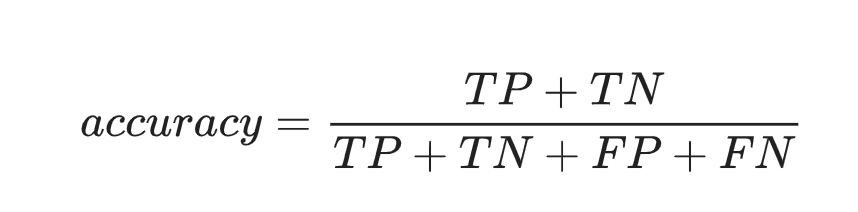
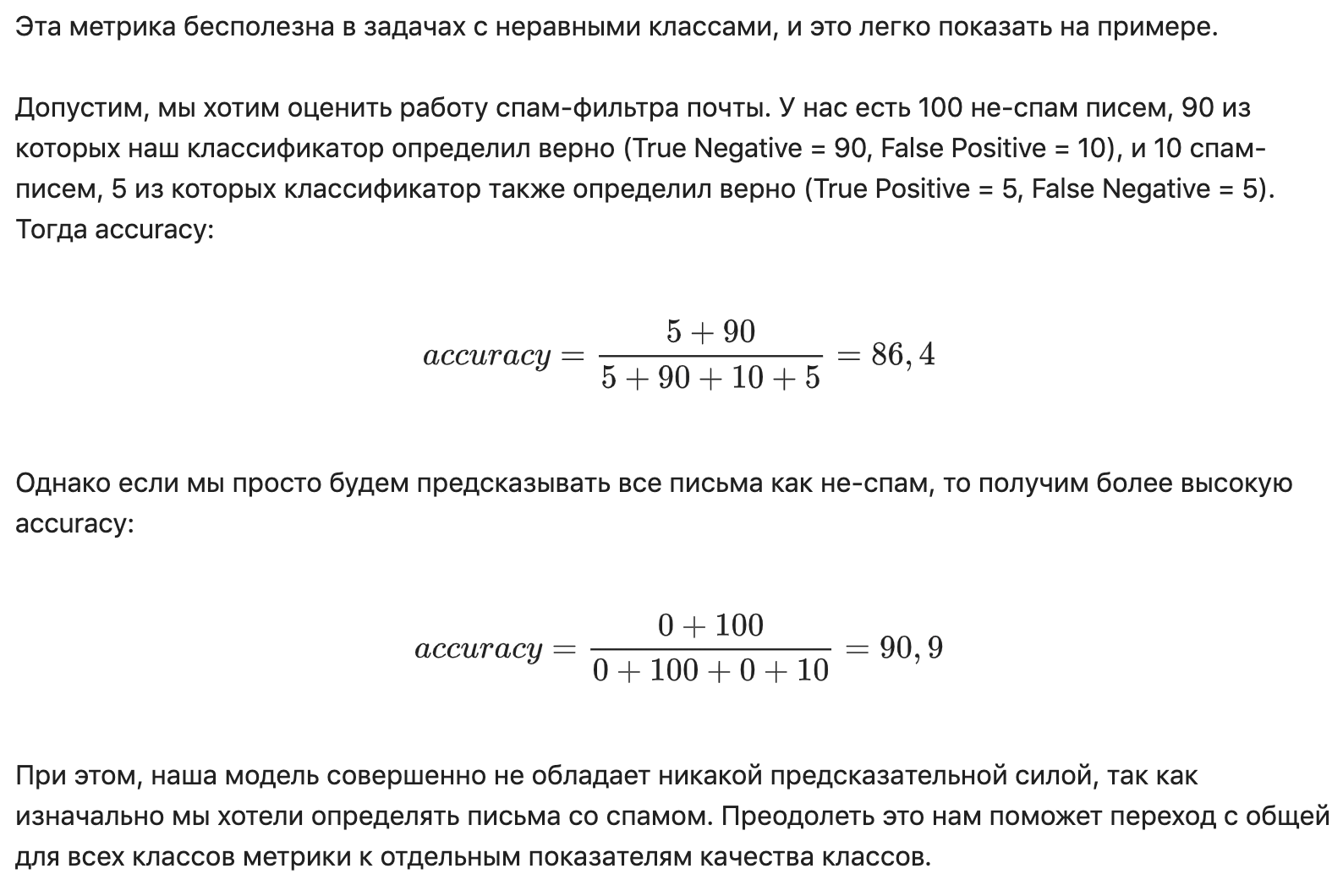
**Вопросы с летучек**

#### Что такое точность(accuracy)? В каких случаях она плохо отражает качество классификации?



Пример с хабра:



#### Что такое стохастический градиентный спуск? Почему для него нужно поддерживать текущее аппроксимированное значение эмпирического риска?

**Стохастический градиентный спуск** (англ. *stochastic gradient descent*)

− оптимизационный алгоритм, отличающийся от обычного [градиентного спуска](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D0%B8%D0%B5%D0%BD%D1%82%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D1%83%D1%81%D0%BA) 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.

#### Что такое RMSprop?

*RMSProp* (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Root Mean Square Propagation*) — это метод, в котором [скорость обучения](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%BA%D0%BE%D1%80%D0%BE%D1%81%D1%82%D1%8C_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D1%8F&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/learning_rate) настраивается для каждого параметра. Идея заключается в делении скорости обучения для весов на сгруппированные средние значения недавних градиентов для этого веса.

#### Что такое MSE? В каких случаях она плохо отражает качество восстановление регрессии?

Для вычисления среднеквадратической ошибки (MSE) все отдельные [остатки регрессии](http://statistica.ru/glossary/general/ostatki-regressii/) возводятся в квадрат, суммируются, сумма делится на общее число ошибок:

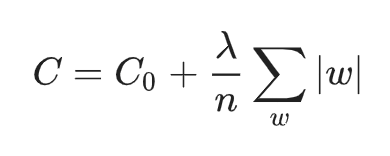


#### Что такое линейный классификатор?

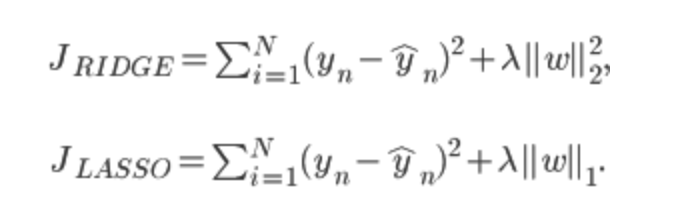
04.03

1. Какие вы знаете аналитические способы решения задачи линейной регрессии? Опишите принципы их работы.
2. Что такое гребневая регрессия?
3. Что такой лассо тибширани?

В данном подходе мы изменяем нерегуляризованную функцию стоимости, добавляя сумму абсолютных значений весов:



Интуитивно это похоже на регуляризацию L2, штрафующую за большие веса и заставляющую сеть предпочитать малые веса. Конечно, член регуляризации L1 не похож на член регуляризации L2, поэтому не стоит ожидать ровно такого же поведения. Давайте попробуем понять, в чём поведение сети, обученной при помощи регуляризации L1, отличается от сети, обученной при помощи регуляризации L2.



1. Зачем вводятся релаксационные переменные (эпсилон) в постановку задачи SVM?
2. Как работает алгоритм одного ближайшего соседа? В чём его преимущества и недостатки? Какая структура данных используется для эффективной работы данного алгоритма?
3. Опишите популярные критерии информативности разделяющих правил в деревьях решений, которые вы знаете? (минимум 2)
4. Опишите принцип работы алгоритма опорных векторов. Как в этом алгоритме определяется лучшая возможная разделяющая поверхность?
5. Что такое кросс-валидация, зачем она нужна? Какие виды кросс-валидации существуют, в чём их особенности?
6. Что такое матрица неточностей? Как вычисляется полнота, точность (precision) и F-мера?

## Деревья решений и алгоритмы бустинга

1. Как работает стэкинг(stacking)?

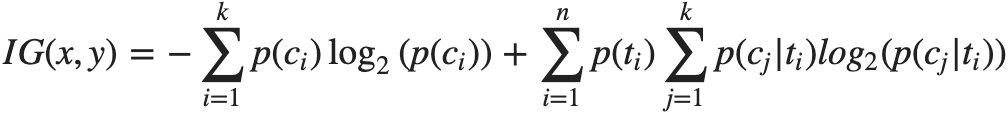
#### Что такое подрезка (pruning)?

Стрижка дерева (*pruning*). При таком подходе дерево сначала строится до максимальной глубины, потом постепенно, снизу вверх, некоторые вершины дерева убираются за счет сравнения по качеству дерева с данным разбиением и без него (сравнение проводится с помощью *кросс-валидации*).

#### Как работает случайный лес? Что такое bagging и subsampling?

Случайный лес (композиция многих деревьев) усредняет ответы деревьев, построенных до максимальной глубины.

1. Как при обучении дерева подбираются разделяющие правила для каждой вершины?
2. Опишите популярные критерии информативности разделяющих правил в максимаstackinдеревьях решений
3. Кратко опишите как работает градиентный бустинг
4. Проблема затухающего градиента
5. Что такое Information gain?

* 

## Байесовский классификатор

#### Как формулируется м максимального правдоподобия и зачем он нужен?максима

Ме́тод максима́льного правдоподо́бия или метод наибольшего правдоподобия в математической статистике — это метод оценивания неизвестного параметра путём максимизации функции правдоподобия. Основан на предположении о том, что вся информация о статистической выборке содержится в функции правдоподобия.

#### Что такое наивный байесовский классификатор?

Наивный байесовский алгоритм – это алгоритм классификации, основанный на теореме Байеса с допущением о независимости признаков. Другими словами, НБА предполагает, что наличие какого-либо признака в классе не связано с наличием какого-либо другого признака. Например, фрукт может считаться яблоком, если он красный, круглый и его диаметр составляет порядка 8 сантиметров. Даже если эти признаки зависят друг от друга или от других признаков, в любом случае они вносят независимый вклад в вероятность того, что этот фрукт является яблоком. В связи с таким допущением алгоритм называется «наивным».

