Введение в машинное обучение в физике элементарных частиц

(Некоторые слайды заимствованы из лекций К.В. Воронцова (МФТИ))

Dmitry Matvienko

22 июля 2025 г.

Оптимизационная постановка задачи

X — пространство *объектов*
Y — множество *ответов* (предсказаний / оценок / прогнозов)
$y(x)$, $y\colon X \to Y$ — неизвестная зависимость (target function)

Дано:
$\{x_1, \ldots, x_\ell\} \subset X$ — *обучающая выборка* (training sample)
$a(x, w)$, $a\colon X \times W \to Y$ — параметрическая модель зависимости

Найти:
$w \in W$ — вектор параметров модели такой, что $a(x, w) \approx y(x)$

Критерий минимизации эмпирического риска:

$$\sum_{i=1}^{\ell} \mathscr{L}(w, x_i) \to \min_w \quad \text{(empirical risk minimization, ERM)}$$

где $\mathscr{L}(w, x)$ — *функция потерь* (loss function) — тем больше, чем сильнее $a(x, w)$ отклоняется от правильного ответа $y(x)$

Оптимизационная постановка задачи

Задачи обучения с учителем (supervised learning):

- заданы «ответы учителя» $y_i = y(x_i)$ на обучающих x_i

— задачи классификации (classification, Y — class labels):

- $Y = \{-1, +1\}$ — на 2 класса (binary classification)
- $Y = \{1, \ldots, M\}$ — на много классов (multiclass c.)
- $Y = \{0, 1\}^M$ — на пересекающиеся классы (multilabel c.)

— задачи ~~восстановления~~ регрессии (regression):

- $Y = \mathbb{R}$ или $Y = \mathbb{R}^m$

— задачи ранжирования (ranking, learning to rank):

- Y — конечное упорядоченное множество

Задачи обучения без учителя (unsupervised learning):

- ответов нет, требуется что-то делать с самими объектами

Функции потерь

- Функции потерь тесно связаны с ММП и статистическими свойствами шума, которые неявно закладываются в модель.
- Среднеквадратичное отклонение для задач регрессии (MSE)

$$y_i = a(x_i, w) + \epsilon_i, \;\; \epsilon_i \sim \mathcal{N}(0, \sigma^2), \;\; i = 1, \ldots, n, \;\; \hat{w} = ?$$

$$\mathcal{L} = \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right)^n exp\left(-\frac{1}{2} \sum_{i=1}^{n} \frac{\epsilon_i^2}{\sigma^2} \right) \to max(w, \sigma^2)$$

$$-log\mathcal{L} = \frac{n}{2} log(2\pi\sigma^2) + \frac{1}{2} \sum_{i=1}^{n} \frac{(y_i - a(x_i, w))^2}{\sigma^2} \to min(w, \sigma^2)$$

- Бинарная кросс-энтропия для задач бинарной классификации (logloss)

$$y_i \sim Bern(p_i), \;\; p_i = a(x_i, w)$$

$$\mathcal{L} = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{(1-y_i)}$$

$$logloss = -log\mathcal{L} = -\sum_{i=1}^{n} [y_i log\, a(x_i, w) + (1 - y_i) log\, (1 - a(x_i, w))] \to min(w)$$

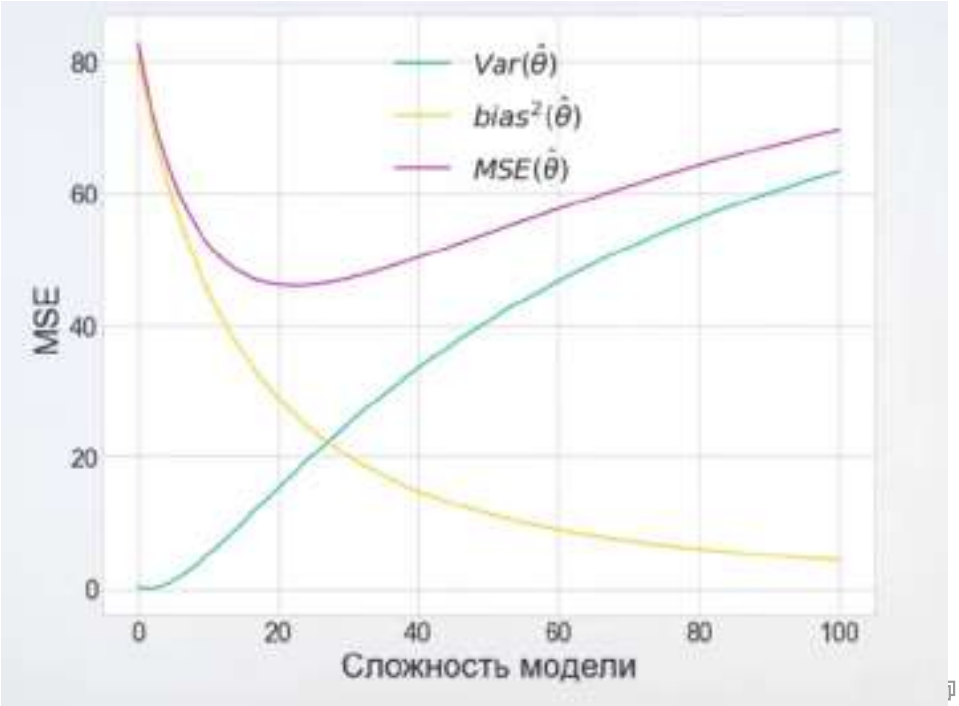
Проблема переобучения

- Дилемма bias-variance.

$$\hat{y} = a(x, w), \; y = \hat{y} + \epsilon$$

$$MSE = \mathbb{E}(y - \hat{y})^2 = Var(\epsilon) + Var(\hat{y}) + Bias^2(\hat{y})$$

- $Var(\epsilon)$ - шум на данных
- $Var(\hat{y})$ - вариативность алгоритма обучения за счет конечной и рандомной обучающей выборки
- $Bias(\hat{y})$ - смещение выборочной оценки модели данных



Регуляризация

- Рассмотрим бинарный классификатор $a(x, w) = sgn < w, x >$. Пусть он обучается на линейно-зависимых признаках $\exists u \in \mathbb{R}^n : \forall x \in X, < u, x >= 0$. Тогда $\forall \beta \in \mathbb{R}, < w, x >=< w, x + \beta u >$ - решение неустойчивое (линейное многообразие).
- Пусть $|\beta| \to \infty$, тогда большие числа сокращаются на обучающей выборке, но разность сильно растет на тестовых данных. Переобучение!
- Штрафуем большие веса $||w|| \leq A_{max}$ ($\mathcal{L}_2$ -регуляризация)

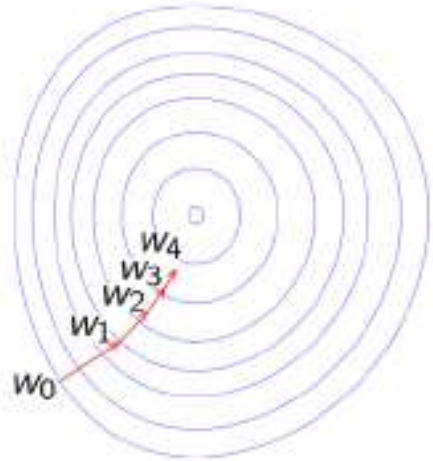
$$\tilde{\mathcal{L}} = \mathcal{L} + \gamma \sum_{i=1}^{n} w_i^2 \to min(w)$$

- γ - гиперпараметр, подбирается кросс-валидацией.
- Можно штрафовать сумму модулей $\sum_i |w_i|$ ($\mathcal{L}_1$ -регуляризация)
- Elastic Net (комбинация $\mathcal{L}_1$ и $\mathcal{L}_2$)

Метод градиентного спуска

Минимизация эмпирического риска:

$$Q(w) = \sum_{i=1}^{\ell} \mathscr{L}(w, x_i) \to \min_w$$



Метод *градиентного спуска*:

$w^{(0)} :=$ начальное приближение;

$$w^{(t+1)} := w^{(t)} - h \cdot \nabla Q(w^{(t)}), \qquad \nabla Q(w) = \left(\frac{\partial Q(w)}{\partial w_j}\right)_{j=0}^{n},$$

где h — *градиентный шаг*, называемый также *темпом обучения*.

