Осенняя школа по информационным технологиям ОИЯИ 2022,
Лаборатория Информационных Технологий им. М.Г.Мещерякова,
ОИЯИ, Дубна 14-19 ноября

# Прикладные аспекты машинного обучения

## в экспериментах физики высоких энергий

**Ососков Геннадий Алексеевич**

Объединенный институт ядерных исследований
Лаборатория Информационных Технологий им. М.Г.Мещерякова
email: ososkov@jinr.ru
http://gososkov.ru

# Искусственный интеллект -
# – Машинное обучение -
# – Глубокое обучение

# Машинное обучение

**Машинное обучение (Machine Learning-ML), это когда компьютер не просто использует заранее написанный алгоритм, а сам обучается решению поставленной задачи на большой выборке данных.**

Три причины синхронного взрыва популярности ML в последние годы:

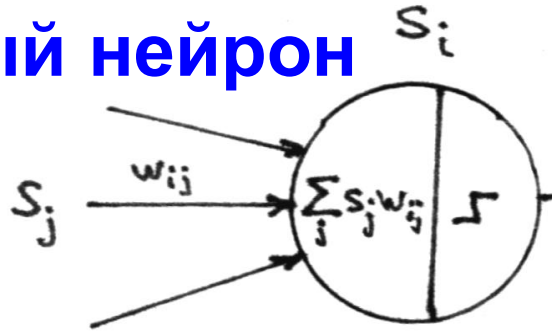
1. **Большие Данные**. Данных стало так много, что новые подходы были вызваны к жизни тем, что растущее разнообразие, как данных, так и возможных решений стало слишком велико для традиционных заранее запрограммированных систем.

2. Снижение стоимости **параллельных вычислений и памяти компьютеров**.

3. Новые алгоритмы **глубокого машинного обучения**.

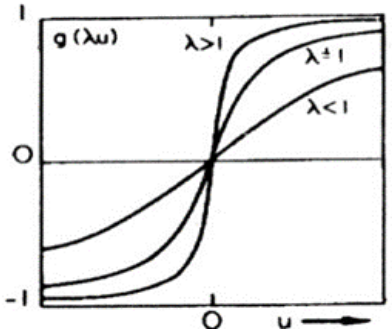
# Искусственный нейрон

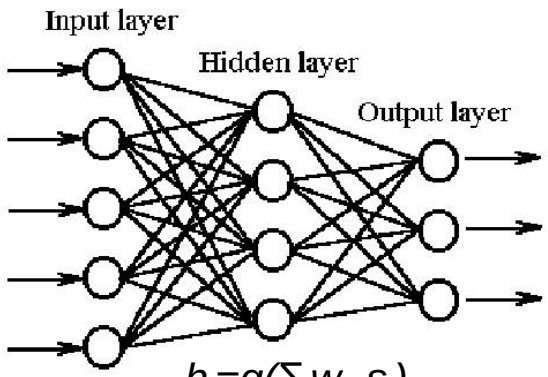
Connection between $i^{th}$ and $j^{th}$ neurons is characterized by synaptic weight $w_{ij}$




Выходной сигнал $h_i=g(\Sigma_j w_{ij}\, s_j)$

**Функция активации g(x).** Как обычно, это сигмоид g(x)=1/(1+exp(-λx)), но не только




$$y_j = f\left(\sum_k w_{kj} h_k\right)$$

$h_i=g(\Sigma_j w_{ij}\, s_j)$

# ИНС в экспериментальной физике

## 1. Прямоточная ИНС или многослойный персептрон (МСП)

**Обучение с учителем.**

**Цель обучения – определить веса так, чтобы обученная сеть решала задачу распознавания или классификации.**

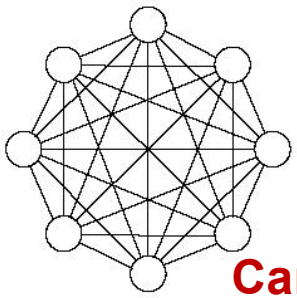
**Этапы применения МСП: 1.** Создать обучающую выборку как набор пар $(X_i,Z_i)$, где $X_i$ – входное значение, $Z_i$ – его целевое значение.

2. Обучить МСП, т.е. так подстроить веса $w_{ij}$, чтобы сеть определяла правильный выход для входов, не использованных при обучении

3. Протестировать МСП на тестовой выборке

4. Обученная сеть реализуется как программа или нейрочип для очень быстрого применения

## 2. Полносвязная ИНС (сеть Хопфилда)

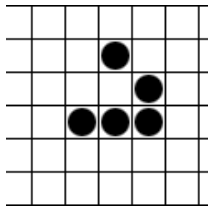


**Самообучение**

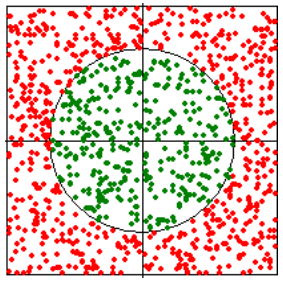
## 3. Клеточные автоматы

**Саморазвитие**

**можно рассматривать как сети с локальными связями**

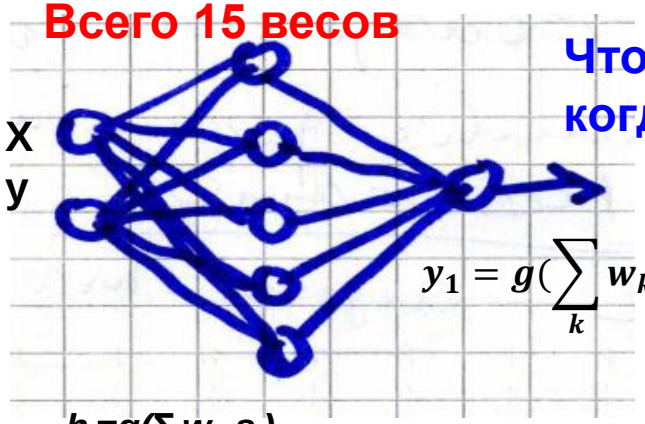


# Тайны обучения. Что внутри черного ящика ИНС?



**Простой пример:** **обучить сеть определять, где точка, - внутри круга или вне него.**
**О**бучающая выборка из 1000 троек чисел: на вход сети подают по 3 числа (X,Y, Z) : координаты точки X,Y и признак Z (Z =1 - внутри круга или Z =0 – вне его).
**Решение: ИНС с одним скрытым слоем из 5 нейронов, два входных и один выходной**

**Всего 15 весов**




$h_i = g(\Sigma_j w_{ij}\, s_j)$

**Чтобы обучить МСП применяют метод обратного распространения ошибки, когда минимизируют по всем весам функцию ошибки сети:**

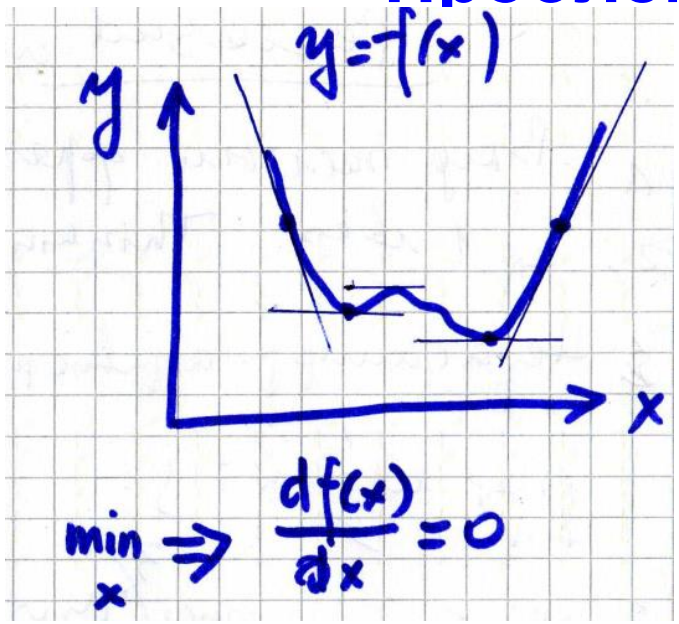
$$E=\Sigma_m \Sigma_{ij} (y_i^{(m)} - z_i^{(m)})^2 \rightarrow min_{\{wij\}}$$

Т.о. надо решать систему из 15 уравнений $\frac{\partial E}{\partial w_{ij}} = 0$
с 15 неизвестными значениями весов $w_{ij}$ $w_{jk}$ , для чего требуется дифференцируемость активационной функции g(x), определяющей выход каждого нейрона. Выбор сигмоидальной функции

$$g(x) = \frac{1}{1+e^{-\lambda x}}$$

обеспечивает к тому же **простое выражение и для ее производных**, $g'(x) = \lambda g(x)(1 - g(x))$ **, входящих в формулы для итеративной (по эпохам обучения) подстройки весов. Для весов выходного слоя имеем** $\Delta w_{kj}(t+1) = -\eta (y_j^t - z_j^t) g'(y_j^t) h_k^t$
**для скрытого слоя -** $\Delta w_{ik}(t+1) = -\eta \sum_j w_{kj} g'(y_j^t) g'(h_k^t) x_k^t$ **, где $\eta$ - параметр скорости обучения. Сеть считается обученной, когда в эпохе обучения $t$ максимальная ошибка обучения $E = max_{t,j} |y_{t,j} - z_{t,j}|$ уменьшится до заданной точности.**

# Проблемы минимизации функции ошибки сети




**Обычный метод решения - антиградиентный спуск Итеративные алгоритмы поиска минимума многомерной функции методом наискорейшего спуска требуют:**

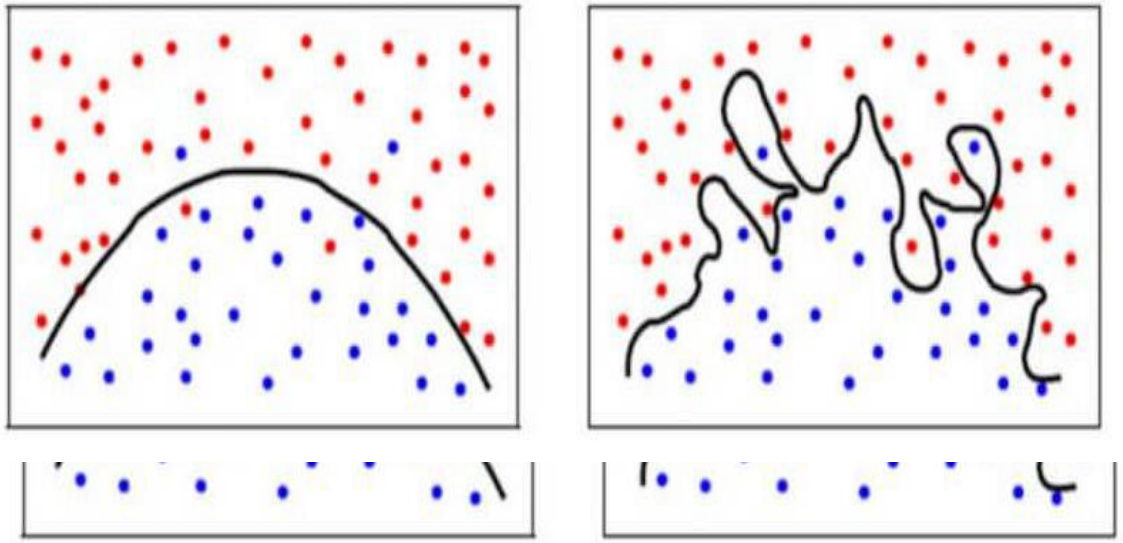
1. **Удачного выбора начального приближения.**

Интервал выбора начальных значений $w^0_{ij}$ ,$w^0_{jk}$ нельзя брать произвольным, он должен соответствовать задаче.

2. **Оптимального выбора шага по параметрам или скорости сходимости $\eta$**

**Проблема переобучения – overfitting**, когда ИНС с переизбытком скрытых нейронов выучивает весь набор данных, и ошибка обучения близка к 0, но знания модели не могут быть обобщены на новые данные, поэтому ошибка тестирования обычно очень высокая.

**Решение:** сравнивать значения ошибки сети при обучении и тестировании. Остановить обучение, когда эти значения разойдутся



**Правильная классификация** **overfitting**

# Почему ИНС так востребованы в физике

**Нейросети с их способностью к обучению и самообучению явились весьма эффективным средством решения многих экспериментальных задач, так что физики уже с 1988 года накопили большой опыт в применении ИНС во многих экспериментах для распознавания изображений, траекторий элементарных частиц и проверки физических гипотез.**

**В физике были особенно популярны, в частности, МСП. Именно физики написали в 80-х программный нейропакет JetNet и были одними из первых пользователей нейрочипов.**

**Причины:**

- **возможность монте-карловской генерации обучающей выборки любой требуемой длины на основе современной физической теории ;**
- **появление в то время на рынке нейрочипов, реализующих обученную нейросеть для ее применения, как сверхбыстрого триггера;**
- **появление удобных в использовании программ для конструирования МСП и реализации их обучения типа черновской программы TMVA – the Toolkit for Multivariate Data Analysis with ROOT.**

# Применение нейронных сетей в прикладных задачах ОИЯИ

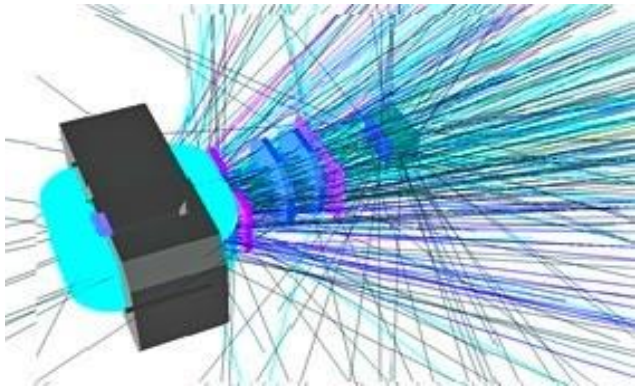
1. **Реконструкции событий по данным детекторов – ключевая проблема экспериментальной физики**
   - **Распознавание траекторий взаимодействующих частиц в экспериментах физики высоких энергий**
   - **Идентификация частиц по данным черенковского излучения**
   - **Прогнозирование работы аппаратуры по данным временных измерений**

**3. Прогнозирование загрязнения атмосферы тяжелыми металлами с использованием космических снимков**

**4. Разработка облачной платформы для выявления и профилактики заболеваний сельскохозяйственных растений**

**5. Анализ свойств тонких структур в распределениях продуктов ядерных реакций по массе**

# Данные, измеренных в экспериментах и постановки задач -1



Эксперимент BM@N. Стриповый GEM-детектор внутри магнита

Трековый детектор TPC внутри магнита MPD. Показано смоделированное событие от взаимодействия ионов золота, порождающее тысячи треков


