.

Поскольку ковариация переменной сама по себе является ее дисперсией (Cov(a,a)=Var(a)), на главной диагонали (сверху слева направо) мы фактически имеем дисперсии каждой исходной переменной. А поскольку ковариация коммутативна (Cov(a,b)=Cov(b,a)), элементы

ковариационной матрицы симметричны относительно главной диагонали, а это означает, что верхняя и нижняя треугольные части равны.

Важен знак ковариации:

- Если положительный, то две переменные увеличиваются или уменьшаются вместе (коррелируют).
- Если отрицательный, то одна переменная увеличивается, когда другая уменьшается (обратно коррелируют)

Шаг 3. Вычисление собственных векторы и собственных значений ковариационной матрицы, чтобы определить главные компоненты.

Собственные векторы матрицы ковариации на самом деле являются направлениями осей, где наблюдается наибольшая дисперсия (наибольшая информация) и которые мы называем главными компонентами. А собственные значения — это коэффициенты, присоединенные к собственным векторам, которые определяют величину дисперсии, содержащуюся в каждом главном компоненте.

Расположив собственные векторы в порядке их собственных значений, от большего к меньшему, получаем главные компоненты в порядке значимости.

Пример анализа главных компонентов: предположим, что наш набор данных двумерный с двумя переменными x, y и что собственные векторы и собственные значения ковариационной матрицы следующие:

$$v1 = \begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix} \qquad \lambda_1 = 1.284028$$

$$v2 = \begin{bmatrix} -0.7351785 \\ 0.6778736 \end{bmatrix} \qquad \lambda_2 = 0.04908323$$

Если мы расположим собственные значения в порядке убывания, мы получим λ1>λ2, что означает, что собственный вектор, соответствующий первому главному компоненту (PC1), равен v1, а собственный вектор, соответствующий второму главному компоненту (PC2), — это v2.

Чтобы вычислить процент дисперсии (информации), приходящейся на каждый компонент, мы делим собственное значение каждого компонента на сумму собственных значений. Если мы применим это к приведенному выше примеру, мы обнаружим, что PC1 и PC2 несут соответственно 96 процентов и 4 процента дисперсии данных.

Шаг 4. Создание вектора признаков

Как мы видели на предыдущем шаге, вычисление собственных векторов и упорядочивание их по собственным значениям в порядке убывания позволяет нам найти главные компоненты в порядке значимости. На этом этапе мы решаем, сохранить ли все эти

компоненты или отбросить те, которые имеют меньшую значимость (с низкими собственными значениями), и сформировать из оставшихся матрицу векторов, которую мы называем вектором признаков.

Итак, вектор признаков — это просто матрица, столбцами которой являются собственные векторы компонентов, которые мы решили сохранить. Это первый шаг к уменьшению размерности, потому что если мы решим сохранить только p собственных векторов (компонентов) из n, окончательный набор данных будет иметь только p измерений.

Продолжая пример из предыдущего шага, мы можем сформировать вектор признаков с обоими собственными векторами v1 и v2:

$$\begin{bmatrix} 0.6778736 & -0.7351785 \\ 0.7351785 & 0.6778736 \end{bmatrix}$$

Или отбросить собственный вектор v2, который имеет меньшую значимость, и сформировать вектор признаков только с помощью v1:

$$\begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix}$$

Отказ от собственного вектора v2 уменьшит размерность на 1 и, следовательно, приведет к потере информации в конечном наборе данных. Но учитывая, что вектор v2 переносил только 4% информации, потеря не будет существенной, и у нас все равно будет 96% информации, которую несет вектор v1.

Сохранить ли все компоненты или отказаться от менее значимых, зависит от цели применения метода. Потому что, если требуется описать данные с точки зрения новых переменных (основных компонентов), которые не коррелируют, не стремясь уменьшить размерность, исключать менее значимые компоненты не нужно.

Шаг 5. Пересчет данных по осям главных компонентов

На предыдущих шагах, кроме стандартизации, не вносится никаких изменений в данные, просто выбираются главные компоненты и формируется вектор признаков, но входной набор данных остается всегда в координатах исходных осей.

Последний этап заключается в том, чтобы использовать вектор признаков, сформированный с использованием собственных векторов ковариационной матрицы, для переориентации данных из исходных осей в те, которые представлены главными компонентами (отсюда и название «Анализ главных компонентов»). Это можно сделать, умножив транспонированный исходный набор данных на транспонированный вектор признаков.

$$FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T$$

Реализация PCA на Python

Реализация [5] приведена в приложении 1.

Перспектива скрытых переменных

Перспектива скрытых переменных в области статистики и машинного обучения означает, что некоторые переменные или факторы могут оказывать влияние на наблюдаемые данные, но не являются прямо измеряемыми или доступными для наблюдения. Вместо этого они остаются скрытыми и могут быть выявлены только через анализ корреляций и зависимостей между наблюдаемыми переменными.