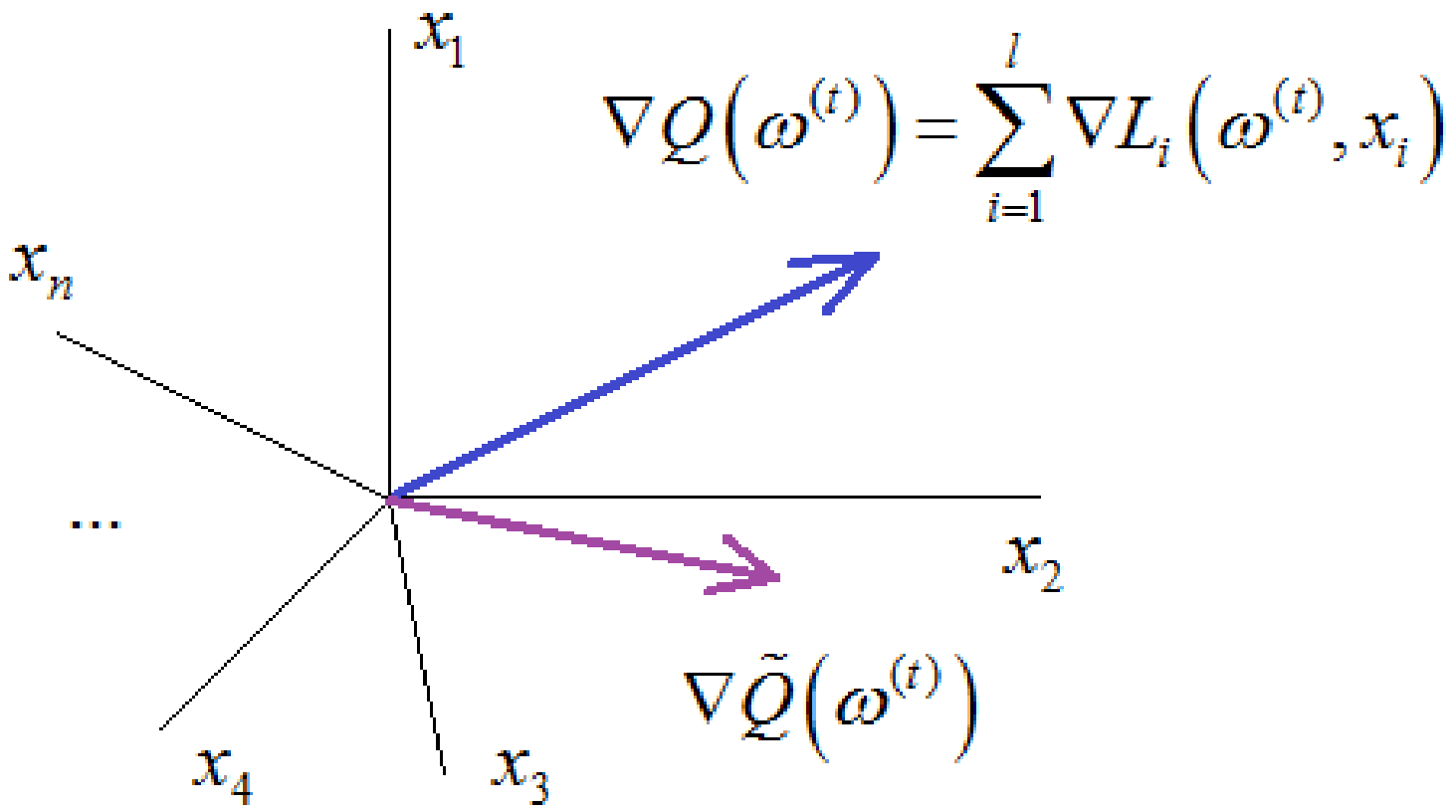
$$w^{(t+1)} := w^{(t)} - h \sum_{i=1}^{\ell} \nabla \mathscr{L}\left(w^{(t)}, x_i\right)$$

Идея ускорения сходимости:

брать объекты x_i по одному и сразу обновлять вектор весов

Метод градиентного спуска

- Псевдоградиент - замена истинного градиента от функционала качества, который в среднем образует острый угол с истинным градиентом.
- Стохастический градиентный спуск - в качестве псевдоградиента берется градиент от одного случайного элемента выборки или батча.



Метод градиентного спуска

Вход: выборка X^ℓ, темп обучения h, темп забывания λ;
Выход: вектор весов w;

1 инициализировать веса w_j, $j = 0, \ldots, n$;
2 инициализировать оценку функционала:
$Q :=$ среднее $\mathscr{L}(w, x_i)$ по случайному подмножеству $\{x_i\}$;
3 **повторять**
4 выбрать объект x_i из X^ℓ случайным образом;
5 вычислить потерю: $\varepsilon_i := \mathscr{L}(w, x_i)$;
6 сделать градиентный шаг: $w := w - h\nabla\mathscr{L}(w, x_i)$;
7 оценить функционал: $Q := \lambda\varepsilon_i + (1 - \lambda)Q$;
8 **пока** значение Q и/или веса w не сойдутся;

- Функционал качества Q обновляется по рекуррентной формуле - экспоненциальное скользящее среднее.

$$Q_m = \frac{1}{m}(\epsilon_m + \ldots \epsilon_1)$$
$$mQ_m - \epsilon_m = \epsilon_{m-1} + \ldots \epsilon_1 = (m-1)Q_{m-1}$$
$$Q_m = \frac{1}{m}\epsilon_m + \left(1 - \frac{1}{m}\right) Q_{m-1}$$

Модели машинного обучения $a(x, w) : X \times W \to Y$

- Обучение с учителем
  - Метод ближайших соседей, KNN
  - Логистическая регрессия
  - Метод опорных векторов, SVM
  - **Деревья решений, DT**
  - Случайный лес, бэггинг
  - **Бустинг**
- Обучение без учителя
  - Кластеризация K-средних, K-means
  - Кластеризация с использованием плотности точек, DBSCAN
  - Метод главных компонент, PCA
- **Нейронные сети**
  - **Полносвязный слой, MLP**
  - **Сверточный слой, CNN**
  - **Рекуррентный слой, RNN**
  - **Графовый слой, GNN**
  - **Автоэнкодеры, AE, VAE**