Схема комплекса NICA с экспериментами MPD, SPD, BM@N

## Задачи: реконструкция событий по данным измерения в трековых и других детекторах

# Данные, измеренные в экспериментах, и постановки задач-2

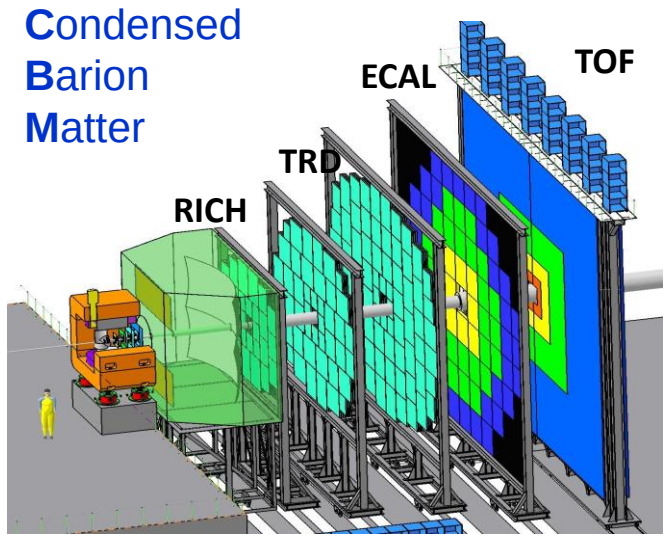



Схема установки CBM

**CBM эксперимент (Германия, GSI, будет запущен в 2023 году)**

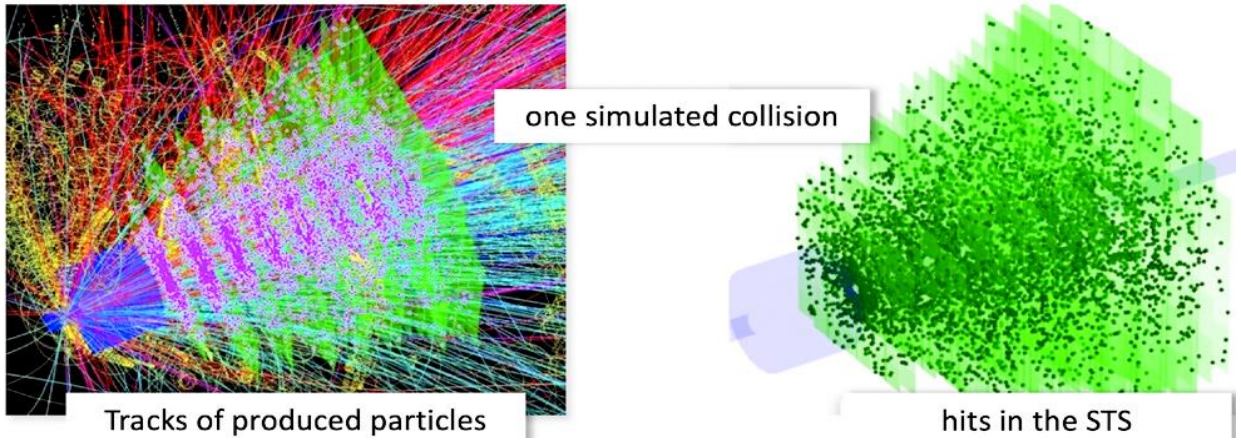
**Скорость передачи данных:**

**$10^7$ событий в сек,**

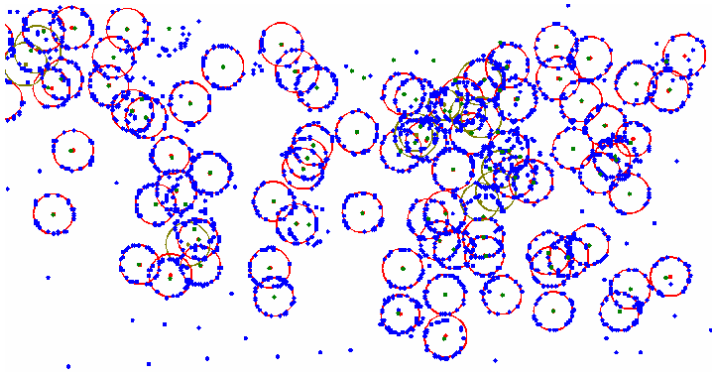
**~1000 треков на событие**

**~100 чисел на трек**

**Итого: 1 терабайт/сек!**




Вид модельного событий взаимодействия Au+Au в вершинном детекторе



Фрагмент данных фотодетектора. В среднем 1200 точек, образующих 75 колец

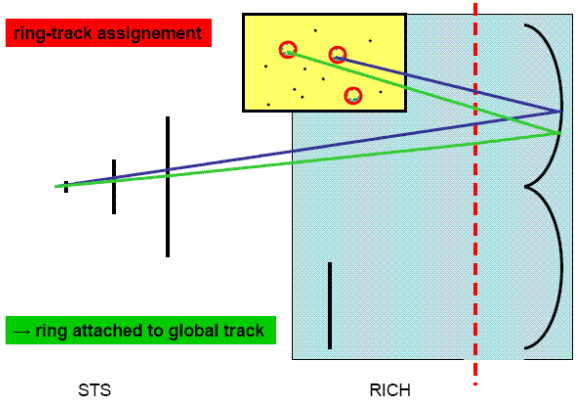
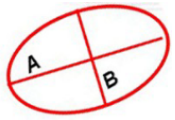



Схема детектораRICH черенковского излучения

**Проблемы CBM, решаемые методами машинного обучения: распознавание всех этих треков и колец RICH и оценка их параметров, с учетом их перекрытий, шумов и оптических искажений, ведущим к эллиптическим формам колец (подгонка эллипса), идентификация частиц, анализ спектров инвариантных масс короткоживущих частиц, поиск резонансов.**




**До 2015 года все эти задачи решались с помощью персептронов с одним скрытым слоем, нейросетей Хопфилда, фильтра Калмана, робастными методами и применением вейвлет-анализа.**

**Глубокое обучение ждало новых компьютерных технологий**

# Пример применения МСП в ФВЭ: идентификация частиц по данным детектора RICH

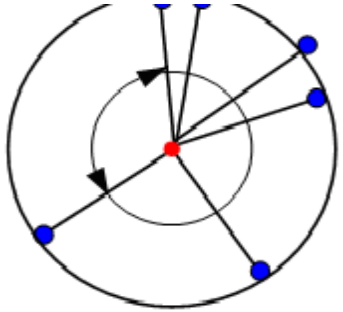
**Следовало выбрать наиболее значимые характеристики полученных колец, чтобы**

- **удалить ложно найденные кольца, - это делает первая нейросеть, после которой среди «хороших» колец вторая нейросеть выполняет**
- **идентификацию частиц**

**В итоге выбрали 10 характеристик:**

1. количество точек в найденном кольце
2. расстояние от центра кольца до ближайшего трека
3. сумма трех наибольших углов между соседними точкам
4. радиальная позиция на плоскости фотодетектора
5. $\chi^2$ эллиптической подгонки кольца
6. большая (A) и (8) малая (B) полуоси эллипса
7. угол поворота эллипса $\varphi$ относительно оси абсцисс
8. азимутальный угол трека
9. импульс трека

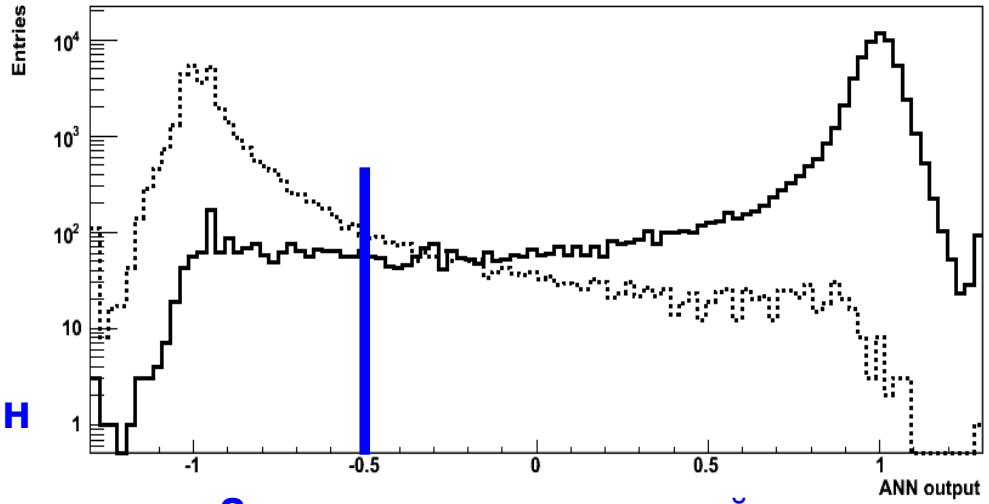
**Первые 5 признаков использовались для ввода в первую нейросеть**



**Обучение 1-й сети велось на Монте-Карло выборке из событий с 3000 электронов (+1) и 3000 пи-мезонов (-1) и показало 93%-й эффективность отбора хороших колец**

**Вторая ИНС имела все 9 входных нейронов, 20 скрытых и один выходной нейрон, обучалась на выборке из хороших колец**

**Порог -0.5 для выхода сети обеспечил идентификацию электронов с приемлемым уровнем подавления пионов**



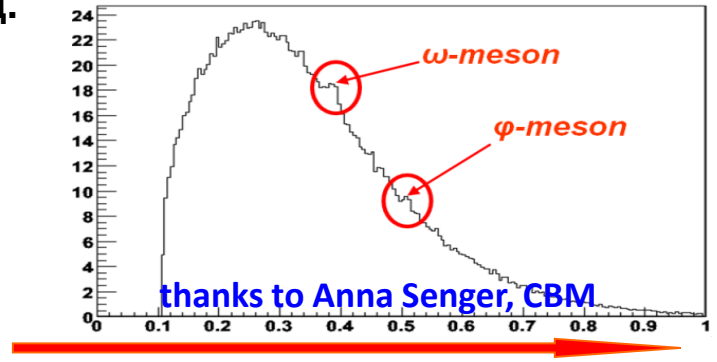

**Значения выходного нейрона Для идетификации частиц**

# Основные этапы анализа данных в экспериментах ФВЭ

- Сбор данных со многих каналов на многих субдетекторах (млн/сек)
- Решить, считывать или отбросить событие (триггеры разных уровней)
- Реконструировать событие (собрать всю информацию)
- Отправить данные на хранение
- Анализировать их
  - корректировка данных с учетом искажений детектора: калибровка, алайнмент
  - нахождение хитов, трекинг, поиск вершин, распознавание черенковских колец,
  - удаление ложных объектов (фейков)
  - алгоритмы анализа физиков-пользователей
  - уменьшение объема данных

Применяемые методы машинного обучения
Преобразования Хафа,
клеточные автоматы,
фильтр Калмана,
искусственные нейронные сети,
робастное оценивание,
вейвлет-анализ и т.д.

- Детальное моделирование всех процессов эксперимента
  - взаимодействия пучка с мишенью или налетающей частицей
  - рассеяния при прохождении частиц через детекторы
  - искажений при оцифровке и т. д.
- Сравнение теории и физических параметров, полученных в эксперименте
  - анализ спектров инвариантных масс короткоживущих частиц резонансов



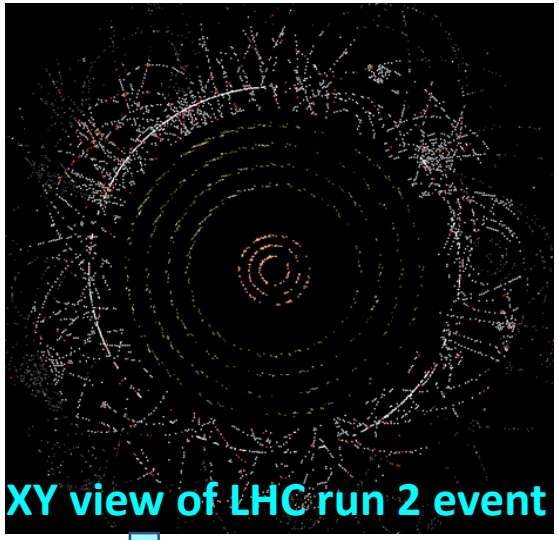

- Использовать современные средства компьютинга для достижения наивысшей скорости и масштабируемости обработки

Неизбежность создания всемирной интернет-сети распределенных вычислений (Worldwide LHC Computing Grid -WLCG)
Parallel programming of optimized algorithms Grid-cloud technologies which changed considerably HEP data processing concept
See *Scientific data management in the coming decade* https://dl.acm.org/doi/10.1145/1107499.1107503

# Восстановление треков – ключевая проблема реконструкции событий ФВЭ

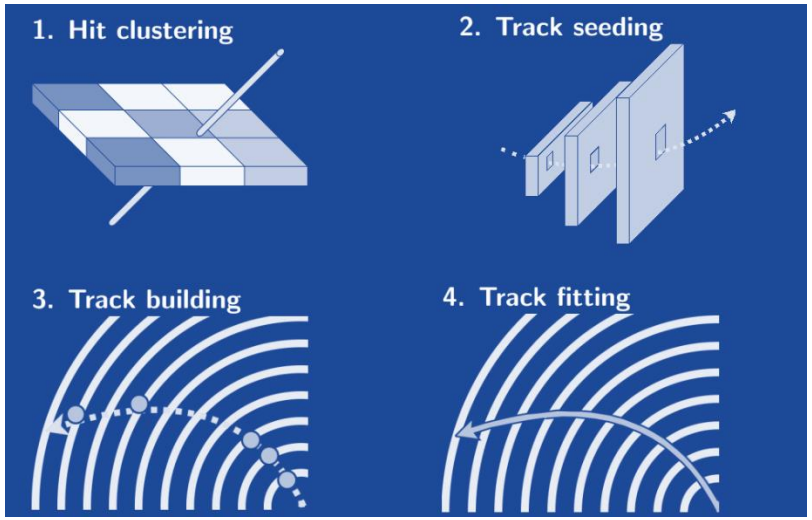
**Реконструкция должна определять параметры вершин и траекторий частиц (треков) для каждого события. Традиционно алгоритмы трекинга, основанные на комбинаторном фильтре Калмана, с большим успехом использовались в экспериментах ФВЭ в течение многих лет.**

## Что такое трекинг?





**Трекинг или распознавание треков - это процесс восстановления траекторий частиц в детекторе ФВЭ путем прослеживания и соединения точек- хитов (*хит* – это реконструированный отклик детектора), которые каждая частица оставляет, проходя через плоскости детектора.**

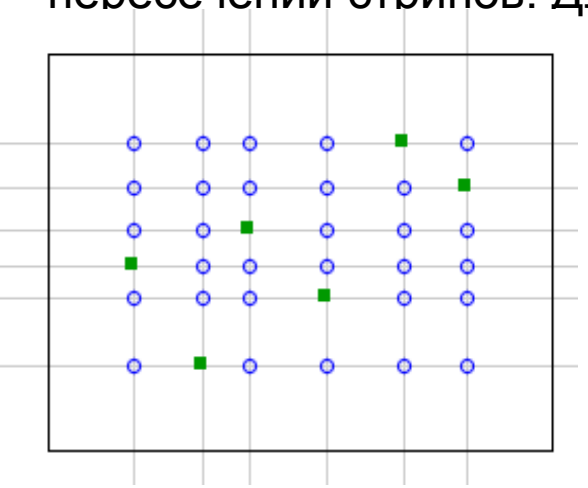



**Процедура трекинга включает в себя фазы: (1) получения хитов (hit clustering), (2) построения треков-кандидатов - наборов хитов с вычисленными параметрами (*англ. seeds*), (3) прослеживания треков и (4) их подгонки уравнением движения частицы в магнитном поле.**

**Главная проблема современного трекинга- высокая светимость пучков ускорителей, т.е. мегагерцовый темп поступления данных и банчевая структура пучка**

# Проблемы трекинга для GEM детектора BM@N

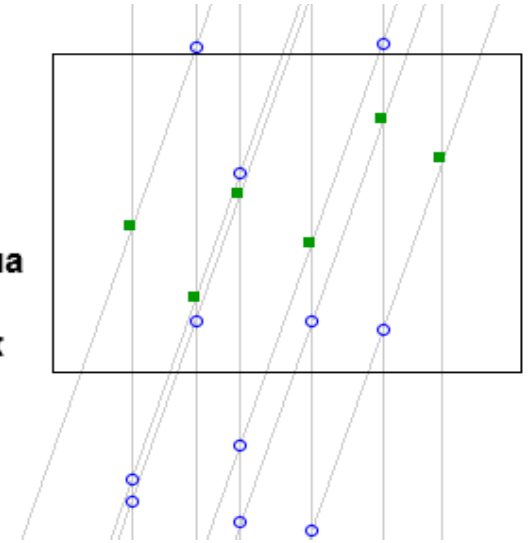
Главная трудность, вызванная спецификой **GEM етектора** – появление ложных отсчетов из-за лишних пересечений стрипов. Для **n** истинных хитов имеем **$n^2$- n фейков!**

