



Машинное обучение: теор. МИН

Блок 1

1. Сильный искусственный интеллект; слабый искусственный интеллект; анализ данных.

Сильный искусственный интеллект (общий) - искусственный интеллект, способный достичь или превзойти человеческие когнитивные способности.

Слабый искусственный интеллект (узкий) - искусственный интеллект, который реализует ограниченную часть разума или сосредоточен на одной узкой задаче. По мере решения задач ИИ их выписывают из задач общего ИИ и относят к узкому ИИ.

Машинное обучение - разработка алгоритмов машинного обучения

Анализ данных - извлечение знаний из данных, в т.ч. с помощью применения алгоритмов машинного обучения

2. Обучение с учителем; обучение без учителя; частичное обучение; обучение с подкреплением; активное обучение; онлайн обучение.

Все это - типы задач машинного обучения:

- **обучение с учителем** (supervised learning) - тренировочная выборка содержит верные ответы, которые алгоритм должен научиться предсказывать для новых данных. *Задачи, которые могут решаться этим способом:* классификация, вероятностная классификация, восстановление регрессии.
- **обучение без учителя** (unsupervised learning) - тренировочная выборка не содержит целевых признаков. Алгоритму требуется самому придумать новые признаки Y на основе существующих X . *Задачи, которые могут решаться этим способом:* кластеризация, мягкая кластеризация, извлечение признаков.
- **частичное обучение** (semi-supervised learning) - задача обучения с учителем, в которой только малая часть тренировочных данных содержит целевой признак. *Базовое решение:* не использовать объекты, у которых пропущен целевой признак, не использовать целевой признак для обучения. Размеченные объекты можно использовать для тестирования. Размеченные объекты могут статистически отличаться от неразмечанных.
- **обучение с подкреплением** (reinforcement learning) - агент взаимодействует со средой, сообщая ей действие для текущего состояния. Среда сообщает агенту награду за действие и новое состояние. Задача агента - максимизировать суммарную награду. (Эта задача больше похожа на то, как происходит обучение в реальном мире). Агент — это то, что имеет логику действий и взаимодействует с окружением.
- **активное обучение** (active learning) - Есть доступ к большому числу объектов, но не у всех есть метки. Данные собираются быстро, а размечиваются медленно и порционно, скорость обучения моделей происходит быстрее, чем разметка. В активном обучении условия такие же, как в частичном обучении, но можно задавать Оракулу вопросы о значении меток. Требуется восстановить $f: X \rightarrow Y$ за наименьшее число обращений к

Оракулу (найти стратегию обращений к Оракулу, оптимизирующую качество аппроксимации f).

- **онлайн обучение** (online learning) - эти алгоритмы имеют возможность дообучить модель на новых данных.

3. Классификация; регрессия; ранжирование; прогнозирование; метод обучения; функция потерь; эмпирический риск; метод минимизации эмпирического риска; гиперпараметры алгоритма.

Классификация - задача обучения с учителем, в которой типом целевого признака Y является категория. Этот признак называется классом, меткой класса, label, типом.

Наивное решение $a(x) = \text{Mode}[Y]$ (чаще всего встретившиеся решение)

Регрессия - задача обучения с учителем, в которой типом целевого признака Y является число.

Наивное решение $a(x) = E[Y]$

Прогнозирование:

- Дано множество значений y_1, y_2, \dots, y_t .
- Требуется предсказать y_{t+1}

Где брать X ?

- $x_t = (y_{t-m}, \dots, y_{t-2}, y_{t-1},)$ — **авторегрессия**.
- $x = f(t)$ — конструирование признаков.

Наивное решение

- Арифметическое скользящее среднее:

$$\hat{y}_t = (y_{t-m+1} + \dots + y_{t-1} + y_t) / m$$

- Экспоненциально взвешенное скользящее среднее:

$$\hat{y}_t = \alpha \cdot y_t + (1 - \alpha) \cdot \hat{y}_{t-1}$$

Ранжирование:

X — множество объектов.

$X^l = \{x_1, \dots, x_l\}$ — обучающая выборка.

$i \prec j$ — правильный **частичный порядок** на парах $(i, j) \in \{1, \dots, l\}^2$. "Правильность" зависит от постановки задачи, а именно запись $i \prec j$ может означать, что объект i имеет ранг выше, чем объект j , так и наоборот.

Задача:

Построить ранжирующую функцию $a : X \rightarrow \mathbb{R}$ такую, что: $i \prec j \Rightarrow a(x_i) < a(x_j)$.

Метод обучения:

Метод обучения – это отображение

$$\mu = \mu^{val} \cdot \mu^A: (X \times Y)^{\dim} \rightarrow A,$$

которое возвращает алгоритм $a \in A$ для набора данных $\mathcal{D} \in (X \times Y)^{\dim}$,
где $(X \times Y)^{\dim} = \bigcup_{i \in \dim N \subseteq \mathbb{N}} (X \times Y)^i$

μ^{val} выбирается и применяется независимо от μ^A (хотя они тесно связаны).

Функция потерь - функция, характеризующая величину отклонения ответа от правильного на произвольном объекте.

Эмпирический риск - функционал качества, характеризующий среднюю ошибку алгоритма a на выборке X^m

$$Q(a, X^m) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(a(x_i), y^*(x_i)).$$

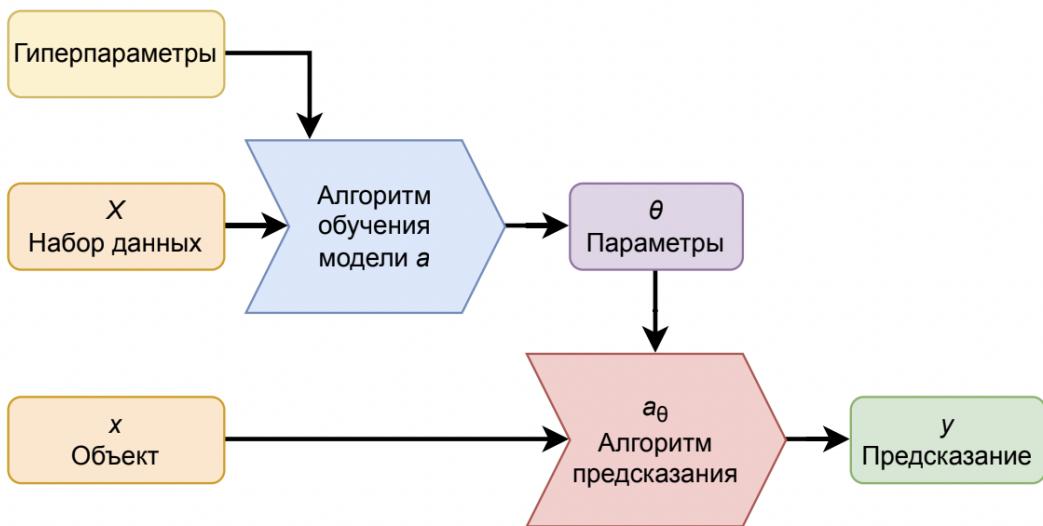
Метод минимизации эмпирического риска заключается в том, чтобы в заданной модели алгоритмов A найти алгоритм, при котором достигается минимальное значение функционала эмпирического риска.

Например найти такие гиперпараметры для алгоритма, чтобы потери (результат применения функции эмпирического риска) были минимальны (т.е. у ответов была минимальная погрешность).

$$a = \arg \min_{a \in A} Q(a, X^m).$$

Гиперпараметры алгоритма:

- Параметры алгоритма обучения
- Параметры, которые не меняются во время обучения



4. Переобучение; валидация; кросс-валидация; leave-one-out; регуляризация

Переобучение (англ. overfitting) — негативное явление, возникающее, когда алгоритм обучения вырабатывает предсказания, которые слишком близко или точно соответствуют конкретному набору данных и поэтому не подходят для применения алгоритма к дополнительным данным или будущим наблюдениям.

Валидация - способ проверки качества обучения нашей модели. Так мы определим что наша модель обучена нормально/переобучена/недообучена. Смысл валидации в том, чтобы выбрать модель, которая лучше всех обучается и меньше всех переобучается на имеющихся данных (или лучше всех обобщает на обучающую выборку). Не в том, чтобы обучить модель без переобучения.

Кросс-валидация - процедура эмпирической оценки обобщающей способности алгоритма. С помощью кросс-валидации эмулируется наличие тестовой выборки, которая не участвует в обучении, но для которой известны правильные ответы.

▼ Виды кросс-валидации:

- K-fold - алгоритм:
 1. Делим выборку на k примерно равных частей
 2. Цикл по $i=1, \dots, k$: используем i -ю часть для вычисления ошибки, объединение остальных для обучения.
 3. усредняем полученные k ошибок
- Leave-one-out - K-fold, где k=количество наблюдений в выборке
 - LeaveOneGroupOut - алгоритм:
 1. Делим выборку на несколько групп в данных (по значениям фичей или по каким-то содержательным соображениям)
 2. Цикл по $i=1, \dots$ количество выделенных групп: используем i -ю группу для вычисления ошибки, а остальные для обучения модели.
 3. Усредняем полученные ошибки (можно усреднять с весами, пропорциональными размеру группы в выборке, если есть необходимость)
 - Out-of-time-контроль - кросс-валидация для временных рядов

leave-one-out - вид кросс-валидации по отдельным объектам. В тестовой выборке находится один объект, в тренировочной — все остальные. Когда алгоритм обучается не на всех объектах, теряется часть информации. Здесь не теряется. Но это долго.

Регуляризация - метод добавления некоторых дополнительных ограничений к условию с целью предотвратить переобучение или повысить вербализуемость моделей. Чаще всего имеет вид штрафа за сложность модели.

Как пример, ограничение вектора шага в градиентном спуске.

▼ Виды регуляризации:

- L2-регуляризация
- L1-регуляризация
- Эластичная сеть - сумма двух предыдущих
- Гребневая регрессия

- Лассо регрессия
- и другие

5. Точность; полнота; accuracy; F-мера; true positive; false positive; ROC-кривая; среднеквадратическое отклонение

В бинарной классификации можно выделить два класса, пусть будут Positive/Negative.

В множественной классификации можно например сказать что один из них == Positive, все остальные вместе == Negative.

Верный ответ == алгоритм предсказал верно

- TP (True Positive) — Число верных ответов положительного класса
- TN (True Negative) — Число верных ответов отрицательного класса
- FP (False Positive) — Число ошибок первого рода
- FN (False Negative) — Число ошибок второго рода
- P = TP + FN - число положительных примеров
- N = FP + TN - число отрицательных примеров

Precision (точность) - доля действительно положительных объектов среди тех, кого отнесли к положительным. Формула: $\frac{TP}{TP+FP}$

Recall (Полнота) - доля положительных объектов, которую алгоритм находит. Формула: $\frac{TP}{P}$

Accuracy (точность) - определяет на сколько хорошо работает классификатор. Формула: $\frac{TP+TN}{P+N}$

F-мера - среднее гармоническое между точностью и полнотой. Формула после преобразований: $\frac{2 \times Precision \times Recall}{Precision + Recall}$. Кроме того, может являться заменой accuracy, если классы несбалансированы.

ROC-кривая - зависимость TPR (sensitivity - True Positive Rate, равна Recall) от TNR (specificity - True Negative Rate, равна $\frac{TN}{TN+FP}$). То есть ROC-кривая – графическая характеристика качества бинарного классификатора, зависимость доли верных положительных классификаций от доли ложных положительных классификаций при варьировании порога решающего правила.

Площадь под ROC кривой (AUC — Area Under Curve) — функция качества бинарной классификации.

Среднеквадратичная ошибка (Mean Squared Error) – Среднее арифметическое квадратов разностей между предсказанными и реальными значениями Модели Машинного обучения

$$MSE = \frac{1}{n} \times \sum_{i=1}^n (y_i - \tilde{y}_i)^2$$

n - количество наблюдений

y_i - фактическая координата наблюдения

\tilde{y}_i с крышкой - предсказанная координата наблюдения

Также есть Root Mean Squared Error (RMSE) - корень из MSE

Если уж на то пошло, еще есть MAPE и SMAPE - (Symmetric) Mean Absolute Percentage Error

НО у них есть проблемы с антисимметричностью

$$M = \frac{1}{n} \sum_{t=1}^n \left| \frac{A_t - F_t}{A_t} \right|$$

A_t - фактическое значение, F_t - предсказанное значение

$$SMAPE = \frac{1}{n} \sum_{t=1}^n \frac{|F_t - A_t|}{(A_t + F_t)/2}$$

6. Метод ближайших соседей

Метод ближайших соседей — простейший метрический классификатор (Метрический классификатор — алгоритм классификации, основанный на вычислении оценок сходства между объектами.), основанный на оценивании сходства объектов. Классифицируемый объект относится к тому классу, которому принадлежат ближайшие к нему объекты обучающей выборки.

