Машинное обучение и пакет **ROOT . TMVA**

Дзюба Алексей \ ПИЯФ НИЦ КИ

Литература

- М.Б. Лагутин «Наглядная математическая статистика», М., 2009
- Профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных: www.machinelearning.ru
  - Курс лекций В.В.Воронцова
- TMVA User Guide
- TMVA Tutorials (C++)
- «Advanced Data Analysis from an Elementary Point of View» by Cosma Rohilla Shalizi
- Простыми словами (1, 2)

Вводные замечания

- В прошлый раз мы рассмотрели **задачу кластеризации**, то есть разбиение множества объектов на компактные группы
  - Метрика / масштаб / типы классов
  - Идеи основных методов
  - Функционалы качества разбиения
  - Дискриминантный анализ
- В практической части курса мы будем рассматривать алгоритмы машинного обучения для **задачи классификации**
  - Есть размеченная на классы выборка (кортеж по нескольким параметрам), на основе которой нужно построить алгоритм для определения класса объекта по параметрам
  - Классификация – частный случай **регрессии**. Бинарная, тернарная и т.п. функция
  - Нам интересен случай двух классов: ***сигнал*** и ***фон***

Свойства данных

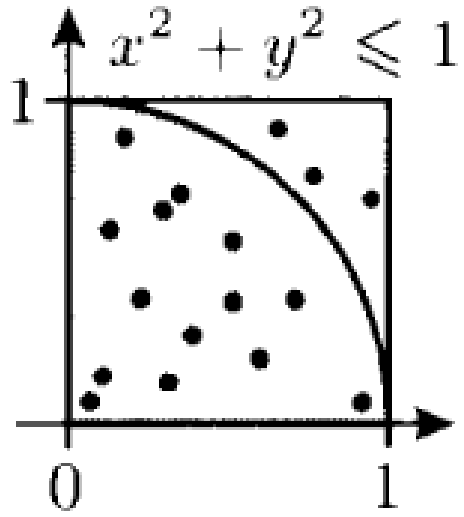
- Переменные могут (не)быть скоррелированы
- Сигнальные и фоновые события могут: занимать весь объем или его часть, или располагаться на гиперповерхностях.
- Переменные могут иметь:
  - Спайки (spikes)
  - Пороги (steps)
  - Хвосты (tails)
  - Полюса (poles)
- Классы различных типов
- Число переменных → «проклятие размерности» (“curse of dimensionality“)

Проклятие размерности

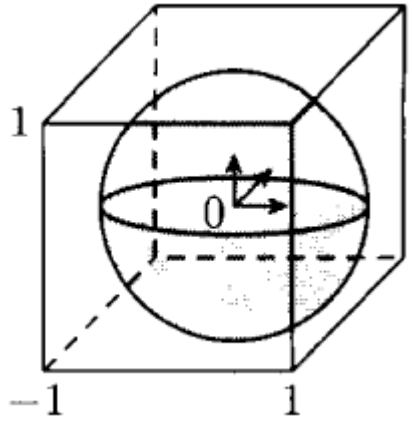
Напротив, метод Монте-Карло не зависит от размерности: чтобы найти приближенное значение интеграла

$$I = \int_0^1 \dots \int_0^1 \varphi(x_1, \dots, x_k)\, dx_1 \dots dx_k$$

с точностью порядка $1/\sqrt{n}$ достаточно случайно набросать n точек в k-мерный единичный куб (разбив псевдослучайные числа на группы из k элементов) и вычислить среднее арифметическое значений φ в этих точках. В частности, если функция — индикатор некоторой области D, то с помощью метода Монте-Карло можно приближенно определить объем этой области. Например, частота случайных точек, попавших под дугу окружности на рис. 6 будет служить приближением к $\pi/4$.



Проклятие размерности

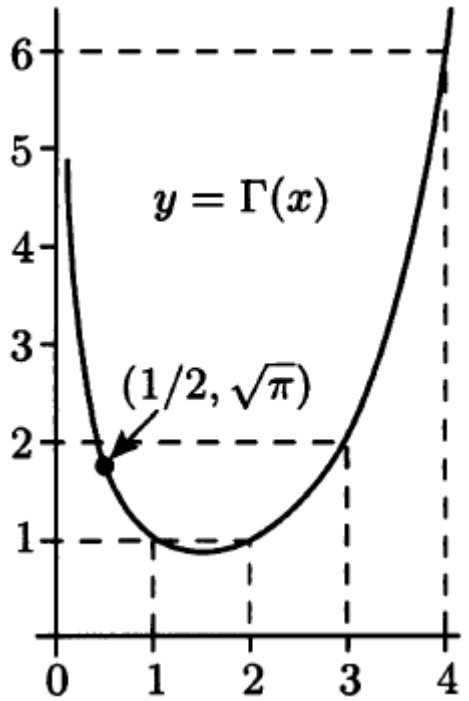


Рассмотрим k-мерный шар $\{\boldsymbol{x}\colon x_1^2 + \ldots + x_k^2 \leqslant 1\}$, вписанный в k-мерный куб $\{\boldsymbol{x}\colon |x_j| \leqslant 1, \quad j = 1, \ldots, k\}$ (рис. 8 для $k = 3$). Вероятность p_k того, что выбранная случайно в кубе точка окажется внутри шара, равна отношению объема k-мерного шара радиуса $r = 1$ к объему k-мерного куба со стороной 2. Очевидно, $p_1 = 1$, $p_2 = \pi r^2/2^2 = \pi/4 \approx 0{,}785$, $p_3 = \frac{4}{3}\pi r^3/2^3 = \pi/6 \approx 0{,}524$.

Оказывается, в общем случае

$$p_k = \left(\sqrt{\pi}/2\right)^k \Big/ \Gamma\left(\frac{k+2}{2}\right).$$

Здесь $\Gamma(x) = \int\limits_0^\infty t^{x-1}e^{-t}dt$ — известная из анализа *гамма-функция Эйлера*, график которой приведен на рис.



Факториал (Г) растет быстрее, чем степенной закон!! Большинство точек будут вне шара!

Проклятие размерности

Рассмотренный пример наглядно показывает, что может потребоваться очень много псевдослучайных точек, чтобы получить удовлетворительное приближение методом Монте-Карло для интеграла от функции, принимающей большие значения на «тощих» многомерных областях. При этом «тощими» могут оказаться области, которые на первый взгляд таковыми не представляются.

Проклятие размерности (ПР) :

- *Экспоненциального роста необходимых экспериментальных данных в зависимости от размерности пространства при решении задач вероятностно-статистического распознавания образов, машинного обучения, классификации и дискриминантного анализа.*
- *Экспоненциального рост числа вариантов в комбинаторных задачах в зависимости от размера исходных данных, что приводит к соответствующему росту сложности переборных алгоритмов.*

ПР действует и на непрерывные оптимизационные методы в силу усложнения многомерной целевой функции

ПР – проблема для cut-based методов классификации!

Борьба с ПР (1 – произведение проекций)

Случайные величины исследуемых векторов независимы или, что более реально, слабо взаимозависимы.

Можно восстановить одномерные распределения случайных величин и построить многомерные распределения как их произведения.

Далее, используя ту же обучающую выборку в режиме скользящего экзамена можно восстановить одномерное распределение отношения функций правдоподобия дифференцируемых классов, что устраняет смещения, связанные с взаимозависимостью в решающем правиле.