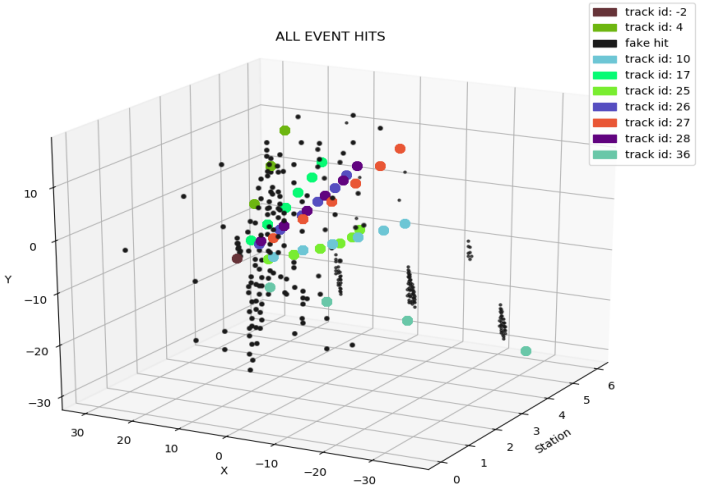


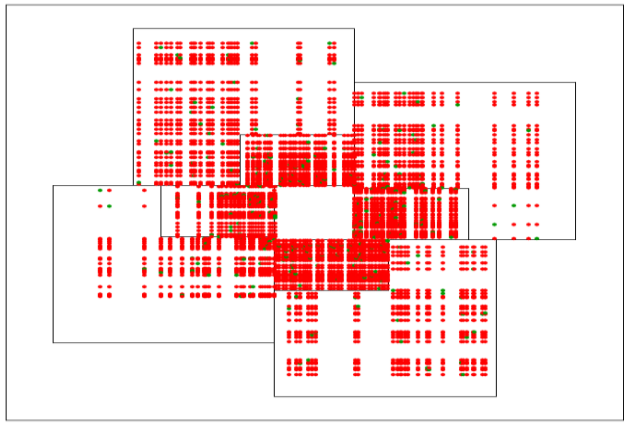
- истинный хит

- ложное пересечение

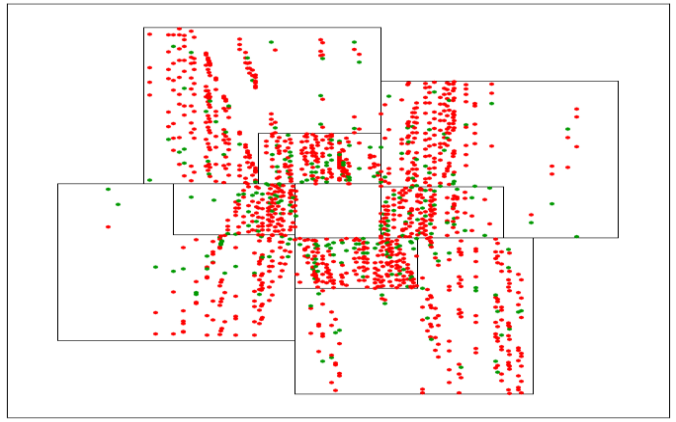
**Можно уменьшить количество фейков – повернуть слой стрипов на маленький угол (5-15 градусов) по отношению к другому слою**





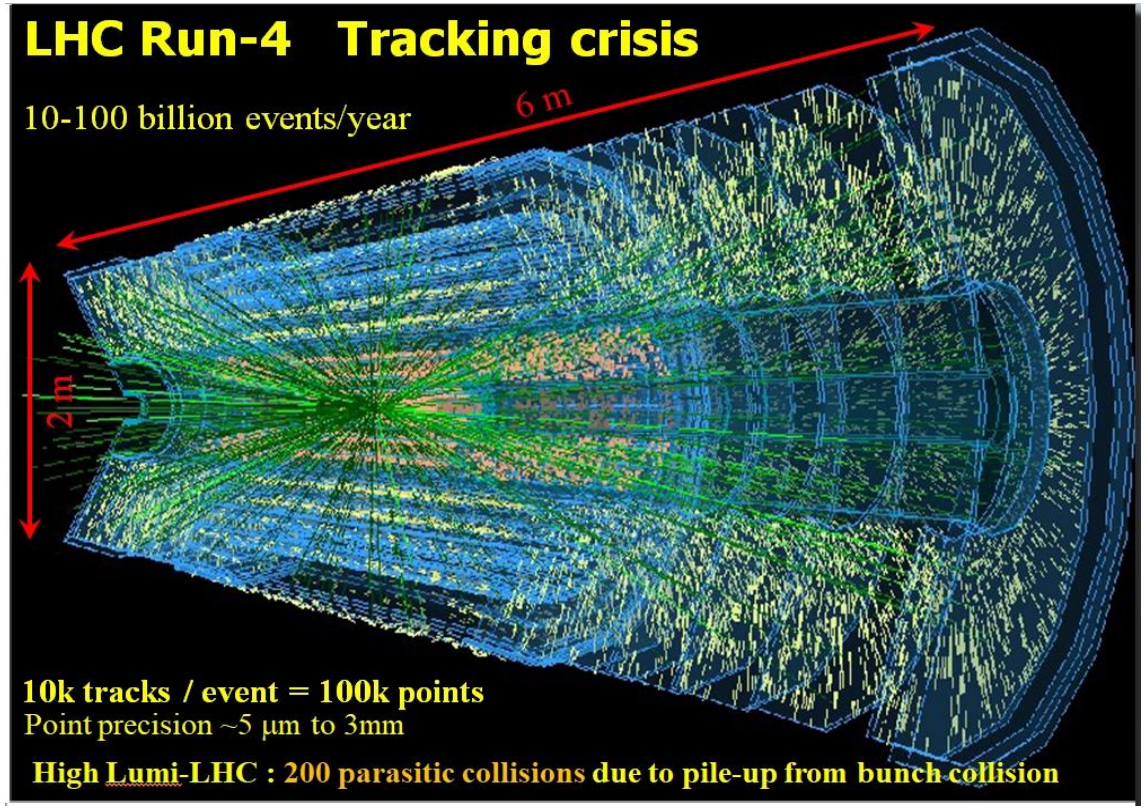



**Наклон 15° избавляет нас от трети фейков, но большая их часть остаётся.**

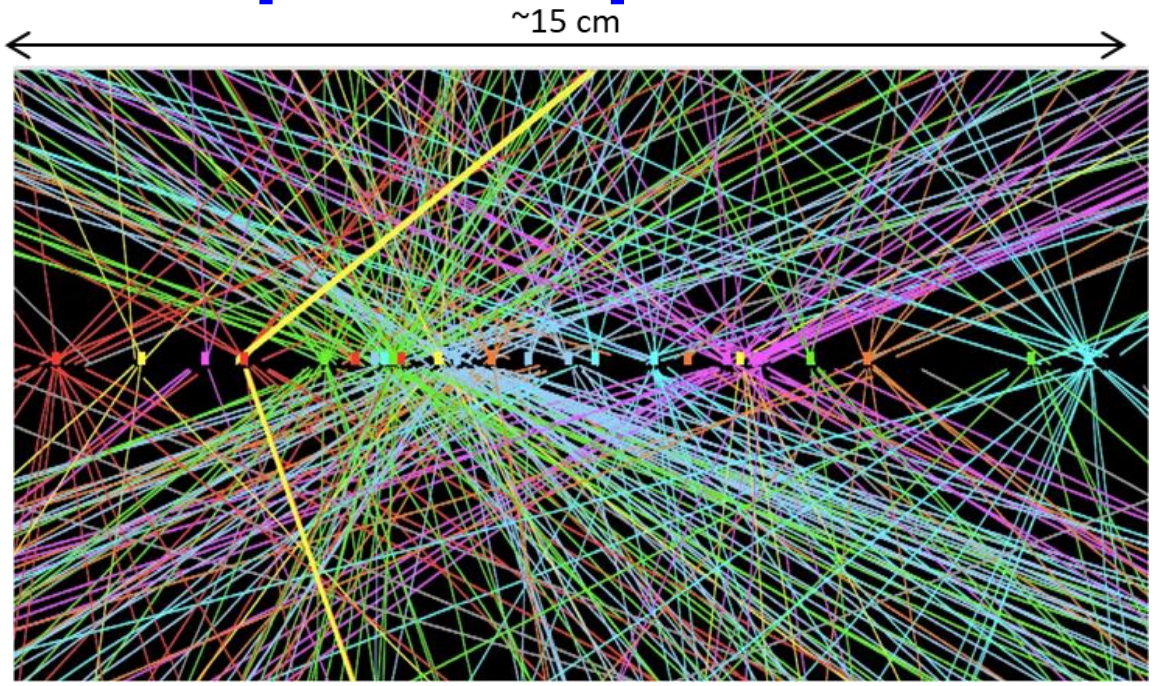


**Эти две проблемы, - наличие фейковых засорений данных и, главное, сверхвысокий темп их поступления из-за высокой светимости неизбежно требуют разработки новых методов трекинга с использованием глубоких нейронных сетей**

# Кризис трекинга




Модельное события в HL-LHC




Столкновение групп ускоренных частиц

**Для достижения большой светимости частицы ускоряются не по отдельности, группами - банчами (англ. bunch) из-за чего моменты столкновений происходят так близко, что треки событий сильно перекрываются**

**Текущая ситуация: 20 паразитных столкновений, High Lumi-LHC: 200 паразитных столкновений.**

**Таким образом, реконструкция треков частиц в плотных средах, таких как детекторы БАК высокой светимости (HL-LHC) и NICA, представляет собой сложную проблему распознавания образов для решения которой необходимо развитие новых алгоритмов глубокого трекинга и их распараллеливание на суперкомпьютерах.**

# Предпосылки и неизбежность появления глубоких нейросетей

**Новые технологические достижения:** Суперкомпьютеры, карты GPU, параллелизм обработки, виртуализация и облачные технологии.

**Биологические открытия о мозге:** многослойность архитектуры, непосредственное восприятие пространственных объектов.

**Эпоха Больших Данных**. Достигнут уже экзабайтный уровень потоков получаемых данных в современных и, особенно, планируемых экспериментах в физике высоких энергий и других научных и экономических областях

Решаемые задачи становятся все более сложными и, чтобы простая нейросеть могла их адекватно моделировать, принимать на вход 2D и 3D объекты, ее архитектура должна становиться все более глубокой, путем добавления больше скрытых слоев, усложнения структуры и параметризации.

Однако простое увеличение числа скрытых слоев приводило к экспоненциальному росту межнейронных соединений и проблеме с «проклятьем размерности», к переобучению сети или застреванию её минимизируемой ошибки в локальном минимуме.

**Возникла необходимость в глубоких нейросетях и новых методах их обучения, обеспечивающих высокую эффективность работы, параллельность и масштабируемость реализации**

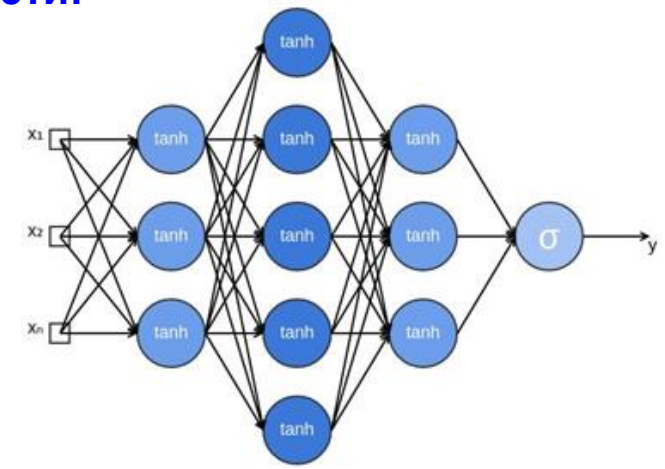
# Различные типы глубоких нейросетей
## и проблемы их обучения

# 1.Многослойная прямоточная нейронная сеть

**Задана обучающая выборка ($X_i$,$Z_i$). Инициируем веса $w_{ij}$, выбираем активационную функцию $g(x)$ (обычно сигмоид $\sigma(x)$) и обучаем сеть.**

**Прежний опыт: чтобы обучить сеть применяют метод обратного распространения ошибки, когда методом градиентного спуска минимизируют по всем весам квадратичную функцию ошибки сети:**

$$E=\Sigma_m\Sigma_{ij}(y_i^{(m)} - z_i^{(m)})^2 \rightarrow min_{\{wij\}}$$




## Возникающие проблемы:

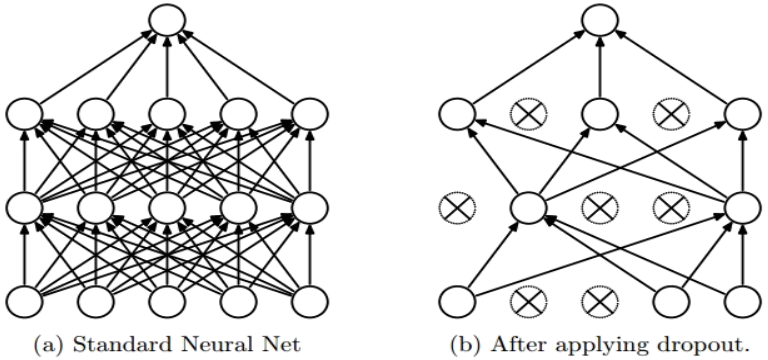
1) проклятье размерности
2) переобучение
3) застревание E в ложном минимуме
4) затухающий или взрывной градиент

## Как их решают

**1. Уменьшение размерности сети, два метода:**

(1) Кардинальное сжатие входных данных, т.к. обычно они сильно скоррелированы. Эффективен нейросетевой метод - автоэнкодер,

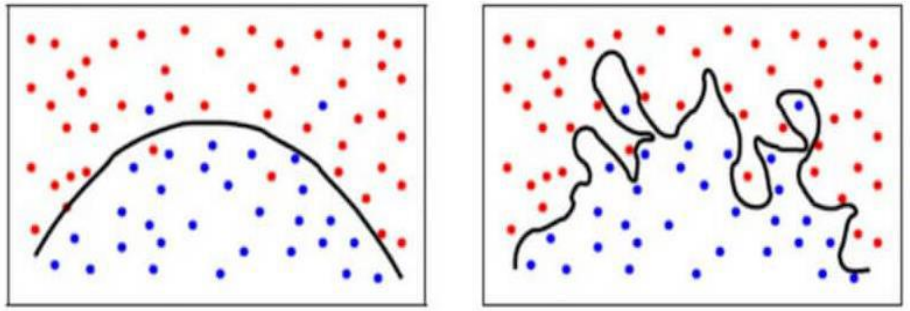
(2) Случайное прореживание нейронов – dropout, когда каждый вес обнуляется с вероятностью p либо умножается на 1/p с вероятностью 1-p. Dropout используют только при обучении сети.


(a) Standard Neural Net

(b) After applying dropout.

# 2) Переобучение глубокой нейросети, как защититься

**Переобучение (overfitting)** — это излишне точное соответствие нейронной сети конкретному набору обучающих примеров, при котором сеть теряет способность к обобщению и не работает на тестовых примерах.