Отсортируем объекты:

$$p(u, x_{(u,1)}) \leq p(u, x_{(u,2)}) \leq \dots \leq p(u, x_{u,|D_{train}|}).$$

Алгоритм (возвращаем тот класс, которому принадлежит большинство соседей):

$$a_{kNN}(u; D_{train}) = \operatorname{argmax}_{y \in Y} \sum_{i=1}^k [y(x_{(u,i)}) = y]$$

Что еще есть для повышения надежности:

Метод k ближайших соседей — Для повышения надёжности классификации объект относится к тому классу, которому принадлежит большинство из его соседей — k ближайших к нему объектов обучающей выборки x_i (x итый). В задачах с двумя классами число соседей берут нечётным, чтобы не возникало ситуаций неоднозначности, когда одинаковое число соседей принадлежат разным классам.

Метод взвешенных ближайших соседей — в задачах с числом классов 3 и более нечётность уже не помогает и ситуации неоднозначности всё равно могут возникать. Тогда i-му соседу приписывается вес w_i (w итый), как правило, убывающий с ростом ранга соседа i. Объект относится к тому классу, который набирает больший суммарный вес среди k ближайших соседей.

7. Метод непараметрической регрессии

Непараметрическая регрессия - в окрестности некоторой точки θ (тета) - константа, пользуясь формулой эмперического риска, пытаемся устремить ее (формулу) к минимуму по всем тета. Для этого будем использовать ядерное сглаживание. В формуле ядерного сглаживания присутствует функция ядра k и h - ширина окна, пользуясь формулой Надарая-Ватсона считаем скользящее среднее

Грубо говоря: Регрессия, которая не описывается конечным числом параметров.

Непараметрическое название весьма условно, так как гиперпараметры есть, которые можно и нужно настраивать

Предположение: пусть $\theta(x) = \theta$ вокруг $x \in X$:

$$\mathcal{L}(\theta, \mathcal{D}_{train}) = \sum_{i=1}^{|\mathcal{D}_{train}|} w_i(x)(\theta - y_i)^2 \rightarrow \min_{\theta \in \mathbb{R}}$$

Основная идея: будем использовать ядерное сглаживание:

$$w_i(x) = K\left(\frac{\rho(x_i, x)}{h}\right),$$

где h – ширина окна.

Ядерное сглаживание Надарадя-Ватсона:

$$\begin{aligned} a_{\text{NonParamReg}h}(x, \mathcal{D}_{train}) &= \frac{\sum_{x_i \in \mathcal{D}_{train}} y_i w_i(x)}{\sum_{x_i \in \mathcal{D}_{train}} w_i(x)} = \\ &= \frac{\sum_{x_i \in \mathcal{D}_{train}} y_i K\left(\frac{\rho(x_i, x)}{h}\right)}{\sum_{x_i \in \mathcal{D}_{train}} K\left(\frac{\rho(x_i, x)}{h}\right)}. \end{aligned}$$

8. Метод стохастического градиентного спуска

Стохастический градиентный спуск - оптимизационный алгоритм, отличающийся от обычного градиентного спуска тем, что градиент оптимизируемой функции считается на каждом шаге не как сумма градиентов от каждого элемента выборки, а как градиент от одного,

случайно выбранного элемента. (Чтобы доказать что стохастический метод сходится, используется теорема Новикова)

Градиентный спуск — метод нахождения локального минимума или максимума функции с помощью движения вдоль градиента (градиент - вектор, своим направлением указывающий направление возрастания (а антиградиент - убывания) некоторой скалярной величины, значение которой меняется от одной точки пространства к другой, образуя скалярное поле, а по величине (модулю) равный скорости роста этой величины в этом направлении.). Для минимизации функции в направлении градиента используются методы одномерной оптимизации, например, метод золотого сечения. Также можно искать не наилучшую точку в направлении градиента, а какую-либо лучше текущей.

Наиболее простой в реализации из всех методов локальной оптимизации. Имеет довольно слабые условия сходимости, но при этом скорость сходимости достаточно мала (линейна). Шаг градиентного метода часто используется как часть других методов оптимизации, например, метод Флетчера — Ривса.

9. Метод линейной регрессии; гребневая регрессия; лассо Тибширани

Метод линейной регрессии - метод восстановления зависимости одной (объясняемой, зависимой) переменной у от другой или нескольких других переменных (факторов, регрессоров, независимых переменных) x с линейной функцией зависимости. Данный метод позволяет предсказывать значения зависимой переменной у по значениям независимой переменной x.

Гребневая регрессия или ридж-регрессия — один из методов понижения размерности. Применяется для борьбы с избыточностью данных, когда независимые переменные коррелируют друг с другом, вследствие чего проявляется неустойчивость оценок коэффициентов многомерной линейной регрессии. (у значений вектора распределение гаусса)

Метод стоит использовать, если:

- сильная обусловленность;
- сильно различаются собственные значения или некоторые из них близки к нулю;
- в матрице X есть почти линейно зависимые столбцы.

Метод регрессии лассо - похож на гребневую регрессию, но он использует другое ограничение на коэффициенты (у значений вектора распределение лапласа)

Основное различие лассо- и ридж-регрессии заключается в том, что первая может приводить к обращению некоторых независимых переменных в ноль, тогда как вторая уменьшает их до значений, близких к нулю.

10. Метод сингулярного векторного разложения

Теорема: Любая матрица F размера $|D| \times n$ может быть представлена в виде **сингулярного разложения**:

$$F = VDU^T,$$

где

- V — матрица $|D| \times n$, где столбцы v_j — собственные векторы матрицы FF^T : $V = (v_1, \dots, v_n)$, является ортогональной ($V^T V = I_n$);
- U — матрица $n \times n$, где строки u_j — собственные векторы $F^T F$, является ортогональной ($U^T U = I_n$);
- $D = diag(\sqrt{\lambda_1}, \dots, \sqrt{\lambda_n})$ размера $n \times n$, $\sqrt{\lambda_j}$ — **сингулярные числа** (т.е. квадраты собственных значений $F^T F$).

Представим какое-то скрытое пространство, в которое мы хотим проецировать данные (матрицу F). Это пространство можно описать базисными векторами. D представляет важность каждого базисного вектора, V представляет, как объекты соответствуют базисным векторам, U показывает, как признаки соответствуют базисным векторам.

С помощью этого метода можно ускорить классический метод наименьших квадратов. Кроме того, **сингулярное разложение** — важный инструмент во многих других задачах машинного обучения, в первую очередь, в снижении размерности.

11. Метод опорных векторов

Метод опорных векторов - линейный алгоритм

Основная идея заключается в построении гиперплоскости, разделяющей объекты выборки оптимальным способом. Алгоритм работает в предположении, что чем больше расстояние (зазор) между разделяющей гиперплоскостью и объектами разделяемых классов, тем меньше будет средняя ошибка классификатора.

Преимущества SVM перед методом стохастического градиента и нейронными сетями:

- Задача выпуклого квадратичного программирования хорошо изучена и имеет единственное решение.
- Метод опорных векторов эквивалентен двухслойной нейронной сети, где число нейронов на скрытом слое определяется автоматически как число опорных векторов.
- Принцип оптимальной разделяющей гиперплоскости приводит к максимизации ширины разделяющей полосы, а следовательно, к более увереной классификации.

Недостатки классического SVM:

- Неустойчивость к шуму: выбросы в исходных данных становятся опорными объектами-нарушителями и напрямую влияют на построение разделяющей гиперплоскости.
- Не описаны общие методы построения ядер и спрямляющих пространств, наиболее подходящих для конкретной задачи.
- Нет отбора признаков.
- Необходимо подбирать константу С при помощи кросс-валидации

12. Ядро; ядерный трюк для метода опорных векторов

Ядро - это непрерывная ограниченная симметричная вещественная функция с единичным интегралом, представляемая в виде скалярного произведения.

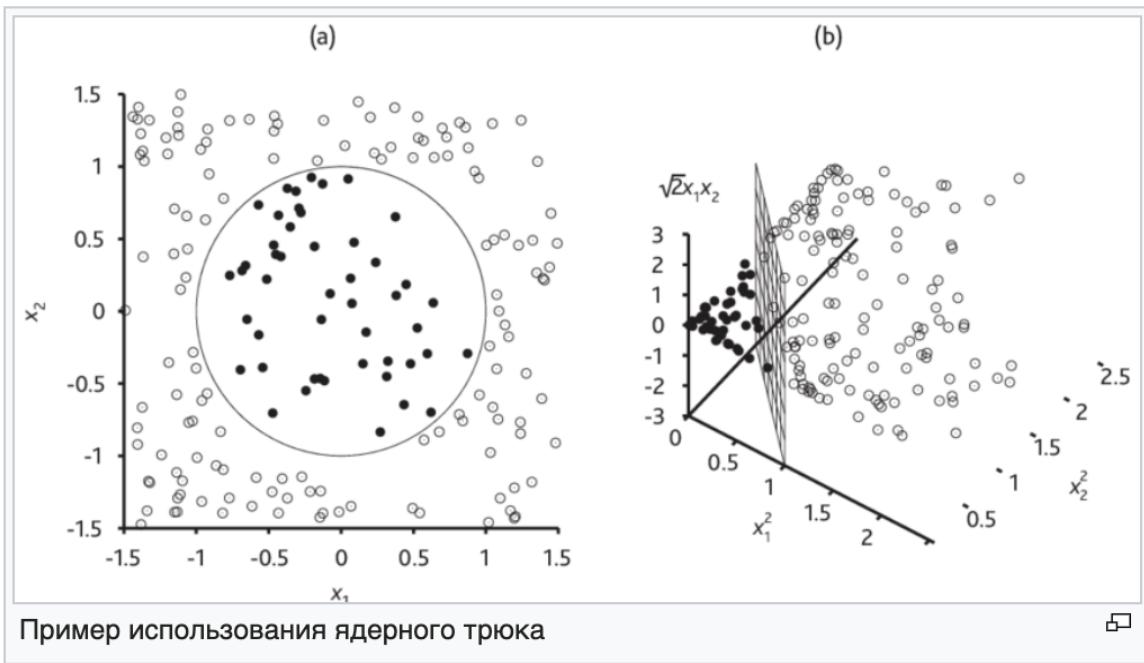
Ядро содержит в себе много информации о спрямляющем пространстве, и позволяет производить в нем различные операции, не зная самого отображения $\phi(x)$.

Ядерный трюк - метод в машинном обучении, позволяющий перевести элементы для случая линейной неразделимости в спрямляющее (т.е. линейно разделимое) пространство. Поскольку для любой непротиворечивой выборки соответствующее пространство большей размерности существует, главной проблемой становится его найти.

Основная идея:

- найти неявное отображение в многомерное пространство, такое, что точки в новом пространстве будут линейно разделимы.
- вычислять ядро быстрее, чем скалярное произведение с явным преобразованием.

Пример: На первой картинке справа можно увидеть, что 2 класса неразделимы линейно, но после преобразования появляется разделяющая плоскость.



13. Вероятностная постановка задачи классификации; правдоподобие класса; метод максимальной апостериорной вероятности; оптимальный байесовский классификатор

Вероятностная постановка задачи классификации:

Вместо неизвестной целевой функции $y^*(x)$ будем думать о неизвестном распределении над $X \times Y$ с плотностью $p(x, y)$.

Простой или независимой одинаково распределенной (independent identically distributed, i.i.d.) называется выборка, содержащая независимые наблюдения из одного распределения.

Задача: найти алгоритм классификации, который минимизирует вероятность ошибки.

Разница с остальным: восстанавливаем именно плотность распределения, а не функцию $x \rightarrow Y$

Почему классификация вероятностная: потому что объект может принадлежать и одному, и другому классу, при том что описание этого объекта одинаковое, то есть вектор признаков, который описывает объект одинаковый, но при этом объект может принадлежать разным классам, поэтому мы и хотим восстановить области распределения, чтобы понимать куда отнести тот или иной объект.

Правдоподобие класса:

$$p(X, Y) = p(x) \Pr(y|x) = \Pr(y) p(x|y) \quad p - \text{правдоподобие}, \Pr - \text{вероятность}$$

$\Pr(y)$ — априорная вероятность класса y : какова вероятность того, что мы можем встретить в выборке объект класса y

$p(x|y)$ — правдоподобие (likelihood) класса y (плотности распределения называются функциями правдоподобия классов)

$\Pr(y|x)$ — апостериорная вероятность класса y : насколько вероятно, что у x будет класс y . С точки зрения задачи классификации это то, что мы и будем пытаться найти, чтобы найти плотность.