Борьба с ПР (2 – уменьшение размерности)

Распределение вектора взаимозависимых случайных величин сосредоточено в подпространстве меньшей размерности, то есть вектор может быть хорошо приближен линейной функцией меньшего числа переменных.

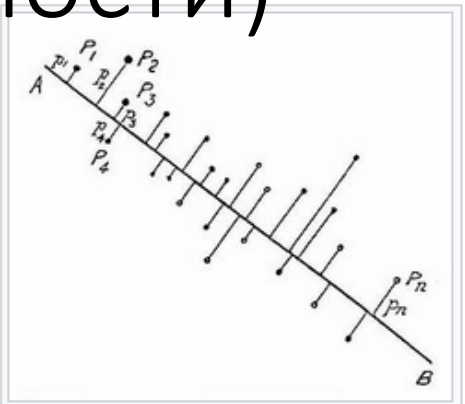


Иллюстрация к знаменитой работе Пирсона (1901): даны точки P_i на плоскости, p_i — расстояние от P_i до прямой AB. Ищется прямая AB, минимизирующая сумму $\sum_i p_i^2$

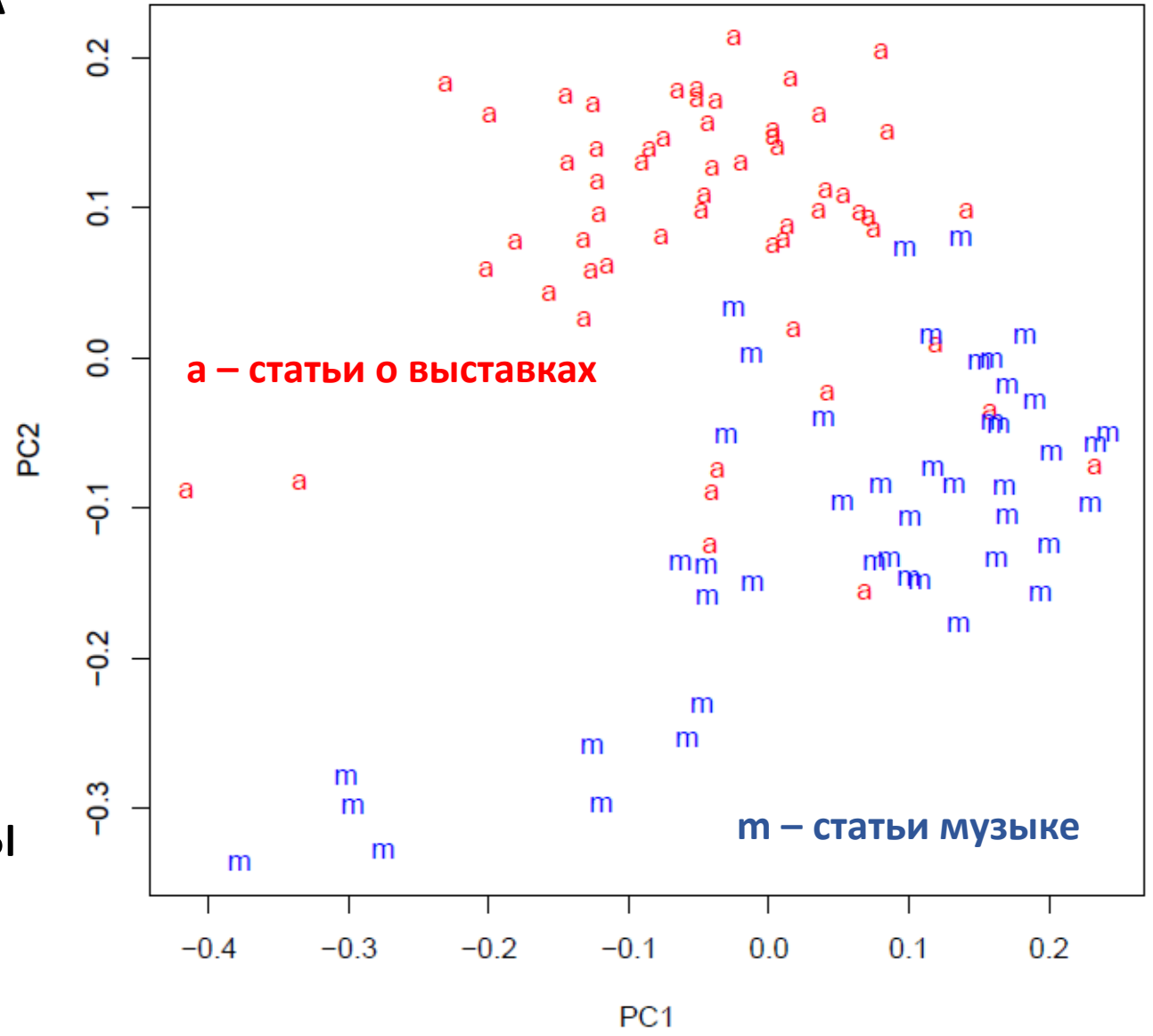
МЕТОД ГЛАВНЫХ КОМПОНЕНТ (PCA principal component analysis) позволяет снизить размерность.

Первая главная компонента максимизирует выборочную дисперсию проекции данных

Поставленные задачи классификации (распознавания, дискриминации) могут решаться уже в пространстве малой размерности.

Пример работы PCA

- New York Times Annotated Corpus (1.8M размеченных статей Таймс с 1987 to 2007)
- 57 историй о выставках (art) и 45 о музыке (m)
- “**bag-of-words**” – для каждой статьи 4431-размерный словарь сколько раз то или иное слово встречается в статье
- Первые две главные компоненты позволяют провести хорошее смысловое разделение историй!



PCA in **ROOT** and **ROOT . TMVA**

- Stand alone: **`TPrincipal`** class of **ROOT**
  - Here is an example
- In **TMVA** PCA is implemented via **`TPrincipal`** as well

Variable transformations to be applied prior to the MVA training (and application) can be defined independently for each MVA method with the booking option `VarTransform=<type>`, where `<type>` denotes the desired transformation (or chain of transformations). The available transformation types are normalisation, decorrelation, principal component analysis and Gaussianisation, which are labelled by `Norm`, `Deco`, `PCA`, `Uniform`, `Gauss`, respectively, or, equivalently, by the short-hand notations `N`, `D`, `P`, `U` , `G`.

Transformations can be *chained* allowing the consecutive application of all defined transformations to the variables for each event. For example, the above Gaussianisation and decorrelation sequence would be programmed by `VarTransform=G,D`, or even `VarTransform=G,D,G,D` in case of two iterations (instead of the comma ",", a "+" can be equivalently used as chain operator, i.e. `VarTransform=G+D+G+D`). The ordering of the transformations goes from left (first) to right (last).

```
factory->BookMethod( TMVA::Types::kLD, "LD_GD", "H:!V:VarTransform=G,D");
```

Метод ближайшего соседа

Задача поиска ближайшего соседа заключается в отыскании среди множества элементов, расположенных в метрическом пространстве, элементов близких к заданному, согласно некоторой заданной функции близости, определяющей это метрическое пространство.