Например, в задачах обучения с учителем скрытые переменные могут включать в себя латентные признаки, которые не непосредственно наблюдаемы, но оказывают влияние на данные. В методах машинного обучения, таких как скрытые модели Маркова или скрытые переменные в моделях глубокого обучения, эти скрытые переменные играют ключевую роль в построении модели и интерпретации данных.

Выявление скрытых переменных и их учет в моделях является важным аспектом анализа данных и обучения моделей, поскольку это позволяет получить более точные и интерпретируемые результаты.

Методы разложения матриц представляют собой класс методов анализа данных, которые находят применение в различных областях, таких как рекомендательные системы, обработка сигналов, анализ изображений и другие. Они позволяют представить исходные данные в виде комбинации более простых компонент, что обычно помогает выделить в данных скрытые закономерности и структуры, поэтому методы разложения матриц могут интерпретироваться как модели со скрытыми переменными. Здесь скрытые переменные представляют собой латентные признаки, которые объясняют структуру данных.

Скрытые переменные в методах разложения матриц используются для описания этой комбинации более простых компонент, которые не являются непосредственно наблюдаемыми в исходных данных. PCA является одним из наиболее распространенных методов разложения матриц. Он пытается найти ортогональные направления в пространстве данных, вдоль которых данные имеют наибольшую дисперсию. Скрытые переменные в PCA представляют собой компоненты, которые объясняют эту дисперсию.

Математическая интуиция, лежащая в основе LDA

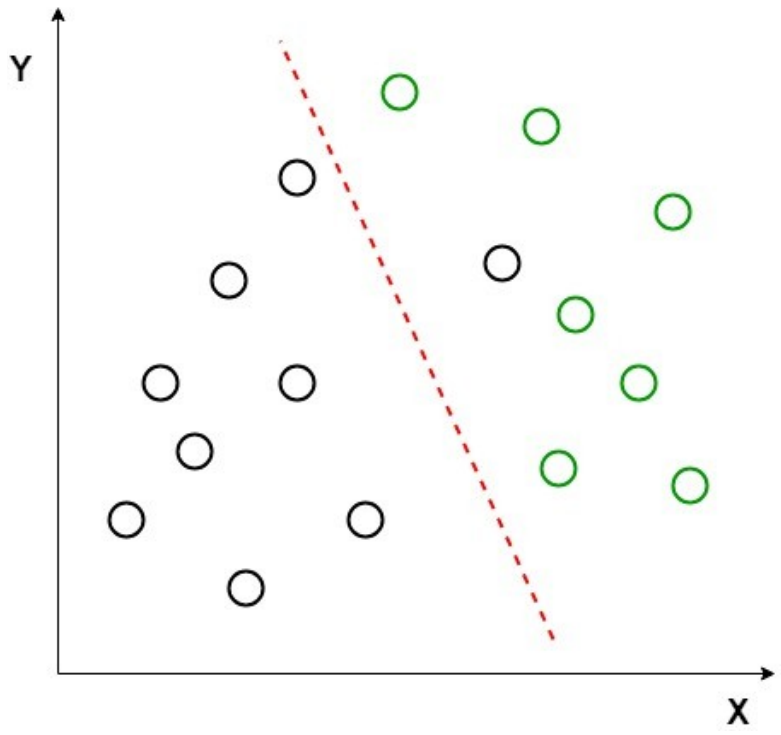
Введение

Линейный дискриминантный анализ (LDA), также известный как нормальный дискриминантный анализ или анализ дискриминантной функции, представляет собой метод уменьшения размерности, который в основном используется в задачах классификации с учителем. Он облегчает моделирование различий между группами, эффективно разделяя два или более класса. LDA работает путем проецирования объектов из пространства более высокого измерения в пространство более низкого измерения. В машинном обучении LDA служит алгоритмом обучения с учителем, специально разработанным для задач классификации и стремящимся определить линейную комбинацию признаков, которая оптимально разделяет классы в наборе данных.

Например, у нас есть два класса, и нам нужно эффективно их разделить. Классы могут иметь несколько признаков. Использование только одного признака для их классификации может привести к некоторому перекрытию, как показано на рисунке ниже [6].

LDA предполагает, что данные имеют распределение Гаусса и что ковариационные матрицы разных классов равны. Также предполагается, что данные линейно разделимы, а это означает, что линейная граница решения может точно классифицировать различные классы.

