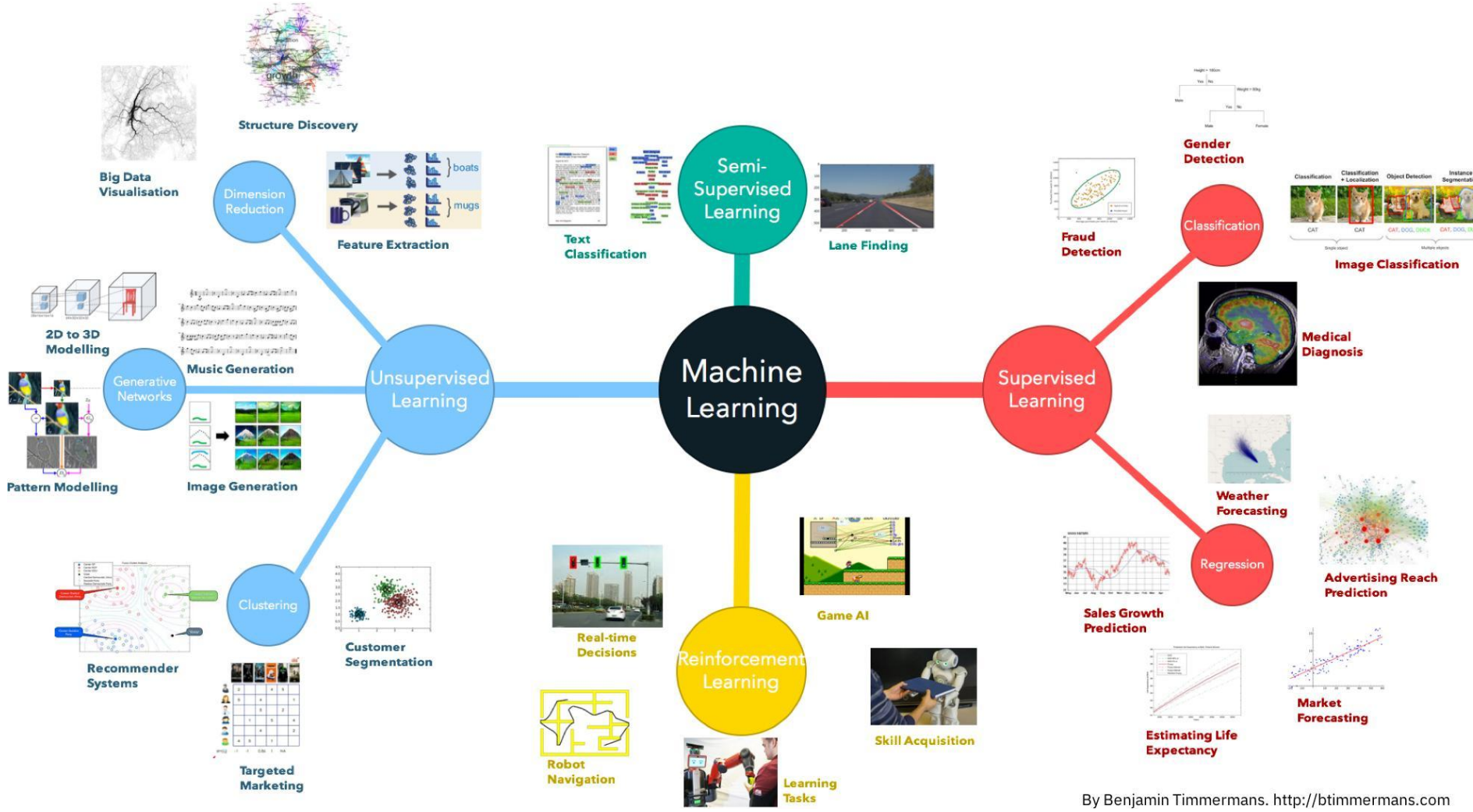
Что мы уже знаем

Методы машинного обучения



By Benjamin Timmermans. http://btimmermans.com

Что мы уже знаем

Этапы обучения

В *формальных логических алгоритмах* все зависимости выявляются и задаются «вручную».

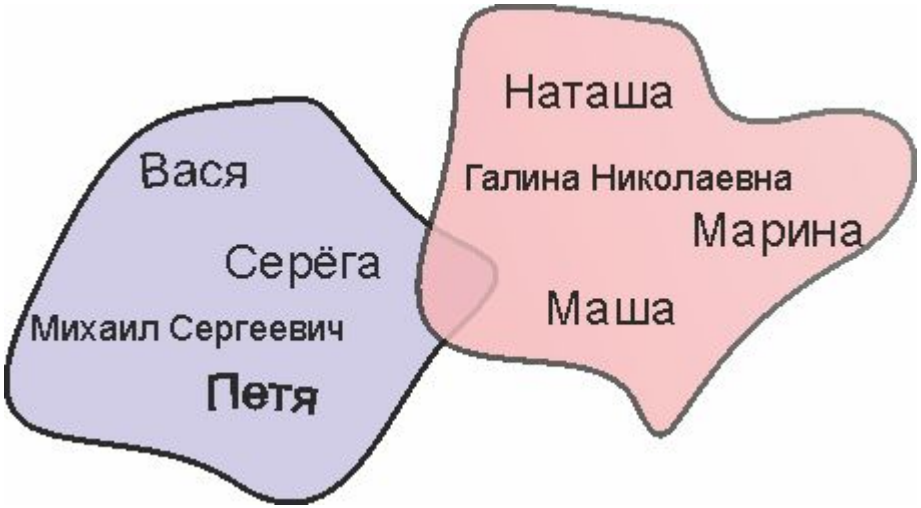
Классическое обучение требует наличия признаков, которые разрабатываются «вручную», но решающее правило строится автоматически в процессе обучения.