Естественное расширение – **метод k ближайших соседей**

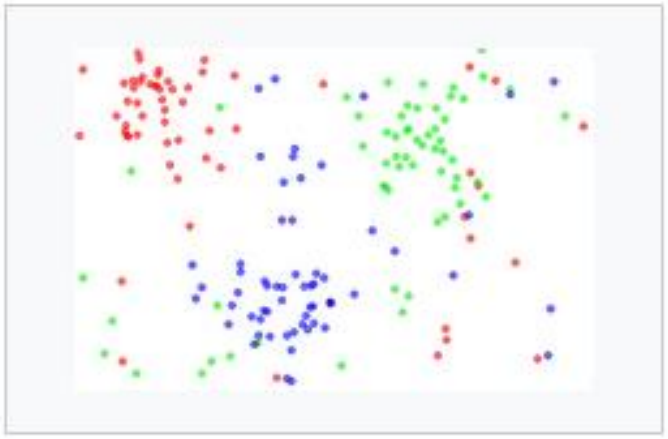


Fig. 1. The dataset.

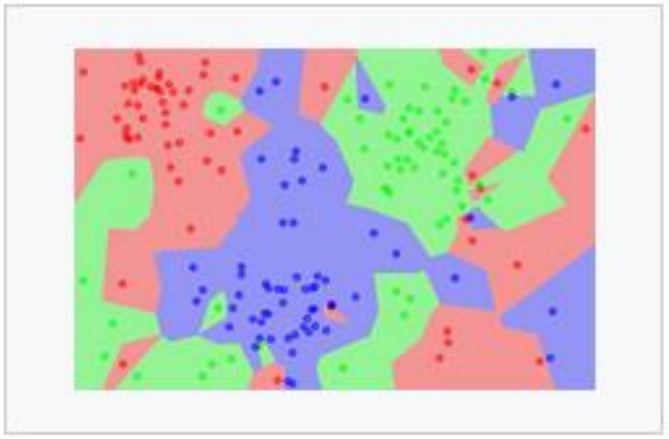


Fig. 2. The 1NN classification map.

Fig. 3. The 5NN classification map.

Метод k ближайших соседей в HEP практически не используется

Выбор параметров классификации / методы

- Насколько большой калибровочный набор данных (training sample) ?
- Какие переменные информативны? Как велика корреляция между ними

Доступные опции:

List of acronyms:
BDT = boosted decision tree, see manual page 103
ANN = articifical neural network
MLP = multi-layer perceptron, a specific form of ANN, also the name of our flagship ANN, manual p. 92
FDA = functional discriminant analysis, see manual p. 87
LD = linear discriminant , manual p. 85
SVM = support vector machine, manual p. 98 , SVM currently available only for classification
Cuts = like in “cut selection“, manual p. 56
Fisher = Ronald A. Fisher, classifier similar to LD, manual p. 83

Выбор метода классификации

universitätbonn

- Number of „parameters“ is limited due to small data sample

→ Use Linear classifier or FDA, small BDT (small MLP)

- Variables are uncorrelated (or only linear corrs) → likelihood
- I just want something simple → Cuts, LD, Fisher
- Methods that usually work out of the box, even for complex problems → BDT, MLP, SVM

Из туториала по **TMVA**

Выбор метода классификации

	CRITERIA	MVA METHOD: Cuts	Likeli-hood	PDE-RS / k-NN	PDE-Foam	H-Matrix	Fisher / LD	MLP	BDT	Rule-Fit	SVM
Perfor-mance	No or linear correlations	⋆	⋆⋆	⋆	⋆	⋆	⋆⋆	⋆⋆	⋆	⋆⋆	⋆
	Nonlinear correlations	○	○	⋆⋆	⋆⋆	○	○	⋆⋆	⋆⋆	⋆⋆	⋆⋆
Speed	Training	○	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆	⋆	⋆	○
	Response	⋆⋆	⋆⋆	○	⋆	⋆⋆	⋆⋆	⋆⋆	⋆	⋆⋆	⋆
Robust-ness	Overtraining	⋆⋆	⋆	⋆	⋆	⋆⋆	⋆⋆	⋆	⋆[39]	⋆	⋆⋆
	Weak variables	⋆⋆	⋆	○	○	⋆⋆	⋆⋆	⋆	⋆⋆	⋆	⋆
Curse of dimensionality		○	⋆⋆	○	○	⋆⋆	⋆⋆	⋆	⋆	⋆	
Transparency		⋆⋆	⋆⋆	⋆	⋆	⋆⋆	⋆⋆	○	○	○	○

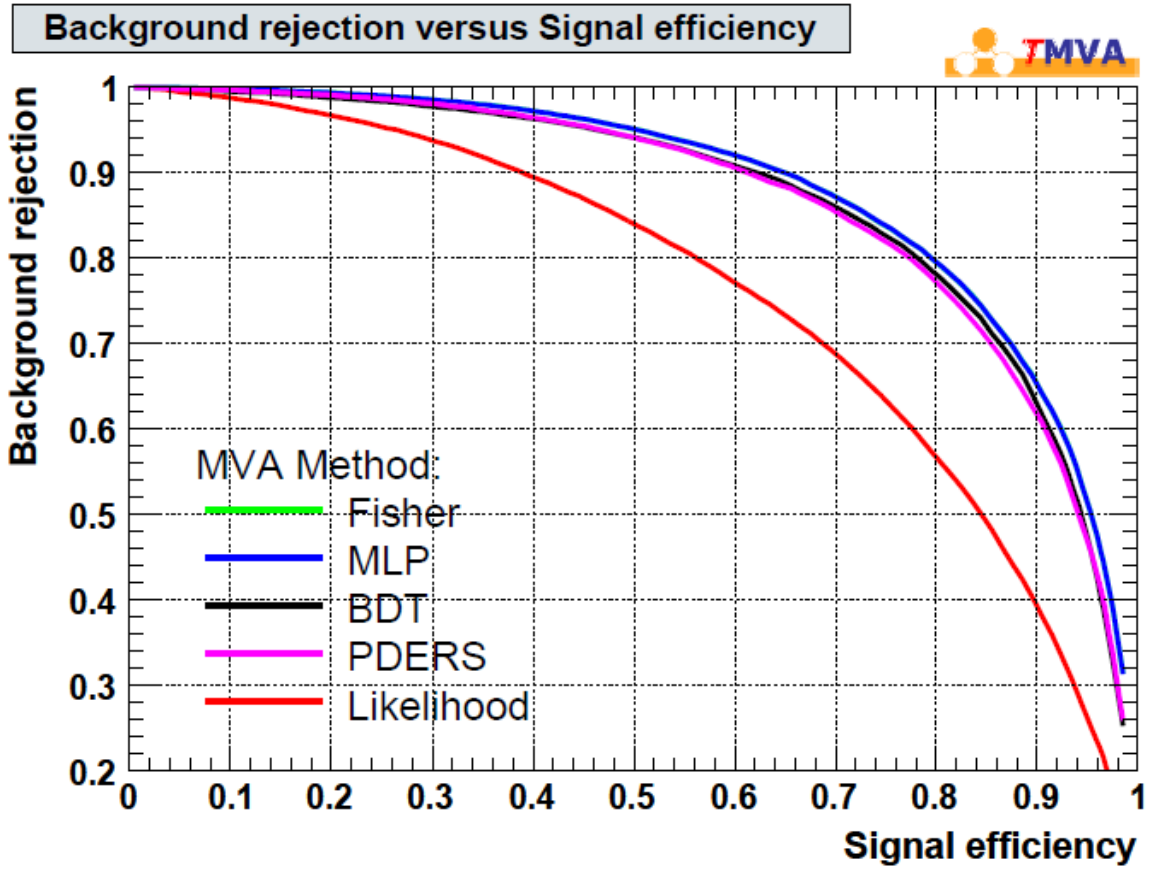
Table 6: Assessment of MVA method properties. The symbols stand for the attributes "good" (⋆⋆), "fair" (⋆) and "bad" (○).