Метод максимальной апостериорной вероятности. Основная идея: для нового объекта будем возвращать класс, к которому он принадлежит с наибольшей вероятностью. Принцип выражается через оценку апостериорного максимума: нужно найти такой y , при значении которого $\Pr(y|x)$ (апостериорная вероятность) будет максимальна

$$a(x) = \operatorname{argmax}_{y \in Y} \Pr(y|x) = \operatorname{argmax}_{y \in Y} \Pr(y) p(x|y).$$

Оптимальный байесовский классификатор - формула такая же как при оценки апостериорного максимума, но добавляется штраф на каждый класс.

$$a_{OB}(x) = \operatorname{argmax}_{y \in Y} \lambda_y \Pr(y) p(x|y).$$

То есть величина штрафа для ошибки первого рода должна быть больше, чем для величины штрафа для ошибки второго рода

Теорема

Если известны $\Pr(y)$ и $p(x|y)$, то минимальный средний риск достигается байесовским классификатором a_{OB}

$$a_{OB}(x) = \operatorname{argmax}_{y \in Y} \lambda_y \Pr(y) p(x|y).$$

Классификатор $a_{OB}(x)$ называется **оптимальным байесовским классификатором**, а достигаемое им значение $R(a_{OB})$ – **байесовским риском**.

где $R(a_{OB})$ - средний риск

14. Метод оценки Парзена-Розенблатта; наивный байесовский классификатор

Метод оценки Парзена-Розенблатта - Алгоритм классификации, в основе которого лежит вероятностная оценка плотности Парзена-Розенблатта. Можно оценить плотность распределения для каждого класса, а затем найти плотность объекта для каждого класса и взять наибольшую. Используется в методе kNN для задания весовой функции.

Алгоритм kNN можно обобщить с помощью функции ядра , с помощью фиксированной и переменной ширины окна.

$$w(i, u) = K\left(\frac{\rho(u, x_{i;u})}{h}\right) \text{ – метод парзеновского окна фиксированной ширины } h;$$

$$w(i, u) = K\left(\frac{\rho(u, x_{i;u})}{\rho(u, x_{k+1;u})}\right) \text{ – метод парзеновского окна переменной ширины};$$

Сравним два этих метода. Сперва запишем классификаторы, полученные при использовании этих методов, в явном виде:

$$\text{Фиксированной ширины: } a_h = a(u, X^m, h, K) = \arg \max_{y \in Y} \sum_{i=1}^m [y_{i;u} = y] K\left(\frac{\rho(u, x_{i;u})}{h}\right)$$

$$\text{Переменной ширины: } a_k = a(u, X^m, k, K) = \arg \max_{y \in Y} \sum_{i=1}^m [y_{i;u} = y] K\left(\frac{\rho(u, x_{i;u})}{\rho(u, x_{k+1;u})}\right)$$

Классификатор фиксированной ширины не будет учитывать соседей на расстояние больше чем h , а всех остальных учет в соответствии с функций

ядра K .

Классификатор переменной ширины является аналогом метода k ближайших соседей (т.к. для всех $k+i$ -ых соседей функция K вернет 0), но при этом чем ближе $k-i$ -ый сосед, тем больший вклад в сторону своего класса он даст.

Чаще используют Классификатор переменной ширины, потому что удобнее оптимизировать целочисленный параметр k , чем вещественный параметр h по некоторой сетке. Плюс существует большое количество задач, где точки разбросаны неравномерно, тогда Классификатор фиксированной ширины ломается, не захватывая в некоторые области ни одного объекта.

Наивный байесовский классификатор - алгоритм, который делает предположение, что все переменные в наборе данных "наивные", т.е. не коррелируют друг с другом. Он своей основе использует Теорему Байеса, которая позволяет рассчитать апостериорную вероятность $P(A | B)$ на основе $P(A)$, $P(B)$ и $P(B | A)$. В основном используется для получения базовой точности набора данных.

Плюсы

- Алгоритм легко и быстро предсказывает класс тестового набора данных. Он также хорошо справляется с многоклассовым прогнозированием.
- Производительность наивного байесовского классификатора лучше, чем у других простых алгоритмов, таких как логистическая регрессия. Более того, вам требуется меньше обучающих данных.
- Он хорошо работает с категориальными признаками(по сравнению с числовыми). Для числовых признаков предполагается нормальное распределение, что может быть серьезным допущением в точности нашего алгоритма.

Минусы

- Если переменная имеет категорию (в тестовом наборе данных), которая не наблюдалась в обучающем наборе данных, то модель присвоит 0 (нулевую) вероятность и не сможет сделать предсказание. Это часто называют нулевой частотой. Чтобы решить эту проблему, мы можем использовать технику сглаживания. Один из самых простых методов сглаживания называется оценкой Лапласа.
- Значения спрогнозированных вероятностей, возвращенные методом *predict_proba*, не всегда являются достаточно точными.

- Ограничением данного алгоритма является предположение о независимости признаков. Однако в реальных задачах полностью независимые признаки встречаются крайне редко.

15. Задача параметрической оценки плотности; принцип максимального правдоподобия

Задача параметрической оценки плотности: задача восстановления плотности распределения, основанная на предположении о том, что плотность $p(x)$ известна с точностью до параметра θ , и является фиксированной функцией $\phi(x, \theta)$. Параметр θ можно оценить по выборке с использованием принципа максимального правдоподобия.

В чем смысл: нам надо найти такие θ , при которых функция правдоподобия достигает максимума.

Производная равна нулю: такие вещи (оптимум) ищутся при помощи градиентного спуска

Принцип максимального правдоподобия:

$$\mathcal{L}(\theta; \mathcal{D}_S) = \sum_{x \in \mathcal{D}_S} \ln \varphi(x; \theta) \rightarrow \max_{\theta},$$

Оптимум θ можно найти в точке, где производная

$$\frac{\partial \mathcal{L}(\theta; \mathcal{D}_S)}{\partial \theta} = 0.$$

Это чаще всего можно искать только итеративно.

Принцип совместного максимального правдоподобия:

Пытаемся построить распределение параметров нашего распределения, которое мы хотим найти.

$$\mathcal{L}(a_\theta, \mathcal{D}) = - \sum_{(x_i, y_i) \in \mathcal{D}} \ln \varphi(x_i, y_i, \theta) \rightarrow \min_{\theta} .$$

$$\varphi(x_i, y_i, \theta) = p(x_i, y_i | w)p(w, \gamma),$$

$p(x_i, y_i | w)$ – вероятностная модель данных,

$p(w, \gamma)$ – априорное распределение параметров модели,

γ – гиперпараметр (в статистическом смысле).

Принцип совместного максимального правдоподобия:

$$\sum_{(x_i, y_i) \in \mathcal{D}} \ln p(x_i, y_i | w) + \ln p(w, \gamma) \rightarrow \max_{w, \gamma}$$

16. Метод логистической регрессии; сигмоида

Метод логистической регрессии - Это алгоритм для решения задачи бинарной классификации (бинарная классификация — это частный случай классификации, когда классов всего два: "0" или "1". Говорят, что целевая переменная здесь — **бинарная величина**).

Модели, обученные алгоритмами для бинарной классификации, могут не просто прогнозировать финальное значение класса для какого-то объекта или клиента, а ещё и оценивать вероятность рассматриваемого события.

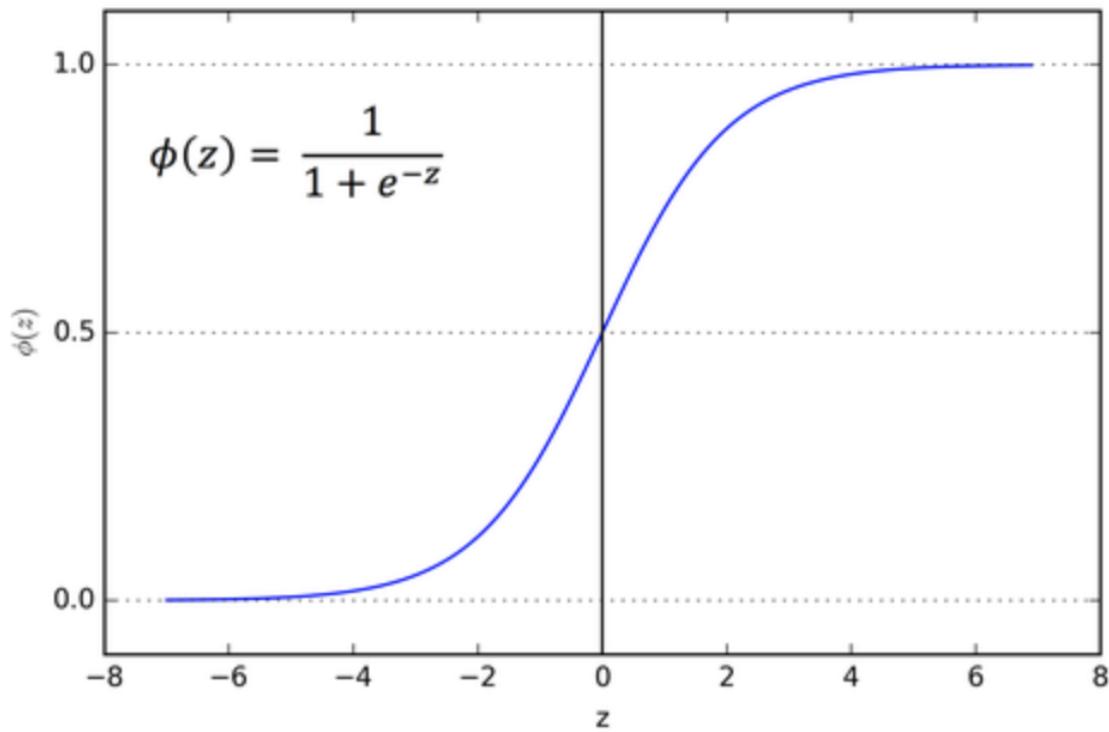
Например:

- вероятность возврата кредита клиентом;
- вероятность возникновения страхового случая;

Алгоритмом для решения таких задач будет логистическая регрессия.

График логистической функции называют **сигмоида**. По нему видно, что логистическая функция принимает значения от 0 до 1. Внешний вид и область допустимых значений сигмоиды хорошо подходит для представления

вероятности наступления какого-то события. Эта кривая «загибается» на бесконечно маленьких и бесконечно больших значениях z .



18. Дерево принятия решений; подрезка

Деревья выбирают ответ сценарным подходом, в отличии от классификации, которая представляет значение целевой функции — как сумму взвешенных значений признаков (линейной функции).

Последовательностью признаков алгоритм пытается так разбить выборку, чтобы группы в конце дерева (в листьях) были максимально отделены на основании наблюдаемых данных.

Чем деревья хороши:

- Они легко интерпретируемые. В бизнесе это часто становится ключевым. Более сложные алгоритмы напоминают «чёрный ящик» и неясно, по какому именно принципу они принимают решения.
- Годятся и для классификации, и для регрессии — в этом случае в листьях дерева будут числа, а не значение класса.
- Достаточно быстры.

- Не требуют серьёзной предобработки данных: деревья не чувствительны к масштабу признаков и устойчивы к их мультиколлинеарности.

Чем деревья плохи:

- Они сильно переобучаются. Это значит, что модель слишком сильно подгонит параметры или разбиение веток под обучающую выборку. На *train* она будет работать очень хорошо, а вот на новых данных нет. Иногда при большом количестве признаков вы можете строить ветку так, что в листе будет один объект, или совсем немного. По сути дерево подстроится под конкретную обучающую выборку, пытаясь оставить в листе объекты лишь одного класса.
- Только верхние вершины влияют на качество алгоритма.

Чтобы избежать переобучения, проводят процедуру «обрезки» дерева — **prunning - обработка созданных деревьев, когда последовательно применяются операторы упрощения, если они улучшают качество по определенному критерию.**

Она заключается в том, чтобы в определённый момент обрезать ветку, не позволяя дереву переобучаться. Эффект переобучения особенно заметен на небольших выборках.

Дерево решений - алгоритм классификации и регрессии. Вершины содержат разделяющие правила (вопросы). Ребро - возможный ответ на вопрос в родительской вершине. Листья содержат решения (класс объекта для задачи классификации и число для задачи регрессии).

Алгоритм примерно такой:

1. Разбить выборку в отношении 2 к 1 на тренировочную и тестовую (валидационную).
2. Для каждой вершины применяем определенный оператор упрощения, который является лучшим с точки зрения выбранного критерия качества:
 - ничего не менять
 - заменить вершину ребенком (для каждого ребенка)
 - заменить вершину листом (для каждого класса)

19. Бустинг алгоритмов; метод градиентного бустинга

Бустинг (англ. boosting — улучшение) — это процедура последовательного построения композиции алгоритмов машинного обучения, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов. На каждом шаге новая модель обучается с использованием данных об ошибках предыдущих.

Одним из недостатков бустинга является то, что он может приводить к построению громоздких композиций, состоящих из сотен алгоритмов. Такие композиции исключают возможность содержательной интерпретации, требуют больших объёмов памяти для хранения базовых алгоритмов и существенных затрат времени на вычисление классификаций.

