Список вопросов к экзамену (Бахшиев)

1. Проблемы формулировки понятия обучения. Формальная постановка задачи машинного обучения. Виды обучения.

Обучение – это накопление новой информации о предметной области. В процессе обучения выводится общее правило на основе частных.

Если отталкиваться от запоминания общих правил и переходе к частным, то есть рассматривать **дедуктивное обучение**, то в процессе формулирования общих правил возникает проблема перехода от сенсорно регулируемой информации об объекте к словесному описанию объекта.

В свою очередь, машинное обучение является **индуктивным обучением**, согласно которому происходит поиск общего решения по частным примерам. Машинное обучение позволяет по частным данным выявить общие зависимости, закономерности, взаимосвязи, присущие не только конкретной выборке, но вообще всем прецедентам, в том числе тем, которые ещё не наблюдались.

Выделяют несколько видов машинного обучения:

* инкрементное;
* неинкрементное;
* метаобучение;
* частичное обучение. Обучение с привлечением учителя.
* обучение с учителем (выборка из объектов, описываемая набором признаков + целевой признак, то, что хотелось бы прогнозировать);
* обучение без учителя (выборка из объектов, описываемая набором признаков);
* обучение с подкреплением.

Выделяют несколько видов задач машинного обучения:

* классификация(обучение с учителем) – отнесение объекта к одной из категорий на основании его признаков;
* регрессия(обучение с учителем) – прогнозирование количественного признака объекта на основании прочих его признаков;
* кластеризация(обучение без учителя) – разбиение множества объектов на группы на основании признаокв этих объектов так, чтобы внутри группы объекты были похожи между собой;

и тд.

Примеры применения:

Техническое зрение (распознавание, классификация объектов, кластеризация, сегментация);

Управление (идентификация объектов управления, синтез адаптивных регуляторов, стратегическое планирование).

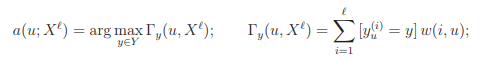
1. Метрические методы классификации. Постановка задачи. Метод К-ближайших соседей.  
    Во многих прикладных задачах измерять степень сходства объектов существенно проще, чем формировать признаковые описания. Метрические методы - это алгоритмы классификации, основанные на вычислении оценок сходства между объектами (анализ сходства объектов). Метрические классификаторы опираются на гипотезу компактности, которая гласит, что *схожие объекты чаще лежат в одном классе, чем в разных*. То есть классы образуют компактно локализованные области в пространстве объектов.

Пусть на множестве объектов X задана функция расстояния ρ: X×X → [0, ∞). Существует целевая зависимость y\*: X → Y, значения которой известны только на объектах обучающей выборки



Множество классов Y конечно. Требуется построить алгоритм классификации a: X → Y, аппроксимирующий целевую зависимость y\*(x) на всём множестве X.

Метрический алгоритм классификации с обучающей выборкой Xℓ относит объект u к тому классу y ∈ Y , для которого суммарный вес ближайших обучающих объектов Γy(u, Xℓ ) максимален:



где весовая функция w(i, u) оценивает степень важности i-го соседа для классификации объекта u. Функция Γy(u, Xℓ ) называется оценкой близости объекта u к классу y.

Выбирая весовую функцию w(i, u), можно получать различные метрические классификаторы:

Метод К-ближайших соседей:

В каждом классе выбирается k ближайших к u объектов, и объект u относится к тому классу, для которого среднее расстояние до k ближайших соседей минимально.

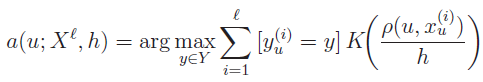
Недостатки:

* необходимо хранить всю выборку целиком;
* не учитываются значения расстояний;
* много совершаемых операций;
* бедный набор параметров, что исключает возможность настройки алгоритма по данным.

1. Метод парзеновского окна.

Один из видов метрических классификаторов. Если в случае метода ближайших соседей весовая функция w(i, u) несла роль задания веса в зависимости от ранга соседа i, то в методе парзеновского окна она является функцией от расстояния.

Введем функцию ядра K(z). Получим алгоритм:



Параметр h называется шириной окна и играет примерно ту же роль, что и число соседей k. Окно-это сферическая окрестность объекта u радиуса h, при попадании в которую обучающий объект xi голосует за отнесение объекта u к классу yi.

Фиксация ширины окна h не подходит для тех задач, в которых обучающие объекты существенно неравномерно распределены по пространству X. В окрестности одних объектов может оказываться очень много соседей, а в окрестности других - ни одного. В этих случаях применяется окно переменной ширины.

1. Отбор эталонов.

Пусть дана [обучающая выборка](http://www.machinelearning.ru/wiki/index.php?title=%D0%9E%D0%B1%D1%83%D1%87%D0%B0%D1%8E%D1%89%D0%B0%D1%8F_%D0%B2%D1%8B%D0%B1%D0%BE%D1%80%D0%BA%D0%B0) X^l=(x_i, y_i)_{i=1}^l, где x_i — объекты, y_i=y^*(x_i) — классы, которым принадлежат эти объекты. Кроме того, задана [метрика](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D1%80%D0%B8%D0%BA%D0%B0) \rho \: X \times X \rightarrow \mathbb{R}, такая, что выполняется [гипотеза компактности](http://www.machinelearning.ru/wiki/index.php?title=%D0%93%D0%B8%D0%BF%D0%BE%D1%82%D0%B5%D0%B7%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BA%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B8). При классификации объектов [метрическим классификатором](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80) a(u, X^l) = \mathrm{arg}\max_{y\in Y} \Gamma_y (u, X^l) = \mathrm{arg}\max_{y\in Y} \sum_{i=1}^l[y_u^(i)=y]w(i, u), например, [методом ближайших соседей](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B1%D0%BB%D0%B8%D0%B6%D0%B0%D0%B9%D1%88%D0%B8%D1%85_%D1%81%D0%BE%D1%81%D0%B5%D0%B4%D0%B5%D0%B9) необходимо вычислять расстояния от классифицируемого объекта до всех объектов обучающей выборки. Время, затрачиваемое на это для каждого классифицируемого объекта, пропорционально размеру обучающей выборки. Кроме того, оказывается необходимым хранить большой объем данных.