Правильная классификация overfitting

Иногда эта проблема возникает, если слишком долго обучать сеть на одних и тех же данных, так что сеть начнет не учиться на данных, а запоминать их шумы. **Простой выход – ранняя остановка обучения** на той эпохе, когда функция потерь ИНС (**loss)** при верификации уйдет вверх от значений loss при обучении

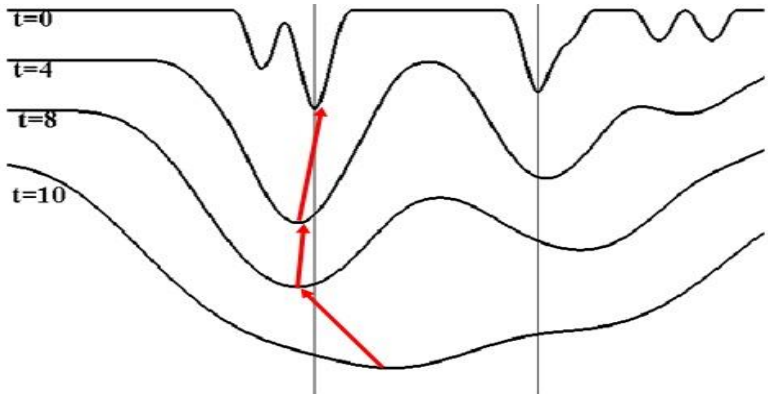
Кроме того, переобучение бывает связано с усложнением модели из-за переизбытка скрытых нейронов при большой глубине сети. Тогда одно из эффективных средств - тот же **dropout**.

Более общим является **метод регуляризации**, ограничивающий реакцию нейросети на влияние шумов добавлением штрафного члена в функцию ошибки сети

$$E(w) = E_0(w) + \frac{1}{2}\lambda \sum_i w_i^2.$$

Коэфициент $\lambda$ не должен быть большим и обычно $\lambda$=0.001.

# 3) Ложные минимумы функции ошибки сети




Для решения этой проблемы используют **метод отжига.**

Более эффективен **Стохастический градиентный спуск (Stochastic Gradient Descent – SGD),** при котором требуется только один проход по обучающим данным, когда значение градиента аппроксимируются градиентом функции ошибки, вычисленном **только на одном элементе обучения**, в то время как при обычном  градиентном спуске на каждой итерации обучающая выборка просматривается целиком, и только после этого изменяется вес. Поэтому для больших массивов данных **SGD работает много быстрее**.

**Выбор точки обучения в SGD происходит случайно**, но попеременно из разных классов, что также повышает вероятность выхода из локального минимума.

Все эти возможности включает метод **ADAM** (**Adaptive Moment Estimation**)**,** который требует только градиентов первого порядка и реализует SGD особенно эффективно путем вычисления адаптивной скорости обучения для каждого параметра, оптимизации шага, а чтобы не «промахнуться» мимо искомого минимума сохраняет экспоненциально убывающее среднее значение прошлых градиентов $\Delta w_i = \eta \cdot Gradw + \alpha \cdot \Delta w_{i-1}$, аналогичное моменту инерции

# 4) Затухающий градиент и паралич нейросети

Проблема затухающего градиента неизбежно проявляется в многослойных нейросетях, особенно в случае сигмоидной активационной функции $\boldsymbol{\sigma(x) = \frac{1}{1+e^{-\mu x}}}$

(заметим: ***σ'(x)≤¼; 0<x<1***). При BackProp обучении ошибка с выходов отправляется на вход, распределяется по всем весам и отправляется дальше по сети к входу. При этом градиент (производная ошибки) — прогоняется через нейросеть обратно с многократным перемножением ***σ'(x)***. Когда в нейросети много слоев, градиент в слоях близких к входу становится исчезающе малым и **веса практически перестанут изменяться.** Происходит «**паралич сети**».

Противоположная картина называется «**взрывной рост градиента**», это случается, если инициализировать веса слишком большими значениями.

Для избежания затухания или взрывного роста градиента нужно

1. **грамотно выбирать функцию активации;**
2. **использовать правильную инициализацию настраиваемых параметров нейросети;**
3. **выбирать функцию ошибки сети, адекватную решаемой задаче.**

# Выбор функции активации

Из-за указанных недостатков **сигмоидальной активационной функции** $\sigma(x) = \frac{1}{1+e^{-\mu x}}$

применяют также

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}.$$

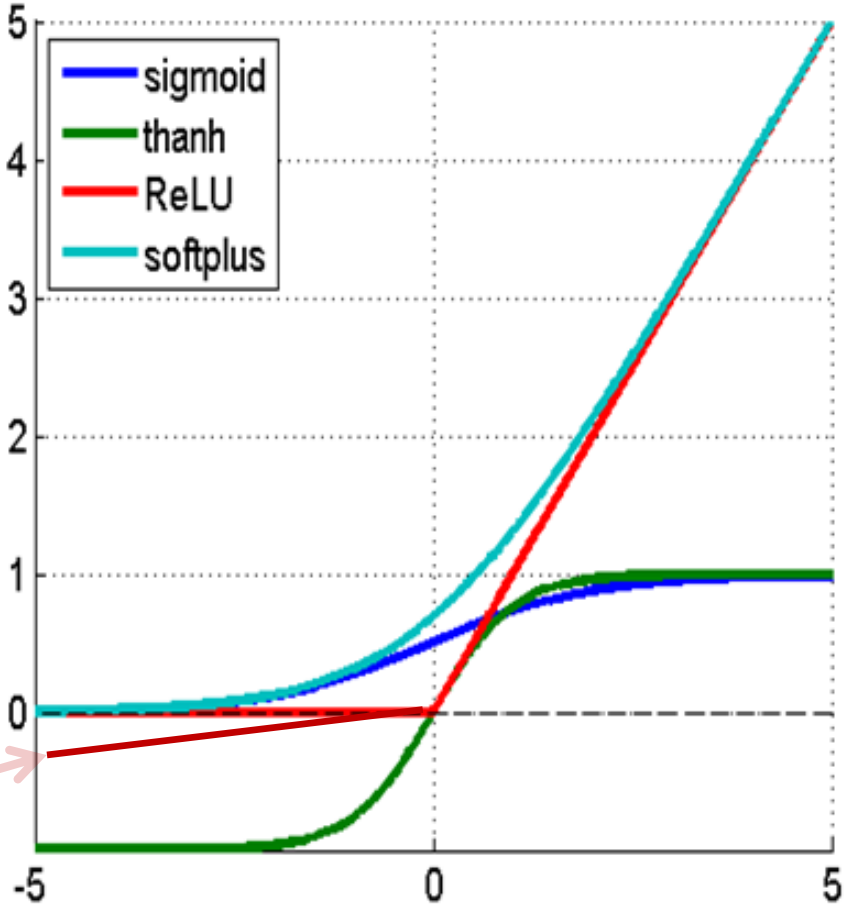
**1. Гиперболический тангенс tanh активация** меняется в интервале от –1 до 1. Подобно сигмоиде, tanh может насыщаться, зато его выход центрирован относительно нуля, что предпочтительней.

2. Функция активации под названием **«выпрямитель» (Rectified Linear Unit – ReLU)** с формулой *f(x) = max(0, x).* ReLU наиболее популярен, т.к. не подвержен насыщению и быстро вычисляется, повышая скорость сходимости. стохастического градиентного спуска в несколько раз. Однако при очень большом градиенте ReLU может так обновить вес, что нейрон никогда больше не активируется, - «умирает». Поэтому появились различные модификации ReLU:

**3. ReLU с «утечкой» (leaky ReLU, LReLU)**
с формулой $f(x) = \alpha x$ при $x < 0$ и $f(x) = x$ при $x \geq 0$,
где α – малая константа.

**4. Softplus** является гладкой аппроксимацией ReLU с формулой

$$f(x) = \ln(1+e^{x})$$



# Инициализация значений весов нейросети

Очевидно, что начальные веса не должны быть все нулевыми или просто одинаковыми, нельзя их брать слишком большими, чтобы избежать взрывного роста градиента. Поэтому обычно рекомендуют в качестве **начальных значений весов** брать случайные значения, **равномерно распределенные в интервале (-0.5,+0.5).**

Однако есть проверенные рекомендации, значительно ускоряющие обучение при использовании активационных функций ReLu и LRelu.

Так **К. Глорот** рекомендует вычислять начальные значения весов по формуле

$$W \sim U\left[-\frac{\sqrt{\{6\}}}{\sqrt{\{n_j + n_{\{j+1\}}\}}}, \frac{\sqrt{\{6\}}}{\sqrt{\{n_j + n_{\{j+1\}}\}}}\right],$$

где ***U[-a,a)*** - равномерное распределение в интервале **(-a,a)**, **n** - размер предыдущего слоя (количество столбцов в матрице весов ***W***), ***j*** –индекс предыдущего слоя.

В другой работе **К.Хе** рекомендует инициализировать веса слоя $n_j$ величинами, нормально распределенными со средним ***0*** и дисперсией $2/n_j$.

В последнее время появился еще один весьма радикальный метод ускорения обучения на очень больших датасетах – **batch-normalization**

https://habr.com/ru/post/309302/

# Функция потерь cross-entropy loss

До сих пор мы использовали минимизацию функцию ошибки сети (loss function) квадратичного вида $E=\Sigma_m\Sigma_{ij}(y_i^{(m)} - z_i^{(m)})^2$, которая согласно матстатистике соответствует случаю, когда ошибки, т.е. разбросы функции ***E*** распределены по нормальному закону, что далеко не всегда верно. **Для задач классификации**, например, характерен результат в виде вероятностей того, что предъявленный сети объект принадлежит тому или иному классу, т.е. нам надо оценить вероятность того, что $y_i^{(m)}$ отличен от $z_i^{(m)}$ . В этом случае для оценки того насколько близко прогнозируемое распределение p(x) к истинному распределению q(x) используют другую функцию потерь называемую **перекрестной энтропией (cross-entropy)**

$$H(p,q) = -\sum_x p(x)\,\log q(x).$$

Эта формула обобщается и на случай нескольких классов естественным образом:

$$\text{loss} = -\frac{1}{q}\sum_{i=1}^{q}\sum_{j=1}^{l} y_{ij}\log a_{ij}$$

, здесь ***q*** – число элементов в выборке, *l* – число классов, $a_{ij}$ – ответ (вероятность) алгоритма на ***i***-м объекте на вопрос принадлежности его к ***j***-му классу, $y_{ij}$**=1** если ***i***-й объект принадлежит ***j***-му классу, в противном случае $y_{ij}$**=0**.

# 3. Сверточные нейросети для распознавания изображений

**Мотивация: 1. Время обучения многослойных персептронов очень велико**

**2. Прямое применение регулярных ИНС к распознаванию изображений бесполезно из-за двух основных факторов: (i) входное 2D-изображение в виде сканированного 1D-вектора означает потерю топологии пространства изображения; (ii) полносвязность ИНС, где каждый нейрон полностью связан со всеми нейронами предыдущего слоя, слишком расточительна из-за проклятия размерности, кроме того, огромное количество параметров быстро приводит к переобучению**

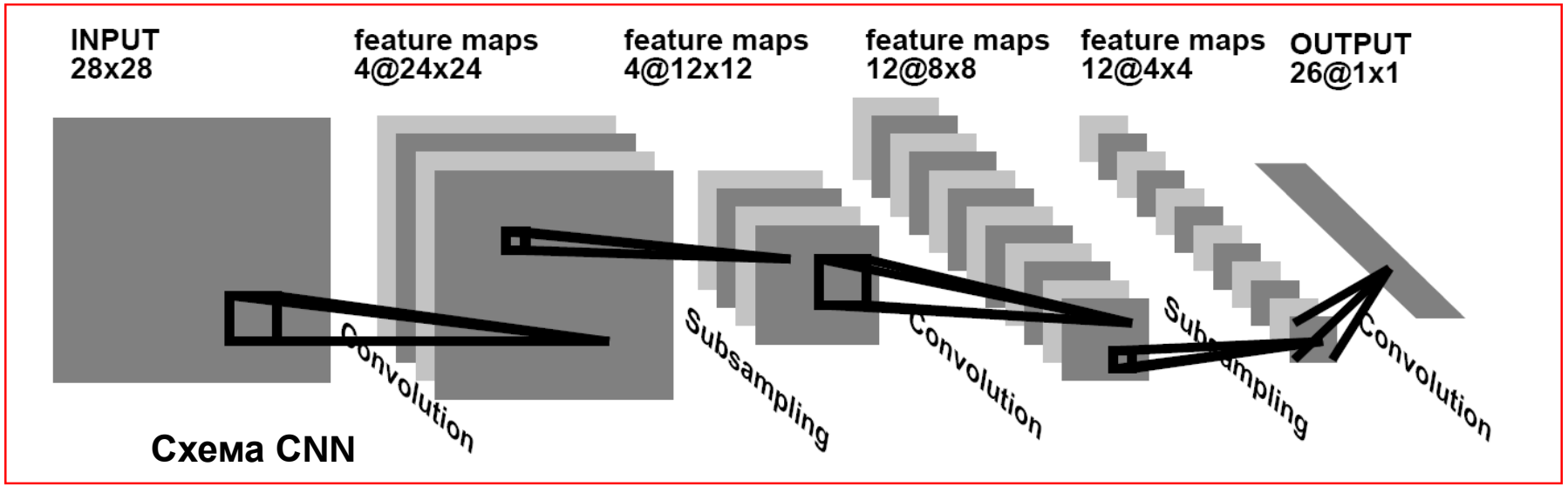


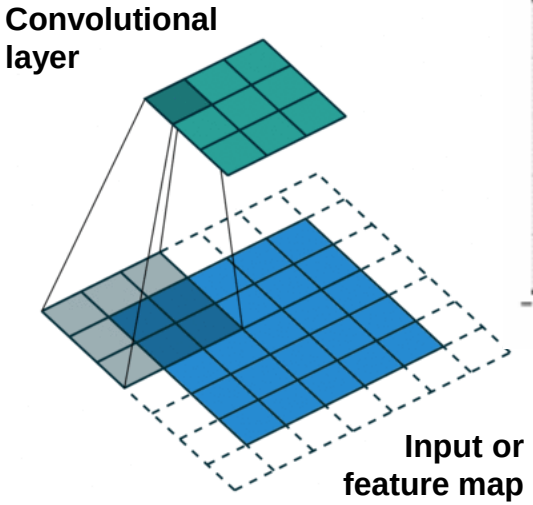

Схема CNN

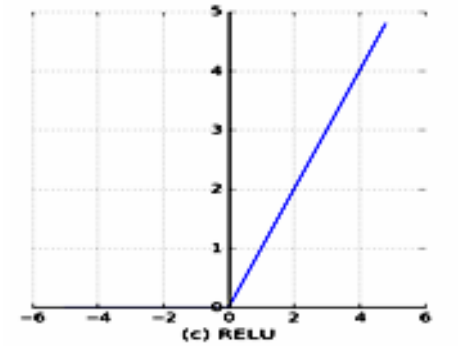
Поясняющая аналогия

**Вместо этого, сверточные сети - Convolutional Neural Networks (CNN) принимают на вход двумерные цветные изображения, а нейроны в слое CNN связаны только с малой областью предыдущего слоя**

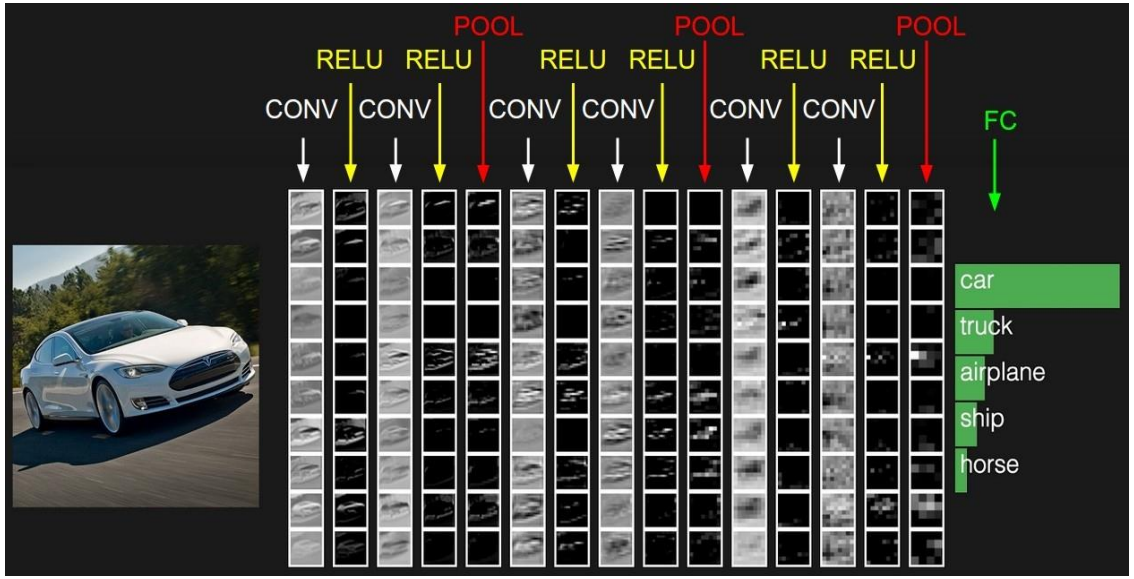
# Основы архитектуры CNN

Архитектура CNN: последовательность слоев, каждый слой преобразует один набор активаций в другой через **фильтр-свертку с ядром.** Основные типы слоев для CNN: **Сверточный слой, Слой объединения** (pooling) и **скрытый слой персептрона с обучением backprop**). Также существуют слои **RELU** (rectified linear unit), выполняющие операцию **max (0, x) .**