Градиентный бустинг - продвинутый алгоритм машинного обучения для решения задач классификации и регрессии. Он строит предсказание в виде ансамбля слабых предсказывающих моделей, которыми в основном являются деревья решений. Из нескольких слабых моделей собираем одну эффективную. Общая идея: последовательное применение предиктора таким образом, что каждая следующая модель сводит ошибку предыдущей к минимуму.

20. Метод AdaBoost

Один из алгоритмов классификации, в процессе обучения строит композицию из базовых алгоритмов обучения для улучшения их эффективности, каждый следующий классификатор **строится (в основном) по объектам, которые плохо классифицируются предыдущими классификаторами**.

То есть мы больше штрафуем алгоритм за ошибки на объектах с большими весами

Алгоритм может использоваться в сочетании с несколькими алгоритмами классификации для улучшения их эффективности. Алгоритм усиливает классификаторы, объединяя их в «комитет». AdaBoost является адаптивным в том смысле, что каждый следующий комитет классификаторов строится по объектам, неверно классифицированным предыдущими комитетами. AdaBoost

чувствителен к шуму в данных и выбросам. Однако он менее подвержен переобучению по сравнению с другими алгоритмами машинного обучения.

AdaBoost вызывает слабые классификаторы. После каждого вызова обновляется распределение весов, которые отвечают важности каждого из объектов обучающего множества для классификации. На каждой итерации веса каждого неверно классифицированного объекта возрастают, таким образом новый комитет классификаторов «фокусирует своё внимание» на этих объектах.

Комитет = ансамбль — серьёзные модели машинного обучения. Они мощные и позволяют отражать сложные зависимости между данными. Однако за « силу » ансамблей приходится платить их интерпретируемостью.

Такие алгоритмы часто называют «чёрными ящиками»: по ним трудно понять, какие же признаки повлияли на прогноз для конкретного наблюдения.

21. Бутстреп; случайный лес; стэкинг

Бутстреп - подход, при котором оценивают различные характеристики или прогнозы на основании множества различных подвыборок из исходной. Используется для бэггинга (усреднение моделей, обученных на разных подвыборках). Основан на многократной генерации выборок на базе уже имеющейся. Используется в алгоритме "Случайный лес".

Случайный лес - алгоритм, в основе которого лежит бэггинг. Учим деревья решений на случайно выбранной части данных (обычно \sqrt{n} , n - число признаков), а затем усредняем их результаты.

Стэкинг - ансамблевый метод машинного обучения, который на части тренировочных данных обучает базовые модели, а затем обучает метамодель на основе предсказаний базовых моделей на оставшейся части тренировочной выборки.

Блок 2

22. Нейрон; перцептрон;

Нейрон - клетка, имеющая множество отростков (дендритов), накапливает сигналы от дендритов, возбуждается и передает сигнал дальше. Один из дендритов (аксон) как раз и передает этот сигнал дальше.

Перцептрон - простейший вид нейронных сетей. В основе лежит математическая модель восприятия информации мозгом, состоящая из сенсоров, ассоциативных и реагирующих элементов. Таким образом, перцептроны позволяют создать набор “ассоциаций” между входными стимулами и необходимой реакцией на выходе.

Перцептрон это input values, weights, sum, activation function (все элементы вместе - модель нейрона)

Функция активации определяет выходное значение перцептрана и применяется после его вычисления. Например, \tanh или сигмоида.

23. Многослойная нейронная сеть; автоматическое дифференцирование

Многослойная нейронная сеть - по сути это матричные вычисления. На входе у нас есть набор нейронов (скалярных произведений), каждый слой это множество скалярных произведений, т.е. произведение матриц. Таким образом можно представить нашу структуру так: получает на вход вектор, его умножает на матрицу, с полученным вектором производят сдвиг и другие операции и идёт дальше: Input \rightarrow * M₁ + b₁ \rightarrow * M₂ \rightarrow f() \rightarrow Output .

Если мы будем выстраивать из искусственных нейронов сеть, которая будет состоять из слоев, то так как нейрон по сути - скалярное произведение, а множество скалярных произведений - умножение на матрицу, тогда нашу послойную архитектуру можно выписать, как функцию, которая поочередно умножает текущий вектор на какую то матрицу, дальше прибавляет к полученному вектору какой то вектор сдвига, дальше применяет к этому какую то не линейную функцию.

Полученная функция выглядит в виде дерева.

Автоматическое дифференцирование – метод обучения нейросетей, основанный на двух шагах:

1. Прямое распространение (forward propagation) – процесс пропуска входных данных через нейросеть, попутно сохраняя промежуточные ответы и значение функции ошибки;

- Обратное распространение (backward propagation) – процесс подсчёта градиентов с помощью chain rule.

Благодаря автоматическому дифференцированию можно обучать нейросети любой сложности, так как функции будут раскладываться в элементарные с помощью chain rule. Более того, мы не будем делать подсчёт лишних операций, так как градиенты с верхних слоёв будут сохраняться и утилизироваться далее.

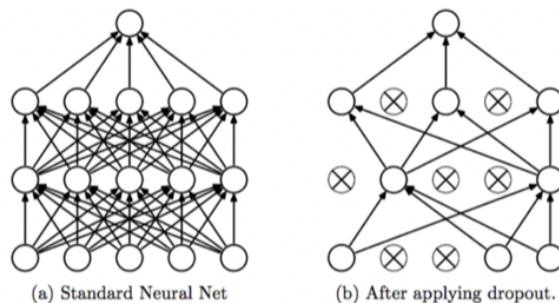
25. Аугментация данных; дропаут; ReLU

Аргументация данных - техники генерации данных с помощью добавления копий уже существующих с небольшими изменениями (или генерации на основе существующих данных), то есть это методика создания дополнительных данных из имеющихся данных. Чаще всего, проблема ограниченного набора данных возникает при решении задач, связанных с обработкой изображений.

Дропаут:

Дропаут: при обучении на итерации обнуляем выходы нейронов с некоторой вероятностью (часто 0,5)

- Обнуленные нейроны не участвуют в обучении, не внося вклад в ошибку
- Для каждого выхода мы фактически строим новую сеть, но у всех этих сетей общие параметры



Делается для избежания переобучения.

ReLU

Rectified Linear Unit — это наиболее часто используемая функция активации при глубоком обучении. Данная функция возвращает 0, если

принимает отрицательный аргумент, в случае же положительного аргумента, функция возвращает само число. То есть она может быть записана как $f(z) = \max(0, z)$. На первый взгляд может показаться, что она линейна и имеет те же проблемы что и линейная функция, но это не так и ее можно использовать в нейронных сетях с множеством слоев. Функция ReLU обладает некоторыми преимуществами перед сигмоидой и гиперболическим тангенсом:

1. Очень быстро и просто считается производная. Для отрицательных значений — 0, для положительных — 1.
2. Разреженность активации. В сетях с очень большим количеством нейронов использование сигмоидной функции или гиперболического тангенса в качестве активационной функции влечет активацию почти всех нейронов, что может оказаться на производительности обучения модели. Если же использовать ReLU, то количество включаемых нейронов станет меньше, в силу характеристик функции, и сама сеть станет легче.

У данной функции есть один недостаток, называющийся проблемой умирающего ReLU. Так как часть производной функции равна нулю, то и градиент для нее будет нулевым, а то это значит, что веса не будут изменяться во время спуска и нейронная сеть перестанет обучаться.

Функцию активации ReLU следует использовать, если нет особых требований для выходного значения нейрона, вроде неограниченной области определения. Но если после обучения модели результаты получились не оптимальные, то стоит перейти к другим функциям, которые могут дать лучший результат.

26. Метод Ньютона-Рафсона

Основная идея состоит в последовательных приближениях к истинному решению уравнения $f(x)=0$, которые вычисляются с помощью производной от $f(x)$.

- Вводится нулевая итерация x_0 .
- В этой точке методом конечных разностей вычисляется производная $f'(x)$.
- Пользуясь разложением Тейлора, $f(x)$ заменяется в окрестности точки касательной - прямой линией $f(x_0) + f'(x_0)(x - x_0)$.
- Определяется точка, в которой прямая пересекает ось X.
- Эта точка принимается за новую итерацию и цикл повторяется, пока корень не будет найден с нужной точностью.

27. Декорреляция; метод Xavier; метод Не

Декорреляция - Процесс, который используется для уменьшения корреляции.

1. Есть матрица X .
2. Матрицу центрировали ($E[X_j] = 0$).
3. Ковариация вычисляется по следующей формуле:

$$\Sigma(X) = \frac{1}{N} X^T X$$

4. Если же матрица нормализована так, что $D[X_j] = 1$, то из произведения мы получим не ковариационную, а корреляционную матрицу
5. Декорреляция вычисляется по формуле:

$$\hat{X} = X \times \Sigma^{-1/2}(X)$$

где $\Sigma^{1/2}$ находится из разложения Холецкого

Утверждение:

После декорреляции: $\Sigma(\hat{X}) = I$

▷

$$\Sigma = \frac{X^T X}{n}$$

$$\hat{X} = X \times \Sigma^{-1/2}$$

$$\frac{\hat{X}^T \hat{X}}{n} = \frac{(X \times \Sigma^{-1/2})^T \times (X \times \Sigma^{-1/2})}{n} = \frac{\Sigma^{-T/2} \times X^T \times X \times \Sigma^{-1/2}}{n} = (\Sigma^{-T/2} \times \Sigma^{T/2}) \times (\Sigma^{1/2} \times \Sigma^{-1/2}) = I \times I = I$$

△

Метод Xavier - Метод инициализации сети. Основная идея этого метода — упростить прохождение сигнала через слой во время как прямого, так и обратного распространения ошибки для линейной функции активации (этот метод также хорошо работает для сигмоидной функции, так как участок, где она ненасыщена, также имеет линейный характер). При вычислении параметров

этот метод опирается на вероятностное распределение (равномерное или нормальное) с дисперсией, равной

$$\text{Var}(W) = \frac{2}{n_{in} + n_{out}}, \text{ где } n_{in} \text{ и } n_{out} — \text{ количества нейронов в предыдущем и последующем слоях соответственно;}$$

Метод Ге (He) — вариация метода Завьера, больше подходящая функции активации ReLU, компенсирующая тот факт, что эта функция возвращает нуль для половины области определения. А именно, в этом случае

$$\text{Var}(W) = \frac{2}{n_{in}}$$

Инициализация — это процесс установки настраиваемых параметров для нашей глубокой сети. Выбор правильного метода инициализации важен для качества обучения нашей модели. Также это позволяет сократить время сходимости и минимизировать функцию потерь. Поэтому важно уметь выбрать правильный метод инициализации.

Принцип выбора начальных значений параметров для слоев, составляющих модель очень важен: установка всех параметров в 0 будет серьезным препятствием для обучения, так как ни один из параметров изначально не будет активен. Присваивать параметрам значения из интервала $[-1, 1]$ — тоже обычно не лучший вариант — на самом деле, иногда (в зависимости от задачи и сложности модели) от правильной инициализации сети может зависеть, достигнет она высочайшей производительности или вообще не будет сходиться. Даже если задача не предполагает такой крайности, удачно выбранный способ инициализации начальных параметров может значительно влиять на способность модели к обучению, так как он предустанавливает параметры модели с учетом функции потерь.

30. Метод Adam

Adam - один из самых эффективных алгоритмов оптимизации в обучении нейронных сетей. Он сочетает в себе идеи RMSProp и оптимизатора импульса. Вместо того чтобы адаптировать скорость обучения параметров на основе среднего первого момента (среднего значения), как в RMSProp, Adam также использует среднее значение вторых моментов градиентов. В частности,

алгоритм вычисляет экспоненциальное скользящее среднее градиента и квадратичный градиент, а параметры beta1 и beta2 управляют скоростью затухания этих скользящих средних.

Преимущества:

- простая реализация
- вычислительная эффективность
- небольшие требования к памяти
- инвариант к диагональному масштабированию градиентов
- хорошо подходит для нестационарных целей
- подходит для задач с очень шумными или разреженными градиентами
- гиперпараметры имеют наглядную интерпретацию и обычно требуют небольшой настройки

31. Метод батчевой нормализации

Пакетная нормализация (англ. batch-normalization) — метод, который позволяет повысить производительность и стабилизировать работу искусственных нейронных сетей. Суть данного метода заключается в том, что некоторым слоям нейронной сети на вход подаются данные, предварительно обработанные и имеющие нулевое математическое ожидание и единичную дисперсию.

Идея

Нормализация входного слоя нейронной сети обычно выполняется путем масштабирования данных, подаваемых в функции активации. Например, когда есть признаки со значениями от 0 до 1 и некоторые признаки со значениями от 1 до 1000, то их необходимо нормализовать, чтобы ускорить обучение. Нормализацию данных можно выполнить и в скрытых слоях нейронных сетей, что и делает метод пакетной нормализации.