Но не все объекты обучающей выборки равноценны. Среди них есть наиболее типичные представители классов, то есть эталоны; неинформативные объекты, при удалении которых из обучающей выборки качество классификации не изменится; выбросы, или шумовые объекты — объекты, находящиеся в гуще «чужого» класса, только ухудшающие качество классификации.

Поэтому необходимо уменьшить объем обучающей выборки, оставив в ней только эталонные объекты для каждого класса.

Эталоны — это такое подмножество выборки X^l, что все объекты X^l (или их большая часть) классифицируются правильно при использовании в качестве обучающей выборки множества эталонов.

Эталонами i-го класса при классификации [методом ближайшего соседа](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B1%D0%BB%D0%B8%D0%B6%D0%B0%D0%B9%D1%88%D0%B5%D0%B3%D0%BE_%D1%81%D0%BE%D1%81%D0%B5%D0%B4%D0%B0) может служить такое подмножество объектов этого класса, что [расстояние](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D1%80%D0%B8%D0%BA%D0%B0) от любого принадлежащего ему объекта из выборки X^l до ближайшего «своего» эталона меньше, чем до ближайшего «чужого» эталона.

Для отбора эталонных объектов для метрического классификатора используется алгоритм STOLP, который позволяет сократить простой перебор.

Вводится понятие величины риска (W), характеризующей степень риска для объекта быть классифицированным не в тот класс, которому он принадлежит.

При использовании [метода ближайшего соседа](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B1%D0%BB%D0%B8%D0%B6%D0%B0%D0%B9%D1%88%D0%B5%D0%B3%D0%BE_%D1%81%D0%BE%D1%81%D0%B5%D0%B4%D0%B0) можно считать W(x_i)=\rho_{in}(x_i)/\rho_{out}(x_i), где \rho_{in} — расстояние от объекта x_i до ближайшего к нему объекта (или эталона) из «своего» класса, \rho_{out} — до ближайшего объекта (или эталона) «чужого» класса.

При использовании любого [метрического метода](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80) можно положить W(x_i)=-M(x_i, \Omega), где M(x_i, \Omega)=\Gamma_{y_i}-\max_{y \in Y \setminus y_i} \Gamma_y (x_i) — [отступ](http://www.machinelearning.ru/wiki/index.php?title=%D0%9E%D1%82%D1%81%D1%82%D1%83%D0%BF&action=edit) на объекте x_i при обучающей выборке \Omega, где \Omega — множество эталонов.

Кроме того, в зависимости от используемого метода классификации можно подобрать и другие оценки величины риска. Главное, чтобы они принимали большие значения на объектах-выбросах, меньшие — на объектах, находящихся на границе класса, и еще меньшие — на объектах, находящихся в глубине своего класса.

Входными данными алгоритма STOLP являются:

* Выборка X^l;
* Допустимая доля ошибок l_0;
* Порог отсечения выбросов δ;
* Алгоритм классификации;
* Формула для вычисления величины риска W.

Алгоритм:

* Отбросить выбросы (объекты X^l с W>δ);
* Сформировать начальное приближение \Omega — из объектов выборки X^l выбрать по одному объекту каждого класса, обладающему среди объектов данного класса максимальной величиной риска[[1]](http://www.machinelearning.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A1%D0%A2%D0%9E%D0%9B%D0%9F#cite_note-0) либо минимальной величиной риска;
* Наращивание множества эталонов (пока число объектов выборки X^l, распознаваемых неправильно, не станет меньше l_0):
* Классифицировать объекты X^l, используя в качестве обучающей выборки \Omega;
* Пересчитать величины риска для всех объектов X^l \setminus \Omega с учетом изменения обучающей выборки;
* Среди объектов каждого класса, распознанных неправильно, выбрать объекты с максимальной величиной риска и добавить их к \Omega.

Результатом работы алгоритма является множество эталонов  \Omega \in X^l в виде набора объектов, находящихся на границе класса и один объект, находящийся в центре класса. В итоге происходит разбиение всего множества объектов на эталонные, шумовые и неинформативные.

Преимущества: сокращается число хранимых объектов, сокращается время классификации, объекты распределяются по величине отступов.

Недостатки: необходимость задавать параметр δ, относительно низкая эффективность.

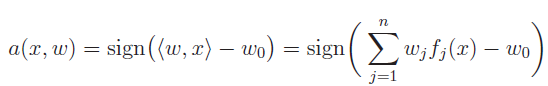
1. Линейные методы классификации. Постановка задачи. Метод стохастического градиента.

В линейных методах классификации решение принимается на основании линейного оператора над входными данными. Линейный классификатор или персептрон сравнивают с простейшей математической моделью нервной клетки – нейроном. Нервная клетка может находиться в двух состояниях: возбужденном и невозбужденном. На основании работы нейрона формируют постановку задачи:

Пусть ­­X - вектор входных данных, Y = {-1, 1} – множество допустимых ответов.

В простом смысле функция является пороговой, отделяющая один класс от другого. В более сложных случаях функция имеет смысл вероятности того или иного решения.

Форма линейного классификатора имеет вид:



Уравнение [w, x] = 0 задаёт гиперплоскость, разделяющую классы в пространстве. Если вектор x находится по одну сторону гиперплоскости с её направляющим вектором w, то объект x относится к классу +1, иначе - к классу −1. Параметр w0 иногда опускают.

Операцию линейной классификации для двух классов можно себе представить как отображение объектов в многомерном пространстве на гиперплоскость, в которой те объекты, которые попали по одну сторону разделяющей линии, относятся к первому классу ("да"), а объекты по другую сторону - ко второму классу ("нет")).

|  |  |
| --- | --- |
|  | https://upload.wikimedia.org/wikipedia/commons/thumb/2/20/Svm_separating_hyperplanes.png/1024px-Svm_separating_hyperplanes.png |

Метод стохастического градиента:

В этом методе градиентный подход применяется для подбора вектора синаптических весов w в линейном классификаторе.

Пусть y^*: \: X \to Y - целевая зависимость, известная только на объектах обучающей выборки: X^l \, = \, (x_i,y_i)_{i=1}^l, \; y_i \, = \, y^*(x_i).

Найдём алгоритм a(x, w), аппроксимирующий зависимость y^*. В случае линейного классификатора искомый алгоритм имеет вид:

a(x, w) = \varphi(\sum_{j=1}^n w_j x^j \, - \, w_0),

где \varphi(z) играет роль функции активации (в простейшем случае можно положить \varphi(z) \, = \, sign(z)).