Оценка качества классификатора. Ошибки.

1. **Ошибка первого рода** — ситуация, когда отвергнута правильная нулевая гипотеза (англ. type I errors, α errors, false positive, ошибочное отвержение).
2. **Ошибка второго рода** — ситуация, когда принята неправильная нулевая гипотеза (англ. type II errors, β errors, false negative, ошибочное принятие).

Выбор классификатора невозможен без учета стоимости ошибок.
Например, оптимизируется (**A*α + B*β**), где A и B известные константы, а α и β – доли ошибок первого и второго рода (свойство классификатора)

ROC-кривая и AUC



Чем больше AUC, тем лучше качество классификатора

ROC-кривая (англ. receiver operating characteristic, рабочая характеристика приёмника) — график, позволяющий оценить качество бинарной классификации, отображает соотношение между долей объектов от общего количества носителей признака, верно классифицированных как несущие признак (англ. true positive rate, TPR, называемой чувствительностью алгоритма классификации), и долей объектов от общего количества объектов, не несущих признака, ошибочно классифицированных как несущие признак (англ. false positive rate, FPR, величина 1-FPR называется специфичностью алгоритма классификации) при варьировании порога решающего правила. Также известна как **кривая ошибок**.

Случайный классификатор – линейная ROC-кривая

AUC (англ. area under ROC curve, площадь под ROC-кривой) — площадь, ограниченная ROC-кривой и осью доли ложных положительных классификаций

Терминология ROC-анализа

condition positive (P)

the number of real positive cases in the data

condition negative (N)

the number of real negative cases in the data

---

true positive (TP)

eqv. with hit

true negative (TN)

eqv. with correct rejection

false positive (FP)

eqv. with false alarm, Type I error

false negative (FN)

eqv. with miss, Type II error

Prevalence Threshold (PT)

$$PT = \frac{\sqrt{TPR(-TNR+1)} + TNR - 1}{(TPR + TNR - 1)}$$

Threat score (TS) or critical success index (CSI)

$$\mathrm{TS} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN} + \mathrm{FP}}$$

sensitivity, recall, hit rate, or true positive rate (TPR)

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = 1 - \mathrm{FNR}$$

specificity, selectivity or true negative rate (TNR)

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{N}} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} = 1 - \mathrm{FPR}$$

precision or positive predictive value (PPV)

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = 1 - \mathrm{FDR}$$

negative predictive value (NPV)

$$\mathrm{NPV} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FN}} = 1 - \mathrm{FOR}$$

miss rate or false negative rate (FNR)

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{P}} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} = 1 - \mathrm{TPR}$$

fall-out or false positive rate (FPR)

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{N}} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} = 1 - \mathrm{TNR}$$

false discovery rate (FDR)

$$\mathrm{FDR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TP}} = 1 - \mathrm{PPV}$$

false omission rate (FOR)

$$\mathrm{FOR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TN}} = 1 - \mathrm{NPV}$$

accuracy (ACC)

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{P} + \mathrm{N}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

balanced accuracy (BA)

$$\mathrm{BA} = \frac{TPR + TNR}{2}$$

F1 score

is the harmonic mean of precision and sensitivity

$$\mathrm{F_1} = 2 \cdot \frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

Matthews correlation coefficient (MCC)

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN})(\mathrm{TN} + \mathrm{FP})(\mathrm{TN} + \mathrm{FN})}}$$

Fowlkes–Mallows index (FM)

$$\mathrm{FM} = \sqrt{\frac{TP}{TP + FP} \cdot \frac{TP}{TP + FN}} = \sqrt{PPV \cdot TPR}$$

informedness or bookmaker informedness (BM)

$$\mathrm{BM} = \mathrm{TPR} + \mathrm{TNR} - 1$$

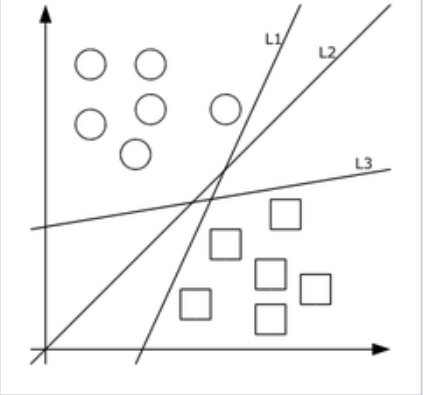
markedness (MK) or deltaP

$$\mathrm{MK} = \mathrm{PPV} + \mathrm{NPV} - 1$$

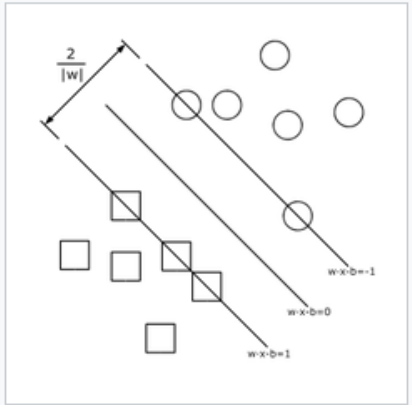
Умение оперировать терминологией – важный навык для исследователя данных!

SVM (Support Vector Machine).

- **Метод опорных векторов** (англ. SVM, support vector machine). См. также SVM.
- Основная идея метода — перевод исходных векторов в пространство более высокой размерности и поиск разделяющей гиперплоскости с максимальным зазором в этом пространстве. Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей классы. *Разделяющей гиперплоскостью* будет гиперплоскость, максимизирующая расстояние до двух параллельных гиперплоскостей.
- Для линейно разделимой выборки алгоритм позволяет получить оптимальную гиперплоскость
- Задача квадратичного программирования



Несколько классифицирующих разделяющих прямых (гиперплоскостей), из которых только одна соответствует оптимальному разделению



Оптимальная разделяющая гиперплоскость для метода опорных векторов, построенная на точках из двух классов. Ближайшие к параллельным гиперплоскостям точки называются опорными векторами