Пакет

Пакет (англ. batch). Возможны два подхода к реализации алгоритма градиентного спуска для обучения нейросетевых моделей: стохастический и пакетный.

- Стохастический градиентный спуск (англ. stochastic gradient descent) — реализация, в которой на каждой итерации алгоритма из обучающей выборки каким-то (случайным) образом выбирается только один объект;
- Пакетный (батч) (англ. batch gradient descent) — реализация градиентного спуска, когда на каждой итерации обучающая выборка просматривается целиком, и только после этого изменяются веса модели.

Также существует "золотая середина" между стохастическим градиентным спуском и пакетным градиентным спуском — когда просматривается только некоторое подмножество обучающей выборки фиксированного размера (англ. batch-size). В таком случае такие подмножества принято называть мини-пакетом (англ. mini-batch). Здесь и далее, мини-пакеты будем также называть пакетом.

Ковариантный сдвиг

Пакетная нормализация уменьшает величину, на которую смещаются значения узлов в скрытых слоях (т.н. ковариантный сдвиг (англ. covariance shift)).

Ковариантный сдвиг — это ситуация, когда распределения значений признаков в обучающей и тестовой выборке имеют разные параметры (математическое ожидание, дисперсия и т.д.). Ковариантность в данном случае относится к значениям признаков.

Простой способ решить проблему ковариантного сдвига для входного слоя — это случайным образом перемешать данные перед созданием пакетов. Но для скрытых слоев нейронной сети такой метод не подходит, так как распределение входных данных для каждого узла скрытых слоев изменяется каждый раз, когда происходит обновление параметров в предыдущем слое. Эта проблема называется внутренним ковариантным сдвигом (англ. internal covariate shift).

Для решения данной проблемы часто приходится использовать низкий темп обучения (англ. learning rate) и методы регуляризации при обучении модели.

Другим способом устранения внутреннего ковариантного сдвига является метод пакетной нормализации.

Свойства пакетной нормализации

Кроме того, использование пакетной нормализации обладает еще несколькими дополнительными полезными свойствами:

- достигается более быстрая сходимость моделей, несмотря на выполнение дополнительных вычислений;

- пакетная нормализация позволяет каждому слою сети обучаться более независимо от других слоев;
- становится возможным использование более высокого темпа обучения, так как пакетная нормализация гарантирует, что выходы узлов нейронной сети не будут иметь слишком больших или малых значений;
- пакетная нормализация в каком-то смысле также является механизмом регуляризации: данный метод привносит в выходы узлов скрытых слоев некоторый шум, аналогично методу dropout;
- модели становятся менее чувствительны к начальной инициализации весов.

32. Свертка; паддинг; пулинг; страйд; тензор

Свертка — операция над парой матриц А (размера $n_x \times n_y$) и В(размера $m_x \times m_y$), результатом которой является матрица С= $A \times B$ размера ($n_x - m_x + 1) \times (n_y - m_y + 1$). Каждый элемент результата вычисляется как скалярное произведение матрицы В и некоторой подматрицы А такого же размера (подматрица определяется положением элемента в результате). Получившееся число записывается в соответствующий элемент результата.

Логический смысл свертки такой — чем больше величина элемента свертки, тем больше эта часть матрицы А была похожа на матрицу В (похожа в смысле скалярного произведения). Поэтому матрицу А называют *изображением*, а матрицу В — *фильтром* или *образцом*.

Можно заметить, что применение операции свертки уменьшает изображение. Также пиксели, которые находятся на границе изображения участвуют в меньшем количестве сверток, чем внутренние. В связи с этим в сверточных слоях используется **дополнение изображения** (англ. *padding*). Выходы с предыдущего слоя дополняются пикселями так, чтобы после свертки сохранился размер изображения. Такие свертки называют *одинаковыми* (англ. *same convolution*), а свертки без дополнения изображения называются *правильными* (англ. *valid convolution*). Среди способов, которыми можно заполнить новые пиксели, можно выделить следующие:

- *zero shift*: 00[ABC]00 ;

- *border extension*: AA[ABC]CC;
- *mirror shift*: BA[ABC]CB;
- *cyclic shift*: BC[ABC]AB.

Еще одним параметром сверточного слоя является **сдвиг** (англ. *stride*). Хоть обычно свертка применяется подряд для каждого пикселя, иногда используется сдвиг, отличный от единицы — скалярное произведение считается не со всеми возможными положениями ядра, а только с положениями, кратными некоторому сдвигу s . Тогда, если если вход имел размерность $w \times h$, а ядро свертки имело размерность $k_x \times k_y$ и использовался сдвиг s , то выход будет иметь размерность

$$\left\lfloor \frac{w-k_x}{s} + 1 \right\rfloor \times \left\lfloor \frac{h-k_y}{s} + 1 \right\rfloor.$$

Пулинговый слой призван снижать размерность изображения. Исходное изображение делится на блоки размером $w \times h$ и для каждого блока вычисляется некоторая функция. Чаще всего используется функция максимума (англ. *max pooling*) или (взвешенного) среднего (англ. *weighted average pooling*). Обучаемых параметров у этого слоя нет. Основные цели пулингового слоя:

- уменьшение изображения, чтобы последующие свертки оперировали над большей областью исходного изображения;
- увеличение инвариантности выхода сети по отношению к малому переносу входа;
- ускорение вычислений.

Тензор — это обобщение векторов и матриц на более высокие измерения. Это объект линейной алгебры, линейно преобразующий элементы одного линейного пространства в элементы другого. Через тензор (который, по сути, трехмерная матрица) можно задать цветное изображение с размерностью: ширина, высота, глубина (она кодирует цвет). Матрицу можно превратить в вектор признаков наивным образом, но тогда потеряются важные инварианты изображения.

33. Сверточная нейронная сеть

Сверточная нейронная сеть - специальная архитектура нейронных сетей, предложенная Яном Лекумом, изначально нацеленная на эффективное распознавание изображений. В сверточной нейронной сети выходы промежуточных слоев образуют матрицу (изображение) или набор матриц (несколько слоев изображения). Так, например, на вход сверточной сети можно подавать три слоя изображения (R-, G-, B-каналы изображения). Основными видами слоев в сверточной нейронной сети являются, пулинговые слои и полно связные слои.

34. Задача семантической сегментации; задача детекции объектов

Задача семантической сегментации - задача, в которой на вход модели подается изображение, а на выходе для каждого пикселя является метка принадлежности этого пикселя к определенной категории.

Задача детекции объектов - задача, в рамках которой необходимо выделить несколько объектов на изображении посредством нахождения координат их ограничивающих рамок и классификации этих ограничивающих рамок из множества заранее известных классов. Число объектов, которые находятся на изображении, заранее неизвестно.

35. Рекуррентная нейронная сеть

Рекуррентная нейронная сеть - вид нейронных сетей, где связи между элементами образуют направленную последовательность. Благодаря этому появляется возможность обрабатывать серии событий во времени или последовательные пространственные цепочки. В отличие от многослойных перцептронов, рекуррентные сети могут использовать свою внутреннюю память для обработки последовательностей произвольной длины. Поэтому эти сети применимы в таких задачах, где нечто целостное разбито на части, например: распознавание речи, генерация описания изображений.

36. Модуль памяти в рекуррентных сетях

Рекуррентные нейронные сети добавляют память к искусственным нейронным сетям, но реализуемая память получается короткой — на каждом шаге

обучения информация в памяти смешиается с новой и через несколько итераций полностью перезаписывается.

Эти модули разработаны специально, чтобы избежать проблемы долговременной зависимости, запоминая значения как на короткие, так и на длинные промежутки времени. Это объясняется тем, что LSTM-модуль не использует функцию активации внутри своих рекуррентных компонентов. Таким образом, хранимое значение не размывается во времени и градиент не исчезает при использовании метода обратного распространения ошибки во времени при тренировке сети.

Ключевые компоненты модуля памяти: состояние ячейки и различные фильтры. О состоянии ячейки можно говорить, как о памяти сети, которая передает соответствующую информацию по всей цепочке модулей. Таким образом, даже информация из ранних временных шагов может быть получена на более поздних, нивелируя эффект кратковременной памяти.

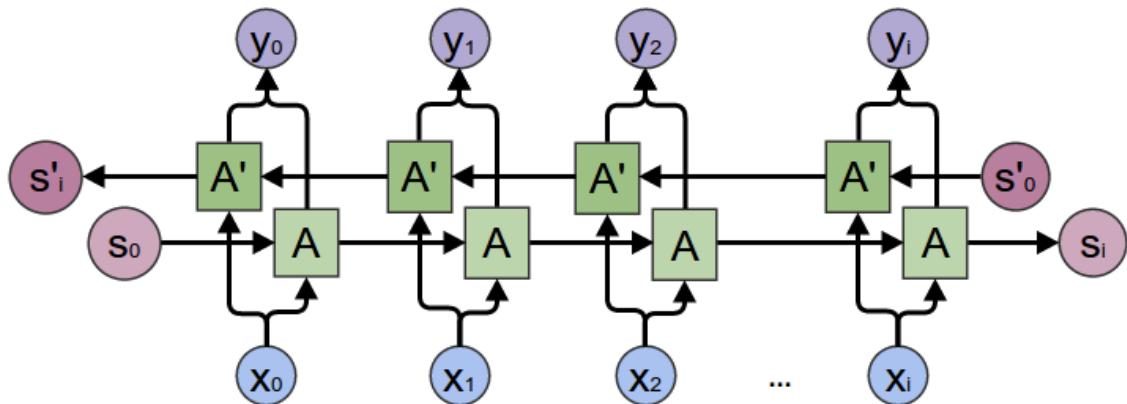
По мере того, как происходит обучение, состояние ячейки изменяется, информация добавляется или удаляется из состояния ячейки структурами, называемыми фильтрами. Фильтры контролируют поток информации на входах и на выходах модуля на основании некоторых условий. Они состоят из слоя сигмоидальной нейронной сети и операции поточечного умножения.

Сигмоидальный слой возвращает числа в диапазоне [0; 1], которые обозначают, какую долю каждого блока информации следует пропустить дальше по сети. Умножение на это значение используется для пропуска или запрета потока информации внутрь и наружу памяти. Например, входной фильтр контролирует меру вхождения нового значения в память, а фильтр забывания контролирует меру сохранения значения в памяти. Выходной фильтр контролирует меру того, в какой степени значение, находящееся в памяти, используется при расчёте выходной функции активации.

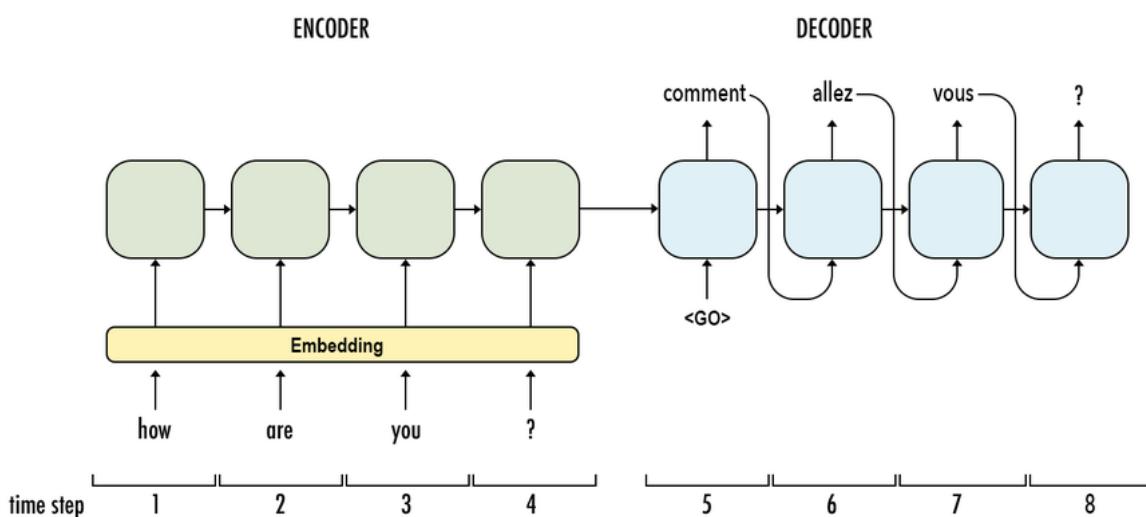
37. Двунаправленная рекуррентная сеть; Seq-2-Seq сеть

Двунаправленная рекуррентная сеть (англ. Bidirectional Recurrent Neural Network, biRNN) представляет собой две односторонние рекуррентные сети, одна из которых обрабатывает входную последовательность в прямом порядке, а другая — в обратном. Таким образом, для каждого элемента входной последовательности считаются два вектора скрытых состояний, на

основе которых вычисляется выход сети. Благодаря данной архитектуре сети доступна информация о контексте как из прошлого, так и из будущего, что решает проблему односторонних рекуррентных сетей. Для обучения biRNN используются те же алгоритмы, что и для RNN.