Согласно принципу минимизации эмпирического риска для этого достаточно решить оптимизационную задачу: Q(w) \, = \, \sum_{i=1}^l L(a(x_i, w), \, y_i) \to \min_w, где L(a,y) - заданная функция потерь.

Для минимизации применим метод градиентного спуска (gradient descent). Это пошаговый алгоритм, на каждой итерации которого вектор w изменяется в направлении наибольшего убывания функционала Q (то есть в направлении антиградиента):

w \, {:=} \, w \, - \, \eta \nabla Q(w),

где \eta - положительный параметр, называемый темпом обучения (learning rate).

В случае реализации стохастического градиентного спуска на каждой итерации алгоритма из обучающей выборки случайным образом выбирается только один объект. Вектор w настраивается на каждый вновь выбираемый объект.

Входными данным являются:

* X^l - обучающая выборка
* \eta - темп обучения
* \lambda - параметр сглаживания функционала Q

Выходными данными является вектор весов w.

Алгоритм:

1. Инициализировать веса w_j \; j = 0, \dots, n;
2. Инициализировать текущую оценку функционала: Q \, {:=} \, \sum_{i=1}^l L(a(x_i, w), \, y_i)
3. Повторять:
   1. Выбрать объект x_i из X^l (например, случайным образом);
   2. Вычислить выходное значение алгоритма a(x_i, w) и ошибку: \varepsilon_i \, {:=} \, L(a(x_i, w), \, y_i)
   3. Сделать шаг градиентного спуска: w \, {:=} \, w \, - \, \eta L_a^\prime (a(x_i, w), \, y_i) \varphi^\prime (<w, x_i>)x_i
   4. Оценить значение функционала: Q \, {:=} \, (1 \, - \, \lambda)Q \, + \, \lambda\varepsilon_i
4. Пока значение Q не стабилизируется и/или веса w не перестанут изменяться.

Преимущества: Метод приспособлен для динамического (online) обучения, когда обучающие объекты поступают потоком, и надо быстро обновлять вектор w. Алгоритм способен обучаться на избыточно больших выборках за счёт того, что случайной подвыборки может хватить для обучения.

Недостатки: может не сходиться или сходиться слишком медленно.

1. Метод опорных векторов.

Метод относится к обучению с учителем, то есть имеются заранее образцы каждого класса, по которым известно заранее к какому классу они принадлежат. (Если классы не заданы изначально, то перед нами задача кластеризации). Используется для задач классификации и регрессии. Принадлежит к семейству линейных классификаторов. Задача классификации состоит в определении к какому классу из, как минимум, двух изначально известных относится данный объект. Обычно таким объектом является вектор в n-мерном вещественном пространстве https://habrastorage.org/storage/habraeffect/ec/9e/ec9e94384dbf5d14049bc5fa7e7d0aeb.png. Координаты вектора описывают отдельные аттрибуты объекта.

Часто в алгоритмах машинного обучения возникает необходимость классифицировать данные. Каждый объект данных представляется как вектор (точка) в  p-мерном пространстве (упорядоченный набор  p чисел). Каждая из этих точек принадлежит только одному из двух классов. Вопрос состоит в том, можно ли разделить точки гиперплоскостью размерности (p−1).

Метод относится изначально к бинарным классификаторам.

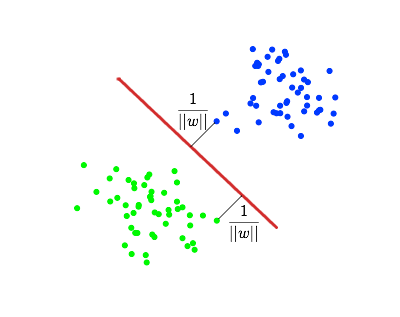
Идею метода удобно проиллюстрировать на следующем простом примере: даны точки на плоскости, разбитые на два класса (рис. 1). Проведем линию, разделяющую эти два класса (красная линия на рис. 1). Далее, все новые точки (не из обучающей выборки) автоматически классифицируются следующим образом: точка выше прямой попадает в класс A, точка ниже прямо— в класс B.

|  |  |
| --- | --- |
| https://habrastorage.org/storage/habraeffect/8c/98/8c98d4824065028420f290d88e52b40e.png | https://habrastorage.org/storage/habraeffect/7c/5f/7c5f4284e204a7c4b544a9ca175a2b13.png |

Такую прямую назовем разделяющей прямой. Однако, в пространствах высоких размерностей прямая уже не будет разделять наши классы, так как понятие «ниже прямой» или «выше прямой» теряет всякий смысл. Поэтому вместо прямых необходимо рассматривать гиперплоскости — пространства, размерность которых на единицу меньше, чем размерность исходного пространства. В https://habrastorage.org/storage/habraeffect/3e/89/3e89720bd70e886270233185080bfd1a.png, например, гиперплоскость — это обычная двумерная плоскость.

С точки зрения точности классификации лучше всего выбрать прямую, расстояние от которой до каждого класса максимально. Другими словами, выберем ту прямую, которая разделяет классы наилучшим образом (красная прямая на рис.2). Такая прямая, а в общем случае — гиперплоскость, называется оптимальной разделяющей гиперплоскостью.  
Вектора, лежащие ближе всех к разделяющей гиперплоскости, называются опорными векторами(support vectors).

Пусть имеется обучающая выборка: https://habrastorage.org/storage/habraeffect/ec/0f/ec0fd2147020136102f199d847315336.png.Метод опорных векторов строит классифицирующую функцию F в виде https://habrastorage.org/storage/habraeffect/9e/39/9e39396ca18b921d9afbef6d92607ddb.png,где https://habrastorage.org/storage/habraeffect/bc/aa/bcaa9f6be3acdb74579d883fa63c21f3.png — скалярное произведение, w — нормальный вектор к разделяющей гиперплоскости, b — вспомогательный параметр. Те объекты, для которых F(x) = 1 попадают в один класс, а объекты с F(x) = -1 — в другой. Выбор именно такой функции неслучаен: любая гиперплоскость может быть задана в виде https://habrastorage.org/storage/habraeffect/bb/77/bb77637fa821ab46507330ed645ceb33.png для некоторых w и b.