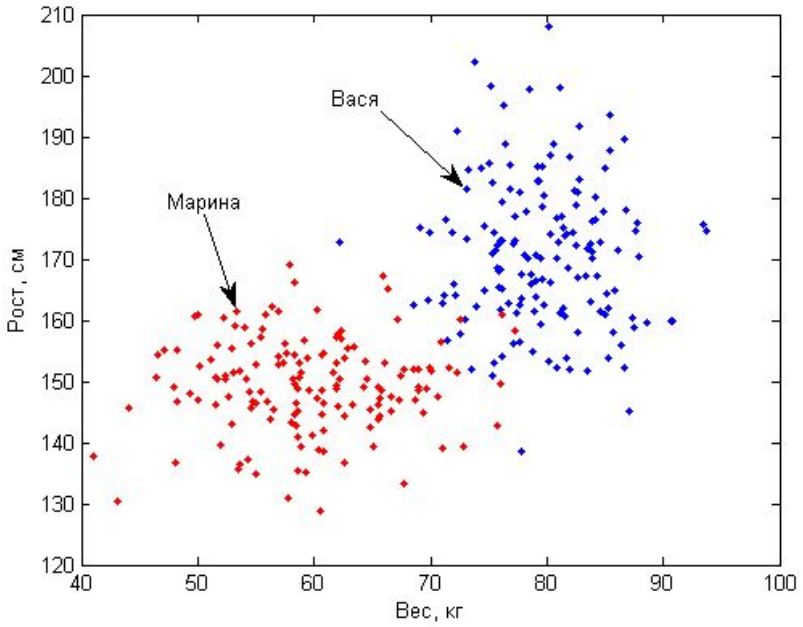
Глубокие модели предполагают автоматическую генерацию и признаков и решающего правила.

Основные понятия теории распознавания образов

Классификация (распознавания образов, объектов, сигналов, ситуаций, явлений или процессов) – задача идентификации объекта т.е. определения его принадлежности к группе однотипных, по его описанию т.е. набору характеристик или признаков.

Множество – набор неповторяющихся однотипных элементов.

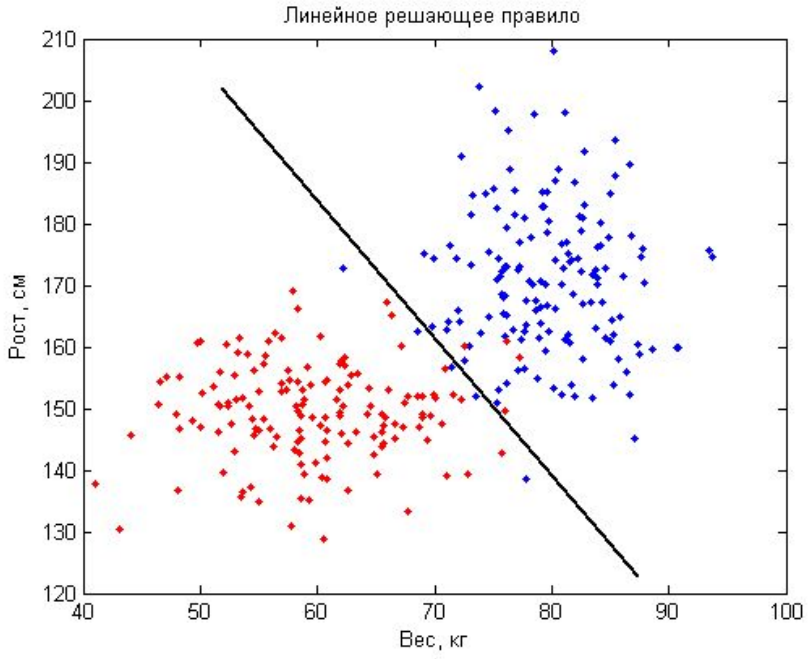


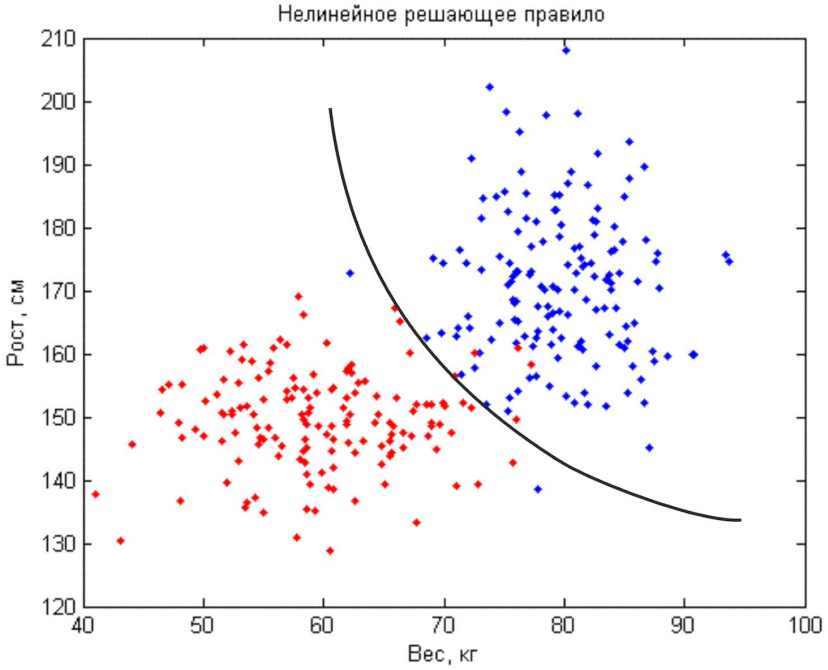


Признаковое пространство – набор характеристик в системе классификации, выделяющий определенную группу (класс) объектов от остальных или разделяющий все объекты на несколько групп (классов).