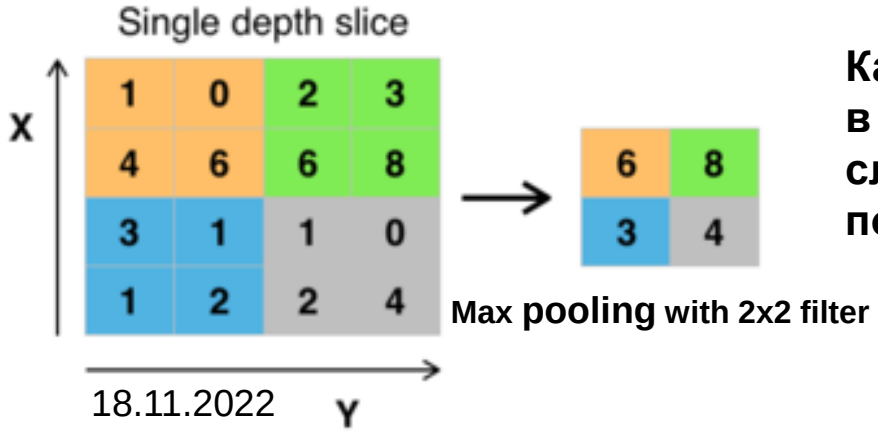



Activation: ReLU




Example of classifying by CNN

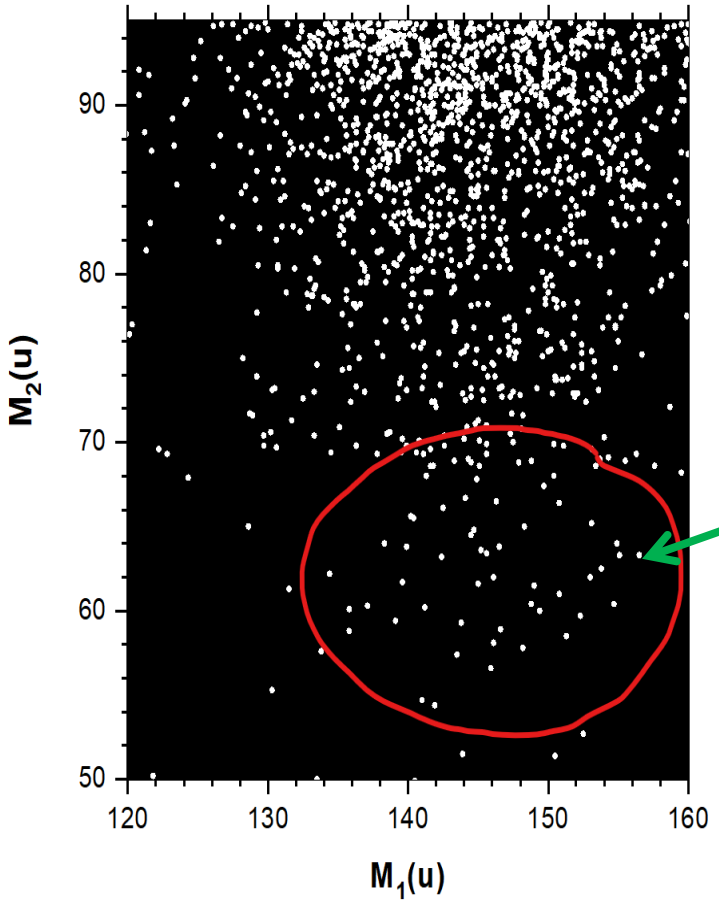


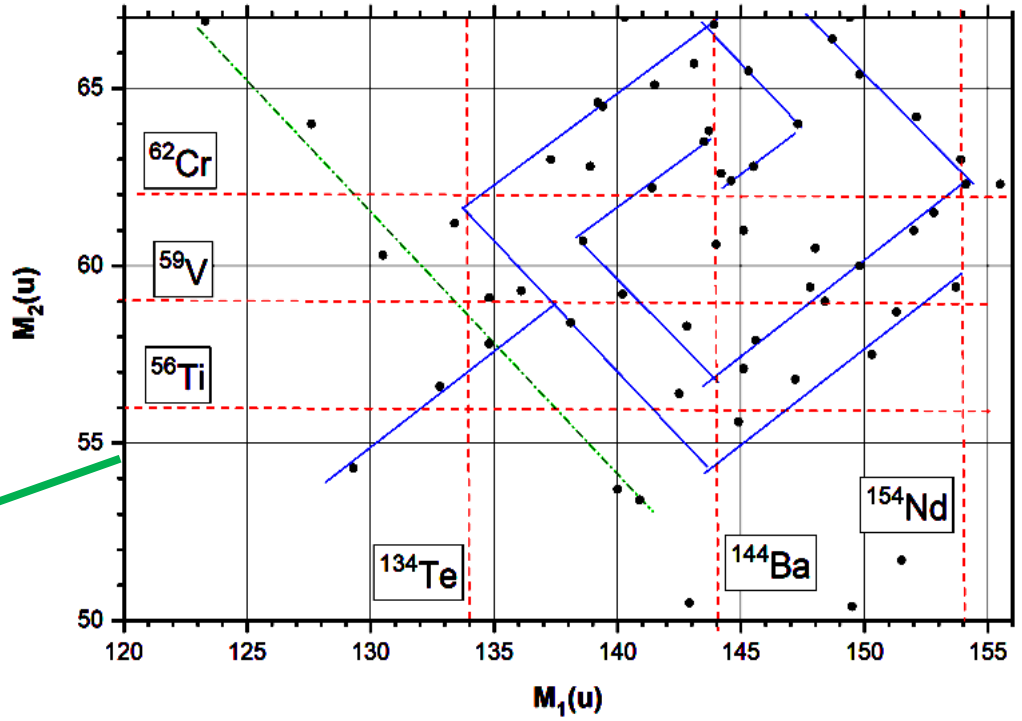

Max **pooling** with 2x2 filter

**Каждый слой принимает входные 3D данные (x, y, RGB) и преобразует их в выходные 3D данные. Чтобы построить все фильтры сверточных слоев, CNN должна быть обучена на помеченных изображениях с помощью метода обратного распространения ошибки.**

# Пример применения сверточного нейроклассификатора для анализа данных ядерного распада калифорния $^{252}Cf$


**Корреляционная диаграмма масс спонтанного деления калифорния, предоставленная группой проф. Пяткова**




**Замечание:** красные пунктирные линии обозначают массы магических ядер

**Задача: доказать, что эта "ядерная роза" не является случайным набором точек**

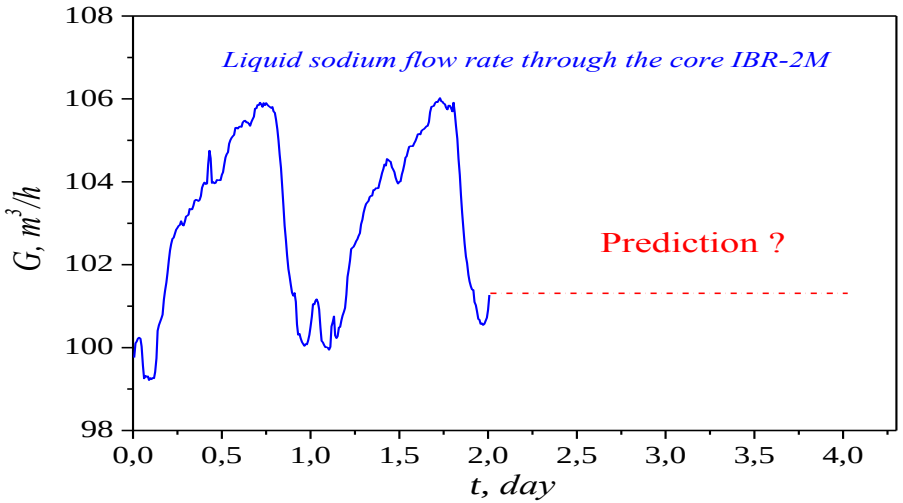
**Постановка задачи:**

- разработать численную модель "розы"
- создать набор размеченных данных для обучения нейроклассификатора
- обучить нейроклассификатор отличать набор шумовых точек от "розы"
- убедиться, что вероятность неверного предсказания "розы" среди набора шумовых точек пренебрежимо мала

# Еще примеры применения глубокоо обучения в ОИЯИ

## 3. Импульсный реактор ИБР-2

**Временной ряд для оперативного прогноза работы реактора**




**Расход жидкого натрия для охлаждения ИБР-2М**

**Задача: спрогнозировать расход жидкого натрия через активную зону ИБР-2М во время работы реактора**

**Решение: применение нелинейной авторегрессионной нейронной сети (NAR) с лагом n-44 для обучения и включения обратной связи для прогнозирования.**

## 4. Нейтронный активационный анализ

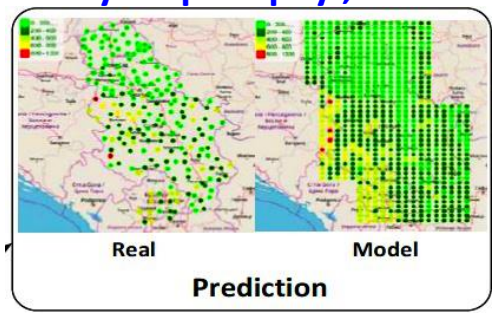
**Международная программа мониторинга и оценки воздействия загрязнителей воздуха на растительность (ICP Vegetation).**

Мхи - идеальные поглотители загрязненного воздуха. Волонтеры собирают мхи в определенных местах, отправляют их в ОИЯИ для проведения нейтронно-активационного анализа, чтобы выяснить, какие вредные вещества загрязняют воздух в данной местности.

**Задача: экстраполировать эти данные из дискретных точек сбора мхов на окружающую местность для каждого из загрязнителей с учетом особенностей окружающей среды, таких как, направления ветра и водостоков**

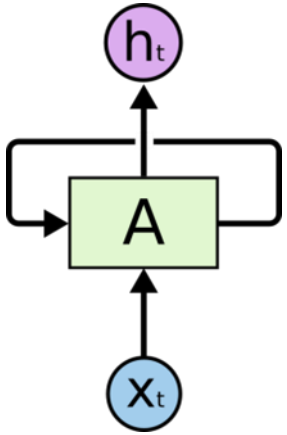
**Решение: Система управления данными (DMS) разработана на облачной платформе ОИЯИ и предоставляет участникам современную единую систему сбора, анализа и обработки данных биологического мониторинга. DMS позволяет организовать гибкий сбор данных с учетом неоднородности исходных данных, их автоматическую проверку ,**

**вычисляет его статистические параметры, геоиндексы и так далее. Сейчас в DMS представлено более 6000 мест отбора проб из 47 регионов разных стран мира**



# 3. Изменяющийся мир требует рекуррентных нейросетей

- **В жизни мы имеем дело с объектами, изменяющимися во времени, и наш мозг, работая, всегда исходит из знания того, что уже было.**
- **Однако обычные нейросети с одним скрытым слоем, глубокие сети и даже такие продвинутые сети, как сверточные, предназначены для работы со статическими объектами. Никакое обучение не поможет традиционной нейросети смоделировать будущее состояние объекта.**
- **Для описания динамического объекта нейронная сеть должна обладать некоей памятью, чтобы исходя не только из настоящего его состояния, но и из прошлого, нейросеть могла бы моделировать его последующее состояние.**




**Эту проблему решает семейство новых глубоких нейросетей, называемых рекуррентными (Recurrent Neural Networks - RNN). Они содержат в себе обратные связи, позволяющие сохранять прошлую информацию.**
**Так модуль *A* принимает входное значение $x_t$ и возвращает значение $h_t$.**
**Внутри этой ячейки обычная нейросеть с одним скрытым слоем.**

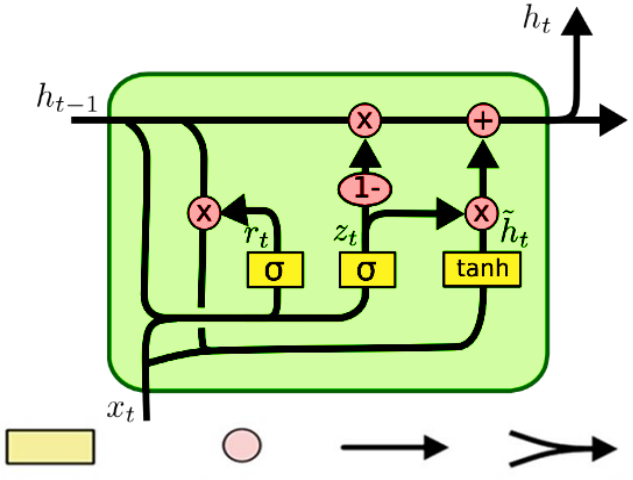
**Чтобы RNN могла связывать предыдущую информацию с текущей задачей, требуется ее усовершенствование до LSTM сети (Long short-term memory - долгая краткосрочная память) с четырьмя взаимодействующих слоями, позволяющими удалять информацию из состояния ячейки с помощью фильтров - gates. Фильтры состоят из слоя сигмоидальной нейронной сети и операции поточечного умножения и позволяют обрабатывать информацию на основании задаваемых условий.**

# Управляемый рекуррентный модуль GRU (Gated Recurrent Unit) как вариант LSTM сети

**Управляемый рекуррентный модуль GRU (Gated Recurrent Unit) является упрощенной версией LSTM.**

**Основной компонент GRU – это своеобразная память сети –линия, проходящая по верхней части схемы. В GRU есть три фильтра, позволяющих защищать и контролировать состояние ячейки.**

1. **Слой фильтра обновление (update gate) объединяет вход и забывание, т.е. определяет какую часть входной информации выбросить.**
2. **Что сохранить: слой входного фильтра + tanh-слой**
3. **Что получать на выходе.**




$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

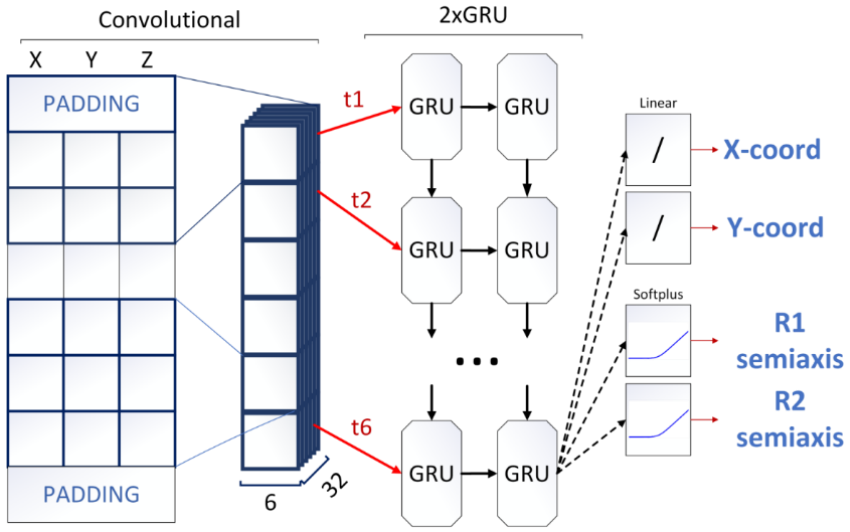
**С помощью именно такой GRU нейросети нам удалось решить задачу восстановления траекторий элементарных частиц в детекторе GEM эксперимента BM@N в ОИЯИ.**
**GRU сеть структурно проще и поэтому обучается в полтора раза быстрее LSTM, практически не уступая в эффективности на таком классе задач.**