Далее, мы хотим выбрать такие w и b, которые максимизируют расстояние до каждого класса. Можно подсчитать, что данное расстояние равно https://habrastorage.org/storage/habraeffect/31/86/3186bd99eb78eadf2af35359b431c58e.png.

1. Оптимальный байесовский классификатор.

Байесовский подход классификации является классическим в теории распознавания образов и опирается на теорему о том, что если плотности распределения классов известны, то алгоритм классификации, имеющий минимальную вероятность ошибок, можно выписать в явном виде. Классификатор основан на принципе максимума апостериорной информации. Для классифицируемого объекта вычисляются функции правдоподобия каждого из классов, по ним вычисляются апостериорные вероятности классов. Объект относится к тому классу, для которого апостериорная вероятность максимальна.

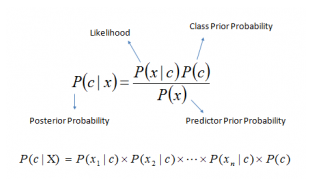
Рассмотрим множество учебных примеров (X,Y ), здесь X ∋ x = (ξ1,…,ξn) – объекты, Y – классы. Классификатор должен отображать объекты в классы с минимальной вероятностью ошибки

a(x ) : X → Y


1. Наивный байесовский классификатор.

Наивный байесовский алгоритм – это алгоритм классификации, основанный на [теореме Байеса](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B5%D0%BC%D0%B0_%D0%91%D0%B0%D0%B9%D0%B5%D1%81%D0%B0) с допущением о независимости признаков, то есть 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.

Теорема Байеса позволяет рассчитать апостериорную вероятность *P(c|x)* на основе *P(c)*, *P(x)* и *P(x|c)*.



P(c|x) – апостериорная вероятность данного класса c (т.е. данного значения целевой переменной) при данном значении признака x.

P(c) – априорная вероятность данного класса.

P(x|c) – правдоподобие, т.е. вероятность данного значения признака при данном классе.

P(x) – априорная вероятность данного значения признака.

С помощью теоремы Байеса рассчитывается апостериорную вероятность для каждого класса при данных погодных условиях. Класс с наибольшей апостериорной вероятностью будет результатом прогноза.

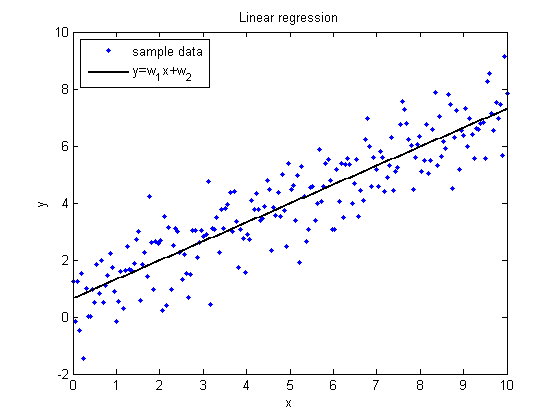
1. Постановка задачи регрессии. Методы восстановления линейной регрессии.

Вычисление ожидаемого значения y, учитывая конкретные значения х. Как задача квалификации сводится к определению объекта класса по его характеристикам, так задача регрессии позволяет определить по известным характеристикам значение его параметра. Только тут параметр имеет значение множества чисел, а не классов. На основании различных признаков предсказать вещественный ответ. Другими словами, ответом может быть 1, 5, 23.575 или любое другое [вещественное число](https://ru.wikipedia.org/wiki/%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), которое, например, может олицетворять стоимость квартиры. **Примеры**: предсказание стоимости акции через полгода, предсказание прибыли магазина в следующем месяце, предсказание качества вина на слепом тестировании.

Регрессия(обучение с учителем) – это зависимость математического ожидания случайной величины от одной или нескольких других случайных величин. Регрессионный анализ – это поиск функции, которая бы описала эту зависимость. Цель регрессионного анализа состоит в том, чтобы оценить значение непрерывной выходной переменной по значениям входных переменных.

Задана [выборка](http://www.machinelearning.ru/wiki/index.php?title=%D0%92%D1%8B%D0%B1%D0%BE%D1%80%D0%BA%D0%B0) — множество \{\mathbf{x}_1,...,\mathbf{x}_N|\mathbf{x}\in\mathbb{R}^M\} значений свободных переменных и множество \{y_1,...,y_N| y\in\mathbb{R}\}соответствующих им значений зависимой переменной. Эти множества обозначаются как D, множество исходных данных \{(\mathbf{x},y)_i\}. Задана [регрессионная модель](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%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C) — параметрическое семейство функций f(\mathbf{w},\mathbf{x}) зависящая от параметров \mathbf{w}\in\mathbb{R} и свободных переменных \mathbf{x}. Требуется найти наиболее вероятные параметры \bar{\mathbf{w}}:

\bar{\mathbf{w}}=\arg\max\limits_{\mathbf{w}\in\mathbb{R}^W}p(y|x,\mathbf{w},f)=p(D|\mathbf{w},f).

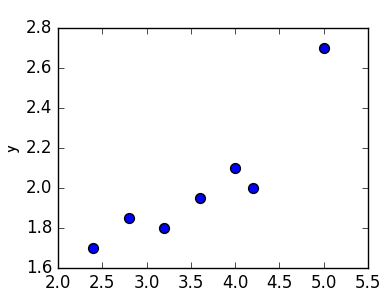


Регрессионный анализ используется для прогноза, выявления скрытых взаимосвязей в данных.

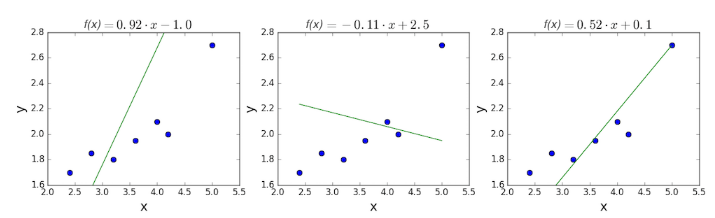
Линейная регрессия – это метод восстановления зависимости между двумя переменными. модель зависимости переменной *x* от одной или нескольких других переменных (факторов, регрессоров, независимых переменных) с линейной функцией зависимости.