Прецедент – характеристики конкретного объекта или его набор признаков (признаковое описание).

Решающим правилом (классификатором) называется методика или правило отнесения объекта к какому-либо классу.





Обучение – определение по имеющимся прецедентам решающего правила.

Т.е. по имеющимся парам **«признаки»–«ответ»** необходимо восстановить зависимость **F(«признаки»)= «ответ»**.

Этапы решения задачи распознавания образов:

1 Выбор свойств, характеристик, признаков класса объектов, наилучшим образом отличающих их от объектов других классов.

Проблема: в реальных задачах «хорошие» признаки либо отсутствуют, либо их сложно измерять (вычислять), поэтому приходится пользоваться многими «не очень хорошими».

2 Формирование априорных наборов описаний объектов в выбранном признаковом пространстве.

Проблема: чем больше размерность признакового пространства, тем большее количество данных необходимо иметь для решения задачи. Возникает **«проклятие размерности»** - экспоненциальный рост размера обучающей выборки от размерности признакового пространства.

3 Выбор алгоритма, реализующего один из методов теории распознавания образов, позволяющего решать задачу различения объектов в рамках принятого их описания.

Проблема: не существует состоятельных рекомендаций по выбору конкретного метода распознавания из множества существующих. Разработчик вынужден полагаться на свой опыт и интуицию, либо реализовывать всё подряд, пока не найдётся приемлемое решение.

4 Выбор функционала качества (функции потерь) функционирования алгоритма. Определяется решаемой задачей и используемым алгоритмом.

Часто используется **эмпирический риск** - это средняя величина ошибки алгоритма на обучающих данных. Обучение проводится с целью минимизации эмпирического риска.

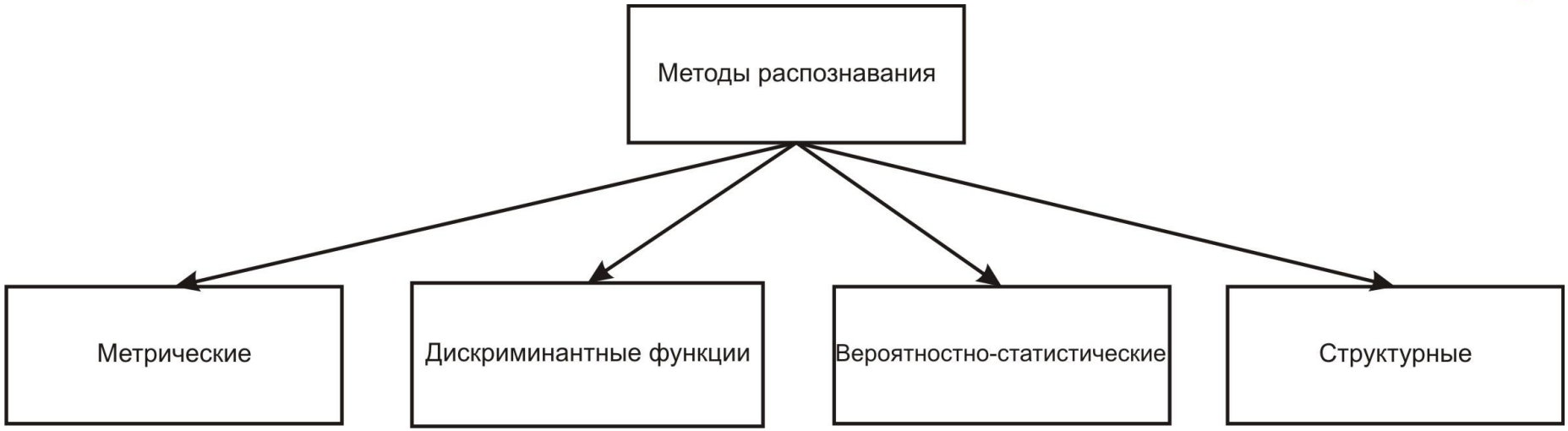
Проблема: возникает явление **переобучения** - когда количество ошибок алгоритма на тестовой выборке оказывается существенно выше, на обучающей.

Например, после простого запоминания всех обучающих объектов каждый новый сравнивается с ними. Правильный ответ выдаётся при полном соответсвии предъявляемого объекта одному из запомненных.

5 Настройка параметров алгоритма, с использованием априорного набора описаний объектов, оптимизирующих выбранный функционал качества. Этот этап часто называют обучением алгоритма.

Проблема: не все параметры алгоритмов являются автоматически настраиваемыми, некоторые из них необходимо задавать разработчику.

Виды алгоритмов распознавания образов



Общая постановка задачи распознавания образов:

Существует множество объектов $\boldsymbol{X}$ и множество имён классов объектов $\boldsymbol{Y}$.