# Локальный и глобальный подходы к трекингу

## Два подхода к реализации «глубокого трекинга»

**1. Локальный трекинг, когда треки восстанавливаются один за другим**, как в алгоритме фильтра Калмана.
**Недостатки:** медленно, нет возможности увидеть зависимость между отдельными треками или группами треков и такие явления как вторичные вершины, необходимость реализации специального этапа для поиска вторичной вершины.
**2. Глобальный трекинг, при котором распознавание треков среди шумов происходит сразу по всей картине события**

**1. Локальный трекинг для детектора GEM эксперимента BM@N особенно сложен из-за наличия гигантского количества фейковых хитов, что крайне затрудняет поиск тех хитов на последующих станциях детектора, которые являются продолжением обрабатываемого трека.**

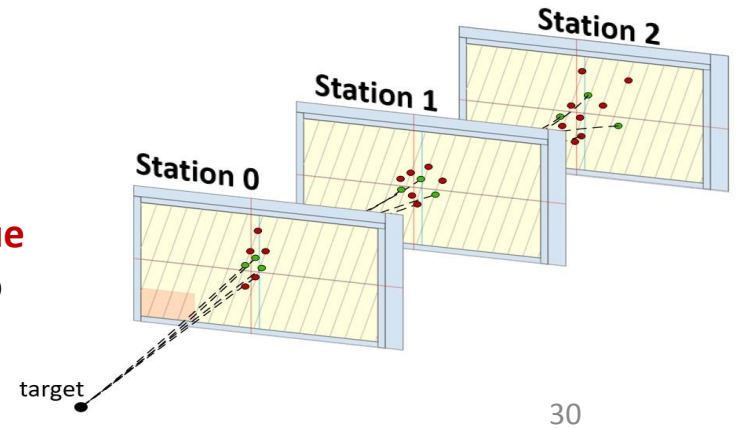



**Scheme of the recurrent TrackNETv2 neural network**

**See** https://doi.org/10.1063/1.5130102

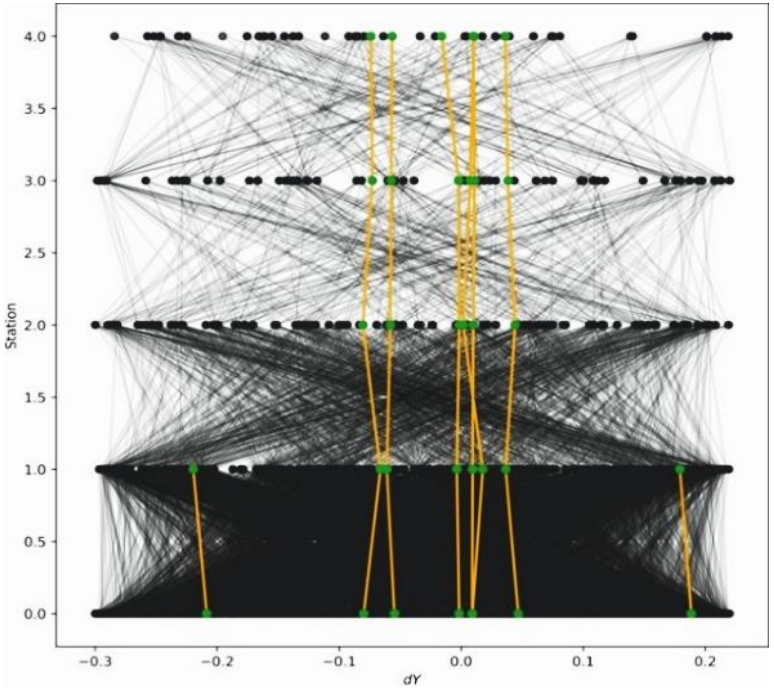
Однако гибкость конструкции RNN позволила нам преодолеть эти трудности и придумать новую сеть, которая объединяет оба этапа в одну сквозную TrackNETv2 с регрессионной частью из четырех нейронов, два из которых предсказывают точку центра эллипса на следующей координатной плоскости, где нужно искать продолжение трека-кандидата, а еще два - определяют полуось этого эллипса.

**Это дает нам возможность обучить одну сквозную модель, используя только истинные треки, которые можно извлечь из симуляции Монте-Карло. Таким образом, мы получили нейронную сеть, выполняющую прослеживание трека подобно фильтру Калмана, хотя и без его части подгонки трека**



# 2. Глобальный трекинг

Глобальное распознавание треков среди шумов осуществляется сразу по всей картине события. Программа **GraphNet** основана на использовании графовых нейронных сетей для трекинга. Событие представляется в виде графа с хитами в качестве узлов, а затем **этот граф инвертируется в линейный диграф, когда ребра представляются узлами, а узлы исходного графа - ребрами.** В этом случае информация о кривизне сегментов трека встраивается в ребра графа, что упрощает распознавание треков в море фейков и шумов **См.** http://ceur-ws.org/Vol-2507/280-284-paper-50.pdf




**Графическое представление события C + C,**
**4 ГэВ эксперимента BM@N. Черные узлы и ребра соответствуют фейкам, зеленые узлы и желтые ребра - найденным трекам**

- В процессе обучения сеть получает на вход инверсный диграф с метками истинных ребер - сегментов реальных путей.
- Уже обученная нейронная сеть RDGraphNet (Reversed Directed Graph Neural Network,) в результате связывает каждое ребро со значением $x \in [0,1]$ на выходе.
- Истинные ребра пути - это те ребра, для которых ***x*** больше некоторого заданного порога ***(> 0,5).***

Отметим, что в отличие от локальной нейронной сети TrackNETv2, измеряющей свою производительность в треках/сек, производительность глобальной нейронной сети RDGraphNet **измеряется в количестве распознанных событий в секунду**, что очень важно при большой множественности событий.

# Нефизические приложения глубокого обучения

- Эра Больших Данных сегодня охватила все области науки и приложений: биологию, экономику, медицину и т.д. Поэтому общий спрос на новые методы обработки данных, такие как глубокое обучение, неизбежно стал актуальным и в этих областях.
- Однако почти фатальным отличием этих областей от физики является отсутствие знаменитой физической "стандартной модели", позволяющей генерировать обучающие выборки любой желаемой длины для надежного обучения разрабатываемых глубоких нейронных сетей.
- **Отсутствие маркированных данных для обучения становится одной из главных проблем**. Помимо отсутствия теоретических предпосылок, ограниченный набор обучающих данных объясняется как высокой стоимостью самих экспериментов, так и стоимостью работы экспертов, маркирующих полученные данные. В качестве примера рассмотрим проблему некоторых сельскохозяйственных приложений CNN

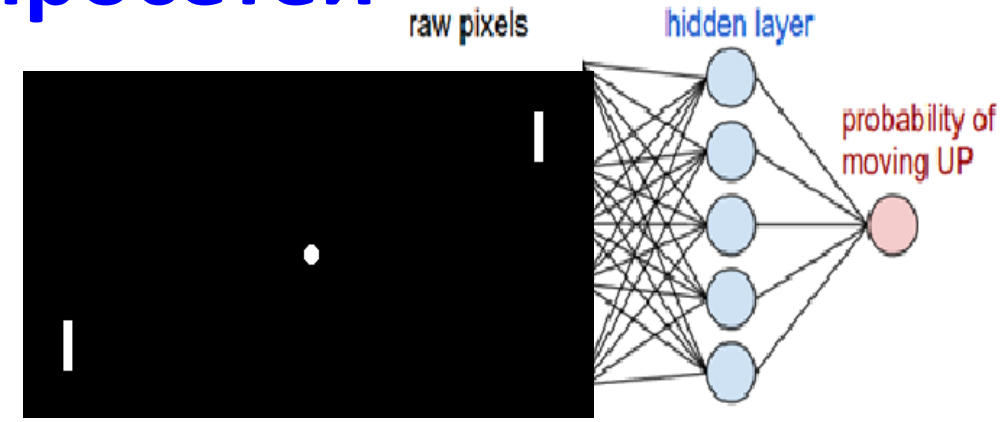
## Определение болезней растений по фото их листьев

Подробности можно найти в нашей статье **DOI: 10.1007/978-3-030-01328-8_16**, где описана веб-платформа http://pdd.jinr.ru , которая позволяет пользователям **отправлять фотографии и текстовые описания больных растений через этот веб-портал или мобильное приложение и выяснять причину заболевания.**

Хотя существует множество мобильных приложений для выявления болезней растений, как, например, знаменитый plantix.net, но ни одно из них не позволило нам получить набор данных, пригодный для использования. Нам пришлось создать свой собственный, чтобы работать с фотографиями, полученными в реальных условиях. **Для преодоления препятствия в виде малой обучающейвыборки мы используем сиамскую CNN со специальной триплетной функцией потерь и персептроном в качестве классификатора на выходном слое, который показывает 98% точности для 25 классов пяти культур**.

# Другие типы глубоких нейросетей

1. **Нейросети Обучаемые с подкреплением Reinforcement learning Networks**. Они реализуют такое обучение, когда агент нейронной сети, находясь в некотором состоянии, взаимодействует с окружающей средой, которая вознаграждает агента за его действия и сообщает, в какое состояние агент перешел после этого, чтобы увеличить общее вознаграждение.
   **Применение обучения с подкреплением**: роботы, самоуправляемые автомобили, торговые боты для игры на фондовом рынке, чат-боты, которые обучаются от диалога к диалогу, разработка игровых программ и т.д.




**Example. Ping-pong game** http://karpathy.github.io/2016/05/31/rl/

2. **Генеративные Состязательные сети** (**Generative Adversarial Networks, GAN**). GAN реализует принцип состязательности между генеративной сетью и сетью дискриминации. Генеративная сеть G генерирует наиболее реалистичный образец, а дискриминационная
3. сеть D обучается различать подлинные и поддельные образцы.

**Применения GAN**

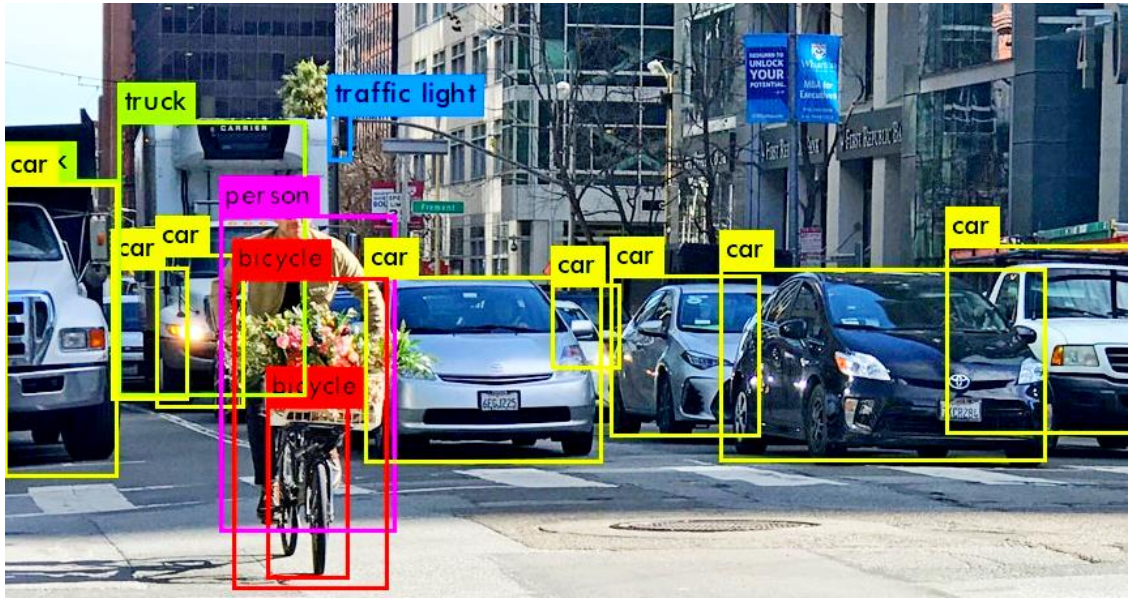
- получения фотореалистичных изображений и картин;
- автоматическое редактирование изображений
- создание фильмов и мультипликаций.
- создание трёхмерной модели объекта с помощью фрагментарных изображений
- моделирование сложных физических процессов в детекторах экспериментальной физики

**Этические проблемы GAN приложений!**

**Опасность дипфейков: политика, мошенничество и шантаж с использованием дипфейков**

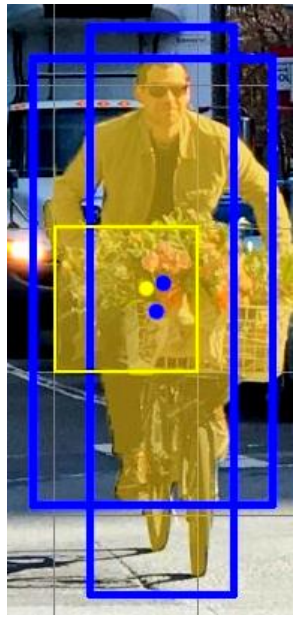
https://www.kaspersky.ru/resource-center/threats/protect-yourself-from-deep-fake

# Детектирование объектов - важный аспект взаимодействия со средой




**Философия YOLO** (от англ. **you only live once — «ты живешь только один раз») — стремление исполнять желания и получать удовольствие прямо сейчас, не отказываясь от этого ради будущего благополучия**.