Seq-2-seq (Sequence to sequence, Seq2seq) сеть является базовой архитектурой many-to-many RNN и используется для трансляции одной последовательности в другую. Она состоит из двух рекуррентных сетей: кодировщика и декодировщика. Кодировщик вычисляет вектор, кодирующий входную последовательность. Далее данный вектор передается декодировщику, который в свою очередь по полученному скрытому представлению восстанавливает целевую последовательность. При этом каждый посчитанный выход используется для обновления скрытого представления.



38. Механизм внимания в рекуррентных сетях

Механизм внимания в рекуррентных сетях - это техника для поиска взаимосвязей между различными частями входных и выходных данных. Изначально механизм внимания был представлен в контексте рекуррентных Seq2Seq сетей для “обращения внимания” блоков декодеров на скрытые состояния RNN для любой итерации энкодера, а не только последней. Несмотря на то, что LSTM и GRU блоки в Seq2Seq сетях используются именно для улучшения передачи информации с предыдущих итераций RNN их основная проблема заключается в том, что влияние предыдущих состояний на текущее уменьшается экспоненциально от расстояния между словами, в то же время механизм внимания улучшает этот показатель до линейного.

39. Векторные представления слов; Word2Vec; skip-gram; CBOW

Векторные представления слов - общее название для различных подходов к моделированию языка и обучению представлений в обработке естественного языка, направленных на сопоставление словам из некоторого словаря векторов небольшой размерности.

Word2Vec - способ построения сжатого пространства векторов слов, использующий нейронные сети. Принимает на вход большой текстовый корпус и сопоставляет каждому слову вектор. Сначала он создает словарь, а затем вычисляет векторное представление слов. Векторное представление основывается на контекстной близости: слова, встречающиеся в тексте рядом с одинаковыми словами (а следовательно, имеющие схожий смысл), в векторном представлении имеют высокое косинусное сходство. В word2vec существуют две основных модели обучения: skip-gram и CBOW.

В модели *Skip-gram* по слову предсказываются слова из его контекста, а в модели *CBOW*

по контексту подбирается наиболее вероятное слово.

В обеих моделях входные и выходные слова подаются в one-hot encoding

One-hot encoding

Пусть число различных слов равно K . Сопоставим слову с номером i вектор длины K , в котором i -тая координата равна единице, а все остальные — нулям. Недостатком one-hot encoding является то, что по векторным представлениям нельзя судить о схожести смысла слов. Также вектора имеют очень большой размер, из-за чего их неэффективно хранить в памяти.

Блок 3

40. Задача кластеризации; внешние меры оценки; внутренние меры оценки

Задача кластеризации - задача обучения без учителя, в которой алгоритму требуется выделить (придумать) новый категориальный признак.

Задача: разделить набор объектов одного типа на группы так, чтобы объекты в этих группах имели похожие свойства.

«Похожесть» формализуется абстрактной метрикой.

D — набор данных, состоящий из объектов из X

$\rho : X \times X \rightarrow [0; +\infty)$ — метрика на X .

Найти алгоритм $\alpha : X \rightarrow Y$, где Y — множество кластеров.

Цели кластеризации:

-
- Уменьшить объем данных
 - Найти группы похожих объектов
 - Найти необычные объекты
 - Найти иерархию объектов (групп)

Пример: разбиение песен по жанрам, но лень придумывать жанры песен

Внутренние меры - такие меры, которые используют для оценки само множество объектов и метки кластеров, которые вернула кластеризация и они не используют никакой внешней информации

Среднее внутрикластерное расстояние:

$$F_0 = \frac{\sum_{i < j} [y_i = y_j] \rho(x_i, x_j)}{\sum_{i < j} [y_i = y_j]} \rightarrow \min.$$

Среднее межкластерное расстояние:

$$F_1 = \frac{\sum_{i < j} [y_i \neq y_j] \rho(x_i, x_j)}{\sum_{i < j} [y_i \neq y_j]} \rightarrow \max.$$

Отношение:

$$\frac{F_0}{F_1} \rightarrow \min.$$

Мы хотим минимизировать среднее внутрикластерное расстояние и максимизировать среднее межкластерное расстояние.

Внешние меры - можем использовать реальные метки классов, если изначальный набор данных, который мы кластеризуем был для задачи классификации, но на прямую меры для задач классификации мы использовать не можем, потому что метка кластера, которую мы придумываем, будет какой то новой категорией объекта, которую нельзя будет сравнивать с метками классов. Кроме того, число классов может отличаться от числа кластеров.

Мы можем взять TP, FP, FN, TN но вычислять их иным способом.

Можно использовать меры для задачи бинарной классификации, которые основываются на TP , FP , FN , TN , но вычислять перебирая пары объектов, в которых:

- TP — элементы принадлежат одному кластеру и одному классу
- FP — элементы принадлежат одному кластеру, но разным классам
- FN — элементы принадлежат разным кластерам, но одному классу
- TN — элементы принадлежат разным кластерам и разным классам

(последний - TN)

Индекс Жаккара

$$\text{Jaccard} = \frac{TP}{TP + FP + FN}$$

Индекс Rand

$$\text{Rand} = \frac{TP + TN}{TP + FP + FN + TN}$$

Индекс Фоулкса-Мэллова

$$FM = \sqrt{\frac{TP}{TP + FP} \frac{TP}{TP + FN}}$$

41. Графовые методы кластеризации

- Основная идея: будем работать с графом. Его вершины являются объектами, а длины его рёбер равны расстояниям между соответствующими объектами.
- Кластеры могут быть хорошо представлены в графическом описании.

Алгоритм

1. Зафиксируем радиус R.
2. Удалим рёбра $(u, v) : \rho(u, v) > R$ (все ребра, превышающие порог R)
3. Посмотреть на компоненты связности графа. Эти самые компоненты связности образуют кластеры, т.е. кластеры соответствуют связанным компонентам.

Получение нужного числа кластеров:

- Нужное число кластеров можно получить бинарным поиском по R.

42. Иерархические методы кластеризации

Иерархическая кластеризация — множество алгоритмов кластеризации, направленных на создание иерархии вложенных разбиений исходного

множества объектов.

Иерархические алгоритмы кластеризации часто называют **алгоритмами таксономии**.

Идея следующая: пусть будем строить **дендrogramму**

— дерево иерархии кластеров, построенное по матрице мер близости между кластерами. В узлах дерева находятся подмножества объектов из обучающей выборки. При этом на каждом ярусе дерева множество объектов из всех узлов составляет исходное множество объектов. Объединение узлов между ярусами соответствует слиянию двух кластеров. При этом длина ребра соответствует расстоянию между кластерами.

Алгоритм иерархической кластеризации

Дерево строится от листьев к корню. В начальный момент времени каждый объект содержится в собственном кластере. Далее происходит итеративный процесс слияния двух ближайших кластеров до тех пор, пока все кластеры не объединятся в один или не будет найдено необходимое число кластеров. На каждом шаге необходимо уметь вычислять расстояние между кластерами и пересчитывать расстояние между новыми кластерами. Расстояние между однозначными кластерами определяется через расстояние между объектами: $R(\{x\}, \{y\}) = \rho(x, y)$ Для вычисления расстояния $R(U, V)$ между кластерами U и V на практике используются различные функции в зависимости от специфики задачи.

Функции расстояния между кластерами

- **Метод одиночной связи** (англ. *single linkage*)

$$R_{\min}(U, V) = \min_{u \in U, v \in V} \rho(u, v)$$

- **Метод полной связи** (англ. *complete linkage*)

$$R_{\max}(U, V) = \max_{u \in U, v \in V} \rho(u, v)$$

- **Метод средней связи** (англ. *UPGMA (Unweighted Pair Group Method with Arithmetic mean)*)

$$R_{\text{avg}}(U, V) = \frac{1}{|U| \cdot |V|} \sum_{u \in U} \sum_{v \in V} \rho(u, v)$$

- **Центроидный метод** (англ. *UPGMC (Unweighted Pair Group Method with Centroid average)*)

$$R_c(U, V) = \rho^2 \left(\sum_{u \in U} \frac{u}{|U|}, \sum_{v \in V} \frac{v}{|V|} \right)$$

- **Метод Уорда** (англ. *Ward's method*)

$$R_{\text{ward}}(U, V) = \frac{|U| \cdot |V|}{|U| + |V|} \rho^2 \left(\sum_{u \in U} \frac{u}{|U|}, \sum_{v \in V} \frac{v}{|V|} \right)$$

Формула Ланса-Уильямса

На каждом шаге необходимо уметь быстро подсчитывать расстояние от образовавшегося кластера $W = U \cup V$ до любого другого кластера S , используя известные расстояния с предыдущих шагов. Это легко выполняется при использовании формулы, предложенной Лансом и Уильямсом в 1967 году:

$$R(W, S) = \alpha_U \cdot R(U, S) + \alpha_V \cdot R(V, S) + \beta \cdot R(U, V) + \gamma \cdot |R(U, S) - R(V, S)|$$

, где $\alpha_U, \alpha_V, \beta, \gamma$ – числовые параметры.

Каждая из указанных выше функций расстояния удовлетворяет формуле Ланса-Уильямса со следующими коэффициентами:

- **Метод одиночной связи** (англ. *single linkage*)

$$\alpha_U = \frac{1}{2}, \alpha_V = \frac{1}{2}, \beta = 0, \gamma = -\frac{1}{2}$$

- **Метод полной связи** (англ. *complete linkage*)

$$\alpha_U = \frac{1}{2}, \alpha_V = \frac{1}{2}, \beta = 0, \gamma = \frac{1}{2}$$

- **Метод средней связи** (англ. *UPGMA (Unweighted Pair Group Method with Arithmetic mean)*)

$$\alpha_U = \frac{|U|}{|W|}, \alpha_V = \frac{|V|}{|W|}, \beta = 0, \gamma = 0$$

- **Центроидный метод** (англ. *UPGMC (Unweighted Pair Group Method with Centroid average)*)

$$\alpha_U = \frac{|U|}{|W|}, \alpha_V = \frac{|V|}{|W|}, \beta = -\alpha_U \cdot \alpha_V, \gamma = 0$$

- **Метод Уорда** (англ. *Ward's method*)

$$\alpha_U = \frac{|S| + |U|}{|S| + |W|}, \alpha_V = \frac{|S| + |V|}{|S| + |W|}, \beta = \frac{-|S|}{|S| + |W|}, \gamma = 0$$

43. k-means. c-means

k-means - Итеративный алгоритм, основанный на минимизации суммарного квадратичного отклонения точек кластеров от центров этих кластеров;

Простой, но в то же время достаточно неточный метод кластеризации в классической реализации. Он разбивает множество элементов векторного пространства на заранее известное число кластеров k . Он стремится минимизировать среднеквадратичное отклонение на точках каждого кластера. Основная идея заключается в том, что на каждой итерации пересчитываются центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике. Алгоритм завершается, когда на какой-то итерации не происходит изменения кластеров.

Проблемы алгоритма k-means:

- **необходимо заранее знать количество кластеров**
- **алгоритм очень чувствителен к выбору начальных центров кластеров**
- не справляется с задачей, когда объект принадлежит к разным кластерам в равной степени или не принадлежит ни одному.

c-means

С последней проблемой k-means успешно справляется алгоритм c-means.

Вместо однозначного ответа на вопрос к какому кластеру относится объект, он определяет вероятность того, что объект принадлежит к тому или иному кластеру.

k-means - итерационный алгоритм, который разбивает наборы на k частей. Центр масс у кластера (среднее внутрикластерное расстояние по каждому признаку) C_j называется центроидом.

Это упрощение EM-алгоритма с сильной ассоциацией только с одним классом.

1. Выбрать k -точек (центроидов) из набора данных $\{c_i\}_{i=1}^k$.
2. Повторять
3. Для каждого x найти ближайший центроид $n(x)$.

$$C_i = \{x | n(x) = c_i\}$$

4. Для каждого C_i найти центральную точку и определить ее центроидом.
5. Пока центроиды не будут изменяться.

Есть и модификация: k-means++

Как предотвратить произвольно плохие локальные минимумы в k-средних?

Делать то же самое, что и в k-средних, но выбирать новый центр i -го кластера с вероятностью, пропорциональной $\|p - c_i\|^2$.

44. Алгоритм DBSCAN

Основная идея метода заключается в том, что алгоритм разделит заданный набор точек в некотором пространстве на группы точек, которые лежат друг от друга на большом расстоянии. Объекты, которые лежат отдельно от скоплений с большой плотностью, будут помечены как шумовые.