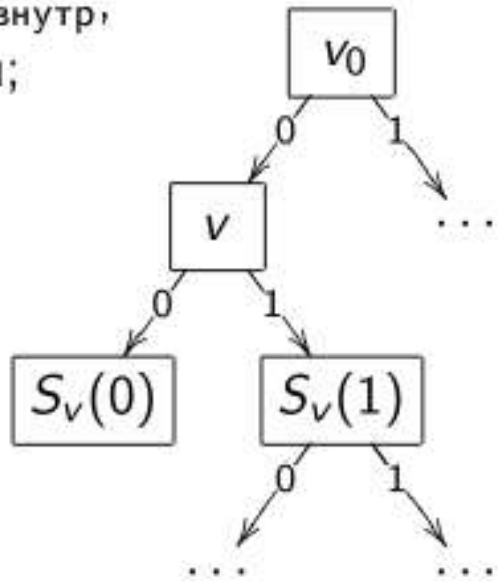
Деревья решений (DT)

Решающее дерево — алгоритм классификации $a(x)$, задающийся *деревом* (связным ациклическим графом) с корнем $v_0 \in V$ и множеством вершин $V = V_{\text{внутр}} \sqcup V_{\text{лист}}$;

$f_v : X \to D_v$ — дискретный признак, $\forall v \in V_{\text{внутр}}$;

$S_v : D_v \to V$ — множество дочерних вершин;

$y_v \in Y$ — метка класса, $\forall v \in V_{\text{лист}}$;



$v := v_0$;
пока $(v \in V_{\text{внутр}})$: $v := S_v(f_v(x))$;
вернуть $a(x) := y_v$;

Чаще всего используются бинарные признаки вида $f_v(x) = [f_j(x) \geqslant a]$

Если $D_v \equiv \{0, 1\}$, то решающее дерево называется *бинарным*

Критерий ветвления в DT

- Нужно выбирать порог a и признак j для разбиения данных в узле дерева
- Gini impurity $G(X)$ - мера порядка в данном узле дерева

$$G(X) = \sum_{c \in C} p_c(1 - p_c)$$

$$p_c = \frac{1}{N_X} \sum_{x_i \in X} [y_i = c]$$

- $G(X) \to 1$ для самого неоднородного случая, $G(X) \to 0$ для самого однородного
- Информационный выигрыш - критерий разбиения данных в узле

$$IG(X) = \frac{N_t}{N_X} \left[G(X) - \frac{N_{tR}}{N_t} G^R(X) - \frac{N_{tL}}{N_t} G^L(X) \right]$$

- Поиск максимума в пространстве $\{J, A\}$:

$$j_*, a_* = \mathtt{arg\,max}_{j \in J, a \in A} IG(j, t)$$

Градиентный бустинг

Линейная (выпуклая) комбинация базовых алгоритмов:

$$a(x) = \sum_{t=1}^{T} \alpha_t b_t(x), \quad x \in X, \quad \alpha_t \in \mathbb{R}_+.$$

Функционал качества с произвольной функцией потерь $\mathscr{L}(a, y)$:

$$Q(\alpha, b; X^\ell) = \sum_{i=1}^{\ell} \mathscr{L}\Big(\underbrace{\underbrace{\sum_{t=1}^{T-1} \alpha_t b_t(x_i)}_{f_{T-1,i}} + \alpha b(x_i)}_{f_{T,i}}, y_i\Big) \to \min_{\alpha, b}.$$

$f_{T-1} = (f_{T-1,i})_{i=1}^{\ell}$ — текущее приближение
$f_T = (f_{T,i})_{i=1}^{\ell}$ — следующее приближение

Градиентный бустинг

Градиентный метод минимизации $Q(f) \to \min$, $f \in \mathbb{R}^\ell$:

$$f_0 := \text{начальное приближение;}$$
$$f_{T,i} := f_{T-1,i} - \alpha g_i, \quad i = 1, \ldots, \ell;$$

$g_i = \mathscr{L}'\big(f_{T-1,i}, y_i\big)$ — компоненты вектора градиента,
α — градиентный шаг.

Наблюдение: это очень похоже на одну итерацию бустинга!

$$f_{T,i} := f_{T-1,i} + \alpha b(x_i), \quad i = 1, \ldots, \ell$$

Идея: будем искать такой базовый алгоритм b_T, чтобы вектор $\big(b_T(x_i)\big)_{i=1}^\ell$ приближал вектор антиградиента $\big(-g_i\big)_{i=1}^\ell$:

$$b_T := \arg\min_b \sum_{i=1}^{\ell} \big(b(x_i) + g_i\big)^2$$

Градиентный бустинг

Вход: обучающая выборка X^ℓ; параметр T;

Выход: базовые алгоритмы и их веса $\alpha_t b_t$, $t = 1, \ldots, T$;

1: инициализация: $f_i := 0$, $i = 1, \ldots, \ell$;

2: **для всех** $t = 1, \ldots, T$

3: базовый алгоритм, приближающий антиградиент:

$$b_t := \arg\min_b \sum_{i=1}^{\ell} \big(b(x_i) + \mathscr{L}'(f_i, y_i)\big)^2;$$

4: задача одномерной минимизации:

$$\alpha_t := \arg\min_{\alpha>0} \sum_{i=1}^{\ell} \mathscr{L}\big(f_i + \alpha b_t(x_i), y_i\big);$$

5: обновление вектора значений на объектах выборки:

$$f_i := f_i + \alpha_t b_t(x_i); \quad i = 1, \ldots, \ell;$$

Градиентный бустинг для бинарной классификации

- Функция потерь - logloss

$$\mathcal{L}(p, y) = -y\, log(p) - (1 - y)\, log(1 - p)$$

$$p = \sigma(f) = \frac{1}{1 + e^{-f}}$$

- Вычисление градиента

$$\frac{\partial \mathcal{L}}{\partial f} = \frac{\partial \mathcal{L}}{\partial p} \frac{\partial p}{\partial f} = \sigma(f) - y$$

- Инициализация $f_i := log(\frac{class\,1}{class\,0})$
- Для всех $t = 1, \ldots, T$
  - вычисляем градиент $r_i := \sigma(f) - y$
  - обучаем дерево $b_t(x_i)$ предсказывать r_i
  - Обновляем вектор ответов на объектах выборки

$$f_i := f_{i-1} + \alpha_t b_t(x_i)$$

- После всех итераций попускаем логиты f_i через сигмоиду $\sigma(f_i)$

XGBoost

Деревья регрессии и классификации (CART):

$$b(x, w) = \sum_{k \in K} w_k B_k(x)$$

где $B_k(x)$ — бинарный индикатор [x попадает в лист k],
w_k — значение в листе k, K — множество листьев дерева.
Для любого x одно и только одно слагаемое не равно нулю.