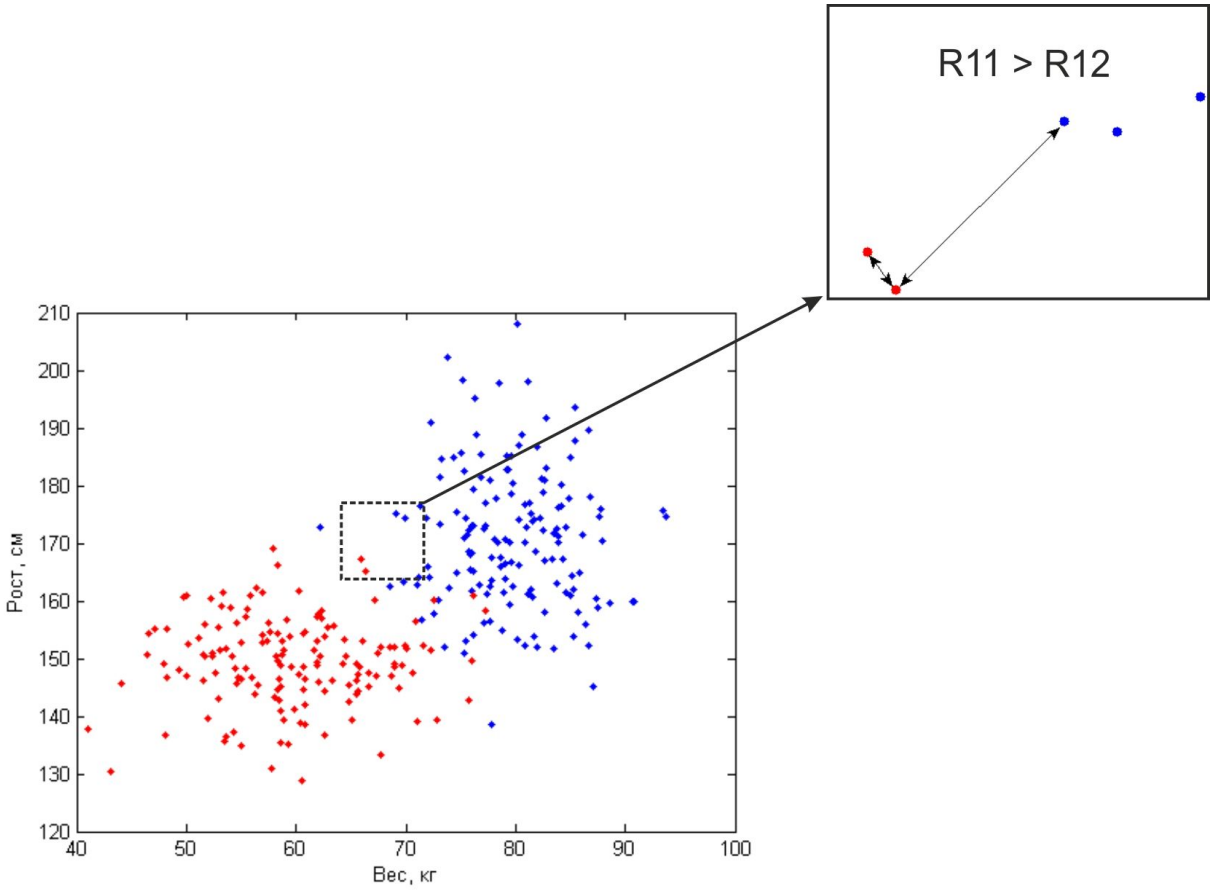
Существует неизвестная функция осуществляющая отображение пространства объектов в пространство имён классов:
$\boldsymbol{y^*: X \to Y}$.

Значения функции y^* известны лишь на объектах обучающей выборки
$\boldsymbol{X^N = (x_i, y_i)_{i=1...N}}$, $\boldsymbol{y_i = y^*(x_i)}$, где N – количество объектов обучающей выборки.

Необходимо построить алгоритм распознавания $\boldsymbol{a: X \to Y}$, аппроксимирующий зависимость $\boldsymbol{y^*(x)}$ на всём множестве $\boldsymbol{X}$.

Метрические методы распознавания образов

Гипотеза компактности – «близкие объекты, как правило, лежат в одном классе».



Формализованным понятием сходства является введённая в пространстве объектов X **функция расстояния** между объектами $\rho : X \times X \to (0,\infty]$.

Евклидово расстояние:

$$\rho(x_i, x_j) = \left(\sum_{k=1}^{M} (x_i^k - x_j^k)^2 \right)^{1/2}$$

$x_i = (x_i^1, \ldots x_i^M)$ - вектор признаков объекта x_i;

$x_j = (x_j^1, \ldots x_j^M)$ - вектор признаков объекта x_j.

Норма векторов:

$$\rho(x_i, x_j) = \left(\sum_{k=1}^{M} \left| x_i^k - x_j^k \right| \right)$$

Взвешенная норма:

$$\rho(x_i, x_j) = \left(\sum_{k=1}^{M} w_k \left| x_i^k - x_j^k \right|^p \right)^{1/p}$$

Расстояние Махаланобиса:

$$\rho(x_i, x_j) = \sqrt{(x_i - x_j)^T S^{-1} (x_i - x_j)}$$

S – ковариационная матрица.