На вход алгоритму подаётся набор точек, параметры ϵ (радиус окружности) и m (минимальное число точек в окрестности). Для выполнения кластеризации потребуется поделить точки на четыре вида: основные точки, прямо достижимые, достижимые и шумовые.

- Точка является основной, если в окружности с центром в этой точке и радиусом ϵ находится как минимум m точек.
- Точка a является *прямо достижимой* из основной точки b , если a находится на расстоянии, не большем ϵ от точки b .
- Точка a является *достижимой* из b , если существует путь p_1, \dots, p_n с $p_1 = a$ и $p_n = b$, где каждая точка p_{i+1} *прямо достижима* из точки p_i .
- Все остальные точки, которые не достижимы из основных точек, считаются *шумовыми*.

Основная точка вместе со всеми достижимыми из нее точками формирует *кластер*. В кластер будут входить как основные, так и неосновные точки. Таким образом, каждый кластер содержит по меньшей мере одну основную точку.

Алгоритм начинается с произвольной точки из набора, которая еще не просматривалась. Для точки ищется ϵ -окрестность. Если она не содержит как минимум m точек, то помечается как шумовая, иначе образуется кластер K , который включает все точки из окрестности. Если точка из окрестности уже является частью другого кластера C_j , то все точки данного кластера добавляются в кластер K . Затем выбирается и обрабатывается новая, не посещённая ранее точка, что ведёт к обнаружению следующего кластера или шума.

На выходе получаем разбиение на кластеры и шумовые объекты. Каждый из полученных кластеров C_j является непустым множеством точек и удовлетворяет двум условиям:

- Любые две точки в кластере попарно связаны (то есть найдется такая точка в кластере, из которой достижимы обе этих точки).
- Если точка достижима из какой-либо точки кластера, то она принадлежит кластеру.

DBSCAN находит практическое применение во многих реальных задачах, например, в маркетинге: необходимо предложить покупателю релевантный товар, который подойдет под его заказ. Выбрать такой товар можно, если посмотреть на похожие заказы других покупателей — в таком случае похожие заказы образуют кластер вещей, которые часто берут вместе. Похожим образом с помощью DBSCAN можно исследовать и находить общие интересы людей, делить их на социальные группы, моделировать поведение посетителей сайта. Алгоритм также может использоваться для сегментации изображений.

45. Уменьшение размерности; синтез признаков; выбор признаков; алгоритмы фильтрации

Под **уменьшением размерности** в машинном обучении подразумевается уменьшение числа признаков набора данных. Наличие в нем признаков избыточных, неинформативных или слабо информативных может понизить эффективность модели, а после такого преобразования она упрощается, и соответственно уменьшается размер набора данных в памяти и ускоряется

работа алгоритмов ML на нем. Уменьшение размерности может быть осуществлено методами выбора признаков или выделения признаков.

Методы **выбора признаков** оставляют некоторое подмножество исходного набора признаков, избавляясь от признаков избыточных и слабо информативных (вторая группа включает в себя все остальные алгоритмы, даже где $k > n$). Основные преимущества этого класса алгоритмов:

- Уменьшение вероятности переобучения;
- Увеличение точности предсказания модели;
- Сокращение времени обучения;
- Увеличивается семантическое понимание модели.

Все методы выбора признаков можно разделить на 5 типов, которые отличаются алгоритмами выбора лишних признаков:

- **фильтры** - измеряют релевантность признаков на основе функции μ , и затем решают по правилу k , какие признаки оставить в результирующем множестве.
- **Оберточные методы** - находят подмножество искомых признаков последовательно, используя некоторый классификатор как источник оценки качества выбранных признаков, т.е. этот процесс является циклическим и продолжается до тех пор, пока не будут достигнуты заданные условия останова.
- **Встроенные методы** - похожи на оберточные методы, но для выбора признаков используется непосредственно структуру некоторого классификатора. В оберточных методах классификатор служит только для оценки работы на данном множестве признаков, тогда как встроенные методы используют какую-то информацию о признаках, которую классификаторы присваивают во время обучения.
- **Правила обрезки** - Для признаков, у которых найдено качество, можно выкинуть ненужное число признаков.
- **Гибридные методы** (комбинируют несколько разных методов выбора признаков, например, некоторое множество фильтров, а потом запускают оберточный или встроенный метод. Таким образом, гибридные методы сочетают в себе преимущества сразу нескольких методов, и на практике повышают эффективность выбора признаков) **Ансамблевые методы** (применяются больше для наборов данных с очень большим

числом признаков. В данном подходе для начального множества признаков создается несколько подмножеств признаков, и эти группы каким-то образом объединяются, чтобы получить набор самых релевантных признаков. Это довольно гибкая группа методов, т.к. для нее можно применять различные способы выбора признаков и объединения их подмножеств).

Фильтры могут быть:

- Одномерные (англ. *univariate*) — функция μ определяет релевантность одного признака по отношению к выходным меткам. В таком случае обычно измеряют "качество" каждого признака и удаляют худшие;
- Многомерные (англ. *multivariate*) — функция μ определяет релевантность некоторого подмножества исходного множества признаков относительно выходных меток.

Распространенными вариантами для μ являются:

- Коэффициент ранговой корреляции Спирмена
- Information gain

Преимуществом группы фильтров является простота вычисления релевантности признаков в наборе данных, но недостатком в таком подходе является игнорирование возможных зависимостей между признаками.

46. Алгоритмы-обертки; встроенные методы выбора признаков

Алгоритмы-обертки - такие алгоритмы, которые используют классификатор или регрессор для оценки качества получаемого подмножества признаков и использует алгоритмы дискретной оптимизации для поиска оптимального подмножества признаков.

Существует несколько типов оберточных методов: детерминированные, которые изменяют множество признаков по определенному правилу, а также стохастические - сводят задачу выбора признаков к задаче оптимизации в пространстве бинарных векторов.

Среди детерминированных алгоритмов самыми простыми являются:

- SFS (Sequential Forward Selection) — жадный алгоритм, который начинает с пустого множества признаков, на каждом шаге добавляя лучший из еще не выбранных признаков в результирующее множество;
- SBS (Sequential Backward Selection) — алгоритм обратный SFS, который начинает с изначального множества признаков, и удаляет по одному или несколько худших признаков на каждом шаге.

Среди стохастических:

- Поиск восхождения на холм
- Генетические алгоритмы

Встроенные методы выбора признаков - методы выбора признаков, при которых выбор осуществляется в процессе работы других алгоритмов (классификаторов и регрессоров).

Популярным оберточным методом является SVM-RFE (SVM-based Recursive Feature Elimination), который иногда также обозначается как **встроенный**. Этот метод использует как классификатор SVM и работает итеративно: начиная с полного множества признаков обучает классификатор, ранжирует признаки по весам, которые им присвоил классификатор, убирает какое-то число признаков и повторяет процесс с оставшегося подмножества фичей, если не было достигнуто их требуемое количество. Таким образом, этот метод очень похож на встроенный, потому что непосредственно использует знание того, как устроен классификатор.

Одним из примеров встроенного метода является реализация на случайном лесе: каждому дереву на вход подаются случайное подмножество данных из датасета с каким-то случайнм набор признаков, в процессе обучения каждое из деревьев решений производит "голосование" за релевантность его признаков, эти данные агрегируются, и на выходе получаются значения важности каждого признака набора данных. Дальнейший выбор нужных нам признаков уже зависит от выбранного критерия отбора.

Встроенные методы используют преимущества оберточных методов и являются более эффективными, при этом на отбор тратится меньше времени, уменьшается риск переобучения, но т.к. полученный набор признаков был отобран на основе знаний о классификаторе, то есть вероятность, что для другого классификатора это множество признаков уже не будет настолько же релевантным.

47. Алгоритм РСА

Бывает линейным и нелинейным. Линейный быстрее и более интерпретируемый, нелинейный находит более сложные признаки.

РСА (метод главных компонент) - наиболее известный метод для извлечения признаков.

Метод осуществляет линейное отображение данных в пространство меньшей размерности таким образом, что дисперсия данных в малоразмерном представлении максимизируется. На практике строится матрица ковариации (а иногда корреляции) данных и вычисляются собственные вектора этой матрицы. Собственные векторы, соответствующие наибольшим собственным значениям (главные компоненты) теперь можно использовать для восстановления большей части дисперсии исходных данных. Более того, первые несколько собственных векторов часто можно интерпретировать в терминах крупномасштабного поведения системы. Исходное пространство (с размерностью, равной числу точек) редуцируется (с потерей данных, но надеждой, что остается наиболее важная дисперсия) до пространства, натянутого на несколько собственных векторов.

48. Алгоритм t-SNE

t-SNE (Стохастическое вложение соседей) — это алгоритм нелинейного снижения размерности, хорошо подходящий для визуализации. Каждый объект укладывается в пространство малой размерности таким образом, что похожим объектам будут соответствовать близко расположенные точки, и наоборот, непохожие объекты представляются далеко расположенными друг от друга точками.

1. Определим вероятность для точки “выбрать ближайшим соседом” другую точку в пространстве
2. Построим такие распределения для высокоразмерных и низкоразмерных представлений
3. Минимизируем расстояние между двумя распределениями

49. Автокодировщик

Автокодировщик - глубокая нейронная сеть, способная строить низкоразмерные представления данных за счет нелинейной трансформации.

Основная идея: заставим сеть предсказывать (восстанавливать) то, что подается ей на вход, ограничив возможность обучиться тривиальному преобразованию.

Два варианта реализации:

- структурный: между входными и выходными слоями должен быть слой меньшей размерности, т.н. бутылочное горлышко. Это недополненный автокодировщик.
- регуляризационный: добавим регуляризационную константу к выходам этого слоя, уменьшающим его размерность. Это разреженный кодировщик.

50. Алгоритм EM

Алгоритм EM (англ. *expectation-maximization*) — итеративный алгоритм поиска оценок максимума правдоподобия модели, в ситуации, когда она зависит от скрытых (ненаблюдаемых) переменных.

Алгоритм ищет параметры модели итеративно, каждая итерация состоит из двух шагов:

E (Expectation) шаг — поиск наиболее вероятных значений скрытых переменных.

M (Maximization) шаг — поиск наиболее вероятных значений параметров, для полученных на шаге E значений скрытых переменных.

EM алгоритм подходит для решения задач двух типов:

1. Задачи с неполными данными.
2. Задачи, в которых удобно вводить скрытые переменные для упрощения подсчета функции правдоподобия. Примером такой задачи может служить кластеризация.

Плюсы и минусы

Плюсы:

- Сходится в большинстве случаев.
- Наиболее гибкое решение.

- Существуют простые модификации, позволяющие уменьшить чувствительность алгоритма к шуму в данных.

Минусы:

- Чувствителен к начальному приближению. Могут быть ситуации, когда сойдемся к локальному экстремуму.
- Число компонент k является гиперпараметром.

51. Частичное обучение. Алгоритм S3VM

Частичное обучение - задача обучения с учителем, в которой только малая часть тренировочных данных содержит целевой признак.

Базовое решение:

- не использовать объекты, у которых пропущен целевой признак.
- не использовать целевой признак для обучения (задача обучения без учителя). Размеченные объекты можно использовать для тестирования.

Что мы имеем? Функция потерь содержит расстояние и до неразмеченных объектов.

Возникает проблема: использование hat loss приводит к невыпуклой оптимизации.

Подходы к решению:

- использовать алгоритмы невыпуклой оптимизации
- использовать сглаживание и градиентный спуск
- использовать верхнюю оценку и разбить на несколько задач выпуклой оптимизации

S3VM

Полуавтоматические опорные вектора (S3VM)

Полуавтоматические SVM (англ. *Semi-supervised SVMs, S3VMs*), они же трансдуктивные SVM (TSVMs) решают задачу максимизации зазора (*margin*) между неразмеченными данными.