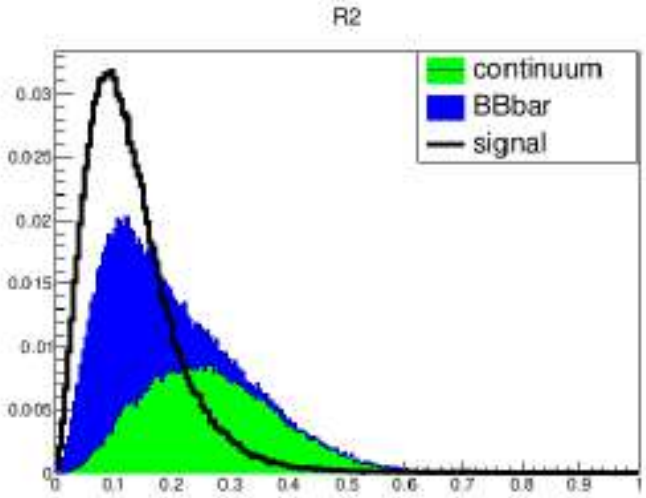
Критерий качества с суммой L_0 и L_2 регуляризаторов:

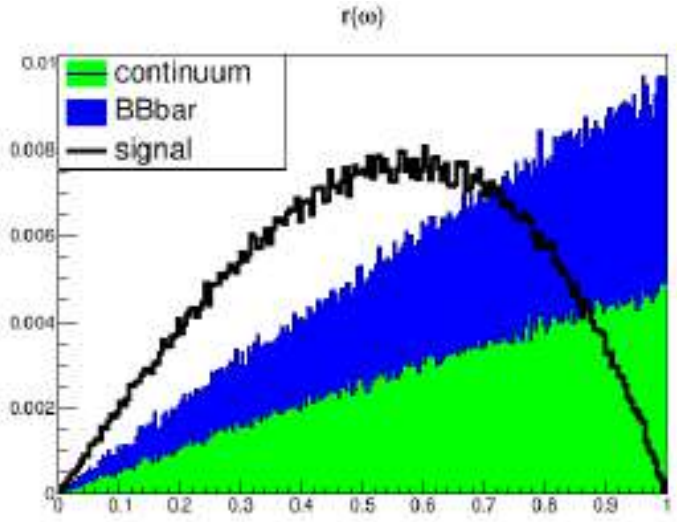
$$Q(w) = \sum_{i=1}^{\ell} \mathscr{L}\big(a(x_i) + b(x_i, w), y_i\big) + \gamma|K| + \frac{\lambda}{2} \sum_{k \in K} w_k^2 \;\to\; \min_w,$$

где $a(x_i) = \sum_{t=1}^{T-1} \alpha_t b_t(x_i)$ — ранее построенная часть ансамбля.
В некоторых случаях задача имеет аналитическое решение.

Пример градиентного бустинга в анализе $B^0 \to D\omega\pi$ (Belle II)

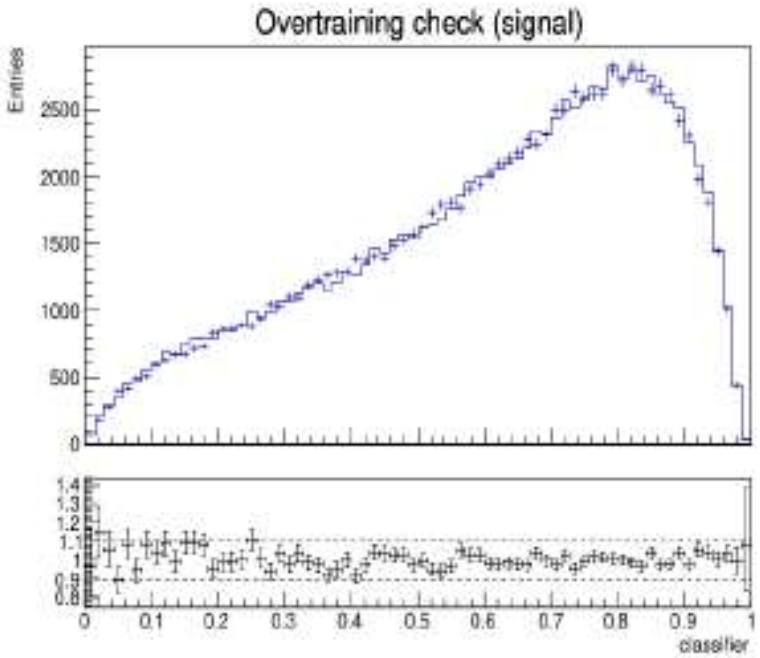
- Используем град. бустинг для разделения сигнальных и всех фоновых ($ee \to q\bar{q}$, $ee \to B\bar{B}$) событий. Задача бинарной классификации.
- Выбираем переменные для тренировки бустинга с учетом их разделяющей способности. Сначала включаем максимальный набор переменных и потом итеративно удаляем те, которые имеют наименьшую важность, feature importance = $Norm(\sum_t IG_t(j,a))$.
- Исключаем переменные с сильной корреляцией со спектаторными перменными (которые будут в дальнейшем использоваться в анализе)

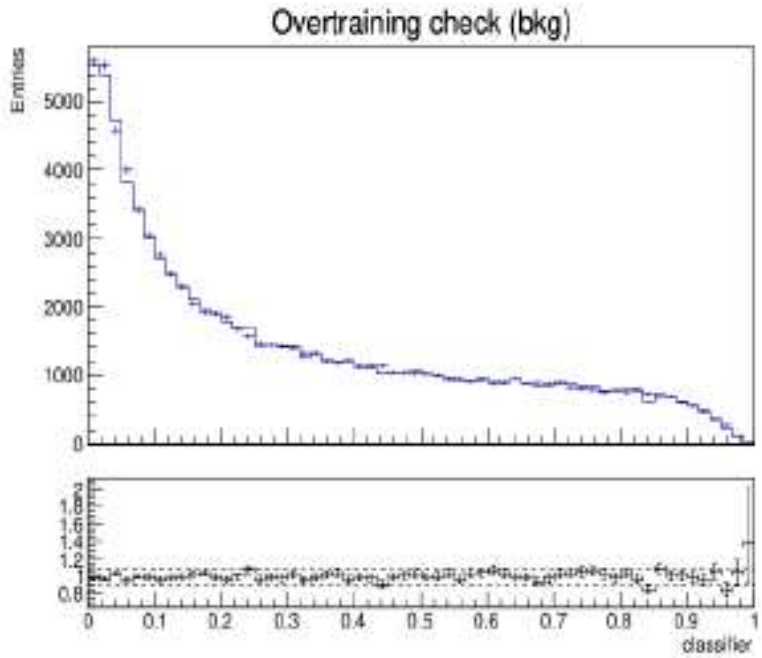




Пример градиентного бустинга в анализе $B^0 \to D\omega\pi$ (Belle II)