Линейная регрессия предполагает, что функция f зависит от параметров \mathbf{w} линейно. При этом линейная зависимость от свободной переменной \mathbf{x} необязательна. Построение линейной регрессии заключается в расчете её коэффициентов w [методом наименьших квадратов](https://basegroup.ru/material/glossary/lsm/).

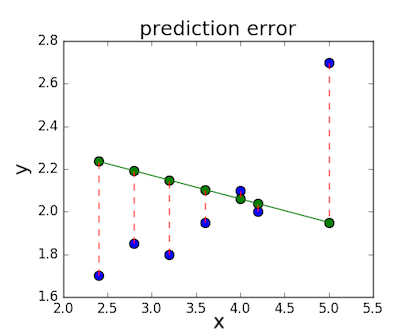
 y=f(\mathbf{w},\mathbf{x})+\nu=\sum_{j=1}^N w_jg_j(\mathbf{x})+\nu. 



Цель линейной регрессии — поиск линии, которая наилучшим образом соответствует этим точкам. Напомним, что общее уравнение для прямой есть f (x) = mx + b, где m — наклон линии, а b — его y-сдвиг. Таким образом, решение линейной регрессии определяет значения для m и b, так что f (x)приближается как можно ближе к y. Попробуем несколько случайных кандидатов:



Довольно очевидно, что первые две линии не соответствуют нашим данным. Третья, похоже, лучше, чем две другие. Но как мы можем это проверить? Формально нам нужно выразить, насколько хорошо подходит линия, и мы можем это сделать, определив функцию потерь, которые вычисляют расстояние между предсказанным значением *y(х)* и его фактическим значением. Например, взяв строку из среднего примера выше, f(x)=−0.11⋅x+2.5, мы выделяем дистанцию ошибки между фактическими и прогнозируемыми значениями  красными пунктирными линиями. Одна очень распространенная функция потерь называется **средней квадратичной ошибкой (MSE)**. Чтобы вычислить MSE, мы просто берем все значения ошибок, считаем их квадраты длин и усредняем.



1. Непараметрическая регрессия.

Непараметрическая регрессия использует модель, которая не описывается конечным числом параметров.

1. Логические методы классификации. Закономерности и критерии информативности.

Моделирование человеческой логики и принятия решений.

1. Решающие деревья.

Решение задачи обучения с учителем. Решающие деревья воспроизводят логические схемы, позволяющие получить окончательное решение о классификации объекта с помощью ответов на иерархически организованную систему вопросов. Причём вопрос, задаваемый на последующем иерархическом уровне, зависит от ответа, полученного на предыдущем уровне.

Дерево начинает строиться от корня к листьям. Каждой из вершин дерева за исключением листьев соответствует некоторый вопрос, подразумевающий несколько вариантов ответов, соответствующих выходящим рёбрам. В зависимости от выбранного варианта ответа осуществляется переход к вершине следующего уровня.  Иными словами в каждой вершине сравниваем значение некоторого признака с неким порогом, получаем ветвление. На каждом шаге выбирается тот признак, при разделении по которому прирост информации оказывается наибольшим. Концевым вершинам поставлены в соответствие метки, указывающие на отнесение распознаваемого объекта к одному из классов. Решающее дерево называется бинарным, если каждая внутренняя или корневая вершина инцидентна только двум выходящим рёбрам. Бинарные деревья удобно использовать в моделях машинного обучения.

Пусть в вершину m попала выборка Xm. Нужно разбить Xm -> Xl, Xr. Требуется метрика, которая бы характеризовала объекты в Xl, Xr. В идеале будет, когда в каждом подмножестве будут содержаться объекты только одного класса. В реальности же будет разброс ответов в каждом подмножестве, мерить который можно с помощью критерия информативности. Перебираем все признаки j и пороги t, оставляя только те j,t для которых метрика качества разбиения, чем меньше, тем лучше, для которых критерий информативности минимален.

**??? Когда останавливать построение дерева ???**

Можно ограничить дерево по глубине, можно разбивать до тех пор, пока в листе не останется n объектов.

**??? А как завершить классификацию, когда остановились ???**

Если это задача регрессии, то мы предсказываем среднее по объектам в листе. Если это задача классификации, то предсказываем самый популярный класс в листе.

**!!! Дерево решений не может содержать в себе циклические элементы !!!**

Несмотря на свою интерпретируемость и высокую выразительную способность, деревья крайне трудны для оптимизации из-за свой дискретной структуры — дерево нельзя продифференцировать по параметрам и найти с помощью градиентного спуска хотя бы локальный оптимум.

Достоинства:

1. Прости в понимании и интерпретации;
2. Не требует подготовки данных;
3. Работа с большим объемом информации.

Недостатки: Слишком громоздкие и сложные конструкции

1. Кластеризация данных. Постановка задачи. Агломеративная кластеризация и визуализация кластерной структуры.

Кластеризация – это обучение без учителя. С одной стороны это позволяет обрабатывать большие объемы данных, поскольку их не надо размечать руками для обучения, с другой – становится неясным измерение качества методов, так как отсутствуют метрики.

Это задача разбиения множества объектов на группы, называемые кластерами. Внутри каждой группы должны оказаться «похожие» объекты, а объекты разных группы должны быть как можно более отличны. Главное отличие кластеризации(обучение без учителя) от классификации(обучение с учителем) состоит в том, что перечень групп четко не задан и определяется в процессе работы алгоритма.

Пусть Х – множество объектов, Y – множество номеров кластеров. Задана функция расстояния между объектами p(x,x’). Имеется конечная обучающая выборка объектов Xm ={х1…xm}. Требуется разбить выборку на непересекающиеся подмножества так, чтобы каждый кластер состоял из объектов, близких по метрике p, а объекты разных кластеров существенно отличались. При этом каждому объекту приписывается номер кластера yi.

Алгоритм кластеризации – это функция а:X->Y, которая любому объекту х ставит в соответствие номер кластера y. Ставится задача определить оптимальное число кластеров.

Цель кластеризации:

* упростить дальнейшую обработку данных;
* сократить объем хранимых данных;
* выделить нетипичные объекты;
* построить иерархию множества объектов.

Агломеративная(объединительная, начинаем с индивидуальных, затем объединяем) кластеризация:

1. Считаем каждую точку кластером;
2. Сортируем попарные расстояния между центрами кластеров по возрастанию;
3. Берем пару ближайших кластеров, склеиваем их в один и пересчитываем центр кластера;
4. Повторяем до тех пор, пока все данные не склеятся в один кластер.
5. Получается дерево.