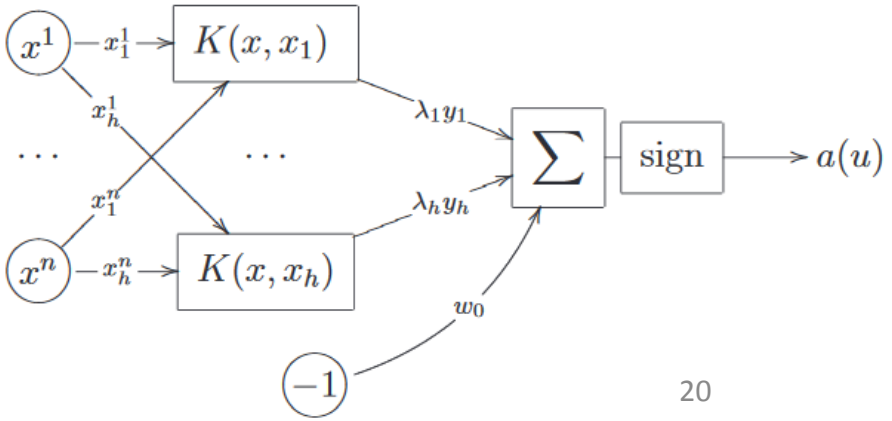
SVM для линейно неразделимых выборок

1. Чтобы обобщить SVM на случай линейной неразделимости, позволим алгоритму допускать ошибки на обучающих объектах, но при этом постараемся, чтобы ошибок было мало. Этот вариант алгоритма называют SVM с мягким зазором (soft-margin SVM), тогда как в линейно разделимом случае говорят об SVM с жёстким зазором (hard-margin SVM). Константу ***C*** обычно выбирают по критерию скользящего контроля. Это трудоёмкий способ, так как задачу приходится решать заново при каждом значении ***C***.

$$\begin{cases} -\mathscr{L}(\lambda) = -\sum\limits_{i=1}^{\ell}\lambda_i + \frac{1}{2}\sum\limits_{i=1}^{\ell}\sum\limits_{j=1}^{\ell}\lambda_i\lambda_j y_i y_j \langle x_i, x_j\rangle \to \min\limits_{\lambda}; \\ 0 \leqslant \lambda_i \leqslant C, \quad i = 1, \ldots, \ell; \\ \sum\limits_{i=1}^{\ell}\lambda_i y_i = 0. \end{cases}$$

2. Переход от исходного пространства признаковых описаний объектов ***X*** к новому пространству ***H*** с помощью некоторого преобразования $\psi: X \to H$. Если пространство ***H*** имеет достаточно высокую размерность, то можно надеяться, что в нём выборка окажется линейно разделимой (легко показать, что если выборка X^ℓ не про-тиворечива, то всегда найдётся пространство размерности не более ℓ, в котором онабудет линейно разделима). Пространство ***H*** называют *спрямляющим*.

Если $X=R^n$, то алгоритм ***a(x)*** можно рассматривать как двухслойную нейронную сеть, имеющую ***n*** входных нейронов и ***h*** нейронов в скрытом слое. ***K*** – ядро преобразования из 2.



Преимущества и недостатки SVM

- это наиболее быстрый метод нахождения решающих функций;
- метод сводится к решению задачи квадратичного программирования в выпуклой области, которая всегда имеет единственное решение;
- метод находит разделяющую полосу максимальной ширины, что позволяет в дальнейшем осуществлять более уверенную классификацию;
- метод чувствителен к шумам и стандартизации данных;
- не существует общего подхода к автоматическому выбору ядра (и построению спрямляющего подпространства в целом) в случае линейной неразделимости классов.

SVN в **ROOT . TMVA**

The SVM classifier is booked via the command:

```
factory->BookMethod( TMVA::Types::kSVM, "SVM", "<options>" );
```

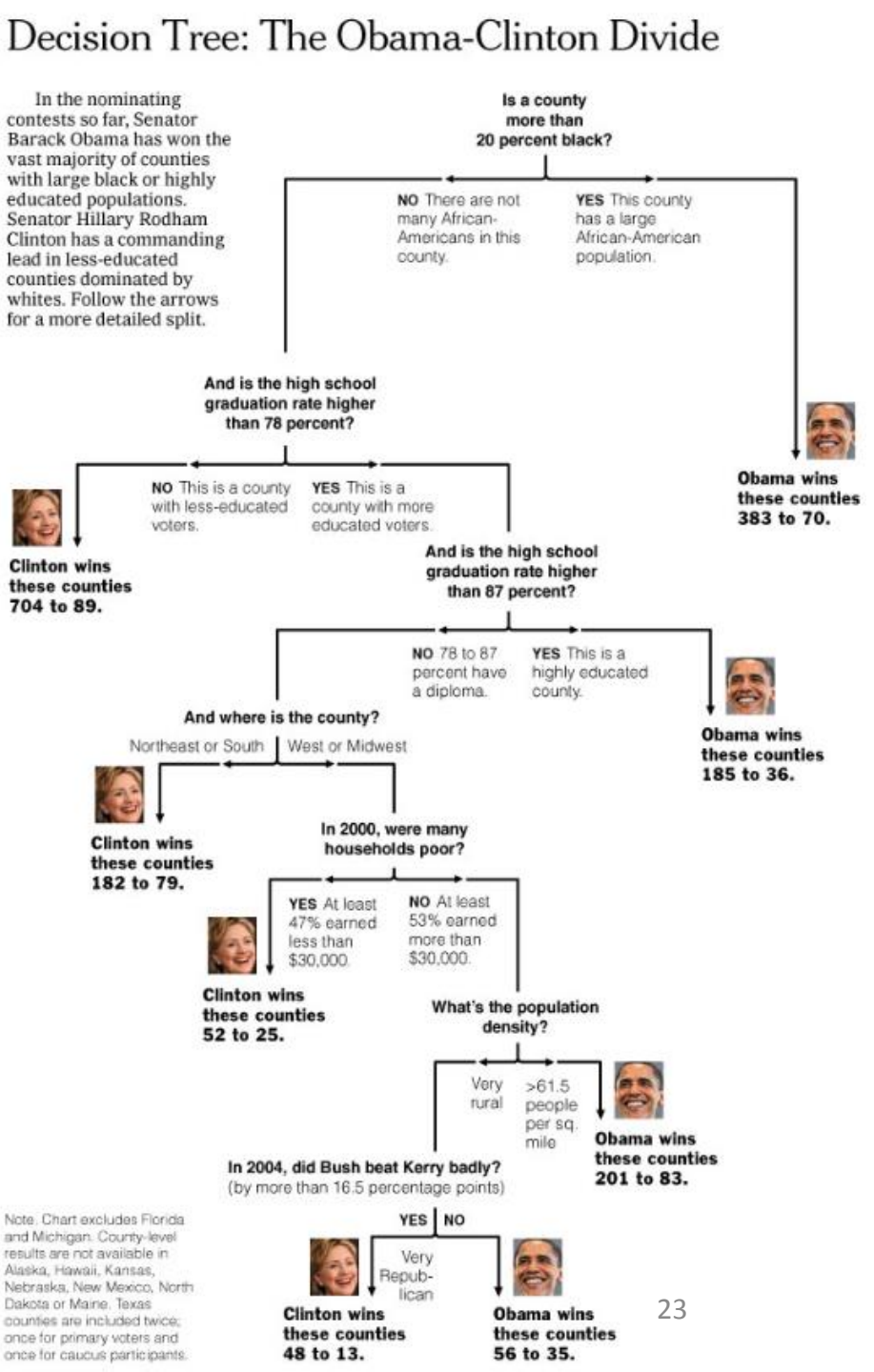
Option	Array	Default	Predefined Values	Description
`Gamma`	–	1	–	RBF kernel parameter: Gamma (size of the Kernel)
`C`	–	1	–	Cost parameter
`Tol`	–	0.01	–	Tolerance parameter
`MaxIter`	–	1000	–	Maximum number of training loops

The TMVA SVM algorithm comes currently only with the Gaussian kernel function. With sufficient training statistics, the Gaussian kernel allows to approximate any separating function in the input space. It is crucial for the performance of the SVM to appropriately tune the kernel parameters and the cost parameter `C`. In case of a Gaussian, the kernel is tuned via option `Gamma` which is related to the width σ by $\Gamma = 1/(2\sigma^2)$. The optimal tuning of these parameters is specific to the problem and must be done by the user.