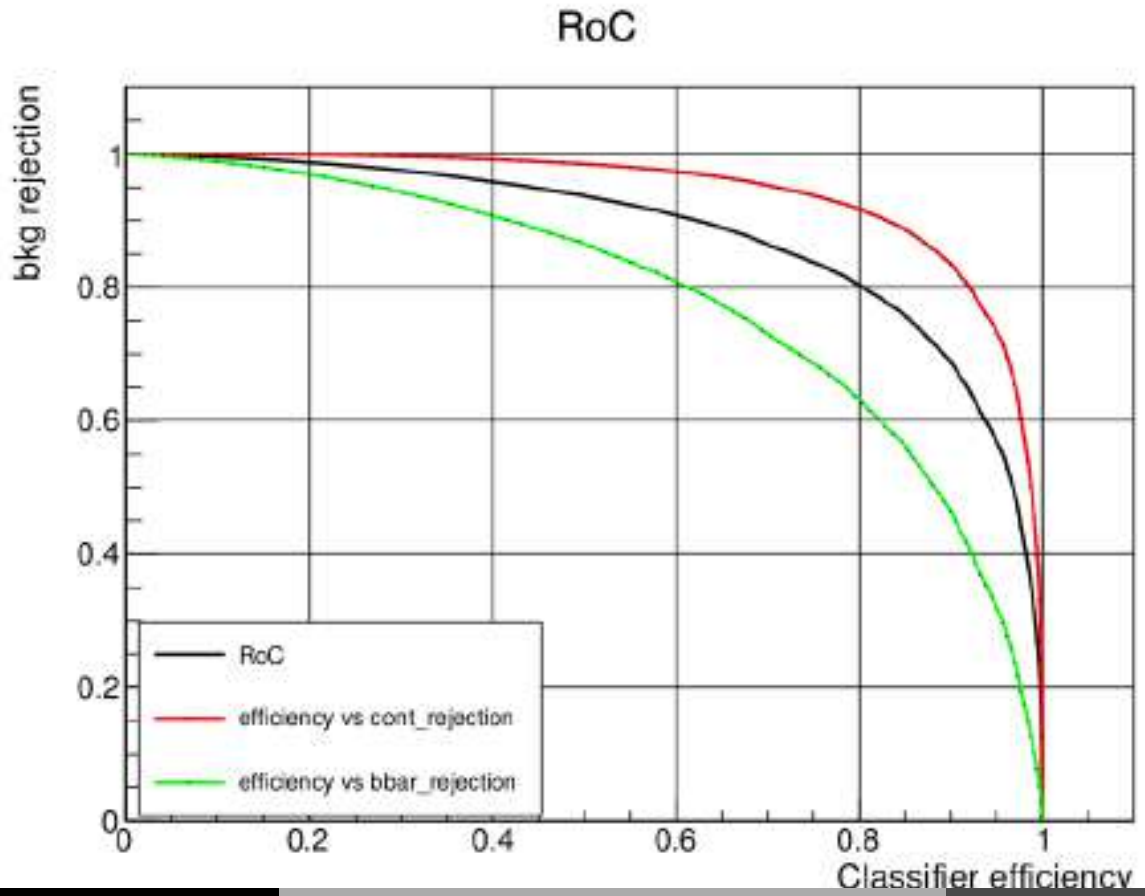
- Контролируем переобучение на тестовой выборке отдельно для сигнальных (класс с меткой 1) и фоновых (класс с меткой 0) событий





Пример градиентного бустинга в анализе $B^0 \to D\omega\pi$ (Belle II)

- Проверяем качество классификации: зависимость эффективности классификатора от доли выброшенного фона
- В идеале площадь под кривой AuC = 1. В данном случае AuC = 0.9

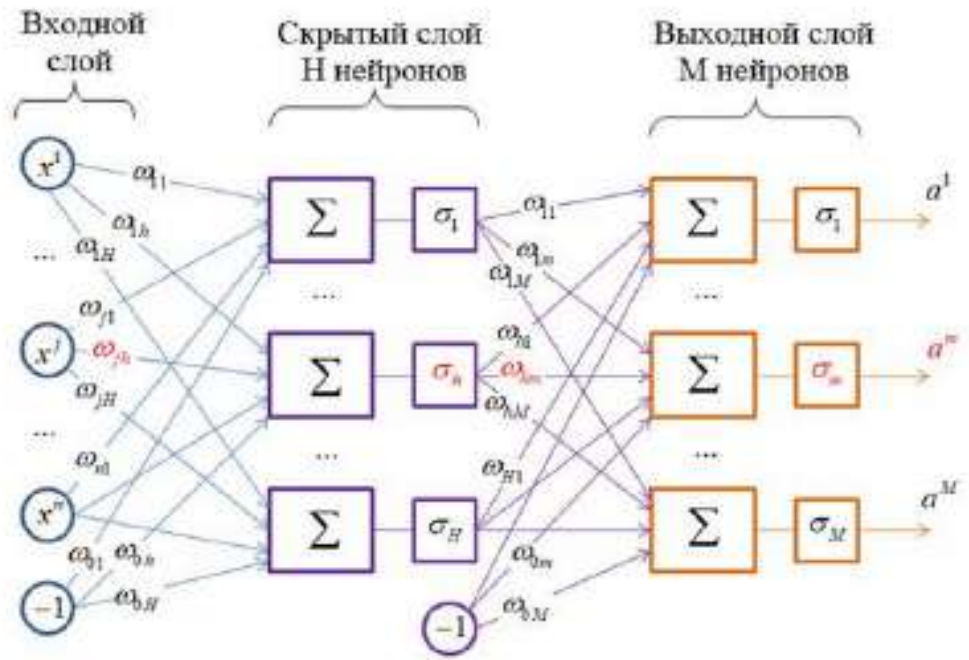


Нейронные сети или бустинг?

- Бустинг хорошо работает и конкурирует с полносвязными NN на табличных данных: строки - события, столбцы - переменные. Такой формат не учитывает связи между строками. Если строки - это векторы (эмбеддинги), описывающие хиты в дрейфовой камере или калориметре, то, чтобы связать хиты в трек или собрать в кластер, необходимо учесть пространственную и/или иную структуру данных.
- NN более гибкие, они умеют работать с разным форматом данных, учитывают связи между эмбеддингами. Более того, они умеют решать другие задачи, например, генерировать новые данные, учитывая характерные особенности обучающей выборки.

Многослойная нейронная сеть

- Входной вектор $x = [-1, x_1, \ldots, x_n]$
- Линейная комбинация признаков на каждом сумматоре $\sum h = \sum_{j=0}^{n} \omega_{jh} x_j,\ h = 1, \ldots H$
- Нелинейная функция активации $u_h = \sigma_h \left(\sum_{j=0}^{n} \omega_{jh} x_j \right),\ h = 1, \ldots, H$
- Выходной вектор $a_m = \sigma_m \left(\sum_{h=0}^{H} \omega_{hm} u_h \right)\ m = 1, \ldots, M$



Алгоритм BackProp

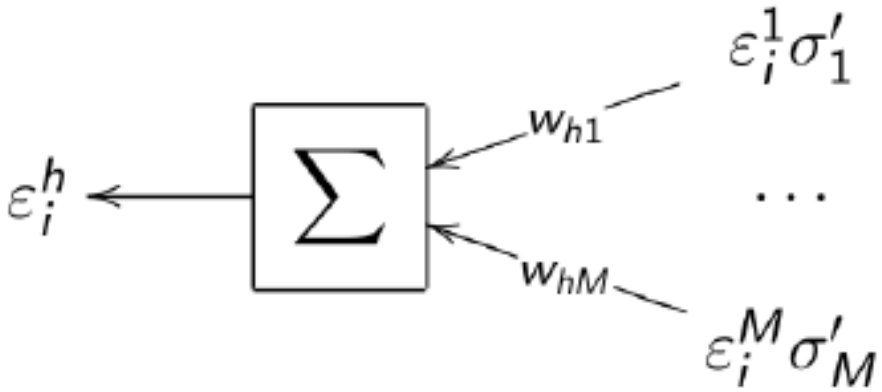
- В алгоритме градиентного спуска производная суперпозиции функций вычисляется методом обратного распространения ошибки
- Пусть $\mathcal{L}_i(w) = \sum_{m=1}^{M}(a^m(x_i) - y_i^m)^2$

$$\frac{\partial \mathcal{L}_i(w)}{\partial a^m} = a^m(x_i) - y_i^m = \epsilon_i^m$$

$$\frac{\partial \mathcal{L}_i(w)}{\partial u^h} = \sum_{m=1}^{M}(a^m(x_i) - y_i^m)\sigma'_m\omega_{hm} = \sum_{m=1}^{M}\epsilon_i^m\sigma'_m\omega_{hm} = \epsilon_i^h$$

$$\frac{\partial \mathcal{L}_i(w)}{\partial \omega_{hm}} = \epsilon_i^m\sigma'_m u^h(x_i),\ m = 1,\ldots,M,\ h = 0,\ldots,H$$