Положительные стороны:

Классификация, в том числе многоклассовая, выполняется легко и быстро.

Когда допущение о независимости выполняется, НБА превосходит другие алгоритмы, такие как логистическая регрессия (logistic regression), и при этом требует меньший объем обучающих данных.

НБА лучше работает с категорийными признаками, чем с непрерывными. Для непрерывных признаков предполагается нормальное распределение, что является достаточно сильным допущением.

Отрицательные стороны:

Если в тестовом наборе данных присутствует некоторое значение категорийного признака, которое не встречалось в обучающем наборе данных, тогда модель присвоит нулевую вероятность этому значению и не сможет сделать прогноз. Это явление известно под названием «нулевая частота» (zero frequency). Данную проблему можно решить с помощью сглаживания. Одним из самых простых методов является сглаживание по Лапласу (Laplace smoothing).

Хотя НБА является хорошим классификатором, значения спрогнозированных вероятностей не всегда являются достаточно точными. Поэтому не следует слишком полагаться на результаты, возвращенные методом predict\_proba.

Еще одним ограничением НБА является допущение о независимости признаков. В реальности наборы полностью независимых признаков встречаются крайне редко.

1. Как формулируется принцип максимизации апостериорной вероятности и зачем нужен?

#### Что такое оптимальный байесовский классификатор?

оптимальный байесовский классификатор в отличие от наивного не имеет допущения о независимости признаков

Свёрточные нейронные сети

* (A) Семантическая сегментация
* (Б) Классификация с кластеризацией
* (В) Детектирование объектов
* (Г) Сегментация объектов

1. В чём сходство и различия свёртки и пулинга?
2. Методы заполнения краёв при свёртке
3. Разница между А и Б
4. Разница между В и Г
5. Разница между А и В
6. Разница между Б и Г

Кластеризация

1. Как устроен алгоритм DBScan?
2. Как ставится задача кластеризации?
3. Как устроен алгоритм среднего сдвига?
4. Как устроен EM-алгоритм
5. Что такое внешние меры оценки качества алгоритмов кластеризации?
   1. Что такое внутренние меры оценки качества алгоритмов кластеризации?
6. Что такое дендрограмма для иерархической кластеризации?
7. Опишите принцип работы иерархической кластеризации? Опишите различные меры расстояния между кластерами.
8. Как устроен алгоритм алгоритм k-средних для чёткой и для нечёткой кластеризации?

# Теоретический минимум к экзамену

# по предмету «Машинное обучение»

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

теория **сильного** искусственного интеллекта предполагает, что компьютеры могут приобрести способность [мыслить](https://ru.wikipedia.org/wiki/%D0%9C%D1%8B%D1%88%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5) и [осознавать](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BC%D0%BE%D1%81%D0%BE%D0%B7%D0%BD%D0%B0%D0%BD%D0%B8%D0%B5) себя как отдельную личность (в частности, понимать собственные мысли), хотя и не обязательно, что их мыслительный процесс будет [подобен](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D0%BE%D0%B1%D0%B8%D0%B5_(%D1%84%D0%B8%D0%B7%D0%B8%D0%BA%D0%B0)) человеческому.

теория **слабого** искусственного интеллекта отвергает такую возможность.

методы и алгоритмы извлечения знаний из экспериментальных (в широком смысле) данных; процесс исследования, фильтрации, преобразования и моделирования [данных](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5) с целью извлечения полезной информации и принятия решений.

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

## Классификация; регрессия; ранжирование; прогнозирование; метод обучения; функция потерь; эмпирический риск; метод минимизации эмпирического риска;

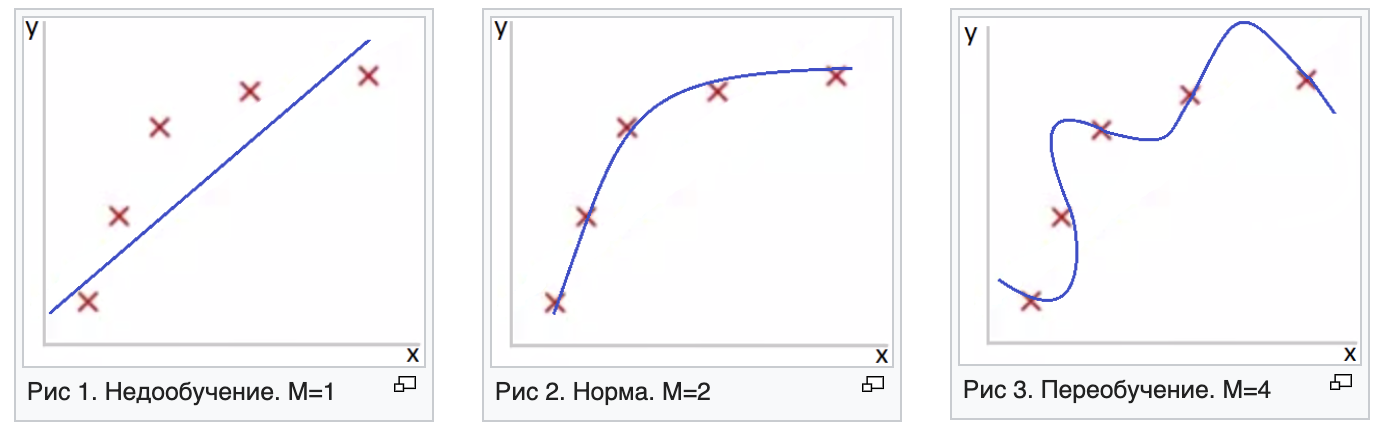
## Гиперпараметры алгоритма.

Гиперпараметрами называют параметры алгоритмов, значения которых устанавливаются перед запуском процесса обучения. В этом смысле они и отличаются от обычных параметров, вычисляемых в процессе обучения.

## 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.

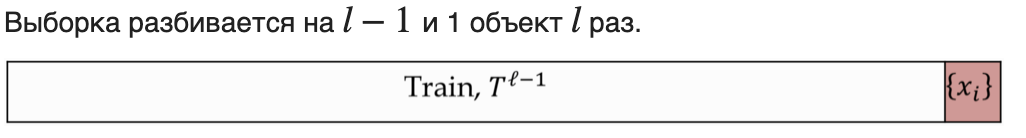
Представьте задачу линейной регрессии. Красные точки представляют исходные данные. Синие линии являются графиками полиномов различной степени M, аппроксимирующих исходные данные.



**Валидация?**

**Кросс-валидация** или **скользящий контроль** — процедура эмпирического оценивания обобщающей способности алгоритмов. С помощью кросс-валидации эмулируется наличие тестовой выборки, которая не участвует в обучении, но для которой известны правильные ответы.

Кросс-валидация по отдельным объектам(Leave-One-Out)



Преимущества LOO в том, что каждый объект ровно один раз участвует в контроле, а длина обучающих подвыборок лишь на единицу меньше длины полной выборки.

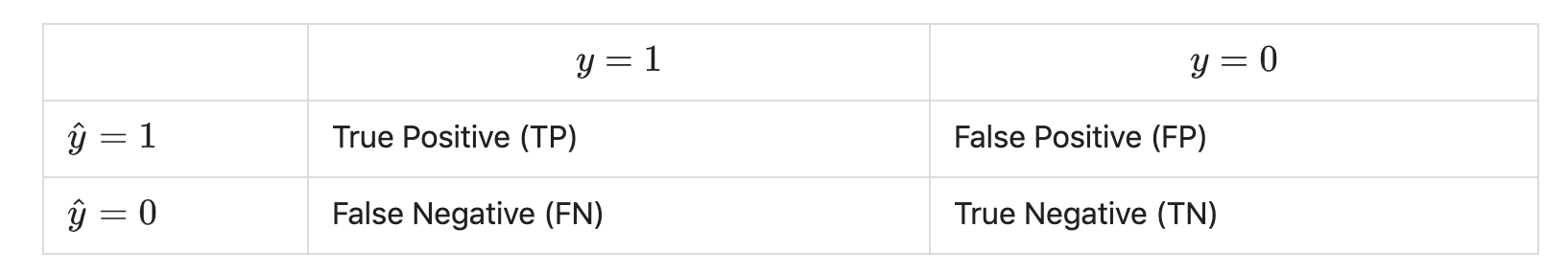
Недостатком LOO является большая ресурсоёмкость, так как обучаться приходится

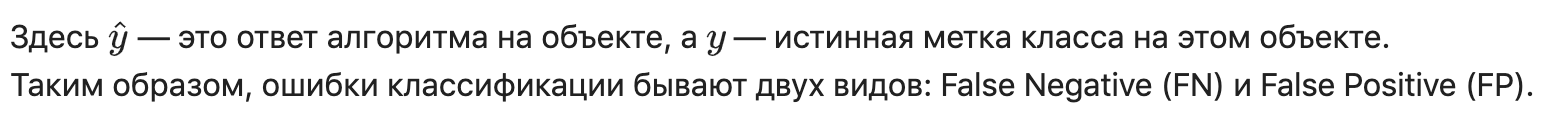
L раз. Некоторые методы обучения позволяют достаточно быстро перенастраивать внутренние параметры алгоритма при замене одного обучающего объекта другим. В этих случаях вычисление LOO удаётся заметно ускорить

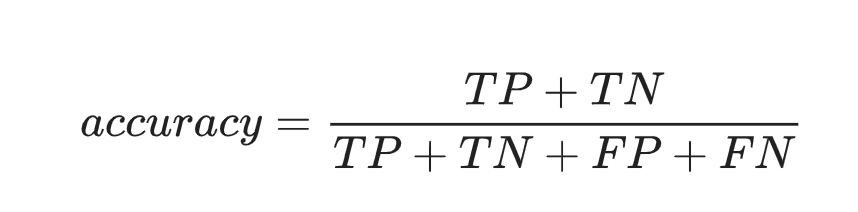
Регуляризация метод добавления некоторых дополнительных ограничений к условию с целью решить некорректно поставленную задачу или предотвратить переобучение. Эта информация часто имеет вид штрафа за сложность модели. Например, это могут быть ограничения гладкости результирующей функции или ограничения по норме векторного пространства.