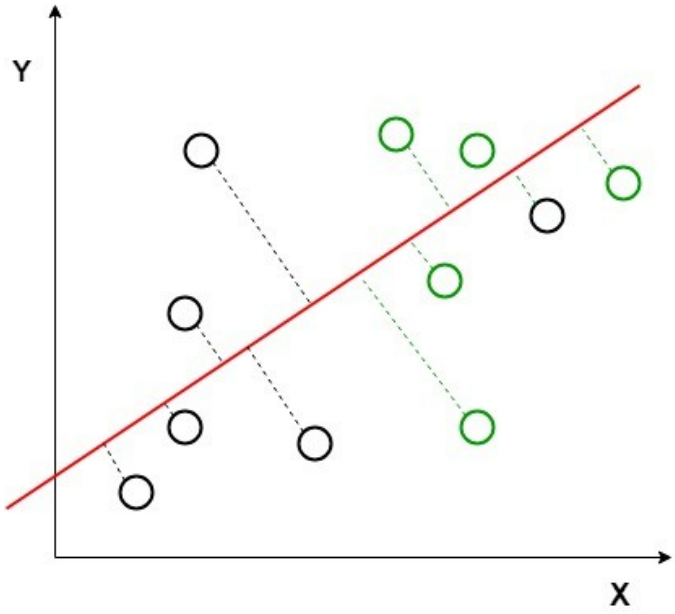
Предположим, у нас есть два набора точек данных, принадлежащих двум разным классам, которые мы хотим классифицировать. Как показано на приведенном двумерном графике, когда точки данных отображаются на двумерной плоскости, не существует прямой линии, которая могла бы полностью разделить два класса точек данных. Следовательно, в этом случае используется LDA (линейный дискриминантный анализ), который сводит 2D-граф к 1D-графу, чтобы максимизировать разделимость между двумя классами.



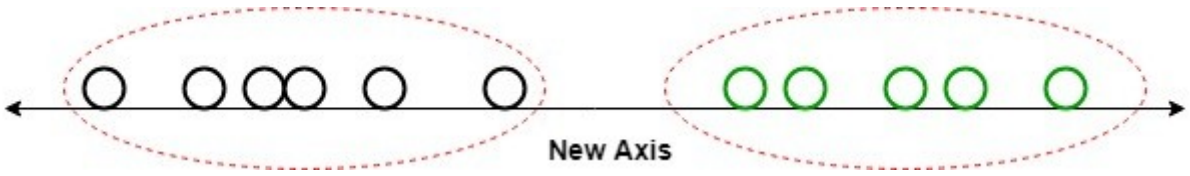
Здесь линейный дискриминантный анализ использует обе оси (X и Y) для создания новой оси и проецирует данные на новую ось таким образом, чтобы максимизировать разделение двух категорий и, следовательно, свести 2D-график к 1D-графику.

LDA использует два критерия для создания новой оси:

1. Максимизируется расстояние между средними значениями двух классов.
2. Минимизируется дисперсия внутри каждого класса.



На приведенном выше графике видно, что новая ось (красная) генерируется и отображается на двумерном графике так, что она максимизирует расстояние между средними значениями двух классов и минимизирует дисперсию внутри каждого класса. Проще говоря, эта вновь созданная ось увеличивает разделение между точками данных двух классов. После создания этой новой оси с использованием вышеупомянутых критериев все точки данных классов наносятся на эту новую ось и показаны на рисунке, приведенном ниже.



Но линейный дискриминантный анализ не работает, когда средние значения распределений являются общими, поскольку для LDA становится невозможным найти новую ось, которая делает оба класса линейно разделимыми. В таких случаях используется нелинейный дискриминантный анализ.

Математическая интуиция

LDA проецирует данные в пространство меньшей размерности, что максимизирует разделение между классами. Он делает это путем нахождения набора линейных

дискриминантов, которые максимизируют отношение дисперсии между классами к дисперсии внутри класса. Другими словами, он находит направления в пространстве признаков, которые лучше всего разделяют разные классы данных.

Предположим, у нас есть два класса и d-мерные образцы, такие как x1, x2… xn, где n1 образцов из класса (c1) и n2 из класса (c2).

Если xi — это точка данных, то ее проекция на линию, представленную единичным вектором v, может быть записана как vTxi.

Давайте рассмотрим u1 и u2 как средние значения выборок класса c1 и c2 соответственно до проецирования, а u1hat обозначает среднее значение выборок класса после проецирования, и его можно вычислить по формуле

$$\widetilde{\mu_1} = \frac{1}{n_1} \sum_{x_i \in c_1}^{n_1} v^T x_i = v^T \mu_1$$

Аналогично,

$$\widetilde{\mu_2} = v^T \mu_2$$

Теперь в LDA нам нужно нормализовать | mu_1 — mu_2 |. Пусть y_i = v^{T}x_i — прогнозируемые выборки, тогда разброс для выборок c1 равен:

$$\widetilde{s_1^2} = \sum_{y_i \in c_1} (y_i - \mu_1)^2$$

Аналогично,

$$\widetilde{s_2^2} = \sum_{y_i \in c_1} (y_i - \mu_2)^2$$

Теперь нам нужно защитить наши данные на линии, имеющей направление v, которое максимизирует,

$$J(v) = \frac{\widetilde{\mu_1} - \widetilde{\mu_2}}{\widetilde{s_1^2} + \widetilde{s_2^2}}$$