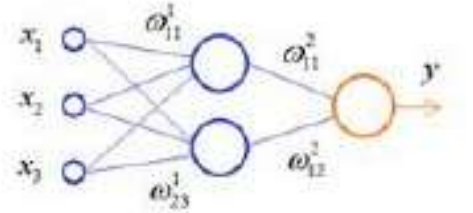
$$\frac{\partial \mathcal{L}_i(w)}{\partial \omega_{jh}} = \epsilon_i^h\sigma'_h x_j\ m = 1,\ldots,M,\ h = 0,\ldots,H$$

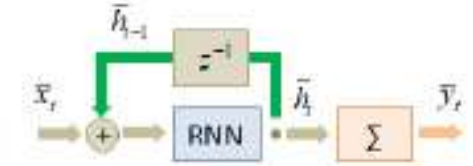


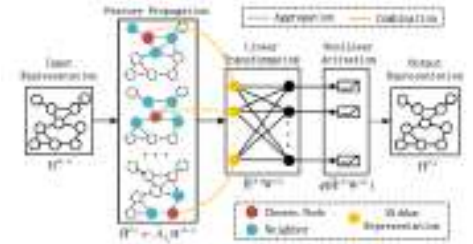
ϵ_i^h вычисляется по ϵ_i^m, если запустить сигнал с конца сети.

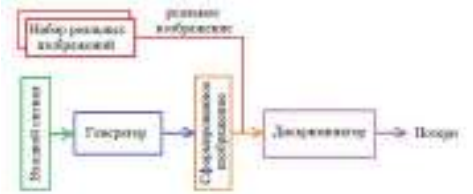
Популярные архитектуры NN

- Полносвязные сети (MLP)
  (Таблицы)
- Сверточные сети (CNN)
  (Картинки)
- Рекуррентные сети (RNN)
  (временные ряды, последовательности)
- Графовые сети (GNN)
  (графы)
- Генеративно-состязательные сети
  (GAN)
  (генерация данных)







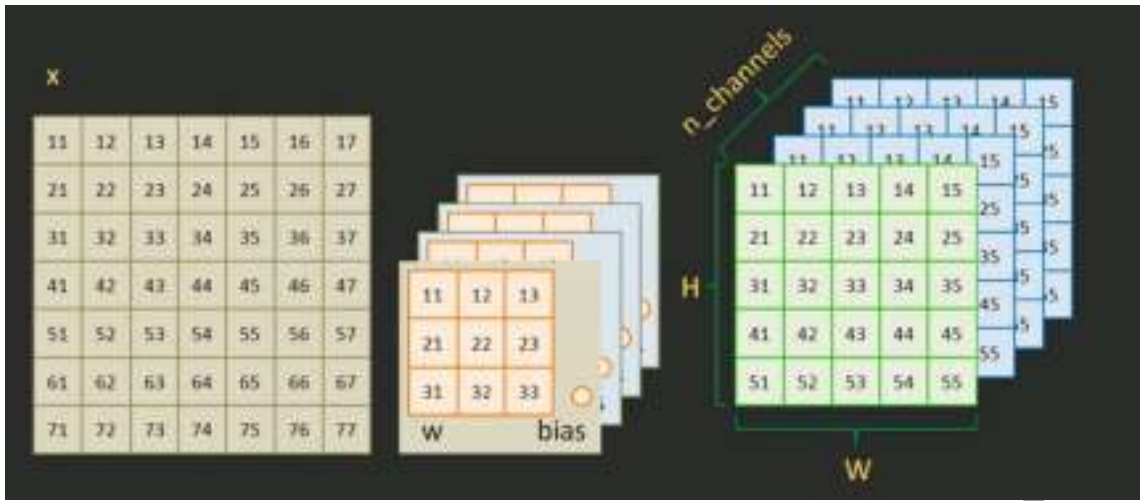


Сверточный слой CNN

- Используется идея свертки двумерных последовательностей. Одна из последовательностей - матрица пикселей входного изображения, вторая последовательность - матрица весовых коэффициентов (ядро) меньшей размерности, чтобы выделять признаки в некоторой области входного изображения.

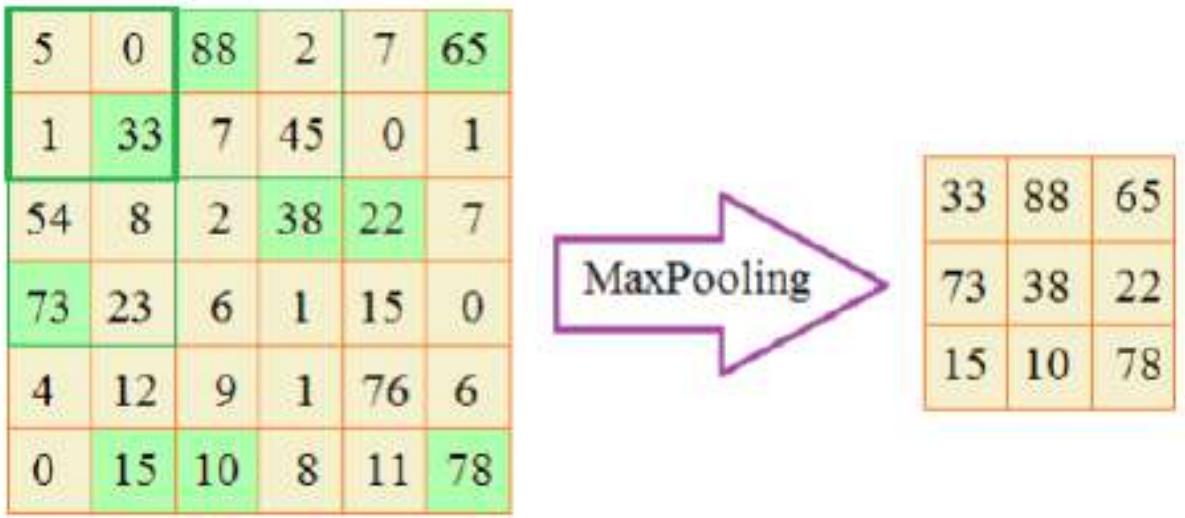
$$(x * w)[i, j] = \sum_{a=-A}^{A} \sum_{b=-B}^{B} w_{ab} x[i + a, j + b]$$

- Такой подход позволяет выделять на изображении характерные участки в соответствии с конфигурацией ядра. Когда на изображении появляется характерный фрагмент, подходящий под ядро, на выходе формируется значимый отклик.