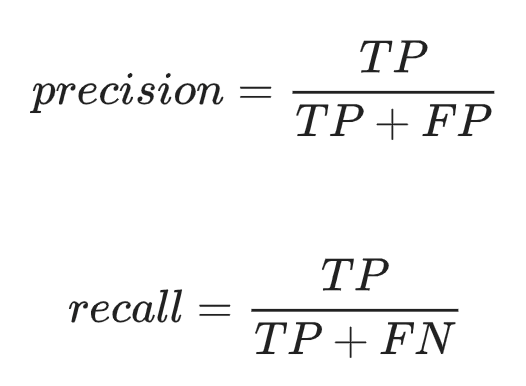
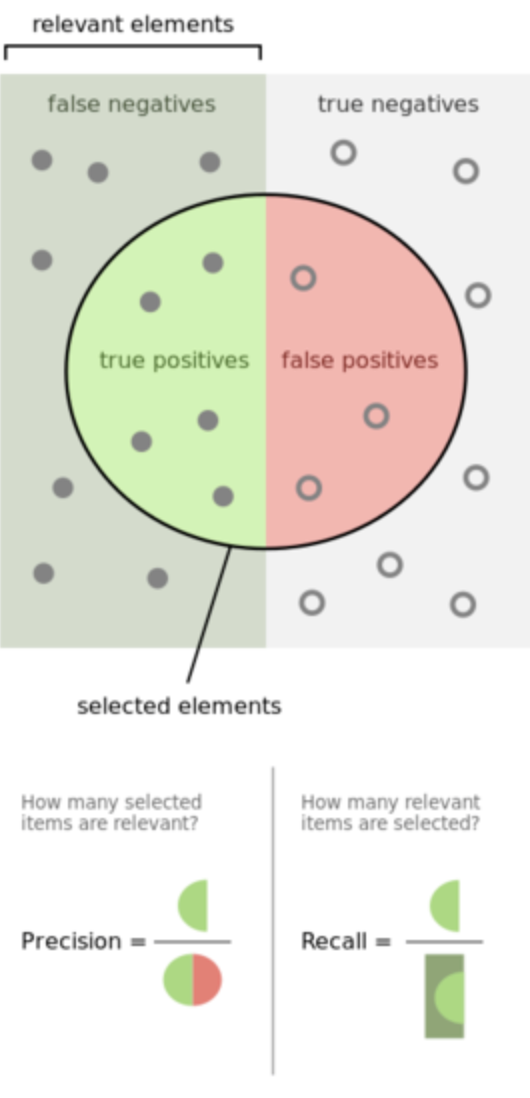
## Точность; полнота; accuracy; F-мера; true positive; false positive; ROC-кривая; среднеквадратическое отклонение

**Матрица ошибок**



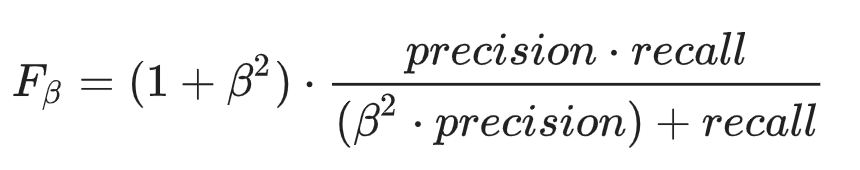




Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а recall показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

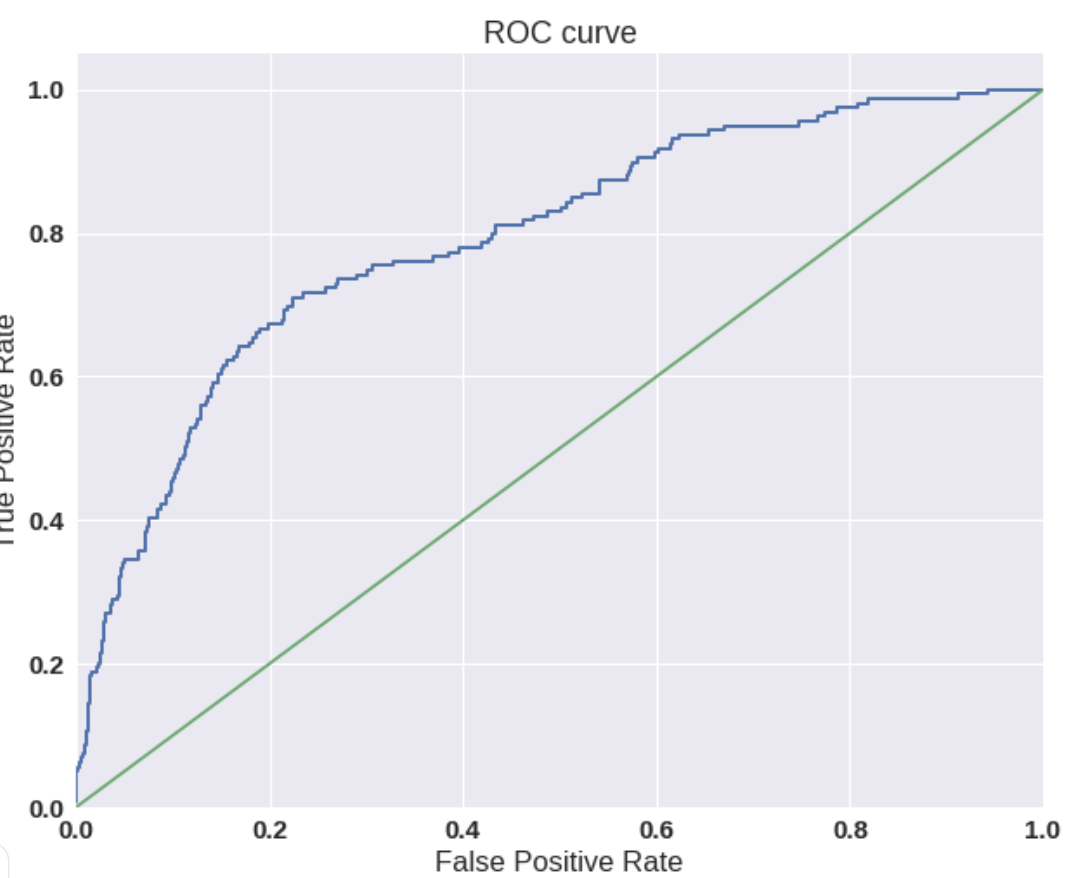
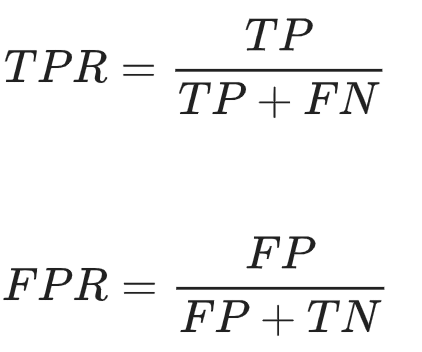
F-мера — среднее гармоническое precision и recall



βв данном случае определяет вес точности в метрике, и при β=1 это среднее гармоническое

ROC

Одним из способов оценить модель в целом, не привязываясь к конкретному порогу, является AUC-ROC (или ROC AUC) — площадь (*A*rea *U*nder *C*urve) под кривой ошибок (*R*eceiver *O*perating *C*haracteristic curve ). Данная кривая представляет из себя линию от (0,0) до (1,1) в координатах True Positive Rate (TPR) и False Positive Rate (FPR):



Критерий AUC-ROC устойчив к несбалансированным классам (спойлер: увы, не всё так однозначно) и может быть интерпретирован как вероятность того, что случайно выбранный positive объект будет проранжирован классификатором выше (будет иметь более высокую вероятность быть positive), чем случайно выбранный negative объект.

**Среднеквадратическое отклонение**

## Метод ближайших соседей

**Метод ближайших соседей** (k Nearest Neighbors, или kNN) — популярный метод классификации, также иногда используемый в задачах регрессии. Это, наравне с деревом решений, один из самых понятных подходов к классификации. На уровне интуиции суть метода такова: посмотри на соседей, какие преобладают, таков и ты. Формально основой метода является гипотеза компактности: если метрика расстояния между примерами введена достаточно удачно, то схожие примеры гораздо чаще лежат в одном классе, чем в разных.

Для классификации каждого из объектов тестовой выборки необходимо последовательно выполнить следующие операции:

* Вычислить расстояние до каждого из объектов обучающей выборки
* Отобрать k объектов обучающей выборки, расстояние до которых минимально
* Класс классифицируемого объекта — это класс, наиболее часто встречающийся среди k ближайших соседей

Под задачу регрессии метод адаптируется довольно легко – на 3 шаге возвращается не метка, а число – среднее (или медианное) значение целевого признака среди соседей.

**Плюсы и минусы метода ближайших соседей**

Плюсы:

* Простая реализация;
* Неплохо изучен теоретически;
* Как правило, метод хорош для первого решения задачи, причем не только классификации или регрессии, но и, например, рекомендации;
* Можно адаптировать под нужную задачу выбором метрики или ядра (в двух словах: ядро может задавать операцию сходства для сложных объектов типа графов, а сам подход kNN остается тем же).
* Неплохая интерпретация, можно объяснить, почему тестовый пример был классифицирован именно так. Хотя этот аргумент можно атаковать: если число соседей большое, то интерпретация ухудшается (условно: "мы не дали ему кредит, потому что он похож на 350 клиентов, из которых 70 – плохие, что на 12% больше, чем в среднем по выборке").