Для максимизации приведенного выше уравнения нам нужно найти вектор проекции, который максимизирует разницу среднего уменьшения разбросов обоих классов. Теперь матрица разброса s1 и s2 классов c1 и c2 равна:

$$s_1 = \sum_{x_i \in c_1} (x_i - \mu_1)(x_i - \mu_1)^T$$

$$s_2 = \sum_{x_i \in c_2} (x_i - \mu_2)(x_i - \mu_2)^T$$

После упрощения приведенного выше уравнения мы получаем разброс внутри классов (sw) и разброс между классами (sb)

$$s_w = s_1 + s_2$$

$$s_b = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

Теперь мы попытаемся упростить числитель J(v):

$$J(v) = \frac{|\widetilde{\mu_1} - \widetilde{\mu_2}|}{\widetilde{s_1^2} + \widetilde{s_2^2}} = \frac{v^T s_b v}{v^T s_w v}$$

Теперь, чтобы максимизировать приведенное выше уравнение, нам нужно вычислить дифференцирование по v:

$$\frac{dJ(v)}{dv} = s_b v - \frac{v^t s_b v (s_w v)}{v^T s_w v}$$

$$= s_b v - \lambda s_w v = 0$$

$$s_b v = \lambda s_w v$$

$$s_w^{-1} s_b v = \lambda v$$

$$Mv = \lambda v$$

$$where,$$

$$\lambda = \frac{v^T s_b v}{v^T s_w v} and$$

$$M = s_w^{-1} s_b$$

Здесь в качестве максимального значения J(v) мы будем использовать значение, соответствующее наибольшему собственному значению. Это предоставит нам лучшее решение для LDA.

Реализация линейного дискриминантного анализа (LDA)

[6] см. Приложение 2.

Математическая интуиция, лежащая в основе GDA (KFDA)

Дискриминантный анализ Фишера с ядром (KFDA, Kernel Fisher Discriminant Analysis, также известный как обобщенный дискриминантный анализ GDA, Generalized Discriminant Analysis) - нелинейный эквивалент линейного дискриминантного анализа (LDA), представляет собой метод уменьшения размерности, который проецирует векторы на меньшее подпространство. Это подпространство оптимизировано для максимизации разброса между классами и минимизации разброса внутри классов, что делает его эффективным методом уменьшения размерности и классификации. KFDA усовершенствовано по сравнению с LDA за счет включения нелинейных подпространств с использованием трюка ядра.

Для расширения LDA на нелинейные отображения данные, представленные в виде ℓ точек xi, могут быть отображены в новое пространство признаков F с помощью некоторой функции ϕ. В этом новом пространстве признаков функция, которую необходимо максимизировать, имеет вид:

$$J(\mathbf{w}) = \frac{\mathbf{w}^{\mathrm{T}} \mathbf{S}_B^{\phi} \mathbf{w}}{\mathbf{w}^{\mathrm{T}} \mathbf{S}_W^{\phi} \mathbf{w}},$$

где

$$\mathbf{S}_B^{\phi} = \left(\mathbf{m}_2^{\phi} - \mathbf{m}_1^{\phi}\right) \left(\mathbf{m}_2^{\phi} - \mathbf{m}_1^{\phi}\right)^{\mathrm{T}}$$
$$\mathbf{S}_W^{\phi} = \sum_{i=1,2} \sum_{n=1}^{l_i} \left(\phi(\mathbf{x}_n^i) - \mathbf{m}_i^{\phi}\right) \left(\phi(\mathbf{x}_n^i) - \mathbf{m}_i^{\phi}\right)^{\mathrm{T}},$$

и

$$\mathbf{m}_i^{\phi} = \frac{1}{l_i} \sum_{j=1}^{l_i} \phi(\mathbf{x}_j^i).$$

Кроме того, стоит отметить, что $w \in F$ w∈F. Явное вычисление отображений $\phi(xi)$ ϕ(xi) и затем применение LDA может быть вычислительно затратным и во многих случаях неприемлемым. Например, F F может быть бесконечномерным. Таким образом, вместо явного отображения данных в F F, данные могут быть неявно встроены путем переписывания алгоритма в терминах скалярных произведений и использования ядерных функций, в которых скалярное произведение в новом пространстве признаков заменяется ядерной функцией: $k(x,y)=\phi(x) \cdot \phi(y)$

LDA можно переформулировать в терминах скалярных произведений, сначала заметив, что w будет иметь разложение в виде

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i \phi(\mathbf{x}_i).$$

Тогда обратите внимание, что

$$\mathbf{w}^{\mathrm{T}}\mathbf{m}_i^{\phi} = \frac{1}{l_i}\sum_{j=1}^{l}\sum_{k=1}^{l_i}\alpha_j k\left(\mathbf{x}_j, \mathbf{x}_k^i\right) = \alpha^{\mathrm{T}}\mathbf{M}_i,$$

где

$$(\mathbf{M}_i)_j = \frac{1}{l_i}\sum_{k=1}^{l_i} k(\mathbf{x}_j, \mathbf{x}_k^i).$$