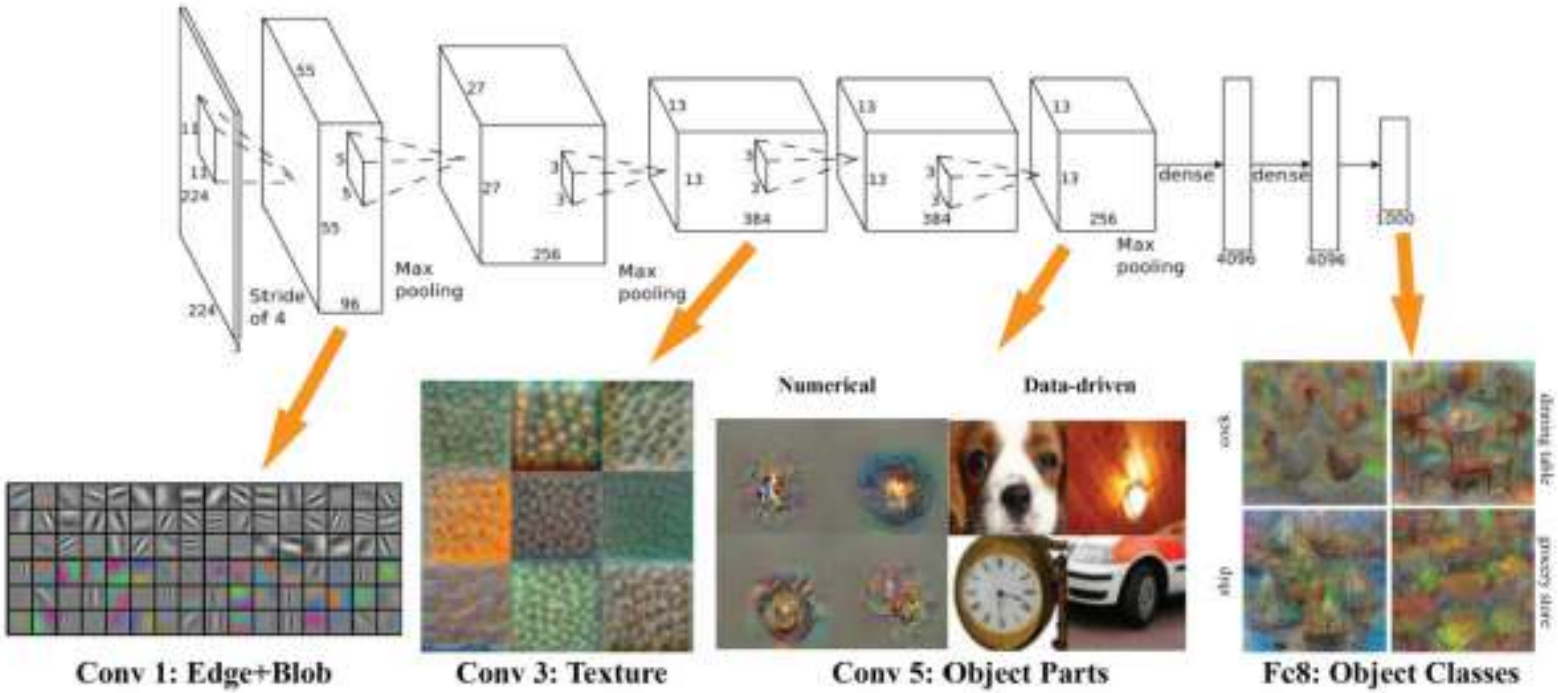


Pooling-слой

- Pooling - объединяющий нейрон, агрегирующий данные (в основном, максимум) на участке карты признаков. Вся карта признаков разбивается на непересекающиеся блоки, в прелах которых отбирается наибольшее значение.
- Pooling обобщает и выделяет все более крупные признаки. В итоге уменьшается размер изображения.
- Последовательное использование сверточных слоев вместе с пулингом приводит к векторизации изображения. Получается вектор признаков, который далее поступает в полносвязный слой, где решается задача классификации.