Минусы:

* Метод считается быстрым в сравнении, например, с композициями алгоритмов, но в реальных задачах, как правило, число соседей, используемых для классификации, будет большим (100-150), и в таком случае алгоритм будет работать не так быстро, как дерево решений;
* Если в наборе данных много признаков, то трудно подобрать подходящие веса и определить, какие признаки не важны для классификации/регрессии;
* Зависимость от выбранной метрики расстояния между примерами. Выбор по умолчанию евклидового расстояния чаще всего ничем не обоснован. Можно отыскать хорошее решение перебором параметров, но для большого набора данных это отнимает много времени;
* Нет теоретических оснований выбора определенного числа соседей — только перебор (впрочем, чаще всего это верно для всех гиперпараметров всех моделей). В случае малого числа соседей метод чувствителен к выбросам, то есть склонен переобучаться;
* Как правило, плохо работает, когда признаков много

## Метод непараметрической регрессии

**Непараметрическая регрессия**, в отличие от параметрических подходов, использует модель, которая не описывается конечным числом параметров.

## Метод стохастического градиентного спуска

**Стохастический градиентный спуск** (англ. *stochastic gradient descent*)

− оптимизационный алгоритм, отличающийся от обычного [градиентного спуска](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D0%B8%D0%B5%D0%BD%D1%82%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D1%83%D1%81%D0%BA) 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.

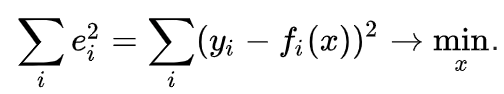
при обучении машин необходимость установки [скорости обучения](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%BA%D0%BE%D1%80%D0%BE%D1%81%D1%82%D1%8C_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D1%8F&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/learning_rate) (размера шага) была признана проблематичной. Установка этого параметра слишком высоким может привести алгоритм к отсутствию сходимости. Установка же слишком низким приводит к медленной сходимости.

*AdaGrad* (адаптивный градиентный алгоритм, [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *adaptive gradient algorithm*) является модификацией стохастического алгоритма градиентного спуска с отдельной для каждого параметра [скоростью обучения](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%BA%D0%BE%D1%80%D0%BE%D1%81%D1%82%D1%8C_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D1%8F&action=edit&redlink=1)

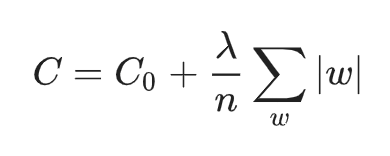
*RMSProp* (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Root Mean Square Propagation*) — это метод, в котором [скорость обучения](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%BA%D0%BE%D1%80%D0%BE%D1%81%D1%82%D1%8C_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D1%8F&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/learning_rate) настраивается для каждого параметра. Идея заключается в делении скорости обучения для весов на сгруппированные средние значения недавних градиентов для этого веса

*Adam*[[26]](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%BE%D1%85%D0%B0%D1%81%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B3%D1%80%D0%B0%D0%B4%D0%B8%D0%B5%D0%BD%D1%82%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D1%83%D1%81%D0%BA#cite_note-DeBa-27) (сокращение от «метод Адаптивной Оценки Моментов», [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Adaptive Moment Estimation*) — это обновление *RMSProp* оптимизатора. В этом оптимизационном алгоритме используются сгруппированные средние как градиентов, так и вторых моментов градиентов.

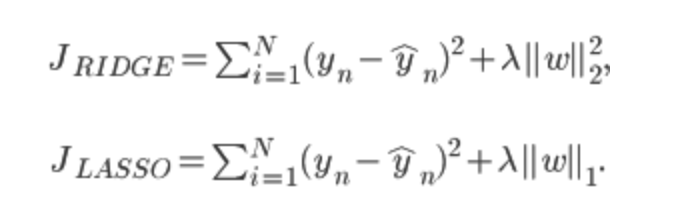
## Метод линейной регрессии; гребневая регрессия; лассо Тибширани

Один из способов вычислить значения параметров модели является метод наименьших квадратов(МНК), который минимизирует среднеквадратичную ошибку между реальным значением зависимой переменной и прогнозом, выданным моделью:

**Лассо Тибширани**. В данном подходе мы изменяем нерегуляризованную функцию стоимости, добавляя сумму абсолютных значений весов:



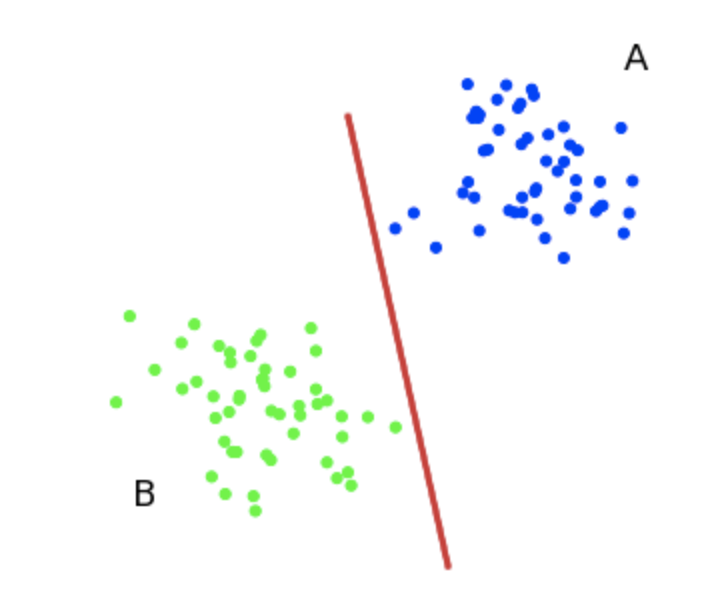
Интуитивно это похоже на регуляризацию L2, штрафующую за большие веса и заставляющую сеть предпочитать малые веса. Конечно, член регуляризации L1 не похож на член регуляризации L2, поэтому не стоит ожидать ровно такого же поведения. Давайте попробуем понять, в чём поведение сети, обученной при помощи регуляризации L1, отличается от сети, обученной при помощи регуляризации L2.



## Метод сингулярного векторного разложения

**Cингулярное разложение** (Singular Value Decomposition, SVD) — декомпозиция [вещественной](http://www.machinelearning.ru/wiki/index.php?title=%D0%92%D0%B5%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE&action=edit) [матрицы](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0&action=edit) с целью ее приведения к [каноническому виду](http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D0%B0%D0%BD%D0%BE%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B2%D0%B8%D0%B4&action=edit). Сингулярное разложение является удобным методом при работе с матрицами. Оно показывает геометрическую структуру матрицы и позволяет наглядно представить имеющиеся данные. Сингулярное разложение используется при решении самых разных задач — от приближения [методом наименьших квадратов](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BD%D0%B0%D0%B8%D0%BC%D0%B5%D0%BD%D1%8C%D1%88%D0%B8%D1%85_%D0%BA%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D0%BE%D0%B2) и решения систем уравнений до [сжатия изображений](http://www.machinelearning.ru/wiki/index.php?title=%D0%A1%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D0%B8%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9&action=edit). При этом используются разные свойства сингулярного разложения, например, способность показывать [ранг матрицы](http://www.machinelearning.ru/wiki/index.php?title=%D0%A0%D0%B0%D0%BD%D0%B3_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D1%8B&action=edit), приближать матрицы данного ранга. SVD позволяет вычислять обратные и [псевдообратные матрицы](http://www.machinelearning.ru/wiki/index.php?title=%D0%9F%D1%81%D0%B5%D0%B2%D0%B4%D0%BE%D0%BE%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0) большого размера, что делает его полезным инструментом при решении задач [регрессионного анализа](http://www.machinelearning.ru/wiki/index.php?title=%D0%A0%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7).

## Метод опорных векторов (общая идея)

**Метод опорных векторов** (англ. *support vector machine*, *SVM*) — один из методов обучения, который применяется для решения задач классификации и регрессии. Основная идея метода 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 уверенной 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 C при помощи кросс-валидации.

## Ядро; ядерный трюк для метода опорных векторов(SVM)

## Вероятностная постановка задачи классификации; правдоподобие класса; метод максимальной апостериорной вероятности; оптимальный наивныйбайесовский классификатор

## Метод оценки Парзена-Розенблатта;

**Метод парзеновского окна** — метод байесовской классификации, основанный на непараметрическом восстановлении плотности по имеющейся выборке.

После ввода метрики, метод парзеновского окна можно использовать, не опираясь на вероятностную природу данных.

В основе подхода лежит идея о том, что плотность выше в тех точках, рядом с которыми находится большое количество объектов выборки.

Если мощность множества элементарных исходов много меньше размера выборки, то в качестве восстановленной по выборке плотности мы вполне можем взять и гистограмму значений выборки.

В противном случае (например, непрерывном) данный подход не применим, так как плотность концентрируется вблизи обучающих объектов, и функция распределения претерпевает резкие скачки. Приходится использовать восстановление методом Парзена-Розенблатта.

## Наивный байесовский классификатор

## Задача параметрической оценки плотности; принцип максимального правдоподобия

## Метод логистической регрессии; сигмоида

**Логистическая регрессия** (англ. *logistic regression*) — метод построения линейного классификатора, позволяющий оценивать апостериорные вероятности принадлежности объектов классам.

Логистическая [регрессия](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) применяется для прогнозирования вероятности возникновения некоторого события по значениям множества признаков. Для этого вводится так называемая *зависимая переменная* y, принимающая лишь одно из двух значений — как правило, это числа 0 (событие не произошло) и 1 (событие произошло), и множество *независимых переменных* (также называемых признаками, предикторами или регрессорами) — вещественных x1...xn,на основе значений которых требуется вычислить вероятность принятия того или иного значения зависимой переменной.