**YOLO — современный алгоритм глубокого обучения “You Only Look Once” (“Стоит только раз взглянуть”), который широко используется для обнаружения объектов. Называется так, потому что подразумевает ввод сразу всего изображения, которое проходит через сверточную нейросеть (СНС) только один раз. Именно поэтому он очень быстро опознает объекты в режиме реального времени. Для детектирования изображение разбивают на ячейки и обычная СНС по многу раз смотрит наличие объекта в каждой ячейке, в YOLO же это делается всего один раз. Вокруг клетки рисуются несколько прямоугольников для определения объекта (их рисуют сразу несколько и разных форм), и их позиции, ширина и высота вычисляются относительно центра этой клетки. После чего специальный блок анализирует все возможные прямоугольники и выбирает объединение тех, куда объект вписывается наилучшим образом (см. https://habr.com/ru/post/514450/ ) Алгоритм YOLOv3 был обучен на датасете “Coco”, состоящем из 80 различных классов**



# Темы магистерских работ на 2022-23 уч.год

1. **Развитие алгоритмов в программе для поиска треков-кандидатов в трековых детекторах методами теории графов.**
2. **Разработка модели интеллектуального триггера для эксперимента SPD на основе методов глубокого обучения**
3. **Разработка программы на языке C++ для реконструкции траекторий элементарных частиц в эксперименте BM@N на основе действующей нейросетевой модели, выполненной на языке Python.**
4. **Разработка программы классификации изображений свёрточными нейронными сетями с применениями в задачах диагностирования заболеваний в биологии и медицине.**
5. **Применение информационных методов анализа социальных сетей для определения структуры малых социальных групп.**
6. **Обработка текстовой информации для сбора данных о заболеваемости и проверки их корреляции с данными по загрязнениям воздуха тяжелыми металлами по Московской области (2019)**
7. **Моделирование центров хранения и обработки данных с учетом экономических составляющих**

## Требования к студентам:

1. **Представление об искусственных нейронных сетях и теории графов**
2. **Умение программировать на Python и/или C++**
3. **Представление о пользовании библиотеками PyTorch и NumPy.**
4. **Знание английского языка, хотя бы на уровне беглого чтения.**

# Темы бакалаврских работ на 2022-23 уч.год

1. **Разработка программного модуля для визуализации событий в рамках библиотеки нейросетевого трекинга Ariadne**
2. **Исследование по выбору наиболее оптимального формата хранения данных Монте-Карло моделирования для решения задач трекинга.**
3. **Исследование по обработке текстовой информации в мобильном приложении в задаче классификации болезней растений.**
4. **Исследование методов глубокого обучения нейросетевого классификатора в условиях сильного дисбаланса обучающей выборки.**
5. **Разработка программы сжатия информации с использованием нейронных сетей.**
6. **Разделение и параметризации перекрывающихся сигналов на основе вейвлет-анализа**
7. **Программа вычисления гауссова вейвлет-преобразования и его применение для распознавания зашумленных сигналов**
8. **Автоматический подсчет стоимости заказа в университетской столовой по фото со смартфона.**

**Т.е. клиент подходит к автоматической кассе, ставит поднос в специальное место, поднос фотографируют, сеть — определяет позиции и считает счет заказа, потом идет оплата через терминал.**

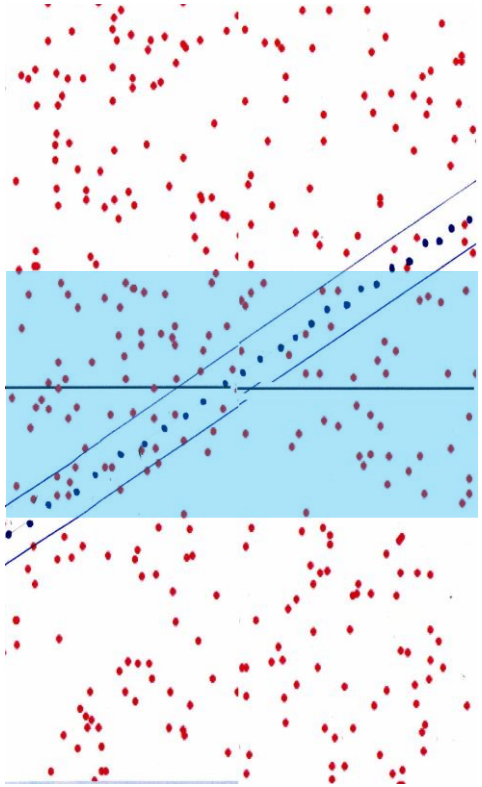
**Требования к студентам:**

1. Представление об искусственных нейронных сетях и теории графов
2. Представление о вейвлет-преобразованиях в их непрерывном и дискретном представлениях.
3. Умение программировать на Python и/или C++
4. Представление о пользовании библиотеками TensorFlow, Keras, NumPy или PyTorch.

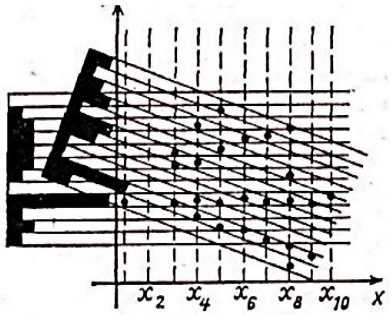
# 3 метода машинного обучения, о которых надо знать!

**Это - Вейвлет-анализ (на след слайде);**

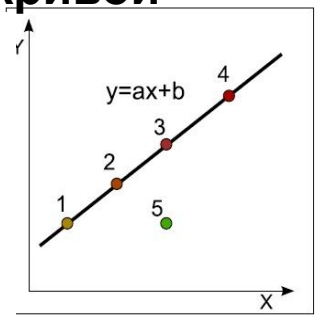
## Преобразования Хафа (Hough Transform)

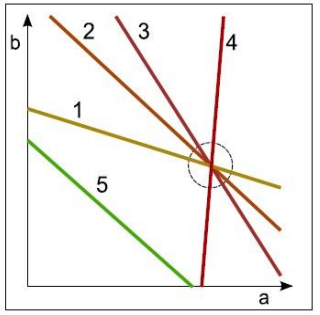


- **это преобразование для поиска линий – прямых, окружностей, когда точки из пространства координат преобразуются в точки в пространстве параметров кривой**

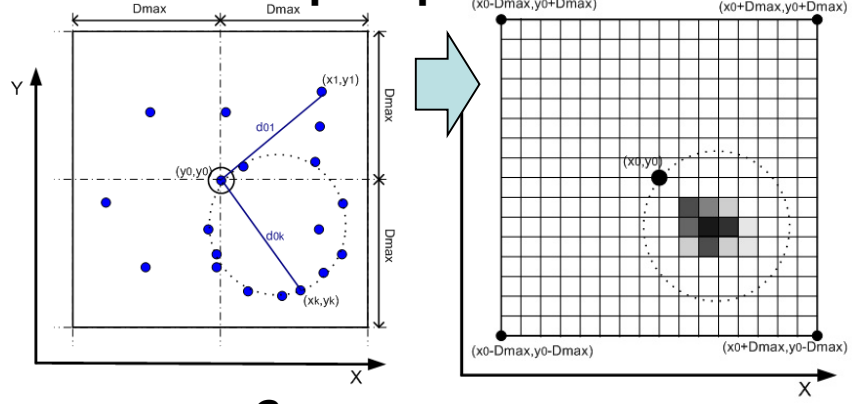


**Поворотные гистограммы**




**Прямая y=ax+b**




***Окружность $(x-x_0)^2+(y-y_0)^2=R^2$***

**Робастная регрессия – это специальный итеративный**

**МНК:** $\sum_i w_i(y_i - ax_i - b)^2 = \sum_i w_i e_i^2$ **-->min $_{a,b}$, где вместо постоянных весов** $w_i=1/\sigma^2$ **вводится весовая функция, чаще всего это бивесовая функция Тьюки**

$$w(e)=\begin{cases}\left(1-\left(\frac{e}{c_T\sigma}\right)^2\right)^2, & \text{если } |e|<c_T\sigma,\\ 0, & \text{в остальных случаях}\end{cases}$$

**, где обычно $c_T$ =3÷5, а значение $\sigma$ также каждой *k*-й итерации по формуле**

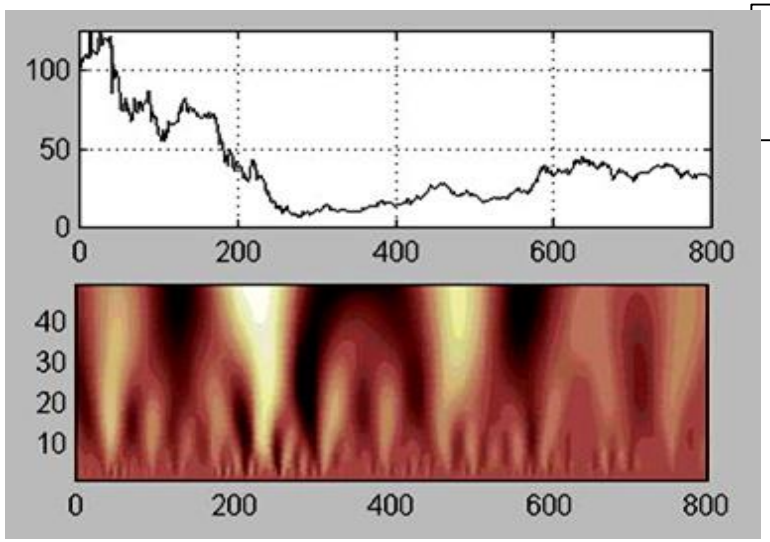
$$\sigma^{(k)}=\sqrt{\sum_i w_i^{(k-1)}\left(e_i^{(k-1)}\right)^2 \Big/ \sum_i w_i^{(k-1)}}.$$

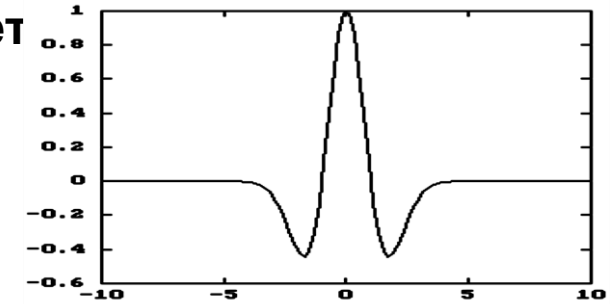
# Вейвлет-анализ

**Одномерное вейвлет-преобразование сигнала *f(x)* является двумерной функцией частоты и времени,**

$$W_\psi(a,b)f = \frac{1}{\sqrt{C_\Psi}}\int_{-\infty}^{\infty}\frac{1}{\sqrt{|a|}}\Psi\left(\frac{b-x}{a}\right)f(x)dx,$$

**где функция Ψ - вейвлет, b - сдвиг (shift), a – масштаб (scale). Условие $C_\psi < \infty$ обеспечивает существование Ψ и обратного вейвлет-преобразования. Произвол в выборе Ψ позволил предложить много разных типов вейвлетов. Семейство непрерывных вейвлетов можно получить как вторую производную гауссиана** $g_2(x) = (1-x^2)e^{-\frac{x^2}{2}}$ **Полученный гауссов вейвлет стал известен как "Мексиканская шляпа"**



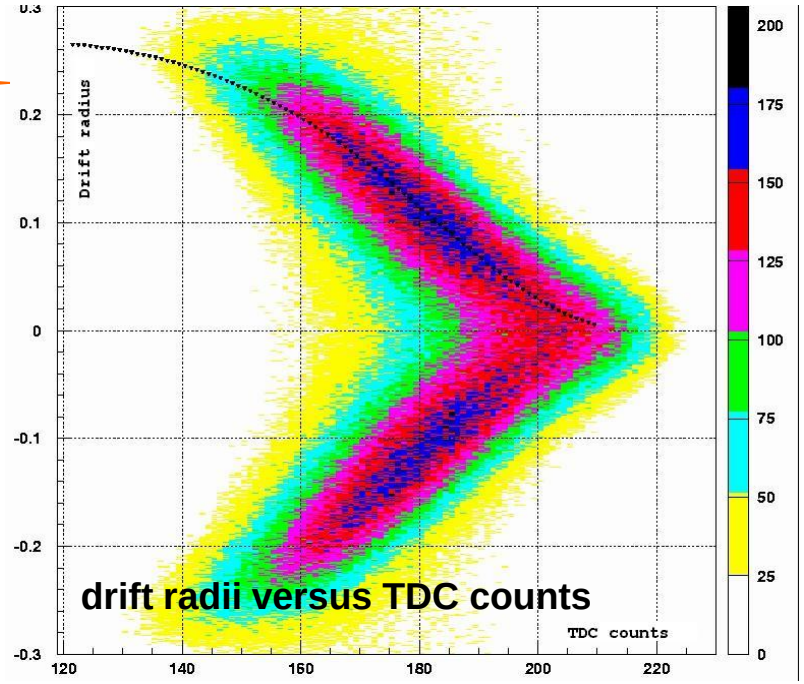

**На этом примере видно, что разные Вертикальные уровни вейвлет-спектра соответствуют разным частотам сигнала, а по горизонтали видно в какое время сигнал изменял свое поведение.**

**Непрерывные вейвлеты очень устойчивы к шумам, однако они неортогональны. поэтому после обратного преобразования происходят недопустимые искажения сигнала. Кроме того, реальные сигналы, подлежащие компьютерному анализу, всегда дискретны. Поэтому непрерывные вейвлеты практически не используют, а применяют ортогональные дискретные вейвлеты .**

# Пример применения МО для коррекции аппаратных измерений

## 1. Калибровка дрейфовых камер

**Пример калибровки дрейфовой камеры HERA-B OTR:**
**Время дрейфа, измеренное в отсчетах TDC (время-цифровой преобразователь), следует перевести в радиусы дрейфа для получения калибровочной функции r(t). r(t) была получена путем робастной подгонки кубических сплайнов непосредственно к 2D гистограмме отсчетов TDC со многими тысячами бинов**




## 2. Проблемы алайнмента.

**Электронные детекторы состоят из многих тысяч модулей, которые, даже при самой тщательной сборке в экспериментальную установку, неизбежно оказываются подвержены различным мелким дисторсиям, таким как сдвиги и наклоны, ведущим к искажающим результаты измерений.**

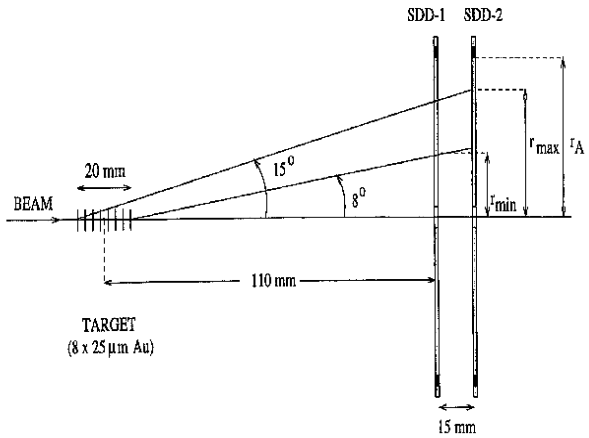
**Поэтому для обнаружения и устранения возможных искажений необходима процедура алайнмента. Такие дисторсии модулей детектора можно найти путем анализа расхождений между измеренными значениями и подогнанными координатами трека. Эти расхождения функционально зависят от двух типов параметров: самой модели трека и параметров процедуры алайнмента.**

**Таким образом, задача математически проблема алайнмента формулируется, как задача минимизации функционала, суммирующего квадраты всех расхождений по обоим типам параметров, число которых достигает миллионов.**

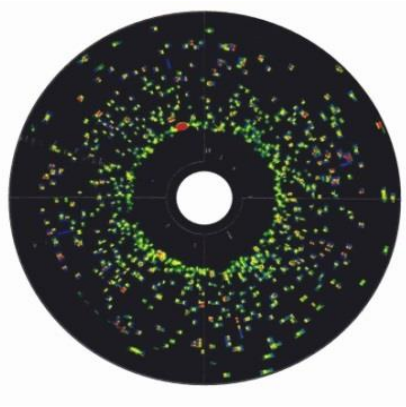
**В качестве решения получающейся задачи с неограниченным числом степеней свободы предложен метод разложения по сингулярным значениям. В результате мы получили Data-driven алайнмент!**

# Примеры применения методов робастной подгонки

## 1. NA-45. Определение координат вершин событий по только двум измерениям

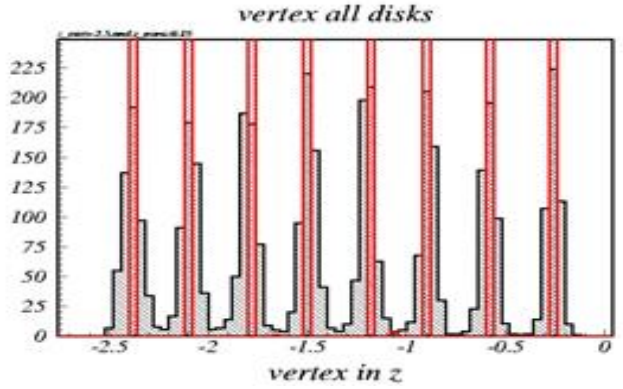



NA-45 координатный детектор

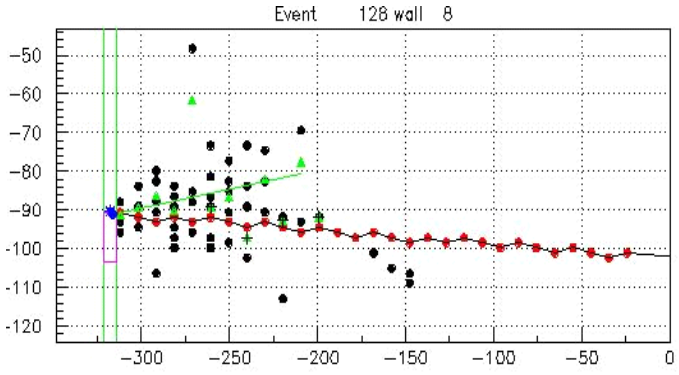


Один из двух силиконовых дисков

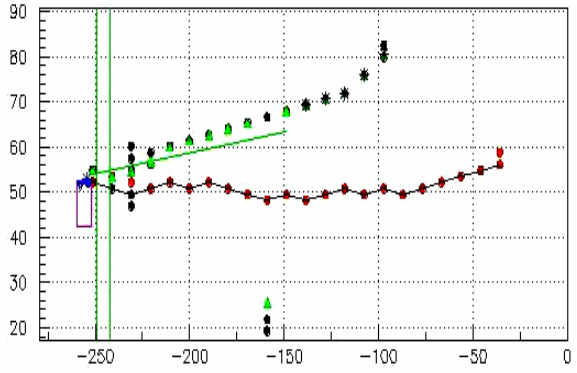
Мишень состояла из восьми 25-мк золотых дисков. 700 трековых событий в узком угловом приеме и большое количество шумовых отсчетов не позволили распознать отдельные треки. Однако метод робастной подгонки сходился всего за пять итераций, хотя за начальную аппроксимацию была грубо взята середина всей области мишени.