The SVM training time scales with n^2, where n is the number of vectors (events) in the training data set. The user is therefore advised to restrict the sample size during the first rough scan of the kernel parameters. Also increasing the minimisation tolerance helps to speed up the training.

Решающее дерево

- **Решающее дерево (Decision tree)** — решение задачи обучения с учителем, основанный на том, как решает задачи прогнозирования человек
- В общем случае — это ***k***-дерево с решающими правилами в нелистовых вершинах (узлах) и некотором заключении о целевой функции в листовых вершинах (прогнозом).
- Решающее правило — некоторая функция от объекта, позволяющее определить, в какую из дочерних вершин нужно поместить рассматриваемый объект.
- В листовых вершинах могут находиться разные объекты: класс, который нужно присвоить попавшему туда объекту (в задаче классификации), вероятности классов (в задаче классификации), непосредственно значение целевой функции (задача регрессии).
- Чаще всего используются **двоичные (бинарные) решающие деревья**.



Бустинг

- Возможно ли, имея множество плохих (незначительно отличающихся от случайных) алгоритмов обучения, получить хороший? ДА!
- **Бустинг** (англ. boosting — улучшение) — это процедура последовательного построения композиции алгоритмов машинного обучения, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов.
- Бустинг представляет собой жадный алгоритм построения композиции алгоритмов.
- Бустинг над решающими деревьями в **TMVA**

Бустинг

Преимущества:

- высокая обобщающая способность
- простота,
- универсальность,
- гибкость (возможность построения различных модификаций)

Бустинг над решающими деревьями считается одним из наиболее эффективных методов с точки зрения качества классификации.

Взвешенное голосование не увеличивает эффективную сложность алгоритма, а лишь сглаживает ответы базовых алгоритмов.

Бустинг

- **Алгоритм AdaBoost** (сокр. от adaptive boosting) —мета-алгоритмом, в процессе обучения строит композицию из базовых алгоритмов обучения для улучшения их эффективности. AdaBoost является алгоритмом адаптивного бустинга в том смысле, что каждый следующий классификатор строится по объектам, которые плохо классифицируются предыдущими классификаторами.
  - **TMVA**: Boosted Decision Trees with adaptive boosting (**BDT**)
- **Алгоритм AnyBoost** - класс алгоритмов, представляющих бустинг как процесс градиентного спуска. В основе алгоритма лежит последовательное уточнение функции, представляющей собой линейную комбинацию базовых классификаторов, с тем чтобы минимизировать функцию потерь. В класс AnyBoost входят практически все алгоритмы бустинга как частные случаи.
  - **TMVA**: Boosted Decision Trees with gradient boosting (**BDTG**)

BDT в **ROOT . TMVA**

```
// Boosted Decision Trees with adaptive boosting
factory->BookMethod( TMVA::Types::kBDT, "BDT",
    "!H:!V:NTrees=400:nEventsMin=400:MaxDepth=3:BoostType=AdaBoost:\
     SeparationType=GiniIndex:nCuts=20:PruneMethod=NoPruning" );

// Boosted Decision Trees with gradient boosting
factory->BookMethod( TMVA::Types::kBDT, "BDTG",
    "!H:!V:NTrees=1000:BoostType=Grad:Shrinkage=0.30:UseBaggedGrad:\
     GradBaggingFraction=0.6:SeparationType=GiniIndex:nCuts=20:\
     PruneMethod=CostComplexity:PruneStrength=50:NNodesMax=5" );
```

Настройки BDT в **ROOT . TMVA**

Option	Array	Default	Predefined Values	Description
NTrees	–	800	–	Number of trees in the forest
MaxDepth	–	3	–	Max depth of the decision tree allowed
MinNodeSize	–	5%	–	Minimum percentage of training events required in a leaf node (default: Classification: 5%, Regression: 0.2%)
nCuts	–	20	–	Number of grid points in variable range used in finding optimal cut in node splitting
BoostType	–	AdaBoost	AdaBoost, RealAdaBoost, Bagging, AdaBoostR2, Grad	Boosting type for the trees in the forest (note: AdaCost is still experimental)
AdaBoostR2Loss	–	Quadratic	Linear, Quadratic, Exponential	Type of Loss function in AdaBoostR2
UseBaggedGrad	–	False	–	Use only a random subsample of all events for growing the trees in each iteration. (Only valid for GradBoost)
Shrinkage	–	1	–	Learning rate for GradBoost algorithm
AdaBoostBeta	–	0.5	–	Learning rate for AdaBoost algorithm

Настройки BDT в **ROOT . TMVA**

`UseRandomisedTrees`	–	`False`	–	Determine at each node splitting the cut variable only as the best out of a random subset of variables (like in RandomForests)
`UseNvars`	–	`2`	–	Size of the subset of variables used with RandomisedTree option
`UsePoissonNvars`	–	`True`	–	Interpret UseNvars not as fixed number but as mean of a Possion distribution in each split with RandomisedTree option
`BaggedSampleFraction`	–	`0.6`	–	Relative size of bagged event sample to original size of the data sample (used whenever bagging is used (i.e. UseBaggedGrad, Bagging,)
`UseYesNoLeaf`	–	`True`	–	Use Sig or Bkg categories, or the purity=S/(S+B) as classification of the leaf node -> Real-AdaBoost

Оптимизация параметров настройки классификатора – необходимая (немного нудная) работа!

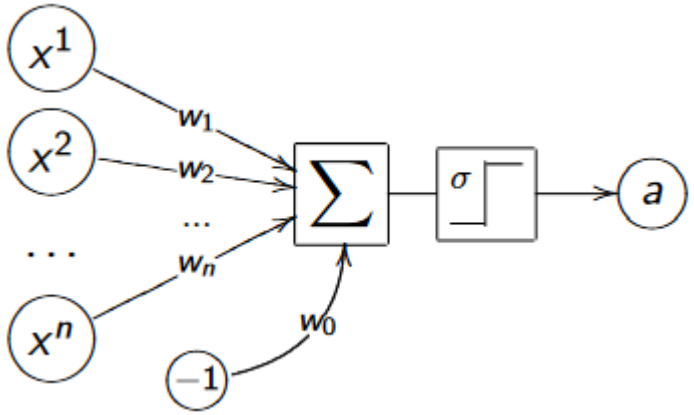
ссылка

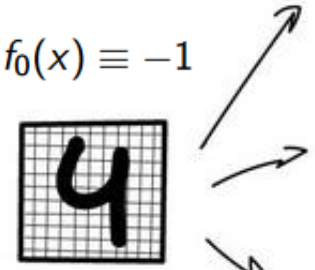
Нейронные сети

Линейная модель нейрона МакКаллока-Питтса [1943]:

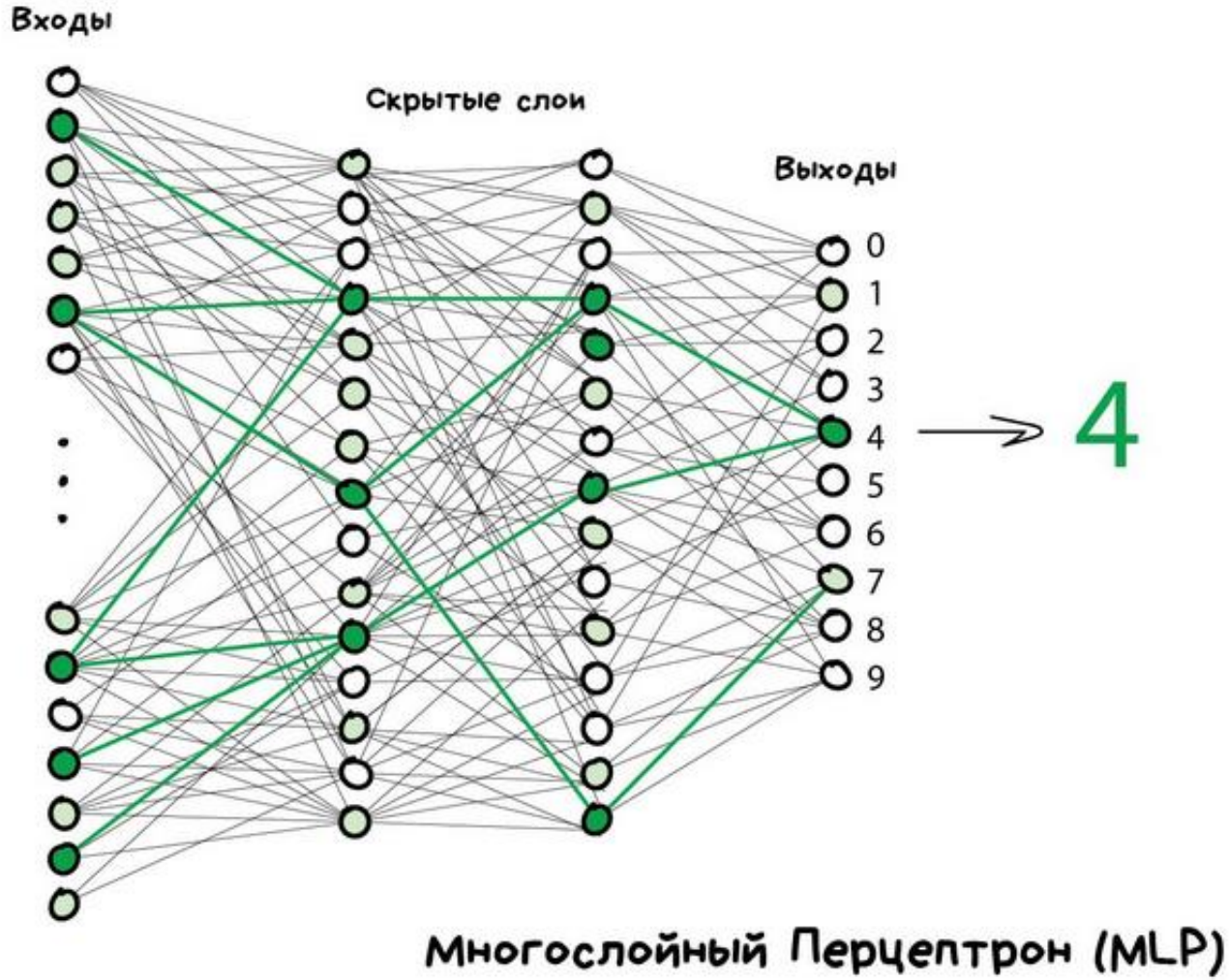
$$a(x, w) = \sigma(\langle w, x \rangle) = \sigma\left(\sum_{j=1}^{n} w_j f_j(x) - w_0 \right),$$

$\sigma(z)$ — функция активации (например, sign),
w_j — весовые коэффициенты синаптических связей,
w_0 — порог активации,
$w, x \in \mathbb{R}^{n+1}$, если ввести константный признак $f_0(x) \equiv -1$





1 пиксель
=
1 вход нейрона
первого слоя

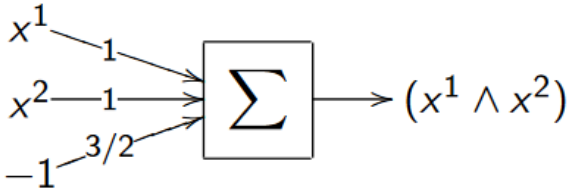


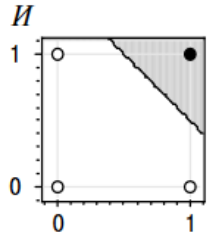
ссылка

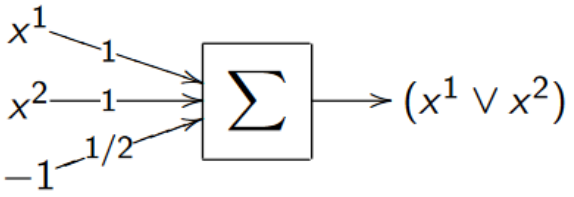
Нейронная реализация логических функций

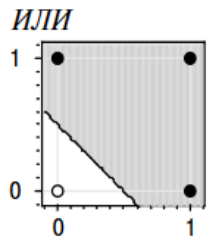
Функции И, ИЛИ, НЕ от бинарных переменных x^1 и x^2:

$$x^1 \wedge x^2 = \left[x^1 + x^2 - \tfrac{3}{2} > 0\right];$$
$$x^1 \vee x^2 = \left[x^1 + x^2 - \tfrac{1}{2} > 0\right];$$
$$\neg x^1 = \left[-x^1 + \tfrac{1}{2} > 0\right];$$





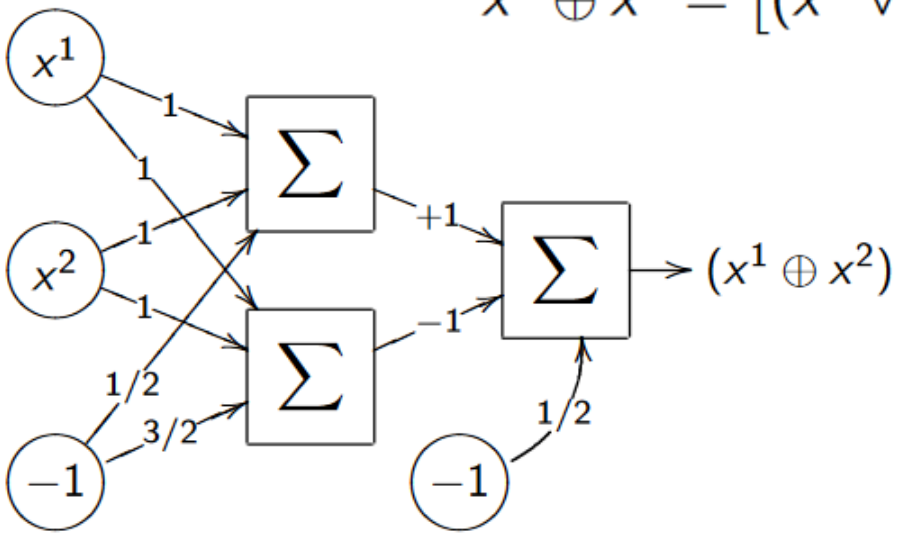


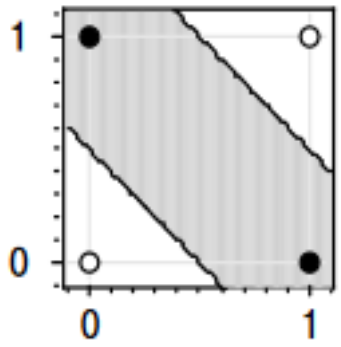


Реализация исключающего ИЛИ (XOR)

Сетью (двухслойной суперпозицией) функций И, ИЛИ, НЕ:

$$x^1 \oplus x^2 = \left[(x^1 \vee x^2) - (x^1 \wedge x^2) - \tfrac{1}{2} > 0\right].$$





Любую ли функцию можно представить нейросетью?