Сигмоида:

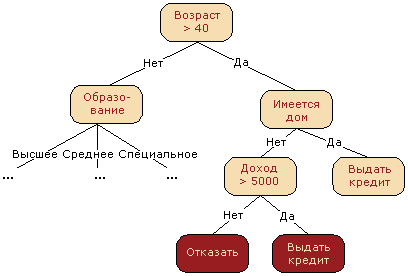


## Понятие (concept), правильно (rule)

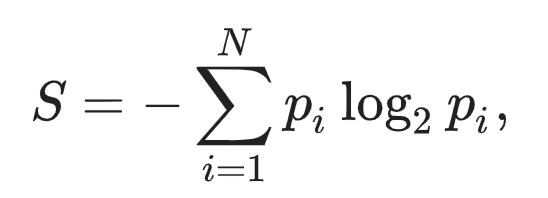
Концепт это предикат на множестве объектов X. Концепт покрывает объект, если возвращает true(1) на нем

Правило это предикат, который покрывает много объектов из одного класса, мало объектов из других классов и который может быть легко интерпретирован.

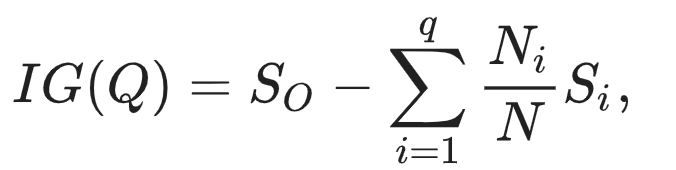
## Дерево принятия решений; подрезка



Энтропия



Поскольку энтропия – по сути степень хаоса (или неопределенности) в системе, уменьшение энтропии называют приростом информации. Формально прирост информации (information gain, IG) при разбиении выборки по признаку Q определяется как



где q – число групп после разбиения, Ni – число элементов выборки, у которых признак Q имеет i-ое значение.

В основе популярных алгоритмов построения дерева решений,лежит принцип жадной максимизации прироста информации – на каждом шаге выбирается тот признак, при разделении по которому прирост информации оказывается наибольшим. Дальше процедура повторяется рекурсивно, пока энтропия не окажется равной нулю или какой-то малой величине (если дерево не подгоняется идеально под обучающую выборку во избежание переобучения).

В разных алгоритмах применяются разные эвристики для "ранней остановки" или "отсечения", чтобы избежать построения переобученного дерева.

Основные способы борьбы с переобучением в случае деревьев решений:

* искусственное ограничение глубины или минимального числа объектов в листе: построение дерева просто в какой-то момент прекращается;
* стрижка дерева

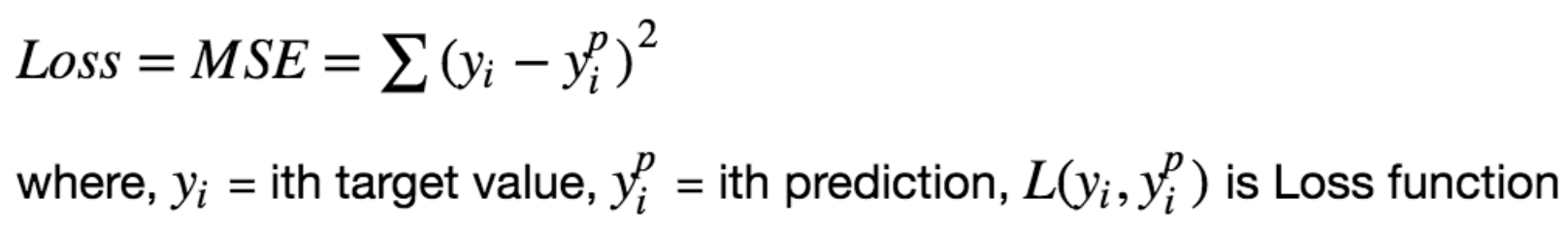
## Бустинг алгоритмов; метод градиентного бустинга

**Бустинг** — это техника построения ансамблей, в которой предсказатели построены не независимо, а последовательно

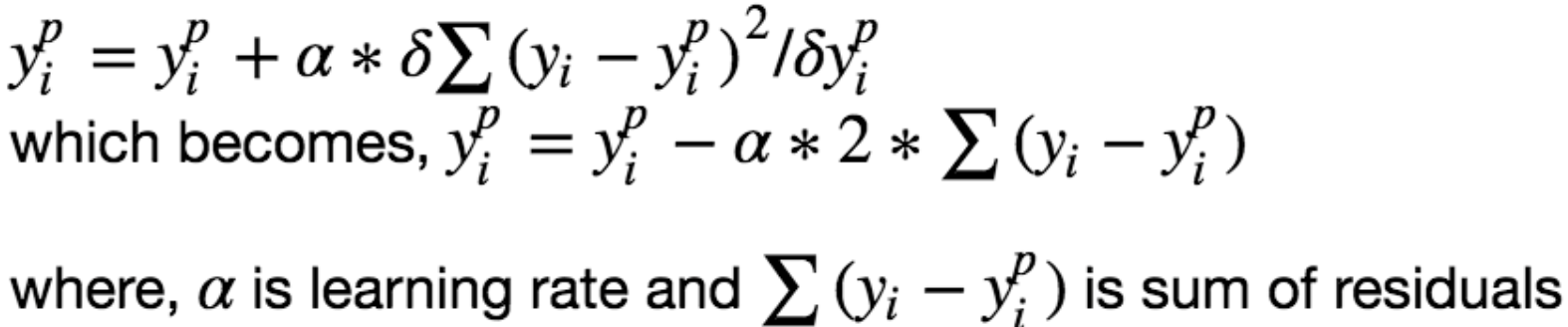
Это техника использует идею о том, что следующая модель будет учится на ошибках предыдущей. Они имеют неравную вероятность появления в последующих моделях, и чаще появятся те, что дают наибольшую ошибку. Предсказатели могут быть выбраны из широкого ассортимента моделей, например, [деревья решений](https://neurohive.io/ru/osnovy-data-science/derevo-resheniy/), [регрессия](https://neurohive.io/ru/osnovy-data-science/linejnaja-regressija/), классификаторы и т.д. Из-за того, что предсказатели обучаются на ошибках, совершенных предыдущими, требуется меньше времени для того, чтобы добраться до реального ответа. Но мы должны выбирать критерий остановки с осторожностью, иначе это может привести к переобучению. Градиентный бустинг — это пример бустинга.

**Градиентный бустинг** — это техника машинного обучения для задач классификации и регрессии, которая строит модель предсказания в форме ансамбля слабых предсказывающих моделей, обычно деревьев решений.

Цель любого алгоритма [обучения с учителем](https://neurohive.io/ru/osnovy-data-science/obuchenie-s-uchitelem-bez-uchitelja-s-podkrepleniem/) — определить функцию потерь и минимизировать её. Давайте обратимся к математике градиентного бустинга. Пусть, например, в качестве функции потерь будет среднеквадратичная ошибка (MSE):



Мы хотим, чтобы построить наши предсказания таким образом, чтобы MSE была минимальна. Используя градиентный спуск и обновляя предсказания, основанные на скорости обучения (learning rate), ищем значения, на которых MSE минимальна.



Итак, мы просто обновляем предсказания таким образом, что сумма наших отклонений стремилась к нулю и предсказанные значения были близки к реальным.

## Метод AdaBoost

**Алгоритм AdaBoost** (сокр. от adaptive boosting) — алгоритм машинного обучения, в процессе обучения строит композицию из базовых алгоритмов обучения для улучшения их эффективности. AdaBoost является алгоритмом адаптивного [бустинга](http://www.machinelearning.ru/wiki/index.php?title=%D0%91%D1%83%D1%81%D1%82%D0%B8%D0%BD%D0%B3) в том смысле, что каждый следующий классификатор строится по объектам, которые плохо классифицируются предыдущими классификаторами.

AdaBoost вызывает слабый классификатор в цикле. 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 классификатор «фокусирует своё внимание» на этих объектах.

### **Достоинства**

* Хорошая обобщающая способность. В реальных задачах (не всегда, но часто) удаётся строить композиции, превосходящие по качеству базовые алгоритмы. Обобщающая способность может улучшаться (в некоторых задачах) по мере увеличения числа базовых алгоритмов.
* Простота реализации.
* Собственные накладные расходы бустинга невелики. Время построения композиции практически полностью определяется временем обучения базовых алгоритмов.
* Возможность идентифицировать объекты, являющиеся шумовыми выбросами.