Числитель ( ) можно записать следующим образом:

$$\mathbf{w}^{\mathrm{T}}\mathbf{S}_B^{\phi}\mathbf{w} = \mathbf{w}^{\mathrm{T}}\left(\mathbf{m}_2^{\phi} - \mathbf{m}_1^{\phi}\right)\left(\mathbf{m}_2^{\phi} - \mathbf{m}_1^{\phi}\right)^{\mathrm{T}}\mathbf{w} = \alpha^{\mathrm{T}}\mathbf{M}\alpha, \qquad \text{where} \qquad \mathbf{M} = (\mathbf{M}_2 - \mathbf{M}_1)(\mathbf{M}_2 - \mathbf{M}_1)^{\mathrm{T}}.$$

Аналогично знаменатель можно записать как

$$\mathbf{w}^{\mathrm{T}}\mathbf{S}_W^{\phi}\mathbf{w} = \alpha^{\mathrm{T}}\mathbf{N}\alpha, \qquad \text{where} \qquad \mathbf{N} = \sum_{j=1,2}\mathbf{K}_j(\mathbf{I} - \mathbf{1}_{l_j})\mathbf{K}_j^{\mathrm{T}},$$

с *n*-м, *m*-м компонентом *Kj*, определенным как *k*(*xn*,*xmj*), *I* - единичная матрица, и 1*lj* - матрица, все элементы которой равны 1/*lj*.
С учетом этих уравнений для числителя и знаменателя *J*(*w*) уравнение для *J* можно переписать следующим образом:

$$J(\alpha) = \frac{\alpha^{\mathrm{T}}\mathbf{M}\alpha}{\alpha^{\mathrm{T}}\mathbf{N}\alpha}.$$

Тогда дифференцирование и приравнивание к нулю дает

$$(\alpha^{\mathrm{T}}\mathbf{M}\alpha)\mathbf{N}\alpha = (\alpha^{\mathrm{T}}\mathbf{N}\alpha)\mathbf{M}\alpha.$$

Поскольку только направление w, и следовательно направление α важны, вышеприведенное уравнение можно решить для α следующим образом:

$$\alpha = \mathbf{N}^{-1}(\mathbf{M}_2 - \mathbf{M}_1).$$

Заметим, что на практике матрица *N* обычно является вырожденной, поэтому к ней добавляется кратное единичной матрице значение.

$$\mathbf{N}_\epsilon = \mathbf{N} + \epsilon\mathbf{I}.$$

Учитывая решение для α, проекция новой точки данных задается следующим образом:

$$y(\mathbf{x}) = (\mathbf{w}\cdot\phi(\mathbf{x})) = \sum_{i=1}^{l}\alpha_i k(\mathbf{x}_i, \mathbf{x}).$$

Реализация обобщенного дискриминантного анализа (GDA) (KFDA)

[13] см. Приложение 3

Математическая интуиция, лежащая в основе GDA

Гауссовский дискриминантный анализ (GDA, Gaussian Discriminant Analysis) — это алгоритм обучения с учителем, используемый для задач классификации в машинном обучении. Это вариант алгоритма линейного дискриминантного анализа (LDA), который ослабляет предположение о том, что ковариационные матрицы разных классов равны.

GDA работает, предполагая, что данные в каждом классе соответствуют гауссовскому (нормальному) распределению, а затем оценивает среднее значение и ковариационную матрицу для каждого класса. Затем он использует теорему Байеса для вычисления вероятности того, что новая точка данных принадлежит каждому классу, и выбирает класс с наибольшей вероятностью в качестве прогнозируемого класса.

GDA может обрабатывать данные с произвольными ковариационными матрицами для каждого класса, в отличие от LDA, который предполагает, что ковариационные матрицы равны. Это делает GDA более гибким и способным работать с более сложными наборами данных. Однако недостатком является то, что GDA требует оценки большего количества параметров, поскольку для каждого класса необходимо оценивать отдельную ковариационную матрицу.

Одним из недостатков GDA является то, что он может быть чувствителен к выбросам и может переопределять данные, если количество обучающих примеров невелико по сравнению с количеством оцениваемых параметров. Кроме того, GDA может работать неэффективно, если граница принятия решений между классами сильно нелинейна.

В целом, GDA — это мощный алгоритм классификации, который может обрабатывать более сложные наборы данных, чем LDA, но для оценки требуется больше параметров, и он может не работать хорошо во всех ситуациях.

Математическая интуиция [7]:

В машинном обучении для классификации используются два типа алгоритмов контролируемого обучения:

1. Дискриминационные алгоритмы обучения
2. Генеративные алгоритмы обучения

Примером алгоритмов дискриминационного обучения является логистическая регрессия. Эти алгоритмы пытаются определить границу между классами в процессе обучения. Алгоритм дискриминационного обучения можно использовать для решения задачи классификации, которая определит, болен ли пациент малярией. Затем граница проверяется, чтобы увидеть, попадает ли новый пример на границу P(y|X), т. е. для данного набора признаков X какова вероятность его принадлежности к классу «y».