Простейший метрический алгоритм

Для каждого объекта *x* можно вычислить расстояния до всех имеющихся в обучающей выборке объектов, и проранжировав все объекты по значению расстояния:

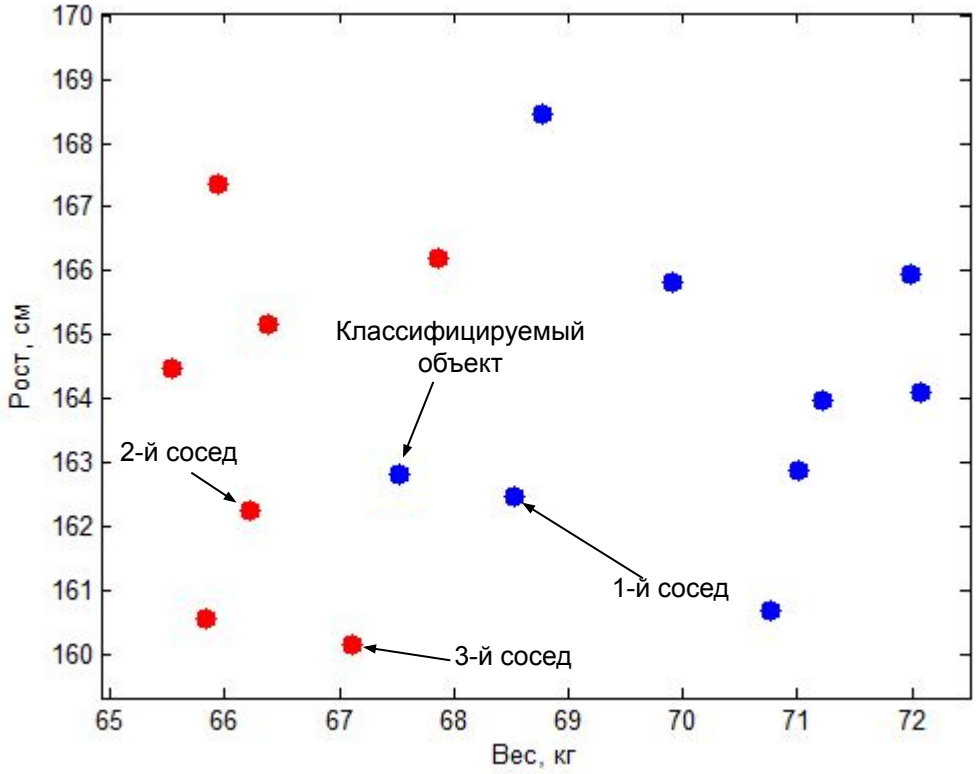
$$\rho(x, x_1) \le \rho(x, x_2) \le \ldots \le \rho(x, x_N)$$

Получим ряд «соседей» объекта *x:* $x_1, x_2, \ldots, x_N$

И ряд ответов *y* на каждом из «соседей»*:* $y_1, y_2, \ldots, y_N$

Будем относить классифицируемый объект к тому классу, к которому относится его ближайший сосед т.е.

$$y = y_1$$



Преимущества:

-прост в реализации;

-решение легко интерпретируется.

Недостатки:

-необходимость хранения всей обучающей выборки;

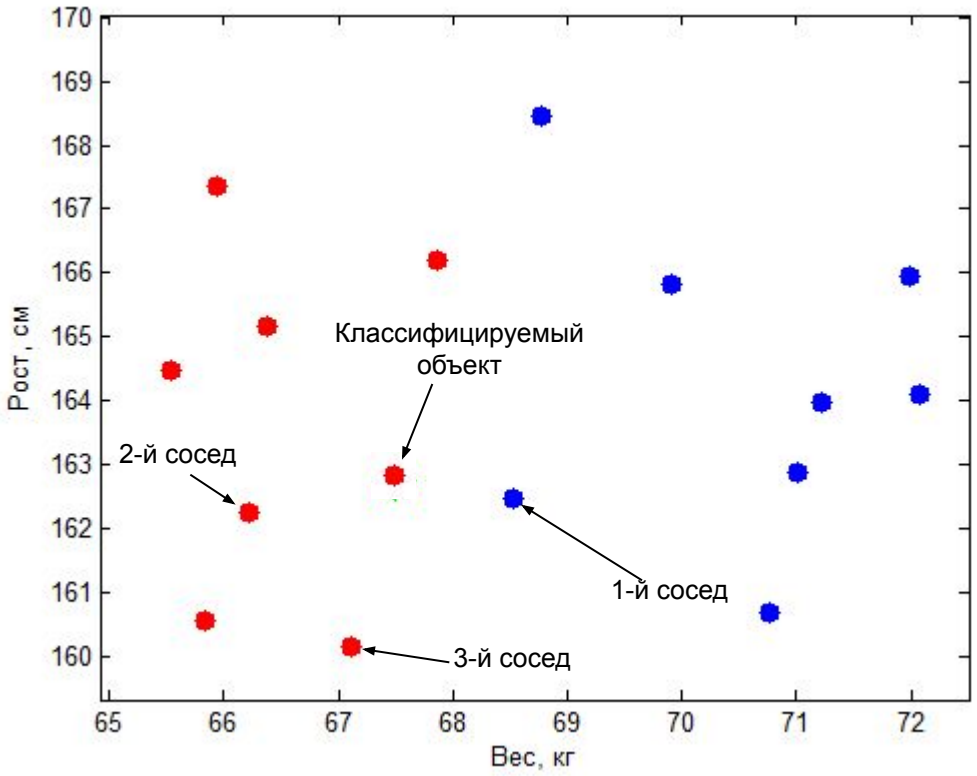
- чувствительность к влиянию «шумовых» данных, и как следствие низкое качество решений.

Алгоритм *k*-ближайших соседей

Алгоритм относит объект *x* к тому классу, к которому относится большее число среди *k* его ближайших соседей.

Это позволяет уменьшить влияние выбросов.

$$a(u; X^N, k) = \arg\max_{y \in Y} \sum_{i=1}^{k} [y_i = y]$$



Преимущества:

-прост в реализации;

-решение легко интерпретируется;

- менее чувствителен к выбросам по сравнению с предыдущим алгоритмом.