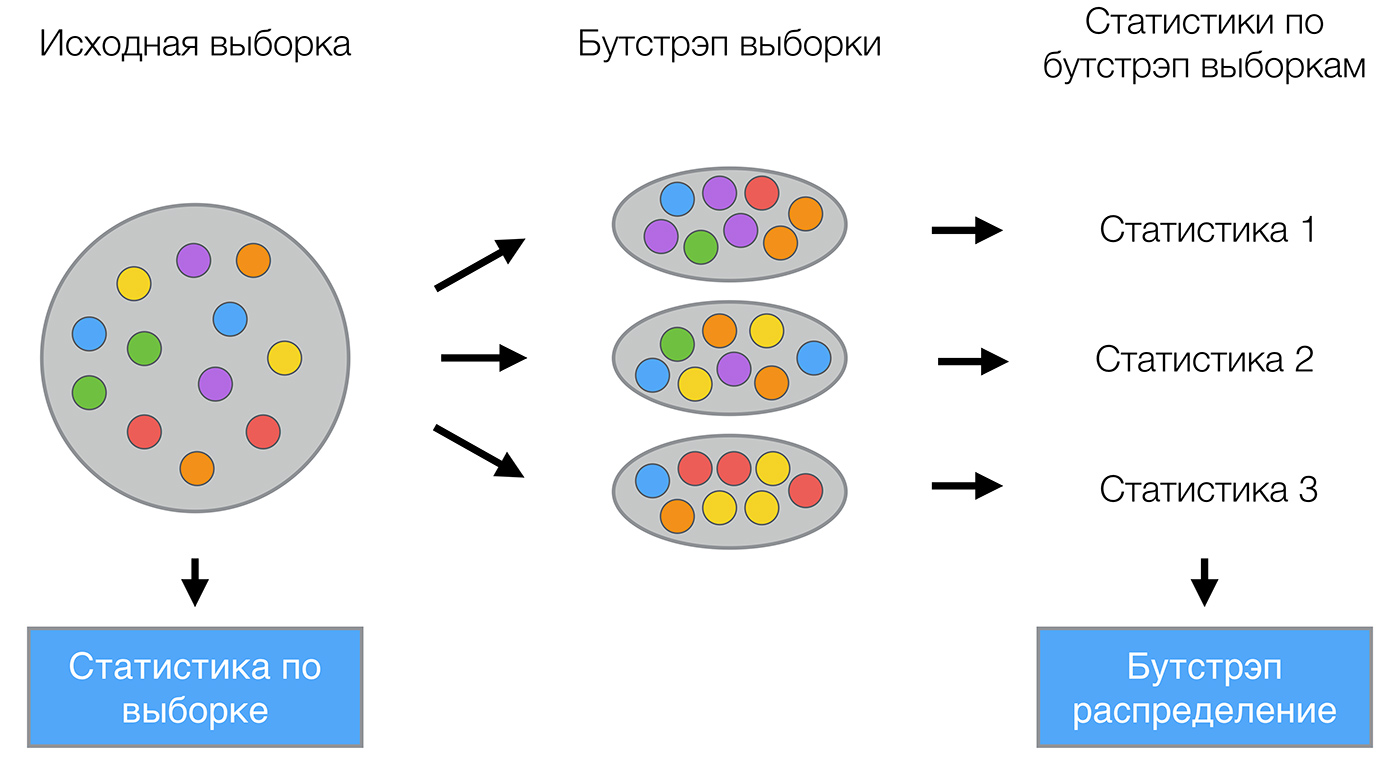
### **Недостатки**

* AdaBoost склонен к переобучению при наличии значительного уровня шума в данных. Экспоненциальная функция потерь слишком сильно увеличивает веса наиболее трудных объектов, на которых ошибаются многие базовые алгоритмы. Однако именно эти объекты чаще всего оказываются шумовыми выбросами. В результате AdaBoost начинает настраиваться на шум, что ведёт к переобучению. Проблема решается путём удаления выбросов или применения менее агрессивных функций потерь.
* AdaBoost требует достаточно длинных обучающих выборок. Другие методы линейной коррекции, в частности, бэггинг, способны строить алгоритмы сопоставимого качества по меньшим выборкам данных.
* Жадная стратегия последовательного добавления приводит к построению неоптимального набора базовых алгоритмов. Для улучшения композиции можно периодически возвращаться к ранее построенным алгоритмам и обучать их заново. Для улучшения коэффициентов можно оптимизировать их ещё раз по окончании процесса бустинга с помощью какого-нибудь стандартного метода построения линейной разделяющей поверхности. Рекомендуется использовать для этой цели[SVM (машины опорных векторов)](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%B0_%D0%BE%D0%BF%D0%BE%D1%80%D0%BD%D1%8B%D1%85_%D0%B2%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%BE%D0%B2).
* Бустинг 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.

## Бустрап; случайный лес; стэкинг

Bagging (от Bootstrap aggregation) — это один из первых и самых простых видов ансамблей. Бэггинг основан на статистическом методе бутстрэпа, который позволяет оценивать многие статистики сложных распределений.

Метод бутстрэпа заключается в следующем. Пусть имеется выборка X размера. Равномерно возьмем из выборки N объектов с возвращением. Это означает, что мы будем N раз выбирать произвольный объект выборки (считаем, что каждый объект «достается» с одинаковой вероятностью 1N), причем каждый раз мы выбираем из всех исходных N объектов. Можно представить себе мешок, из которого достают шарики: выбранный на каком-то шаге шарик возвращается обратно в мешок, и следующий выбор опять делается равновероятно из того же числа шариков. Отметим, что из-за возвращения среди них окажутся повторы. Обозначим новую выборку через X1. Повторяя процедуру M раз, сгенерируем M подвыборок X1,…,XM. Теперь мы имеем достаточно большое число выборок и можем оценивать различные статистики исходного распределения.

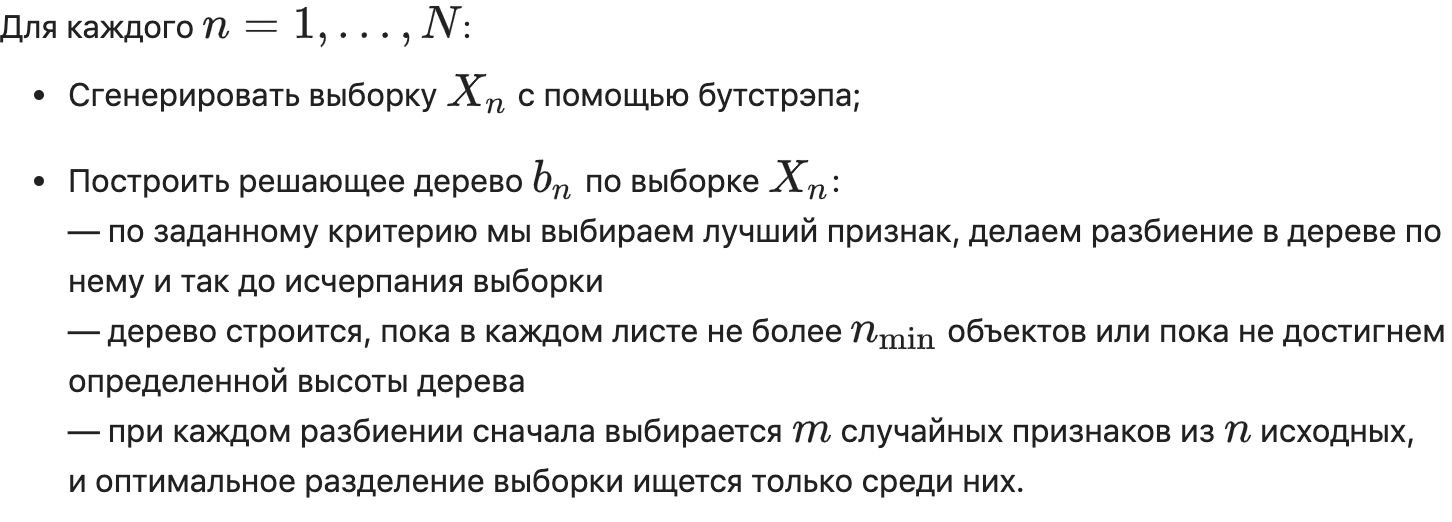


**Случайный лес**  - (композиция многих деревьев) усредняет ответы деревьев, построенных до максимальной глубины.

Решающие деревья являются хорошим семейством базовых классификаторов для бэггинга, поскольку они достаточно сложны и могут достигать нулевой ошибки на любой выборке. Метод случайных подпространств позволяет снизить коррелированность между деревьями и избежать переобучения. Базовые алгоритмы обучаются на различных подмножествах признакового описания, которые также выделяются случайным образом.

Ансамбль моделей, использующих метод случайного подпространства, можно построить, используя следующий алгоритм:

Алгоритм построения случайного леса, состоящего из N деревьев, выглядит следующим образом:

Плюсы и минусы случайного леса

Плюсы:

— имеет высокую точность предсказания, на большинстве задач будет лучше линейных алгоритмов; точность сравнима с точностью бустинга

— практически не чувствителен к выбросам в данных из-за случайного сэмлирования

— не чувствителен к масштабированию (и вообще к любым монотонным преобразованиям) значений признаков, связано с выбором случайных подпространств

— не требует тщательной настройки параметров, хорошо работает «из коробки». С помощью «тюнинга» параметров можно достичь прироста от 0.5 до 3% точности в зависимости от задачи и данных

— способен эффективно обрабатывать данные с большим числом признаков и классов

— одинаково хорошо обрабатывет как непрерывные, так и дискретные признаки

— редко переобучается, на практике добавление деревьев почти всегда только улучшает композицию, но на валидации, после достижения определенного количества деревьев, кривая обучения выходит на асимптоту

— для случайного леса существуют методы оценивания значимости отдельных признаков в модели

— хорошо работает с пропущенными данными; сохраняет хорошую точность, если большая часть данных пропущена

— предполагает возможность сбалансировать вес каждого класса на всей выборке, либо на подвыборке каждого дерева

— вычисляет близость между парами объектов, которые могут использоваться при кластеризации, обнаружении выбросов или (путем масштабирования) дают интересные представления данных

— возможности, описанные выше, могут быть расширены до неразмеченных данных, что приводит к возможности делать кластеризацию и визуализацию данных, обнаруживать выбросы

— высокая параллелизуемость и масштабируемость.