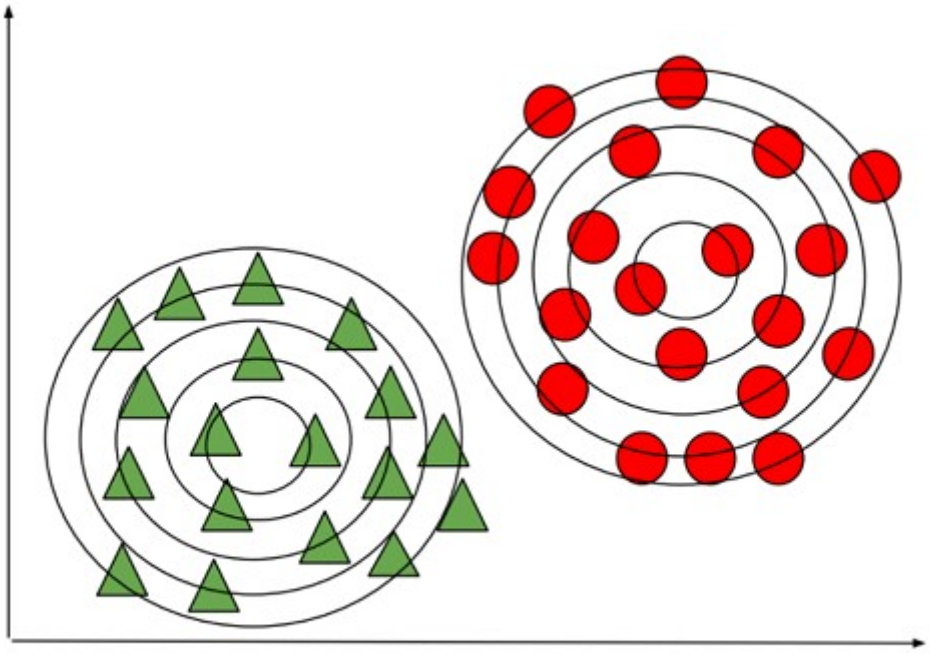
Алгоритмы генеративного обучения используют другой подход. Они пытаются уловить распределение каждого класса отдельно, а не находить границу между классами. Алгоритм генеративного обучения будет отдельно изучать распределение инфицированных и здоровых пациентов, а затем он попытается изучить признаки каждого распределения индивидуально. Когда будет представлен новый пример, он будет сравнен с обоими распределениями и будет присвоен класс, на который он наиболее похож.

Теорема Байеса используется для прогнозирования алгоритмов генеративного обучения:

$$P(y|X) = \frac{P(X|y).P(y)}{P(X)}$$
$$where, P(X) = P(X|y=1).P(y=1) + P(X|y=0).P(y=0)$$

Анализируя только количество P(X|y), а также P(y) в конкретном классе, мы можем определить P(y), т.е. учитывая характеристики выборки, насколько вероятно, что она принадлежит к класс «y».

GDA — это алгоритм генеративного обучения, целью которого является определение распределения каждого класса. Он пытается создать распределение Гаусса для каждой категории данных отдельно. Вероятность результата в случае использования алгоритма, известного как алгоритм генеративного обучения, очень высока, если он находится близко к центру контура, соответствующему его классу. Оно уменьшается при удалении от середины контура. Ниже приведены изображения, иллюстрирующие различия между дискриминативными и генеративными алгоритмами обучения.



(a) Generative Learning Algorithm (GDA)

Давайте рассмотрим случай бинарной задачи классификации, в которой все наборы данных распределены независимо и одинаково. Чтобы определить P(X|y), мы можем использовать многомерное распределение Гаусса для расчета уравнения плотности вероятности для каждого конкретного класса. Чтобы определить P(y) или предшествующий класс для каждого класса, мы можем использовать распределение Бернулли, поскольку все выборочные данные, используемые в двоичной классификации, могут быть равны 0 или 1.

Таким образом, распределение вероятностей, а также предшествующий класс можно определить с использованием общей модели распределений Гаусса и Бернулли:

$$P(x|y=0) = \frac{1}{(2\pi)^{n/2} * |\Sigma|^{1/2}} exp(-1/2(x-\mu_0)^T \Sigma^{-1}(x-\mu_0)) \ - \ \textbf{Eq 1}$$

$$P(x|y=1) = \frac{1}{(2\pi)^{n/2} * |\Sigma|^{1/2}} exp(-1/2(x-\mu_1)^T \Sigma^{-1}(x-\mu_1)) \ - \ \textbf{Eq 2}$$

$$P(y) = \phi^y.(1-\phi)^{1-y} \ - \ \textbf{Eq 3}$$

Чтобы понять распределения вероятностей с точки зрения вышеуказанных параметров, мы можем сформулировать формулу правдоподобия, которая является произведением распределения вероятностей, а также класса перед каждой выборкой данных (принимать распределение вероятностей как произведение разумно, поскольку все выборки данных считаются независимо и одинаково распределенными).