По итогам выполнения такого алгоритма можно также построить дерево склеивания кластеров и глядя на него определить, на каком этапе нам было бы оптимальнее всего остановить алгоритм. В результате получается древообразная иерархическая структура. Вместо номера кластера объект характеризуется перечислением всех кластеров, которым он принадлежит.

1. Методы частичного обучения.

Занимает промежуточное положение(с частичным привлечением учителя) между задачами классификации(с учителем) и кластеризации(без учителя). Есть какая-то размеченная выборка, на которую есть ответы учителя и есть неразмеченная, на которую ответов нет. Причем обе выборки должны содержать данные одних и тех же классов.

Требуется построить алгоритм классификации: отображение из множества объектов множество ответов, которые доразметят неразмеченную выборку.

Постановки задачи:

* Частичное обучение. Классификация каждого нового объекта а:Х-У. Необязательно который находится в неразмеченной выборке, а любой;
* Трансдуктивное обучение. Хотим получить метки только неразмеченных объектов.

Если у нас есть какое-то небольшое количество размеченных данных и намного большее количество неразмеченных, мы можем сначала обучить модель на размеченной выборке (обучение с учителем), а потом использовать эту обученную модель, чтобы предсказать с ее помощью класс (или значение) для неразмеченных данных. Потом — снова обучаем модель с учителем, используя предсказанные значения для неразмеченных данных, как «правильный ответ». Потом — снова предсказываем значения для неразмеченных данных уже с новой моделью, и так по кругу.

Недостаток:

Неразмеченная выборка должна содержать только те же самые классы, что и размеченная. Иными словами, если в размеченной выборке у нас фотографии мотоциклов и машин, и мы знаем, на какой картинке что, то в неразмеченной тоже должны быть только мотоциклы и машины, только мы не знаем, что из них где.

1. Понятие искусственных нейронных сетей. Классификация моделей нейрона. Формальный нейрон. Топологии искусственных нейронных сетей.

Позволяют реализовать глубокое обучение.

Чаще всего ИНС представляют собой систему соединённых и взаимодействующих между собой искусственных нейронов. Искусственный нейрон имитирует в первом приближении свойства биологического нейрона. На вход искусственного нейрона поступает некоторое множество сигналов, каждый из которых является выходом другого нейрона. Нейронные сети имитируют структуру головного мозга: каждый искусственный нейрон соединяется с несколькими другими нейронами. Нейросети имеют многослойную структуру: нейроны на одном слое передают данные нескольким нейронам на следующем и т. д. В конечном счете данные достигают выходного слоя, где сеть выдает предположение о том, как решить задачу, классифицировать объект и т. п.

Во-первых, нейроны можно классифицировать на основе положения нейронов в топологии сети:

1. входные нейроны: принимают исходный вектор, кодирующий входной сигнал. Как правило, не выполняют вычислительных операций, а просто передают полученный входной сигнал на выход, возможно, усилив или ослабив его;
2. выходные нейроны: представляют выходы сети. В выходных нейронах могут производиться какие-либо вычислительные операции;
3. промежуточные нейроны — выполняют основные вычислительные операции.

Во-вторых, классификацию нейронов можно производить по степени разработки математического описания модели нейрона и качественного описания поведения нейрона (Рис.1):



1. формальные (абстрактные) нейроны: отличаются хорошо разработанным математическим описанием, но игнорируют многие свойства своего биологического двойника. Однако вопрос того, какие не описываемые формальными нейронами свойства биологических прототипов могут позволить существенно расширить сферу применения ИНС, или сложность решаемых ими задач, в общем случае остается открытым;
2. физиологические нейроны: отличаются количественным описанием поведения нейронов, порождены из экспериментов над биологическими нейронами;
3. феноменологические модели нейронов: не имеют строгой математической и опытной базы.

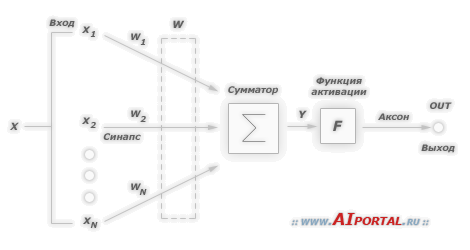
Формальные нейроны в свою очередь тоже могут быть разделены по нескольким признакам:

* 1. по виду функции активации;
  2. по изменяемости весов входов:
* неизменные веса (например, нейрон Макклока-Питтса);
* управляемые веса;
* веса, перестраивающиеся в зависимости от входного по данному каналу сигнала (например, «Instar» Гроссберга.);
* веса типа синапсов Хебба, изменяющиеся под влиянием выходного сигнала нейрона.
  1. в зависимости от вероятности значения выходного сигнала:
* детерминированные модели;
* стохастические модели.

В-третьих, нейроны можно подразделять в зависимости от их аппаратной реализации:

1. цифровые модели – нейроны выполняются на микросхемах средней и высокой степени интеграции; на кристалле реализуется непосредственно нейронная сеть, т. е. отдельные нейроны не выделяются как самостоятельные единицы, т.к. модель разрабатывают для решения конкретных задач;
2. аналоговые модели: позволяют воспроизводить пространственное и временное суммированием возбуждающих сигналов, свойства абсолютной и относительной рефрактерности, процессы переработки информации в отдельном нейроне и при взаимодействии нейронов. Нейрон рассматривается как устройство, состояние которого может непрерывно меняться от полного покоя до некоторого максимального уровня возбуждения.

Первая модель формального нейрона была разработана Маккалоком и Питтсом. Формальный нейрон представляет собой математическую модель простого процессора, имеющего несколько входов и один выход. Вектор входных сигналов, поступающих через дендриты, преобразуется нейроном в выходной сигнал, распространяющийся по аксону, с использованием трех функциональных блоков: локальной памяти, блока суммирования и блока нелинейного преобразования. Нейрон представляет собой единицу обработки информации в нейронной сети.