Минусы:

— в отличие от одного дерева, результаты случайного леса сложнее интерпретировать

— нет формальных выводов (p-values), доступных для оценки важности переменных

— алгоритм работает хуже многих линейных методов, когда в выборке очень много разреженных признаков (тексты, Bag of words)

— случайный лес не умеет экстраполировать, в отличие от той же линейной регрессии (но это можно считать и плюсом, так как не будет экстремальных значений в случае попадания выброса)

— алгоритм склонен к переобучению на некоторых задачах, особенно на зашумленных данных

— для данных, включающих категориальные переменные с различным количеством уровней, случайные леса предвзяты в пользу признаков с большим количеством уровней: когда у признака много уровней, дерево будет сильнее подстраиваться именно под эти признаки, так как на них можно получить более высокое значение оптимизируемого функционала (типа прироста информации)

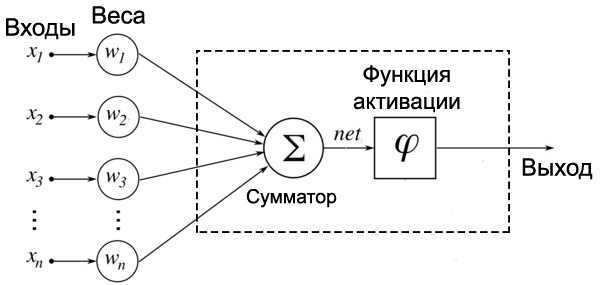
— если данные содержат группы коррелированных признаков, имеющих схожую значимость для меток, то предпочтение отдается небольшим группам перед большими

— больший размер получающихся моделей. Требуется O(NK) памяти для хранения модели, где K — число деревьев.

**Стэкинг?**

## Нейрон; перцептрон;

Схема нейрона



## Многослойная нейронная сеть; метод обратного распространения ошибок

**Многослойная нейронная сеть** (англ. *Multilayer neural network*) — нейронная сеть, состоящая из входного, выходного и расположенного(ых) между ними одного (нескольких) скрытых слоев нейронов.

Помимо входного и выходного слоев эти нейронные сети содержат промежуточные, *скрытые слои*. Такие сети обладают гораздо большими возможностями, чем однослойные нейронные сети, однако методы обучения нейронов скрытого слоя были разработаны относительно недавно.

Работу скрытых слоев нейронов можно сравнить с работой большого завода. Продукт (выходной сигнал) на заводе собирается по стадиям на станках. После каждого станка получается какой-то промежуточный результат. Скрытые слои тоже преобразуют входные сигналы в некоторые промежуточные результаты.

**Метод обратного распространения ошибок** (англ. *backpropagation*) — метод вычисления градиента, который используется при обновлении весов в [нейронной сети](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8,_%D0%BF%D0%B5%D1%80%D1%86%D0%B5%D0%BF%D1%82%D1%80%D0%BE%D0%BD).

## Аугментация данных; дропаут; ReLU

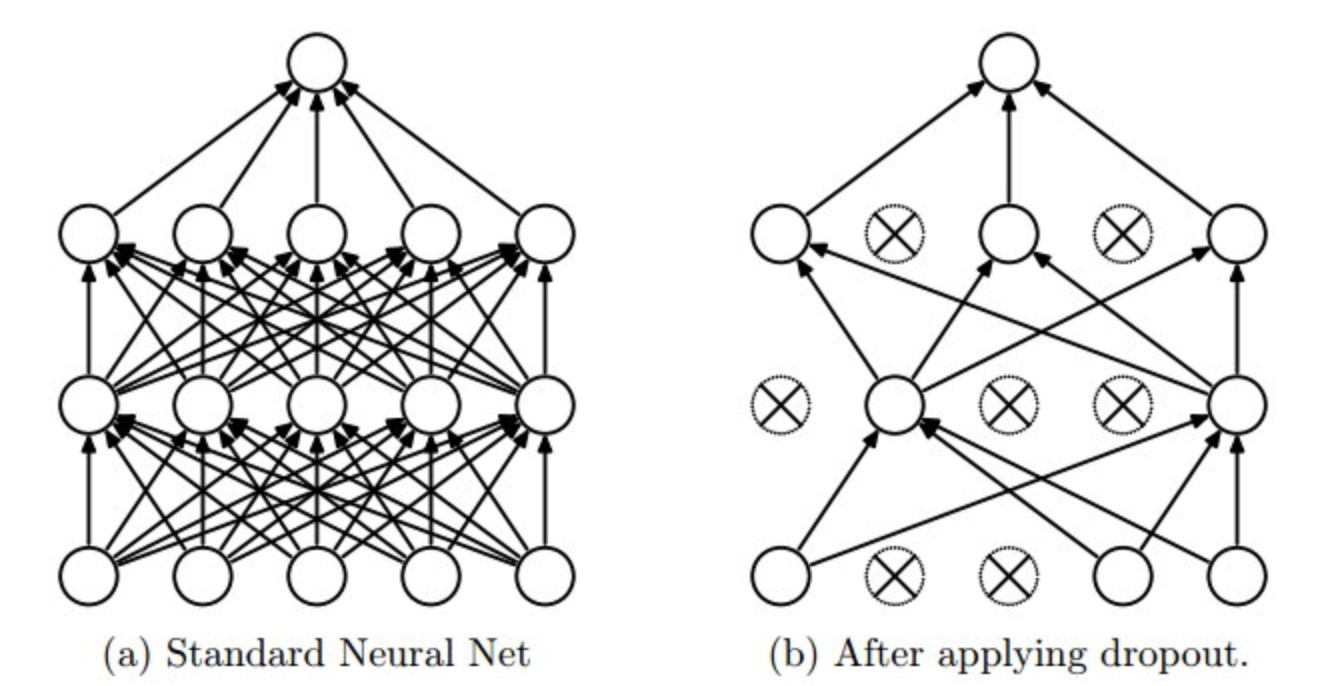
**Аугментация данных**

**Исключение(Dropout)** – совершенно другая техника регуляризации. В отличие от регуляризации L1 и L2, исключение не занимается изменением функции стоимости. Вместо этого мы изменяем саму сеть.

Главная идея Dropout — вместо обучения одной DNN обучить ансамбль нескольких DNN, а затем усреднить полученные результаты.

Сети для обучения получаются с помощью исключения из сети (dropping out) нейронов с вероятностью p, таким образом, вероятность того, что нейрон останется в сети, составляет q=1−p. “Исключение” нейрона означает, что при любых входных данных или параметрах он возвращает 0.

Исключенные нейроны не вносят свой вклад в процесс обучения ни на одном из этапов алгоритма обратного распространения ошибки (backpropagation); поэтому исключение хотя бы одного из нейронов равносильно обучению новой нейронной сети.



**ReLU**

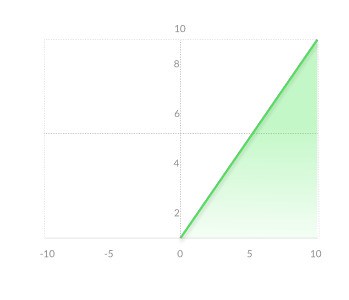
Rectified Linear Unit — это наиболее часто используемая функция активации при глубоком обучении. Данная функция возвращает 0, если принимает отрицательный аргумент, в случае же положительного аргумента, функция возвращает само число. То есть она может быть записана как

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 несколькими преимущества 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 активационный функции влечет активацию почти всех нейронов, что может сказаться 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[[2]](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B8_%D1%80%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D1%85_%D1%81%D0%B5%D1%82%D0%B5%D0%B9#cite_note-2). 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.

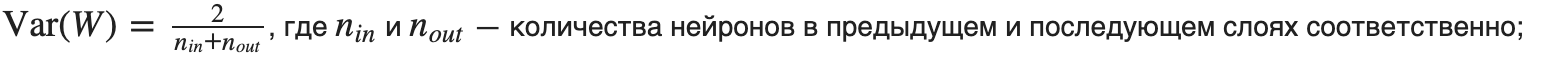


## Метод Ньютона-Рафсона

## Декорреляция; метод Xavier; метод He

**Декорреляция?**

**Метод инициализации Завьера (Xavier)** (иногда — метод Glorot’а)[[2]](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9D%D0%B0%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B0_%D0%B3%D0%BB%D1%83%D0%B1%D0%BE%D0%BA%D0%BE%D0%B9_%D1%81%D0%B5%D1%82%D0%B8#cite_note-2). 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 весов этот метод опирается на вероятностное распределение (равномерное или нормальное) с дисперсией, равной



**Метод инициализации 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



## Метод Adam

A[dam](https://arxiv.org/abs/1412.6980) — adaptive moment estimation, ещё один оптимизационный алгоритм. Он сочетает в себе и идею накопления движения и идею более слабого обновления весов для типичных признаков.

## Метод батчевой нормализации

**Пакетная нормализация** (англ. batch-normalization) — метод, который позволяет повысить производительность и стабилизировать работу [искусственных нейронных сетей](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8,_%D0%BF%D0%B5%D1%80%D1%86%D0%B5%D0%BF%D1%82%D1%80%D0%BE%D0%BD). Суть данного метода заключается в том, что некоторым слоям нейронной сети на вход подаются данные, предварительно обработанные и имеющие нулевое [математическое ожидание](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BE%D0%B6%D0%B8%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5_%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D0%BE%D0%B9_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D1%8B) и единичную [дисперсию](https://neerc.ifmo.ru/wiki/index.php?title=%D0%94%D0%B8%D1%81%D0%BF%D0%B5%D1%80%D1%81%D0%B8%D1%8F_%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D0%BE%D0%B9_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D1%8B).

## Свертка; паддинг; пулинг; страйд; тензор

**Свёртка** – это операция вычисления нового значения выбранного пикселя, учитывающая значения окружающих его пикселей. Алгоритм получения результата свёртки можно описать так: Фильтр накладывается на левую верхнюю часть изображения и производится покомпонентное умножение значений фильтра и значений изображения, после чего фильтр перемещается дальше по изображению до тех пор, подетекка аналогичным образом не будут обработаны все его участки.

Пулинг призван снижать размерность изображения. Исходное изображение делится на блоки размером w×h и для каждого блока вычисляется некоторая функция. Чаще всего используется функция максимума (англ. *max pooling*) или (взвешенного) среднего (англ. *(weighted) average pooling*).