ссылка

- Двухслойная сеть в $\{0,1\}^n$ позволяет реализовать произвольную булеву функцию (ДНФ).
- Двухслойная сеть в $\mathbb{R}^n$ позволяет отделить произвольный выпуклый многогранник.
- Трёхслойная сеть $\mathbb{R}^n$ позволяет отделить произвольную многогранную область, не обязательно выпуклую, и даже не обязательно связную.
- С помощью линейных операций и одной нелинейной *функции активации* σ можно приблизить любую непрерывную функцию с любой желаемой точностью.

Практические рекомендации:

- Двух-трёх слоёв теоретически достаточно.
- Глубокие сети — это встроенное обучение признаков.

Любую ли функцию можно представить нейросетью?

ссылка

Функция $\sigma(z)$ — *сигмоида*, если $\lim\limits_{z\to-\infty}\sigma(z)=0$ и $\lim\limits_{z\to+\infty}\sigma(z)=1$.

Теорема Цыбенко (1989)

Если $\sigma(z)$ — непрерывная сигмоида, то для любой непрерывной на $[0,1]^n$ функции $f(x)$ существуют такие значения параметров $w_h\in\mathbb{R}^n$, $w_0\in\mathbb{R}$, $\alpha_h\in\mathbb{R}$, что двухслойная сеть

$$a(x)=\sum_{h=1}^{H}\alpha_h\sigma\big(\langle x,w_h\rangle+w_0\big)$$

равномерно приближает $f(x)$ с любой точностью ε:

$$\big|a(x)-f(x)\big|<\varepsilon,\ \text{для всех } x\in[0,1]^n.$$

George Cybenko. Approximation by Superpositions of a Sigmoidal function. Mathematics of Control, Signals, and Systems. 1989.

Настройки MLP в **ROOT.TMVA**

This neural network interfaces the **ROOT** class **TMultiLayerPerceptron** and is booked through the Factory via the command line:

```
factory->BookMethod( Types::kTMlpANN, "TMlp_ANN", "<options>" );
```

MLP booking:

```
factory->BookMethod( Types::kMLP, "MLP_ANN", "<options>" );
```

Суммирование:

$$\kappa:\ (y_1^{(\ell)},..,y_n^{(\ell)}|w_{0j}^{(\ell)},..,w_{nj}^{(\ell)}) \rightarrow \begin{cases} w_{0j}^{(\ell)} + \sum_{i=1}^{n} y_i^{(\ell)} w_{ij}^{(\ell)} & \textit{Sum}, \\ w_{0j}^{(\ell)} + \sum_{i=1}^{n} \left(y_i^{(\ell)} w_{ij}^{(\ell)}\right)^2 & \textit{Sum of squares}, \\ w_{0j}^{(\ell)} + \sum_{i=1}^{n} |y_i^{(\ell)} w_{ij}^{(\ell)}| & \textit{Sum of absolutes}, \end{cases}$$

Активация:

$$\alpha:\ x \rightarrow \begin{cases} x & \textit{Linear}, \\ \frac{1}{1+e^{-kx}} & \textit{Sigmoid}, \\ \frac{e^x - e^{-x}}{e^x + e^{-x}} & \textit{Tanh}, \\ e^{-x^2/2} & \textit{Radial}. \end{cases}$$

Настройки MLP в **ROOT . TMVA**

Option	Array	Default	Predefined Values	Description
NCycles	–	500	–	Number of training cycles
HiddenLayers	–	N,N-1	–	Specification of hidden layer architecture
NeuronType	–	sigmoid	–	Neuron activation function type
RandomSeed	–	1	–	Random seed for initial synapse weights (0 means unique seed for each run; default value '1')
EstimatorType	–	MSE	MSE, CE, linear, sigmoid, tanh, radial	MSE (Mean Square Estimator) for Gaussian Likelihood or CE(Cross-Entropy) for Bernoulli Likelihood
NeuronInputType	–	sum	sum, sqsum, abssum	Neuron input function type
TrainingMethod	–	BP	BP, GA, BFGS	Train with Back-Propagation (BP), BFGS Algorithm (BFGS), or Genetic Algorithm (GA - slower and worse)
LearningRate	–	0.02	–	ANN learning rate parameter
DecayRate	–	0.01	–	Decay rate for learning parameter
TestRate	–	10	–	Test for overtraining performed at each #th epochs
EpochMonitoring	–	False	–	Provide epoch-wise monitoring plots according to TestRate (caution: causes big ROOT output file!)
Sampling	–	1	–	Only 'Sampling' (randomly selected) events are trained each epoch
SamplingEpoch	–	1	–	Sampling is used for the first 'SamplingEpoch' epochs, afterwards, all events are taken for training
SamplingImportance	–	1	–	The sampling weights of events in epochs which successful (worse estimator than before) are multiplied with SamplingImportance, else they are divided.
SamplingTraining	–	True	–	The training sample is sampled
SamplingTesting	–	False	–	The testing sample is sampled
ResetStep	–	50	–	How often BFGS should reset history
Tau	–	3	–	LineSearch size step
BPMode	–	sequential	sequential, batch	Back-propagation learning mode: sequential or batch
BatchSize	–	-1	–	Batch size: number of events/batch, only set if in Batch Mode, -1 for BatchSize=number_of_events
ConvergenceImprove	–	1e-30	–	Minimum improvement which counts as improvement (<0 means automatic convergence check is turned off)
ConvergenceTests	–	-1	–	Number of steps (without improvement) required for convergence (<0 means automatic convergence check is turned off)
UseRegulator	–	False	–	Use regulator to avoid over-training
UpdateLimit	–	10000	–	Maximum times of regulator update
CalculateErrors	–	False	–	Calculates inverse Hessian matrix at the end of the training to be able to calculate the uncertainties of an MVA value
WeightRange	–	1	–	Take the events for the estimator calculations from small deviations from the desired value to large deviations only over the weight range

RTFancyM!

Заключение

- «Учиться, учиться и учиться» (В.И. Ленин)
  - Понимать суть (математику) методов классификации
  - Исходя из задачи понимать какой классификатор нужен и какие его настройки важны
  - Обучать классификаторы
- На практике для задач HEP: **`ROOT.TMVA`** → **BDT, BDTG, MLP**
- **`ROOT.TMVA`** – стандарт в нашем комьюнити, но нужно знать и о существовании других программ:
  - TensorFlow, OpenNN, PyTorch, Theano, …
  - Куча мелких библиотек для конкретных методов машинного обучения