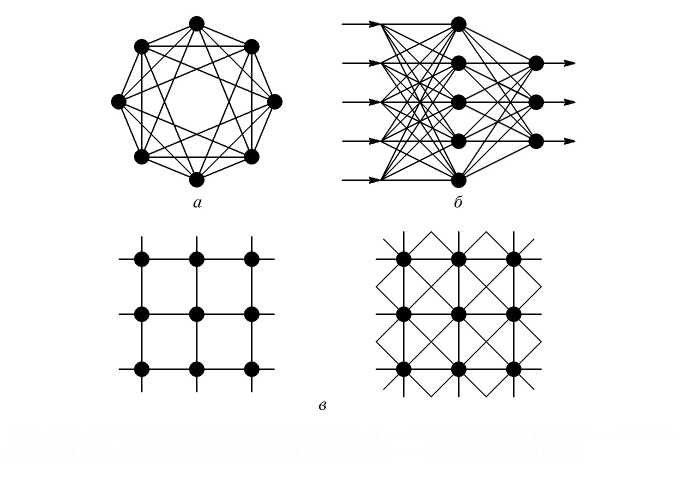
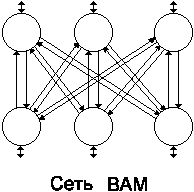


В этой модели нейрона можно выделить три основных элемента:

* синапсы, каждый из которых характеризуется своим весом или силой. Осуществляют связь между нейронами, умножают входной сигнал x_{i} на весовой коэффициент синапса w_{i}, характеризующий силу синаптической связи;
* сумматор, аналог тела клетки нейрона. Выполняет сложение внешних входных сигналов или сигналов, поступающих по синаптическим связям от других нейронов. Определяет уровень возбуждения нейрона;
* функция активации, определяет окончательный выходной уровень нейрона, с которым сигнал возбуждения (торможения) поступает на синапсы следующих нейронов.

Классифицируя [нейронные сети](https://neuronus.com/stat/1271-nejronnye-seti-iskusstvennyj-intellekt.html) по топологии, можно выделить три основных типа таких сетей:

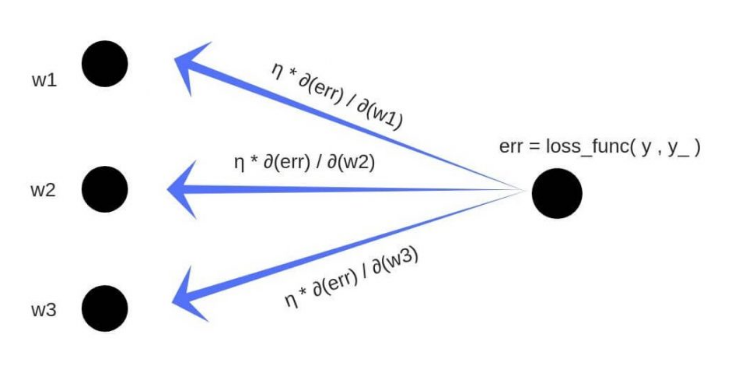
1. полносвязные сети ( рис. 1, а). Она состоит из одного слоя нейронов, которые выполняют функции входов и выходов одновременно. Сеть Хопфилда;
2. многослойные или слоистые сети (рис 1, б)
3. слабосвязные сети. Двунаправленная ассоциативная память (рис. 1, в)



1. Алгоритм обратного распространения ошибки.

Является одним из методов обучения многослойных нейронных сетей. Цели обратного распространения просты: отрегулировать каждый вес пропорционально тому, насколько он способствует общей ошибке. Если мы будем итеративно уменьшать ошибку каждого веса, в конце концов, у нас будет ряд весов, которые дают хорошие прогнозы.

Предполагается два прохода по всем слоям: прямого и обратного. При прямом проходе входной вектор подается на входной слой нейронной сети, после чего распространяется по сети от слоя к слою. В результате генерируется набор выходных сигналов, который и является фактической реакцией сети на данный входной образ. Во время прямого прохода все синаптические веса сети фиксированы. Во время обратного прохода все синаптические веса настраиваются в соответствии с правилом коррекции ошибок, а именно: фактический выход сети вычитается из желаемого, в результате чего формируется сигнал ошибки. Этот сигнал впоследствии распространяется по сети в направлении, обратном направлению синаптических связей.



Достоинства: простота реализации и устойчивость.

Недостатки: неопределенно долгий процесс обучения, уязвимость попадания в локальные минимумы функции ошибки.

1. Композиции классификаторов. AdaBoost

*Ансамблевые методы* — это парадигма машинного обучения, где несколько моделей (часто называемых «слабыми учениками») обучаются для решения одной и той же проблемы и объединяются для получения лучших результатов. Основная гипотеза состоит в том, что при правильном сочетании слабых моделей мы можем получить более точные и/или надежные модели.

Бустинг – это последовательный метод, при котором комбинированные слабые модели не обучаются независимо друг от друга. Наоборот, идея в том, чтобы итеративно обучать модели, чтобы обучение модели на данном этапе зависело от моделей, обученных на предыдущем этапе. Каждая модель в последовательности подбирается, что придает большее значение объектам в датасете, которые плохо обрабатывались предыдущими моделями в последовательности.

Но после того, как слабые ученики выбраны, необходимо определить, как они будут последовательно подгоняться и какую информацию из предыдущих моделей мы будем учитывать при подборе текущей модели, как их создавать и объединять в ходе последовательного процесса. Adaboost или адаптивный бустинг обновляет веса, прикрепленные к каждому из объектов обучающего датасета.

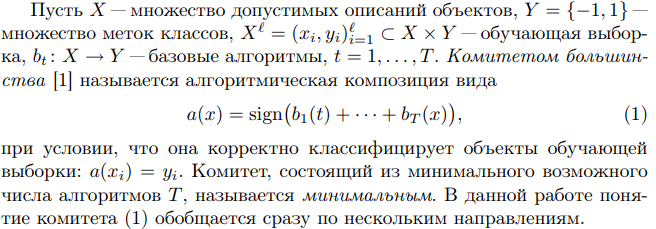
Адаптивный бустинг на каждой новой итерацииобновляет веса объектов в датасете и обучает нового слабого ученика, уделяя особое внимание наблюдениям, неправильно классифицированным текущей ансамблевой моделью. Веса хорошо классифицированных объектов уменьшаются относительно весов неправильно классифицированных объектов. Он добавляет слабого ученика к взвешенной сумме в соответствии с коэффициентом обновления, который выражает эффективность этой слабой модели: чем лучше слабый ученик выполнил свою работу, тем больше он будет учтен в сильном ученике.