**Тензоры** — это 3D массивы чисел, или, проще говоря, массивы матриц чисел.

## Сверточная нейронная сеть

**Сверточная нейронная сеть** (англ. *convolutional neural network*, *CNN*) — специальная архитектура нейронных сетей, изначально нацеленная на эффективное распознавание изображений.

## Задача семантической сегментации; задача детекции объектов

## Задача кластеризации; внешние меры оценки; внутренние меры оценки

Выделяют *внешние* и *внутренние* метрики качества. Внешние используют информацию об истинном разбиении на кластеры, в то время как внутренние метрики не используют никакой внешней информации и оценивают качество кластеризации, основываясь только на наборе данных. Оптимальное число кластеров обычно определяют с использованием внутренних метрик.

[https://neerc.ifmo.ru/wiki/index.php?title=Оценка\_качества\_в\_задаче\_кластеризации](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9E%D1%86%D0%B5%D0%BD%D0%BA%D0%B0_%D0%BA%D0%B0%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%B0_%D0%B2_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B5_%D0%BA%D0%BB%D0%B0%D1%81%D1%82%D0%B5%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8)

Об алгоритмах кластеризации: <https://habr.com/ru/post/101338/>

## Графовые методы кластеризации

Суть таких алгоритмов заключается в том, что выборка объектов представляется в виде графа *G=(V, E)*, вершинам которого соответствуют объекты, а ребра имеют вес, равный «расстоянию» между объектами. Достоинством графовых алгоритмов кластеризации являются наглядность, относительная простота реализации и возможность внесения различных усовершенствований, основанные на геометрических соображениях. Основными алгоритмам являются алгоритм выделения связных компонент, алгоритм построения минимального покрывающего (остовного) дерева и алгоритм послойной кластеризации.

## Иерархические методы кластеризации

|  |
| --- |
| **Иерархическая кластеризация** (англ. *hierarchical clustering*) — множество алгоритмов кластеризации, направленных на создание иерархии вложенных разбиений исходного множества объектов. |

Иерархические алгоритмы кластеризации часто называют **алгоритмами таксономии**. Для визуального представления результатов кластеризации используется **дендрограмма** — дерево, 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.

## k-means. c-means

Алгоритм К-средних, наверное, самый популярный и простой алгоритм кластеризации и очень легко представляется в виде простого псевдокода:

* Выбрать количество кластеров k которое нам кажется оптимальным для наших данных.
* Высыпать случайным образом в пространство наших данных k точек (центроидов).
* Для каждой точки нашего набора данных посчитать, к какому центроиду она ближе.
* Переместить каждый центроид в центр выборки, которую мы отнесли к этому центроиду.
* Повторять последние два шага фиксированное число раз, либо до тех пор пока центроиды не "сойдутся" (обычно это значит, что их смещение относительно предыдущего положения не превышает какого-то заранее заданного небольшого значения).

[Нечеткие методы](http://bourabai.kz/tpoi/fuzzy.htm) - это методы, основанные не на бинарной логике - где все четко - элемент либо принадлежит одному кластеру, либо другому - а на предположении, что каждый элемент в какой-то степени принадлежит определенному кластеру. m - мера нечеткости, она как раз определяет нечеткость алгоритма.

**Метод нечёткой кластеризации C-средних** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *fuzzy clustering, soft k-means, c-means*) позволяет разбить имеющееся множество элементов [мощностью](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D1%89%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B0) N на заданное число нечётких множеств k. Метод нечеткой кластеризации *C*-средних можно рассматривать как усовершенствованный [метод *k*-средних](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_k-%D1%81%D1%80%D0%B5%D0%B4%D0%BD%D0%B8%D1%85), при котором для каждого элемента из рассматриваемого множества рассчитывается степень его принадлежности ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *responsibility*) каждому из кластеров.

## Алгоритм DBSCAN

DBSCAN (Density-based spatial clustering of applications with noise, плотностной алгоритм пространственной кластеризации с присутствием шума), как следует из названия, оперирует плотностью данных

## Уменьшение размерности; синтез признаков; выбор признаков; алгоритмы фильтрации

Под **уменьшением размерности** (англ. *dimensionality reduction*) 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 (англ. *feature selection*) или выделения признаков (англ. *feature extraction*).

**Синтез признаков?**

Методы **выбора признаков** оставляют некоторое подмножество исходного набора признаков, избавляясь от признаков избыточных и слабо информативных. Основные преимущества этого класса алгоритмов:

* Уменьшение вероятности [переобучения](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9F%D0%B5%D1%80%D0%B5%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5);
* 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.

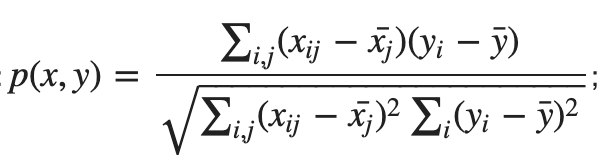
**Фильтры** (англ. *filter methods*) измеряют релевантность признаков на основе функции μ, и затем решают по правилу K, какие признаки оставить в результирующем множестве.

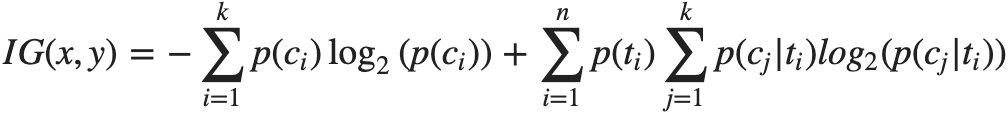
Фильтры могут быть:

* 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.

## Алгоритмы-обертки; встроенные методы выбора признаков

**Оберточные методы** (англ. *wrapper methods*) 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. Оберточные методы учитывают зависимости между признаками, что является преимуществом по сравнению с фильтрами, к тому же показывают большую точность, но вычисления занимают длительное время, и повышается риск [переобучения](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9F%D0%B5%D1%80%D0%B5%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5).

Существует несколько типов оберточных методов: детерминированные, которые изменяют генетимножество признаков по определенному правилу, а также рандомизированные, которые используют генетические алгоритмы для выбора искомого подмножества признаков. 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.

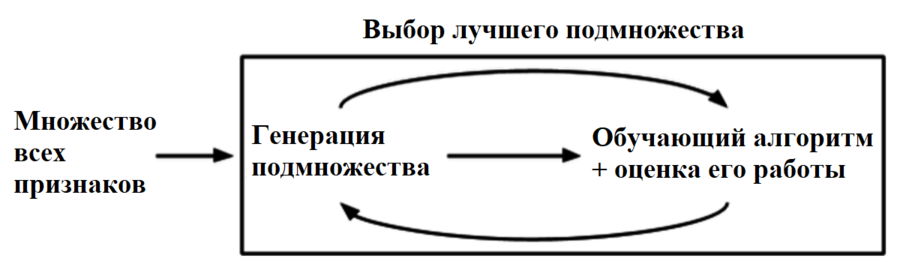
Популярным оберточным методом является SVM-RFE (SVM-based Recursive Feature Elimination), который иногда также обозначается как встроенный. Этот метод использует как классификатор [SVM](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BE%D0%BF%D0%BE%D1%80%D0%BD%D1%8B%D1%85_%D0%B2%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%BE%D0%B2_(SVM)) 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.



Группа **встроенных методов** (англ. *embedded methods*) очень похожа 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.

Одним из примеров встроенного метода является реализация на [случайном лесе](https://neerc.ifmo.ru/wiki/index.php?title=%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9_%D0%B8_%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D0%B9_%D0%BB%D0%B5%D1%81): каждому дереву на вход подаются случайное подмножество данных из датасета с каким-то случайным 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](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9F%D0%B5%D1%80%D0%B5%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), 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.



## Алгоритм PCA

Метод главных компонент (Principal Component Analysis) — один из основных способов уменьшить [размерность](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D0%BC%D0%B5%D1%80%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D1%81%D1%82%D0%B2%D0%B0) данных, потеряв наименьшее количество [информации](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D1%8F).

## Алгоритм t-SNE

**Стохастическое вложение соседей с** [**t-распределением**](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%A1%D1%82%D1%8C%D1%8E%D0%B4%D0%B5%D0%BD%D1%82%D0%B0) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *t-distributed Stochastic Neighbor Embedding*, t-SNE) — это алгоритм [машинного обучения](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) для [визуализации](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Он является техникой [нелинейного снижения размерности](https://ru.wikipedia.org/w/index.php?title=%D0%9D%D0%B5%D0%BB%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%BE%D0%B5_%D1%81%D0%BD%D0%B8%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%80%D0%BD%D0%BE%D1%81%D1%82%D0%B8&action=edit&redlink=1), хорошо подходящей для вложения данных высокой размерности для визуализации в пространство низкой размерности (двух- или трехмерное). В частности, метод моделирует каждый объект высокой размерности двух- или трёхмерной точкой таким образом, что похожие объекты моделируются близко расположенными точками, а непохожие точки моделируются с большой вероятностью точками, далеко друг от друга отстоящими.

Алгоритм t-SNE в упрощенном виде можно представить следующим псевдокодом:

Data: набор данных X = {x1, x2, …, xn},

параметр функции потерь: перплексия Perp,

Параметры оптимизации: количество итераций T, скорость обучения η, момент α(t).

Result: представление данных Y(T) = {y1, y2, …, yn} (в 2D или 3D).

begin

вычислить попарное сходство pj|i c перплексией Perp (используя формулу 1)

установить pij = (pj|i + pi|j)/2n

инициализировать Y(0) = {y1, y2, …, yn} точками нормального распределения (mean=0, sd=1e-4)

for t = 1 to T do

вычислить сходство точек в пространстве отображения qij (по формуле 4)

вычислить градиент δCost/δy (по формуле 5)

установить Y(t) = Y(t-1) + ηδCost/δy + α(t)(Y(t-1) - Y(t-2))

end

end

## 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.

Что такое модель? Что такое параметр и гиперпараметр модели,rabbits stockings в чём их отличие?