$$L(\phi, \mu_0, \mu_1, \Sigma) = \Pi_{i=1}^m P(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma)$$
$$= \Pi_{i=0}^m P(x^{(i)}|y^{(i)}).P(y^{(i)}) \ - \textbf{Eq} \quad \textbf{4}$$

В соответствии с принципом оценки правдоподобия нам необходимо выбрать параметры так, чтобы увеличить функцию вероятности, как показано в уравнении 4. Вместо максимизации функции правдоподобия мы можем повысить логарифмическую функцию правдоподобия, строго растущую функцию.

$$Therefore, \ Log - Likelihood \ function = log(L(\phi, \mu_0, \mu_1, \Sigma))$$
$$On \ maximizing \ Log - Likelihood \ following \ parameters \ are \ obtained$$

$$\phi = \frac{1}{m} \Sigma_{i=1}^m 1\{y^{(i)} = 1\}$$

$$\mu_0 = \frac{\Sigma_{i=1}^m 1\!\!1\{y^{(i)} = 0\}.x^{(i)}}{\Sigma_{i=1}^m 1\!\!1\{y^{(i)} = 0\}}$$

$$\mu_1 = \frac{\Sigma_{i=1}^m 1\!\!1\{y^{(i)} = 1\}.x^{(i)}}{\Sigma_{i=1}^m 1\!\!1\{y^{(i)} = 1\}}$$

$$\Sigma = \frac{1}{m} \Sigma_{i=1}^m (x^{(i)} - \mu_{y^{(i)}}).(x^{(i)} - \mu_{y^{(i)}})^T$$

В приведенных выше уравнениях «1{условие}» — это индикаторная функция, которая возвращает 1, если это условие выполняется; в противном случае возвращает ноль. Например, 1{y = 1} возвращает 1, только если класс выборки данных равен 1 - в противном случае он возвращает 0. Аналогично в случае 1{y = 0} фунция вернет 1 только если класс выборки равен 0. В противном случае возвращается 0.

Полученные параметры можно использовать в уравнениях 1, 2 и 3, чтобы определить распределение вероятностей и класс перед всеми выборками данных. Рассчитанные значения можно дополнительно умножить, чтобы определить функцию правдоподобия, как показано в уравнении 4. Как упоминалось ранее, это функция вероятности, т. е. P(X|y). P(y)

интегрируется в формулу Байеса для расчета P(y|X) (т. е. определения класса «y» выборки данных для указанных признаков «X»).

Таким образом, гауссовский дискриминантный анализ работает очень хорошо с ограниченным объемом данных (скажем, несколькими тысячами примеров) и может быть более надежным, чем логистическая регрессия, если наши фундаментальные предположения относительно распределения данных верны.

Реализация обобщенного дискриминантного анализа (GDA)

[8] см. Приложение 4

Математическая интуиция, лежащая в основе алгоритма t-SNE

T-распределенное стохастическое внедрение соседей (t-SNE, T-distributed Stochastic Neighbor Embedding) — это метод нелинейного уменьшения размерности, хорошо подходящий для внедрения многомерных данных для визуализации в низкомерном двух- или трехмерном пространстве. Этот алгоритм использует некоторый рандомизированный подход для нелинейного уменьшения размерности имеющегося набора данных. При этом больше внимания уделяется сохранению локальной структуры набора данных и в нижнем измерении.

Это помогает нам исследовать многомерные данные, а также отображать их в меньших измерениях, поскольку локальные структуры сохраняются в наборе данных, и мы можем визуально заметить закономерности, построив их график и визуализировав их в двумерном или трехмерном пространстве.

Несмотря на то, что PCA и t-SNE являются алгоритмами без учителя, которые используются для уменьшения размерности набора данных, PCA — это детерминированный алгоритм для уменьшения размерности алгоритма, а алгоритм t-SNE — рандомизированный нелинейный метод для отображения многомерных данные в более низкое измерение (обычно двумерное или трехмерное). Данные, полученные после уменьшения размерности с помощью алгоритма t-SNE, обычно используются только для целей визуализации.

Преимущество использования данных t-SNE заключается в том, что на них не влияют выбросы, но выбросы сильно влияют на алгоритм PCA, поскольку методологии, используемые в этих двух алгоритмах, различны. Алгоритма PCA стремится сохранить дисперсию данных, а алгоритм t-SNE - локальную структуру набора данных.

t-SNE находит закономерности в данных на основе сходства точек данных с признаками, сходство точек рассчитывается как условная вероятность того, что точка A выберет точку B в качестве своего соседа.

Затем он пытается минимизировать разницу между этими условными вероятностями (или сходствами) в многомерном и низкомерном пространстве для идеального представления точек данных в низкомерном пространстве.