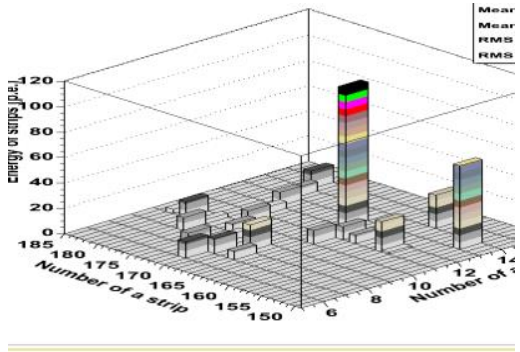

## 2. Opera. Робастная подгонка для адронных ливней и мюонных треков с 2D весами, зависящими не только от расстояний, но и амплитуд трековых хитов



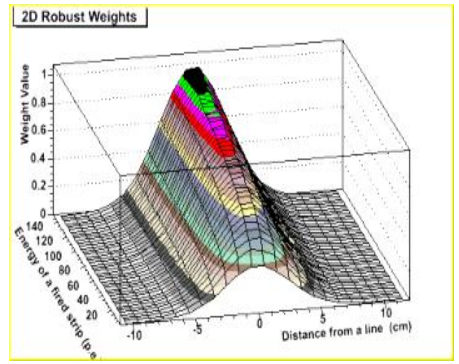

Адронный ливень




1D мюонные треки




2D мюонные треки




Двумерная весовая функция

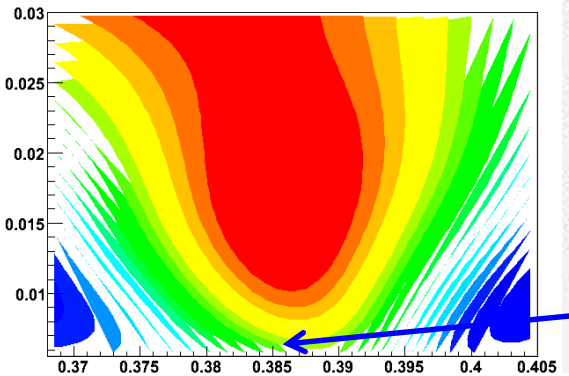
# Непрерывные вейвлет-преобразования

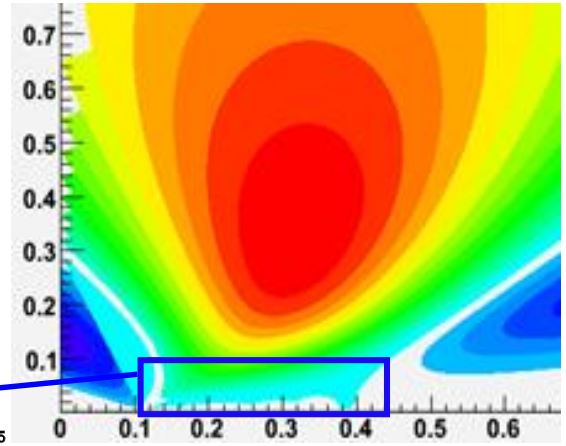
## это путь к вейвлет- домену, где проблемы МО могут быть решены гораздо проще

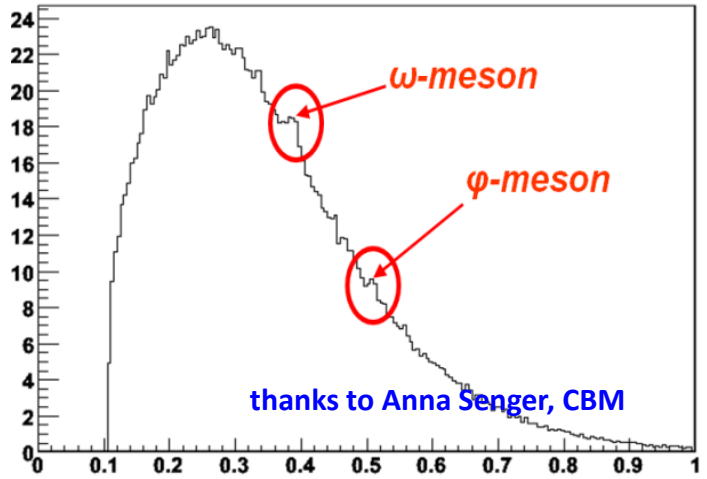
**они нужны для работы с инвариантными масс-спектрами, когда отношение S/B << 1**

**Непрерывный вейвлет G2 преобразует гауссиан g(x;A,xo,σ) в вейвлет того же** $W_{G_2}(a,b)g = \frac{Aa^{5/2}\sigma}{(a^2+\sigma^2)^{3/2}} G_2\left(\frac{b-x_0}{\sqrt{a^2+b^2}}\right)$ **порядка, но с параметрами этого гауссиана. Это справедливо для любого порядка n и позволяет находить параметры пика непосредственно в области G2 без инвертирования**

**1. оценка параметров пиков по спектру инвариантных масс**

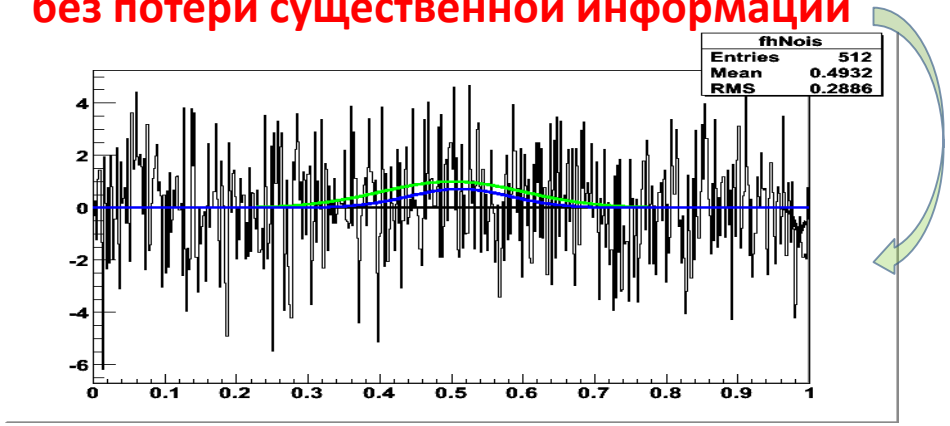


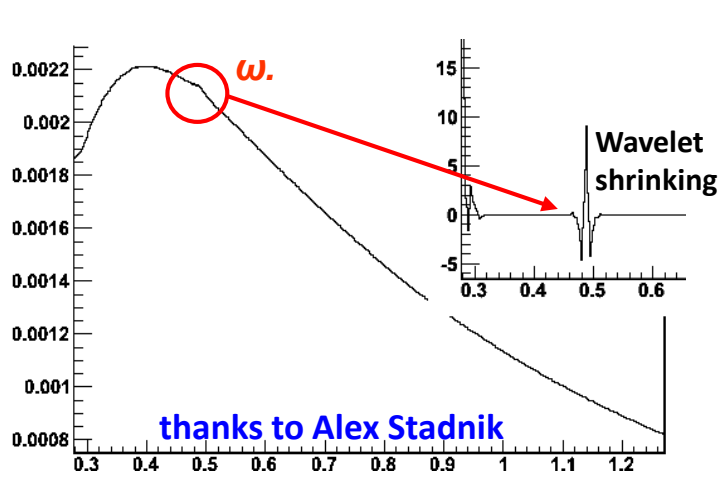




**2. Сглаживание после вычитания фона**

**без потери существенной информации**



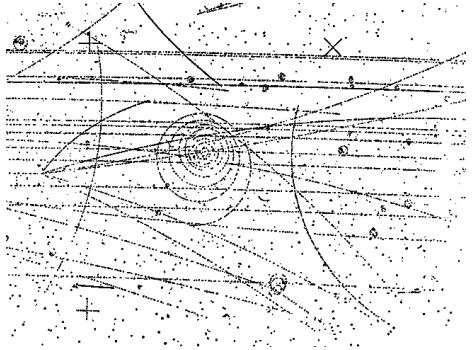

## 3. Дискретные вейвлеты

**Применяют для выявления резонанса даже в присутствии массивного фона**

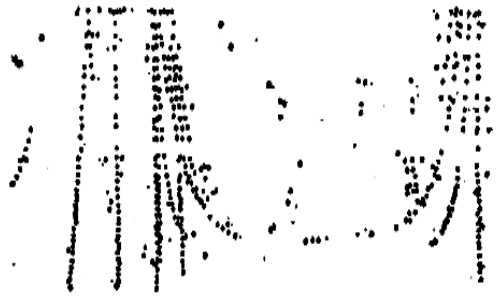
***Thanks for your attention!***

# Эволюция методов трекинга

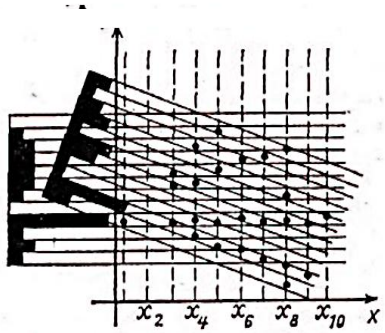
Началось еще в эпоху **пузырьковых камер**, когда события регистрировались на стереофотографиях и вводились в компьютер вручную, полуавтоматами или с помощью сканирующих устройств типа «Спиральный измеритель», в котором оператор ставил точку в вершину события, откуда шло сканирование снимка по спирали



Снимок события.



Его оцифровка в полярных коорд.

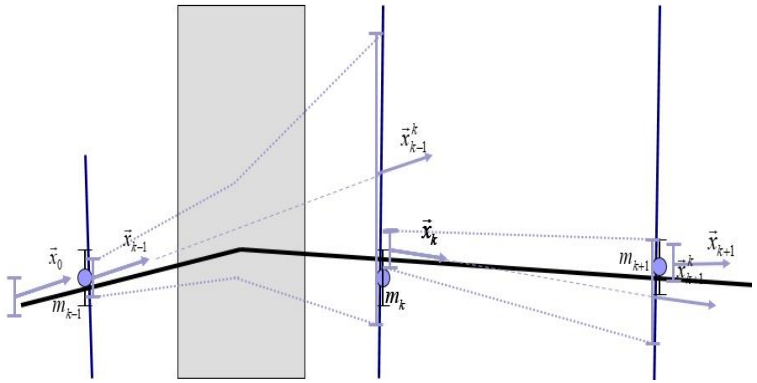



Поворотные гистограммы

Когда пришла **эра электронных экспериментов**, данные измерений стали оцифровываться и сразу поступать прямо в компьютер. После многоэтапной фильтрации и процедур алайнмента, наступало время трекинга. Среди многих методов трекинга, самым эффективным оказался метод, использующий **фильтр Калмана**, поскольку он позволяет легко учитывать неоднородность магнитного поля, многократное рассеяние и потери энергии

**Фильтр Калмана (ФК)** – это эффективный рекурсивный фильтр, оценивающий состояние **линейной динамической системы**, используя ряд неточных измерений

Вектор состояния $\vec{x} = (x, y, t_x, t_y, q/p)^T$ итеративно оценивается для предсказания позиции трека на след. координатной плоскости с учетом изменения ковариационной матрицы и коридоров ошибок.




**Главный недостаток ФК – необходимость знать начальное значение вектора состояния $\vec{X}$, выполнить «сидинг» (англ. seed- семя)**

# Как оценивать результаты трекинга

**Первая наивная оценка эффективности трекинга по процентному отношению числа найденных треков к общему числу треков-кандидатов (accuracy)– бесполезна и даже опасна, т.к. наша выборка очень сильно несбалансирована и среди треков-кандидатов может оказаться слишком много ложных, образованных из шумовых отсчетов и кусков разных треков, так что среди распознанных, кроме тех, что в самом деле являются реальными треками также и те, что ошибочно распознаны как треки, но они – ложные (ghosts- призраки)**

**Поэтому в терминах трекинга принято использовать метрики**

$$recall = \frac{N_{true}^{rec}}{N^{in}}$$

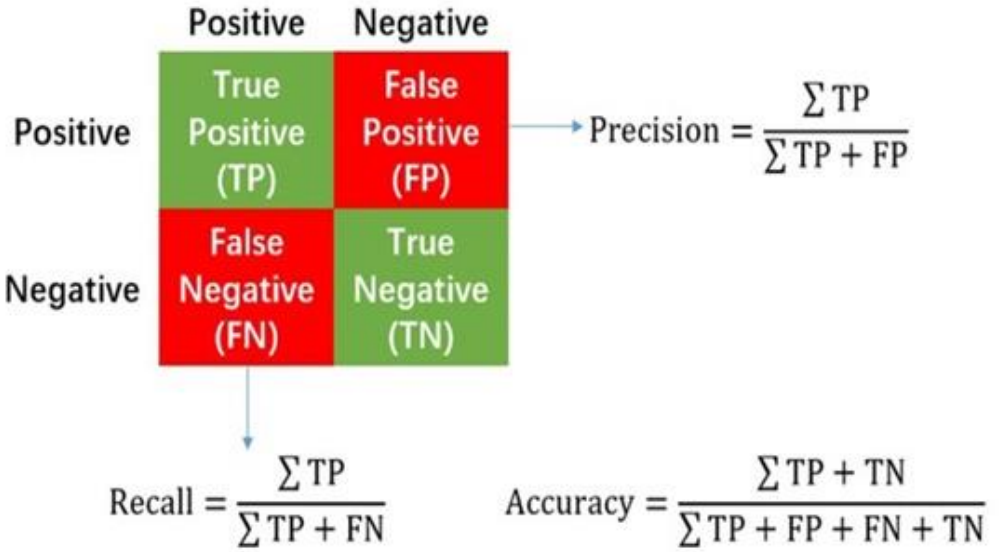
$$precision = \frac{N_{true}^{rec}}{N^{rec}},$$

где $N_{true}^{rec}$ - количество реальных треков, найденных моделью;

$N_{in}$ - количество всех реальных треков, известных из симуляции Монте-Карло;

$N^{rec}$ - количество всех треков, которые модель реконструировала как реальные.

**По существу – это общий вопрос о критериях проверки правильности гипотез, основной и альтернативной. Ошибки классификации бывают двух видов: FP и FN. В статистике FP называют ошибкой I-го рода, а FN — ошибкой II-го рода.**




**Одним из способов оценить модель в целом является AUC-ROC (или ROC AUC) — площадь (*A*rea *U*nder *C*urve) под кривой ошибок(*R*eceiver *O*perating *C*hara cteristic curve ).**