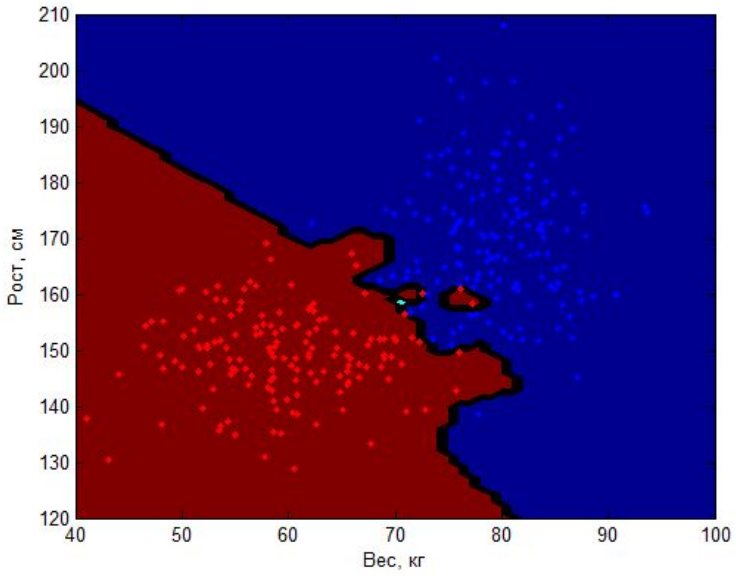
Недостатки:

-необходимость хранения всей обучающей выборки;

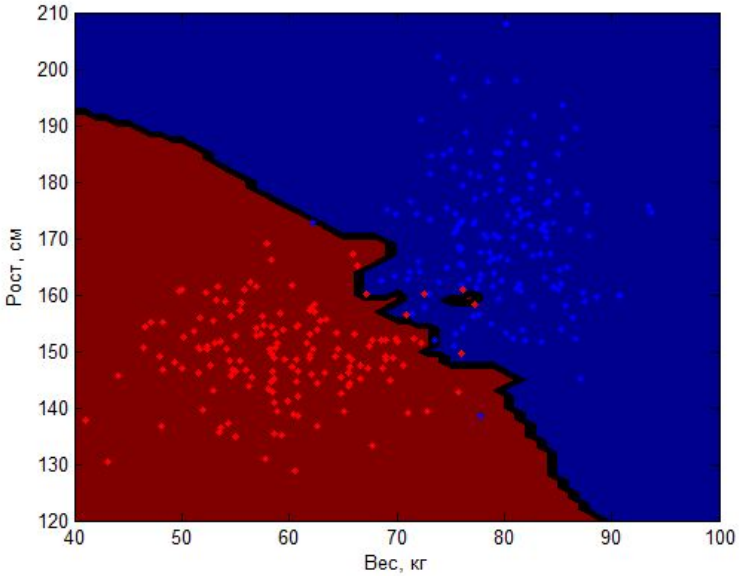
- может возникать неоднозначное решение, когда соседей из разных классов поровну.

Алгоритм *k*-ближайших соседей

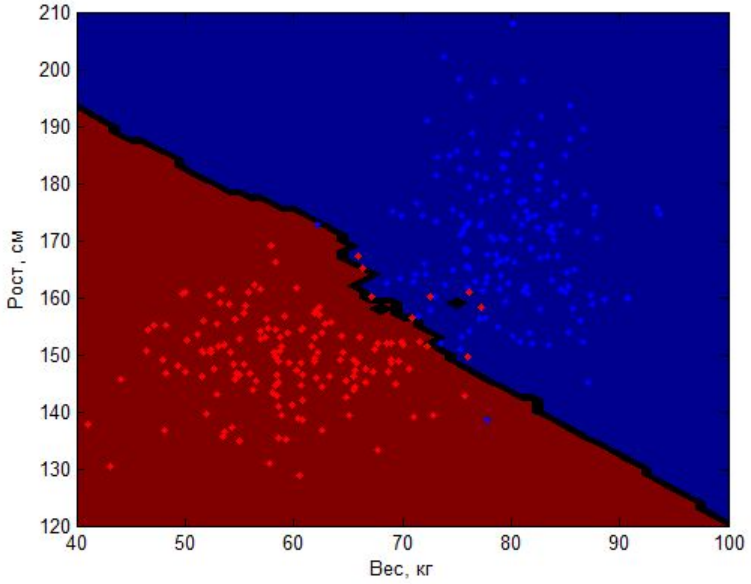
k=1



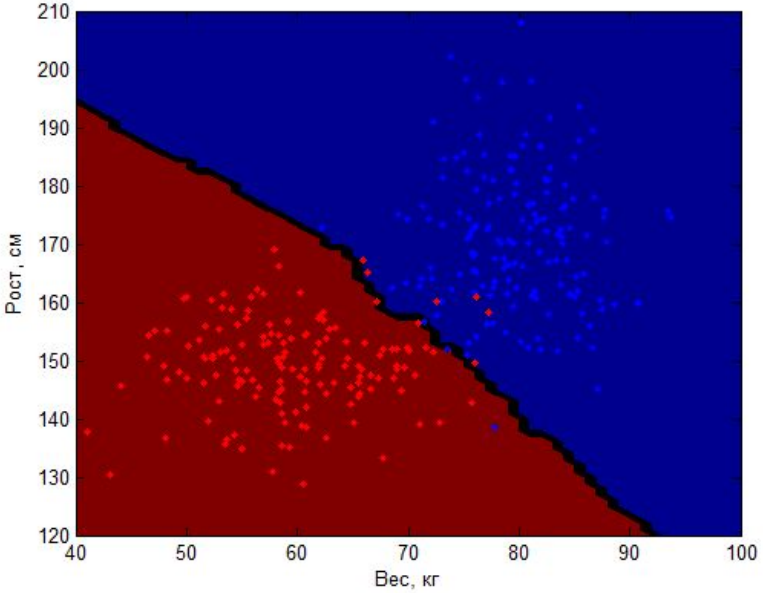
k=3



k=5



k=10



Алгоритм взвешенных *k*-ближайших соседей

Устранение неоднозначных решений возможно путём введения весов для каждого из обучающих объектов, определяющих их важность для решения:

$$a(u; X^N, k) = \arg\max_{y \in Y} \sum_{i=1}^{k} [y_i = y] \omega_i$$

Веса зависят от номера соседа. Неоднозначность снижается при линейно убывающей последовательности весов, и устраняется при нелинейной последовательности:

$w_i = \frac{k+1-i}{k}$ — линейное убывающие веса;
$w_i = q^i$ — экспоненциально убывающие веса, $0 < q < 1$

Недостатки алгоритмов типа *kNN*:

-необходимость хранить обучающую выборку целиком;
- при наличии погрешностей данных снижается точность классификации вблизи границы классов;
- поиск ближайшего соседа предполагает сравнение классифицируемого объекта со всеми объектами выборки.

Метод парзеновского окна

Веса можно задавать в зависимости не от номера соседа, а от значения расстояния до него:

$$w(i,u) = K\left(\frac{\rho(u, x_{i,u})}{h}\right)$$

Функция *K,* называемая ядром, должна быть невозрастающей положительной на отрезке [0,1] и равной нулю вне его.

Параметр *h* - ширина окна, аналог числа соседей k.

Параметр h можно задавать априори. Слишком узкие окна приводят к неустойчивой классификации, а слишком широкие к вырождению алгоритма в константу.
Ширину окна можно определять по скользящему контролю.

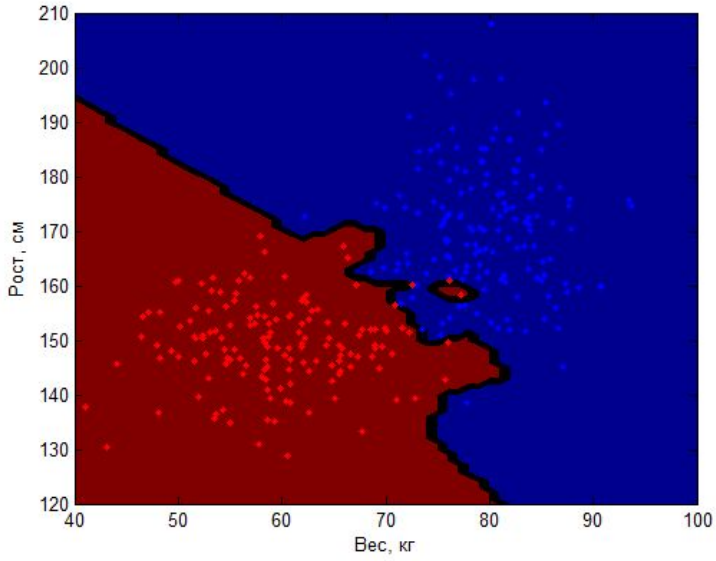
Ширину окна можно задавать в зависимости от обучающего, а не классифицируемого объекта:

$$w(i,u) = \gamma_i K\left(\frac{\rho(u, x_i)}{h_i}\right)$$

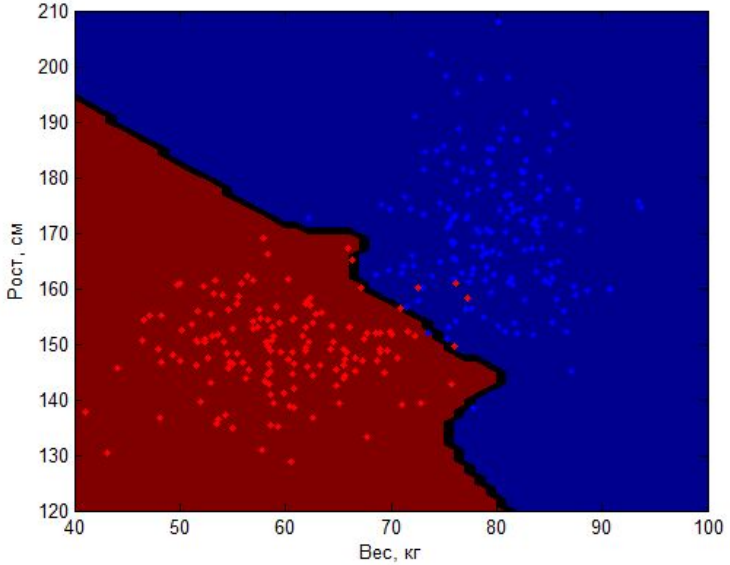
Получаем метод потенциальных функций.