Алгоритм:

* обучить наилучшую возможную слабую модель с текущими весами наблюдений;
* вычислить значение коэффициента обновления, который является своего рода скалярной метрикой оценки слабого ученика, которая показывает, насколько вклад этого слабого ученика должен быть учтен в ансамблевой модели;
* обновить сильного ученика, добавив нового слабого ученика, умноженного на его коэффициент обновления;
* вычислить новые веса объектов, которые показывают, на каких наблюдениях мы хотели бы сосредоточиться на следующей итерации (веса ошибочно прогнозируемых объектов увеличиваются в совокупной модели, а веса правильно предсказанных объектов — уменьшаются).

1. Композиции классификаторов. Комитетный бустинг.

В процессе накопления алгоритмов в композиции классификаторов следует избегать избыточности, то есть стремиться к тому, чтобы композиция состояла из минимального числа базовых алгоритмов. Метод комитетного бустинга позволяет достичь минимизации числа базовых алгоритмов при простом голосовании. Он удаляет наименее удачные алгоритмы из композици.



Рассматриваются базовые алгоритмы bt, которые могут выдавать как +-1, так и числовую оценку принадлежности объекта классу +1. Базовые алгоритмы могут быть как бинарными (возвращать ±1), так и вещественнозначными (возвращать числовую оценку). В отличие от бустинга, не требуется, чтобы метод обучения был чувствителен к весам объектов.

Для многих приложений гарантировать корректность не обязательно. Гораздо важнее, чтобы композиция обладала хорошей обобщающей способностью, т. е. допускала как можно меньше ошибок на независимых контрольных данных. Минимальность также не является жестким требованием; достаточно обеспечить значение, близкое к минимальному.

Вводится понятие степени граничности или отступа объекта x, величина

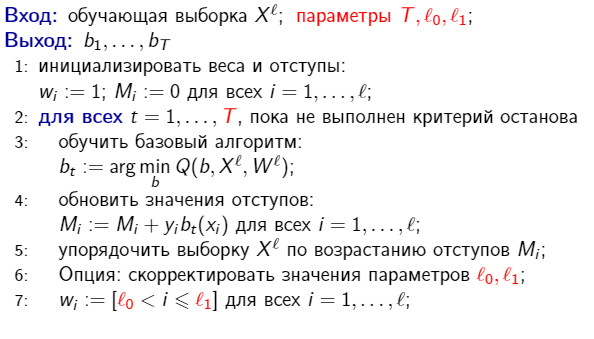


Отступ отрицателен тогда и только тогда, когда а(х) допускает ошибку на объекте xi.

Переходим к методу построения композиций, основанном на идентификации выбросов.

Алгоритм:

1. Представим, что необходимо построить новый алгоритм bt и уже часть базовых алгоритмов есть b1,..,bt-1. Упорядочим выборку Xl по возрастанию отступов mt-1(xi). Объекты с одинаковыми отступами будем располагать в случайном порядке.
2. Обучим алгоритм bt по подборке длины (l2-l1), с l1+1 по l2 объекты включительно. Параметр l1 ограничивает количество выбросов, параметр l2 позволяет найти компромисс между качеством и различностью базовых алгоритмов. Их значения могут либо фиксироваться, либо подбираться на каждом новом шаге по критерию минимума числа ошибок композиции на всей выборке Xl.
3. По мере увеличения числа алгоритмов в композиции повышается надёжность идентификации выбросов. При этом алгоритмы, которые были обучены ранее, оказываются неоптимальными, так как их обучающие выборки содержали выбросы. Поэтому, начиная с некоторого шага t0, перед обучением нового алгоритма «наиболее старый» удаляется.



Вопросы для подготовки к государственному экзамену (Семакова)

1. Матрицы поворота, композиция поворотов.
2. Кватернионы.
3. Прямая задача кинематики: параметры Денавита-Хартенберга.
4. Обратная задача кинематики: формулировка и подходы к решению.
5. Задача планирования пути: алгоритм A\*.
6. Поиск пути в пространстве конфигураций: алгоритм PRM.
7. Метод градиентного спуска.
8. Момент инерции плоских и трехмерных тел. Главные моменты инерции и главные оси.
9. Потенциальная и кинетическая энергия твердого тела.
10. Частотная область. Преобразование Лапласа. Обратное преобразование Лапласа.
11. Передаточная функция. Общий принцип синтеза регулятора.
12. Характеристики переходного процесса систем первого порядка.
13. Характеристики переходного процесса систем второго порядка.
14. Пропорционально-интегрально-дифференцирующий регулятор. Настройка параметров PID-регулятора.
15. Датчики квадрокоптера: акселерометр, гироскоп, УЗ-датчик, GPS.
16. Совместная и условная вероятности. Математическое ожидание и ковариация случайной величины.
17. Правило Байеса. Пример.
18. Предположения Маркова. Марковская цепь. Модель линейного процесса.
19. Алгоритм Байесовской фильтрации. Пример.
20. Алгоритм фильтрации Калмана.

Список вопросов к экзамену (Абросимов)

1. Определение интеллекта через понятие алгоритма. Искусственный интеллект.
2. Интеллектуальные информационные системы. Данные и знания.
3. Понятие о математической логике. Типы неклассических логик.
4. Логика высказываний, логика предикатов, продукционные системы.
5. Понятие о нечёткой логике, нечёткое моделирование.
6. Понятие нечёткого множества. Примеры записи нечёткого множества.
7. Основные характеристики нечёткого множества.
8. Типовые функции принадлежности.
9. Операции над нечёткими множествами: дополнение, равенство, пересечение и объединение (в том числе алгебраические и граничные). T-норма, S-норма.
10. Операции над нечёткими множествами: разность, умножение на число, возведение в степень, концентрация, растяжение, декартово прямое произведение.
11. Нечёткие отношения. Способы задания нечёткого отношения.
12. Свойства нечётких отношений.
13. Операции над нечёткими отношениями.
14. Лингвистическая переменная. Примеры.
15. Нечёткий логический вывод. Типовая структура нечёткого вывода.
16. Основные этапы нечёткого вывода.
17. Алгоритм нечёткого вывода Мамдани. Графический пример.
18. Алгоритм нечёткого вывода Сугено. Графический пример.
19. Гибридные нейро-нечёткие системы. Архитектура ANFIS.
20. Иерархические системы нечёткого вывода.