Идея

- Перечислить все 2^u возможные способы разметки множества X_u
- Построить стандартную SVM для каждой разметки (и для X_l)
- Взять SVM с наибольшим зазором

Постановка задачи

- Два класса $y \in \{+1, -1\}$
- Размеченные данные (X_l, Y_l)
- Ядро K
- Гильбертово пространство функций H_K (RKHS)

С помощью SVM найти функцию $f(x) = h(x) + b$, где $h \in H_K$ и классифицировать x с помощью $\text{sign}(f(x))$

Алгоритм

1. Входные данные: ядро K , веса $\lambda_1, \lambda_2, (X_l, Y_l), X_u$. Решим задачу оптимизации для $f(x) = h(x) + b, h(x) \in H_K$

$$\min_f \sum_{i=1}^l (1 - y_i f(x_i))_+ + \lambda_1 \|h\|_{H_K}^2 + \lambda_2 \sum_{i=l+1}^n (1 - |f(x_i)|)_+$$

такую, что $\frac{1}{n-l} \sum_{i=l+1}^n f(x_i) = \frac{1}{l} \sum_{i=1}^l y_i$

4. Классифицируем новый объект x из тестового множества, используя $\text{sign}(f(x))$

Достоинства S3VM

- Применимо везде, где применимы классические SVM

Недостатки

- Трудности в оптимизации
- Алгоритм может сходиться к неправильной (плохой) целевой функции
- Менее мощный подход, по сравнению с алгоритмами на графах и генеративными моделями, т. е. потенциально менее эффективное обучение

52. Активное обучение. Метод uncertainty sampling

Активное обучение:

- есть доступ к большому числу объектов, но не у всех есть метки
- данные собираются быстро, а размечаются медленно и порционно, скорость обучения моделей происходит быстрее, чем разметка
- в активном обучении условия такие же, как в частичном обучении, но можно задавать Оракулу вопросы о значении меток
- требуется восстановить $f: X \rightarrow Y$ за наименьшее число обращений к Оракулу (найти стратегию обращений к Оракулу, оптимизирующую качество аппроксимации f).

Выбор по степени неуверенности (англ. *uncertainty sampling*) — метод отбора объектов из выборки, где самыми информативными объектами

считываются те, на которых текущий алгоритм меньше всего уверен в верности классификации. Для этого необходимо задать меру неуверенности в классификации на каждом объекте.

Зафиксируем модель на некотором этапе обучения и обозначим за $P(y|x)$ вероятность того, что объект x принадлежит классу y . Приведем основные меры неуверенности для текущей классификации:

- **Максимальная энтропия** (англ. *maximum entropy*)

Энтропия классификации на объекте x :

$$\Phi_{ENT}(x) = - \sum_y P(y|x) \log P(y|x).$$

Чем больше энтропия — тем больше неуверенность в классификации.

- **Минимальный отступ** (англ. *smallest margin*)

Отступ (англ. *margin*) от y_1 — самого вероятного класса до y_2 — второго по вероятности класса:

$$\Phi_M(x) = P(y_1|x) - P(y_2|x).$$

Очевидно, что если отступ велик, то велика и уверенность, потому что один класс заметно выигрывает у всех остальных. Поэтому имеет смысл запрашивать оракула на объектах с минимальным отступом.

- **Минимальная уверенность** (англ. *least confidence*)

Функция неуверенности:

$$\Phi_{LC}(x) = 1 - P(y_1|x),$$

y_1 — наиболее вероятный класс. Интересующие нас объекты — объекты с минимальной уверенностью, то есть с максимальным Φ_{LC} .

Заметим, что в случае бинарной классификации эти методы эквивалентны.

53. Сэмплирование; алгоритм SMOTE

Сэмплирование (англ. *data sampling*) — метод корректировки обучающей выборки с целью балансировки распределения классов в исходном наборе данных. Нужно отличать этот метод от сэмплирования в активном обучении для отбора кандидатов и от сэмплирования в статистике для создания подвыборки с сохранением распределения классов.

Неравномерное распределение может быть следующих типов:

- Недостаточное представление класса в *независимой переменной*;
- Недостаточное представление класса в *зависимой переменной*.

Стратегии сэмплирования

- **Субдискретизация** (англ. *under-sampling*) — удаление некоторого количества примеров мажоритарного класса.
- **Передискретизация** (англ. *over-sampling*) — увеличение количества примеров миноритарного класса.
- **Комбинирование** (англ. *combining over- and under-sampling*) — последовательное применение субдискретизации и передискретизации.

- **Ансамбль сбалансированных наборов** (англ. *ensemble balanced sets*) — использование встроенных методов сэмплирования в процессе построения ансамблей классификаторов.

Алгоритм SMOTE

1. Случайно выбрать точку a
2. Выбрать k ближайших точек из ее класса
3. Случайно выбрать одну из них, b
4. Случайно выбрать точку на отрезке (a, b) .
5. Добавить ее с той же меткой класса, что у a

54. Обучение с первого взгляда; сиамская сеть

Обучение с первого взгляда

Сценарий:

- число классов может увеличиться
- в некоторых классах сравнительно мало объектов

Тогда можем прибегнуть к следующему:

Обучение на одном примере - это постановка, в которой алгоритм должен дообучиться классификации на новый класс, содержащий всего один объект.

Обучение на нескольких примерах - предполагает все то же, но с несколькими объектами.

Сиамская сеть - состоит из двух идентичных сетей, возвращающие векторные представления входов. Для обучения используется в таком случае triplet loss:

$$L(a, p, n) = \max(\text{dist}(a, p) - \text{dist}(a, n) + \epsilon, 0)$$

Обучение: сеть обучается по батчам троек градиентным спуском.

Вывод: для каждого класса хранится объект (центроид), с которыми сравнивается вход по косинусному расстоянию. Чтобы добавить класс, добавляется новый объект в качестве центроида.

55. Аномалии; шумы

Аномалии (выбросы) - плохие объекты для построения модели

Виды выбросов

На основе размерности изучаемого массива данных выбросы подразделяют на одномерные и многомерные.

Одномерные выбросы

Точка является выбросом только по одной из своих координат.

Многомерные выбросы

Точка является выбросом сразу по нескольким координатам.

Другой подход классификации выбросов — по их окружению.

Точечные выбросы

Единичные точки, выбивающиеся из общей картины. Точечные аномалии часто используются в системах контроля транзакций для выявления мошенничества, например, когда с украденной карты совершается крупная покупка.

Контекстуальные выбросы

Для того, чтобы определить, является ли точка выбросом необходим контекст. Например, в Петербурге +15 градусов Цельсия. Зимой такая температура является выбросом, а летом нет.

Коллективные выбросы

Здесь выбросом является не точка, а группа точек. Примером таких выбросов могут служить, например, задержки поставок на фабрике. Одна задержка не является выбросом. Но если их много, значит это может стать проблемой.

Причины возникновения выбросов

- Сбой работы оборудования;
- Человеческий фактор;
- Случайность;
- Уникальные явления;
- и др.

Шум – записи в наборе данных, не укладывающиеся в ту или иную концепцию Классификации (Classification). Такие Наблюдения (Observation) вызваны человеческой ошибкой при создании Датасета (Dataset).

или иными причинами. Зашумленный набор наносит ущерб всему Пайплайну (Pipeline).

. Зашумленность измеряется как отношение чистых данных – сигнала к шуму. Существует много методов, используемых для искоренения шума.

Про шум подробнее: <https://www.helenkapatsa.ru/shum/>

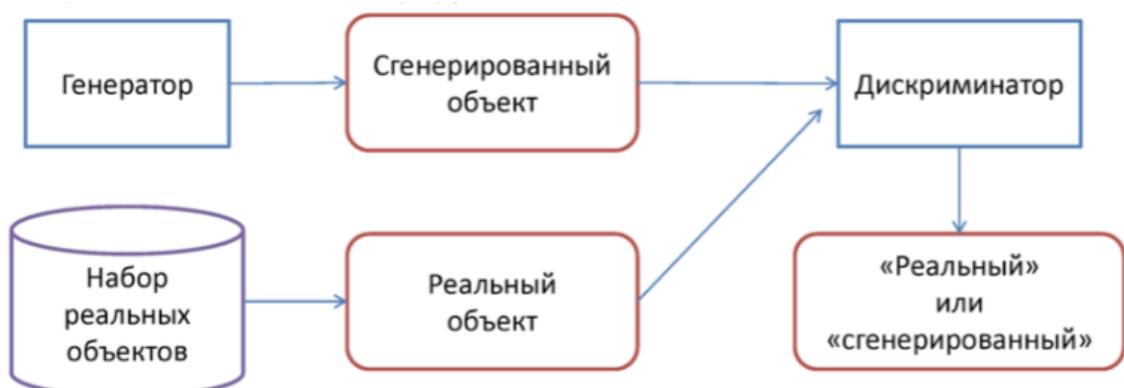
56. Задача генерации объектов

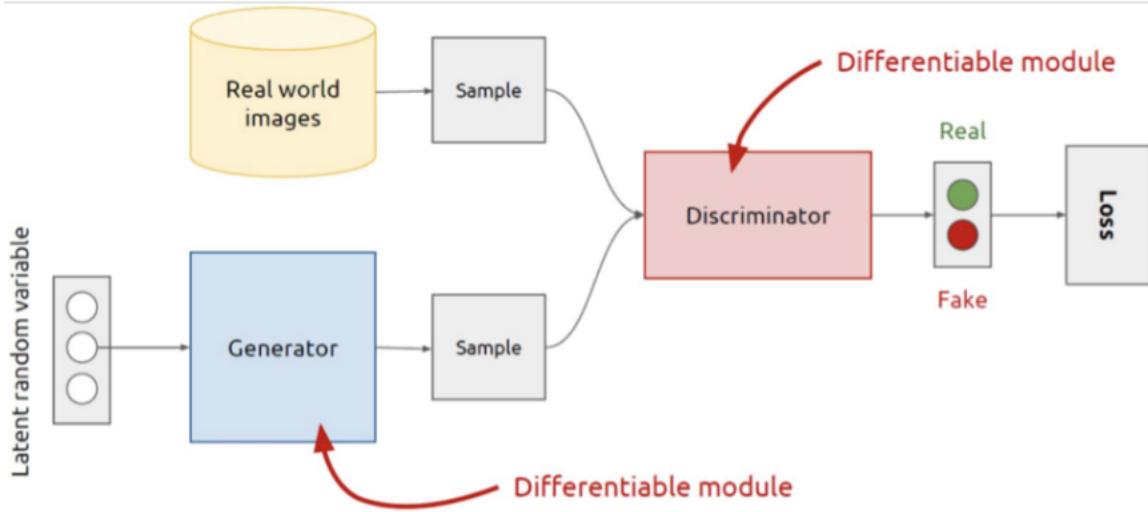
Задача генерации объектов — задача, связанная с машинным обучением, заключающаяся в создании новых правдоподобных объектов на основании заданной выборки. Полученные объекты могут быть использованы как для прикладных целей (в таком случае, это чаще всего изображения), так и для генерации объектов для тренировочной выборки, когда размечать настоящие данные — долго и дорого, или их нужно анонимизировать. В зависимости от того, для какой из этих целей используется генерация объектов, постановка задачи и методы её решения несколько отличаются. Для достижения данной цели обычно используются порождающие модели (т.е. используется генератор - сеть, которая генерирует объекты, и дискриминатор - сеть, которая будет будущий отличать сгенерированные объекты от настоящих).

57. GANs

Генеративно-состязательная сеть (GAN): производная по объекту используется как производная по выходу генератора, который эти объекты генерирует.

Схема выглядит примерно вот так:





Можно обучать совместно в постановке минимаксной игры.

Минимаксная целевая функция:

$$\min_{\theta_g} \max_{\theta_d} [\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$$

где D_{θ_d} — дискриминатор с параметрами θ_d

(пытается максимизировать целевую функцию и сделать так, чтобы $D(x)$ был близок к 1 (настоящий) и $D(G(z))$ был близок к 0 (сгенерированный))

и G_{θ_g} — генератор с параметрами θ_g

(пытаемся минимизировать целевую функцию и сделать $D(G(z))$ близким к 1)

58. CGANs

Те же генеративно-состязательные модели (GANs), но с модификацией:

- идея состоит в том, чтобы добавить несколько меток, чтобы дискриминатор мог работать как классификатор по отношению к некоторым меткам.
- в этом случае у нас разное распределение для каждого класса
- тогда у нас меняется целевая функция

$$\min_{\theta_g} \max_{\theta_d} [\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x, y) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z, y), y))]$$

59. Вариационный автокодировщик

Вариационный автокодировщик — автокодировщик

(генеративная модель, которая учится отображать объекты в заданное скрытое пространство (и обратно)) основанный на вариационном выводе.

При попытке использования обычного автокодировщика для генерации новых объектов (желательно из того же априорного распределения, что и датасет) возникает следующая проблема. Случайной величиной с каким распределением проинициализировать скрытые векторы, для того, чтобы картинка, после применения декодера, стала похожа на картинки из датасета, но при этом не совпадала ни с одной из них? Ответ на этот вопрос не ясен, в связи с тем, что обычный автокодировщик не может ничего утверждать про распределение скрытого вектора и даже про его область определения. В частности, область определения может быть даже дискретной.

Вариационный автокодировщик в свою очередь предлагает пользователю самому определить распределение скрытого вектора.

Благодаря тому, что пользователь сам устанавливает нужное распределение скрытого вектора, вариационный кодировщик хорошо подходит для генерации новых объектов (например, картинок). Для этого достаточно разыграть скрытый вектор согласно его распределению и подать на вход декодера. Получится объект из того же распределения, что и датасет.

Подробнее тут: http://neerc.ifmo.ru/wiki/index.php?title=Вариационный_автокодировщик