Метод парзеновского окна

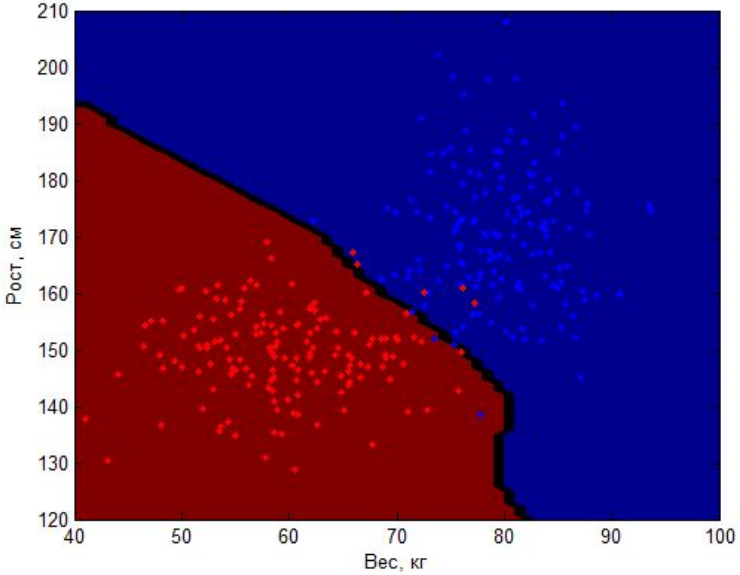
h=1



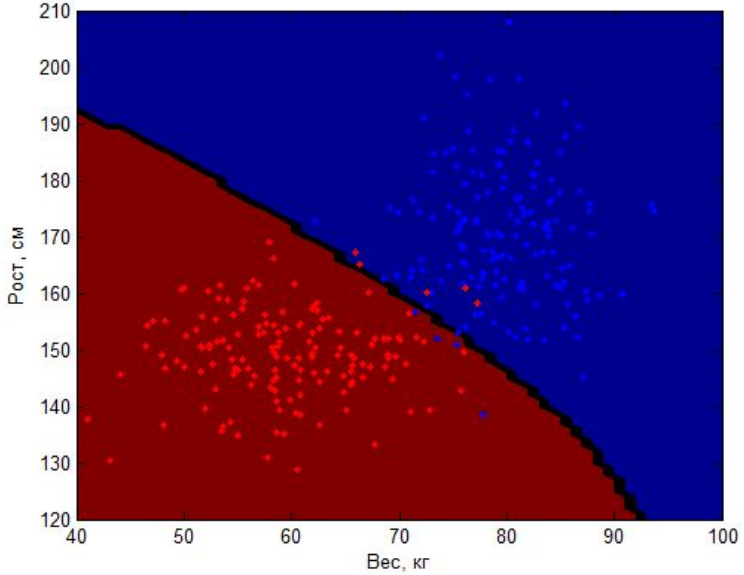
h=3



h=5



h=10

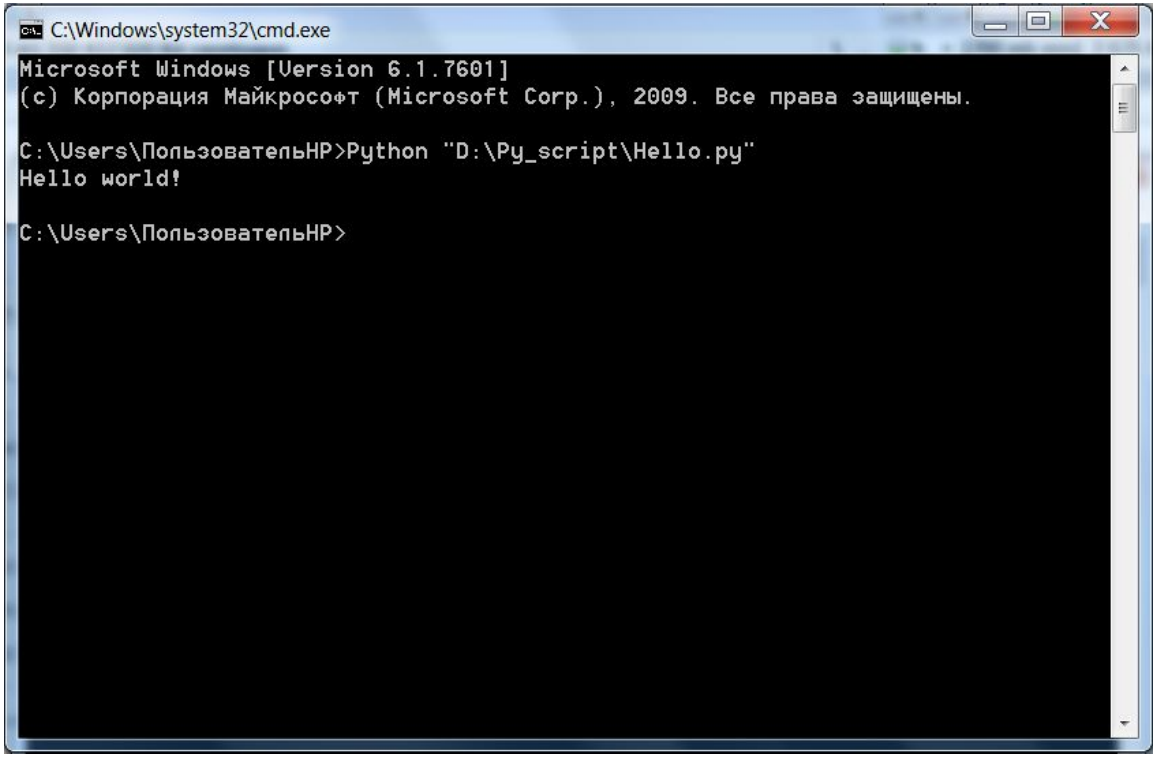


Домашнее задание

1 Установить Python3 или Anaconda с python3.

2 Установить пакеты расширения Pandas, Numpy, Matplotlib, Scipy, Scikit-learn.

3 Запустить скрипт “Hello.py” через командную строку:



или в IDE Spyder.

Если всё сделано правильно, то в консоли появится надпись «Hello world!»