Алгоритм вычисляет попарные условные вероятности и пытается минимизировать сумму разности вероятностей в высших и низших измерениях. Это требует большого количества расчетов и вычислений. Таким образом, алгоритм требует много времени и места для вычислений. t-SNE имеет квадратичную временную и пространственную сложность по количеству точек данных [9].

Математическая интуиция [10]:

Первый этап — вычисление евклидовых расстояний каждой точки от всех остальных точек.

После этого расстояния преобразуются в условные вероятности, которые представляют сходство между всеми двумя точками. Таким образом мы хотим оценить, насколько похожи каждые две точки в данных или, другими словами, насколько вероятно, что они будут соседями.

Условная вероятность того, что точка x_j окажется рядом с точкой x_i, представлена гауссианом

с центром в x_i и стандартным отклонением σ_i. Математически это записывается следующим образом:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}$$

Причина деления на сумму всех остальных точек, помещенных в гауссиану с центром в x_i, заключается в том, что нам, возможно, придется иметь дело с кластерами с разной плотностью. Из созданных условных распределений мы рассчитываем совместное распределение вероятностей, используя следующее уравнение:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

Использование совместного распределения вероятностей вместо условной вероятности является одним из улучшений метода t-SNE по сравнению с прежним SNE. Свойство симметричности парных сходств ($p_{ij} = p_{ji}$) помогает упростить расчет на третьем этапе алгоритма.

Второй этап — Создание данных в низком измерении

На этом этапе мы создаем набор данных точек в низкоразмерном пространстве и также вычисляем для них совместное распределение вероятностей.

Для этого мы создаем случайный набор данных из точек с тем же количеством точек, что и в исходном наборе данных, и K объектов, где K — наше целевое измерение. Обычно K будет равно 2 или 3, если мы хотим использовать уменьшение размеров для визуализации.

Для этого набора точек мы создадим их совместное распределение вероятностей, но на этот раз мы будем использовать t-распределение, а не распределение Гаусса, как мы это делали для исходного набора данных. Это еще одно преимущество t-SNE по сравнению с прежним SNE (t в t-SNE означает t-распределение). Обозначим здесь вероятности q, а точки — y.

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}$$

Причиной выбора t-распределения вместо распределения Гаусса является свойство тяжелых хвостов t-распределения. Это качество приводит к тому, что умеренные расстояния между точками в многомерном пространстве становятся экстремальными в низкомерном пространстве, и это помогает предотвратить «скопление» точек в нижнем измерении. Еще одним преимуществом использования t-распределения является улучшение процесса оптимизации в третьей части алгоритма.

Третий этап — изменение набора данных в низкомерном пространстве.

Теперь используем расхождение Кульбака-Лейбера, чтобы сделать совместное распределение вероятностей точек данных в низком измерении максимально похожим на

распределение из исходного набора данных. Если это преобразование пройдет успешно, мы получим хорошее уменьшение размерности.

Расхождение Кульбака-Лейбера — это мера того, насколько два распределения отличаются друг от друга. Для распределений P и Q в вероятностном пространстве χ расхождение Кульбака-Лейбера определяется следующим образом:

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

В результате этой оптимизации мы получаем значения точек в наборе данных низкой размерности и используем их для визуализации.

Реализация алгоритма t-SNE

[11] см. Приложение 5

Список источников

1. A Survey of Dimensionality Reduction Techniques Based on Random Projection - Haozhe Xie, Jie Li, Hanqing Xue
2. A Review of Dimensionality Reduction Techniques for Efficient Computation - S.Velliangiria, S.Alagumuthukrishnanb , S Iwin Thankumar joseph
3. Generalized Low Rank Approximations of Matrices — Jieping Ye
4. A Step-by-Step Explanation of Principal Component Analysis (PCA) - https://builtin.com/data-science/step-step-explanation-principal-component-analysis
5. Principal Component Analysis(PCA) - https://www.geeksforgeeks.org/principal-component-analysis-pca/
6. Linear Discriminant Analysis in Machine Learning - https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/
7. Gaussian Discriminant Analysis - https://www.javatpoint.com/gaussian-discriminant-analysis
8. Gaussian Discriminative Analysis - https://colab.research.google.com/github/gmortuza/machine-learning-scratch/blob/master/machine_learning/bayesian/gaussian_discriminative_analysis/Gaussian%20Discriminative%20analysis.ipynb
9. ML | T-distributed Stochastic Neighbor Embedding (t-SNE) Algorithm - https://www.geeksforgeeks.org/ml-t-distributed-stochastic-neighbor-embedding-t-sne-algorithm/
10. T-SNE Explained — Math and Intuition - https://medium.com/swlh/t-sne-explained-math-and-intuition-94599ab164cf
11. t-SNE from Scratch (ft. NumPy) - https://towardsdatascience.com/t-sne-from-scratch-ft-numpy-172ee2a61df7
12. Kernel Fisher discriminant analysis - https://en.wikipedia.org/wiki/Kernel_Fisher_discriminant_analysis
13. Kernel FDA - https://github.com/concavegit/kfda