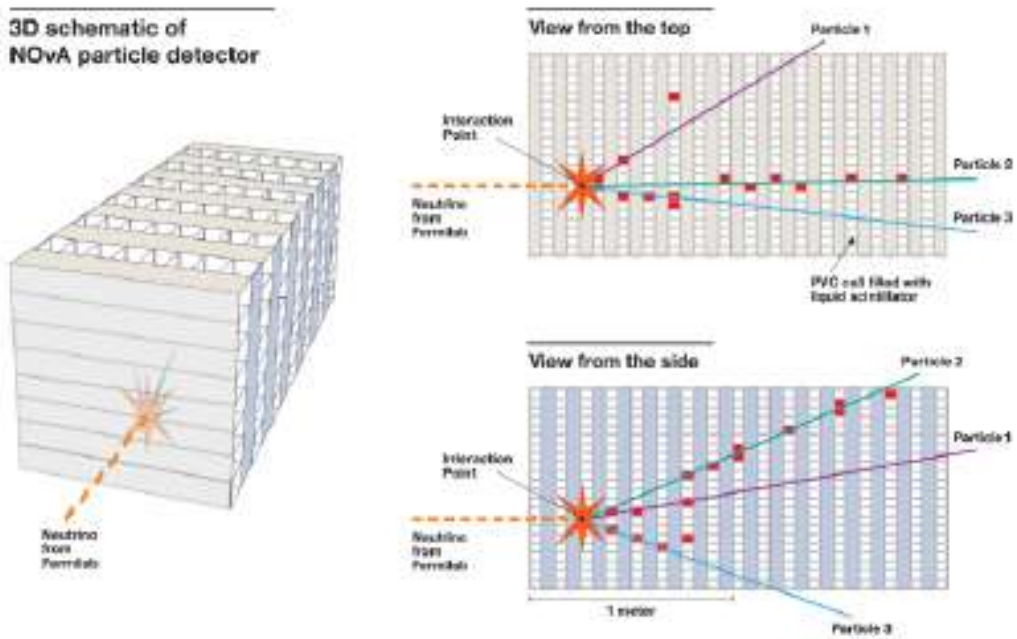


Архетиктура AlexNet



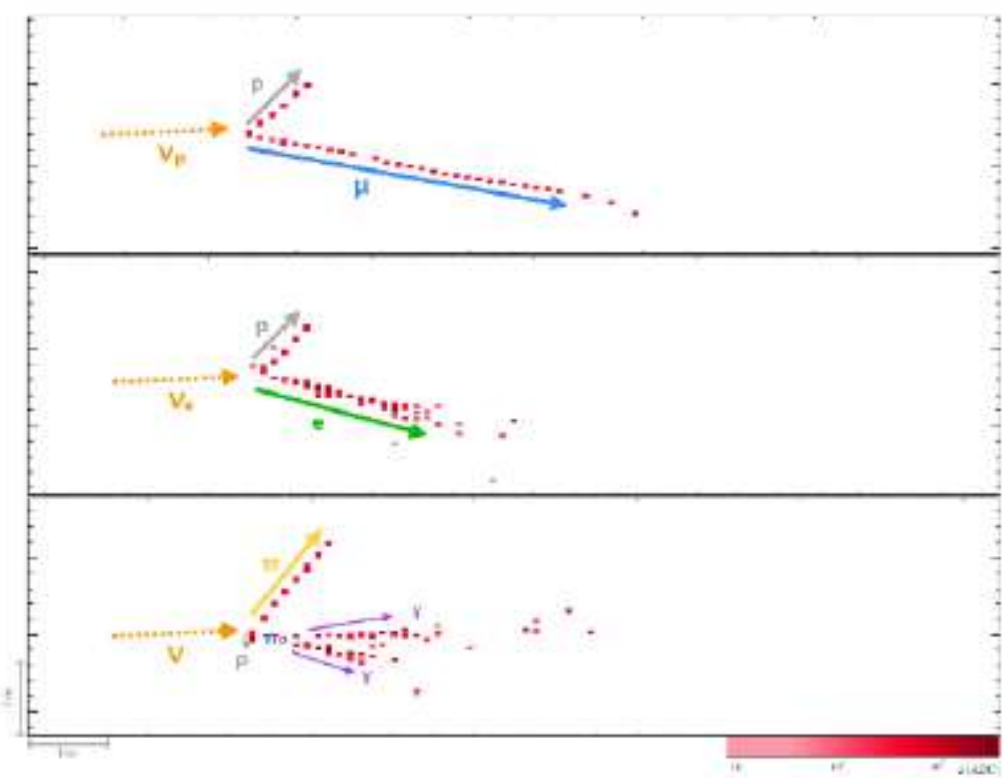
Сверточные сети в нейтринных экспериментах (NOvA)

NOvA эксперимент (Фермилаб) изучает осцилляции $\nu_\mu \to \nu_e$ и иерархию масс. Главный инжектор разгоняет протоны (E=120 ГэВ), которые при столкновении с графитовой мишенью образуют π и K-мезоны. Через распадный механизм выделяются ν_μ ($\bar{\nu}_\mu$)-нейтрино. Нейтрино взаимодействуют с ядром вещества детектора: $\nu_\mu(\nu_e) + n \to \mu^-(e^-) + p$ (канал заряженного тока, появляется лептон $(\mu^-/(e^-))$ и $\nu_\mu + n/p \to \nu_\mu + n/p$ (канал нейтрального тока, возможен распад ядра с вылетом π^0). Используется ячеистый детектор с жидким сцинтиллятором. Группировка сигналов от соседних ячеек формирует 2D-проекцию трека.



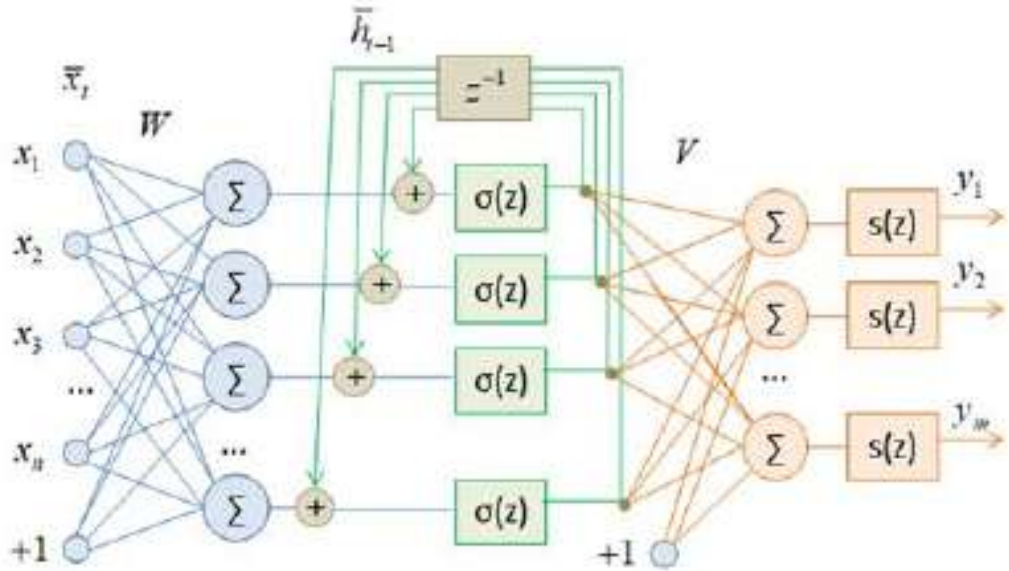
Сверточные сети в нейтринных экспериментах (NOvA)

Сверточные сети принимают данные как 2D (или 3D) изображения, где пиксели - это ячейки детектора с сигналом от сцинтиллятора. CNN классифицируют тип нейтринного события ($\nu_\mu \rightarrow \nu_e$ или $\nu_\mu \rightarrow \nu_\mu$) и отбрасывают фон (космика). Мюонный трек - длинный прямой, электронный трек - короткий каскад, нейтральный пион - неупорядоченное скопление энергии.



Рекуррентный слой RNN

- $x_1, \ldots, x_t, \ldots, x_n$ - временной ряд (последовательный поток входных векторов со сложной корреляцией между ними).
- $y_1, \ldots, y_t, \ldots, y_n$ - поток выходных векторов (скаляров)
- $h_1, \ldots, h_t, \ldots, h_n$ - поток векторов скрытого состояния. Каждый такой вектор отвечает за накопление информации во времени. Он учитывает все корреляции ряда до момента t. $h_t = f(x_t, h_{t-1})$ - реккурентное задание.
- $h_t = \sigma(Wx_t + h_{t-1})$
- $y_t = s(Vh_t)$
- Стохастичность градиентного спуска теряется из-за порядка входных векторов.

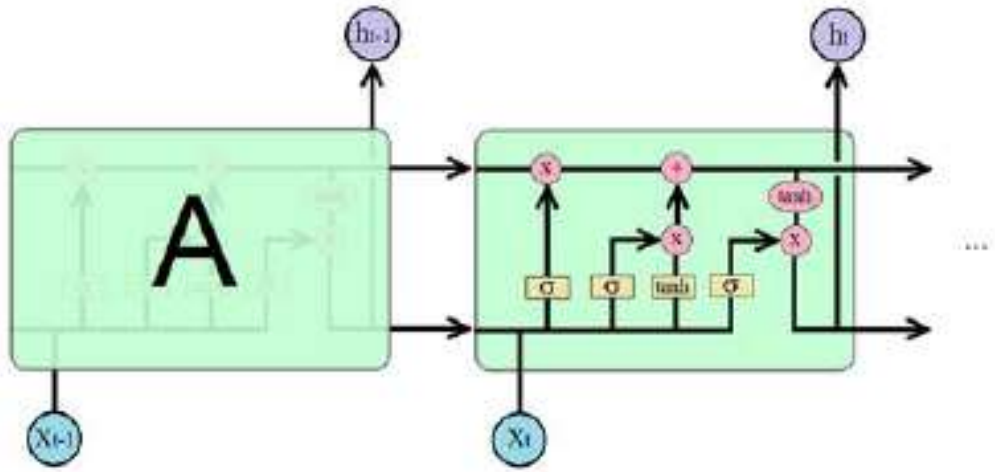


Проблемы рекуррентного слоя

- Обучение рекуррентного слоя алгоритмом BackProp имеет квадратичную сложность по длине сигнала

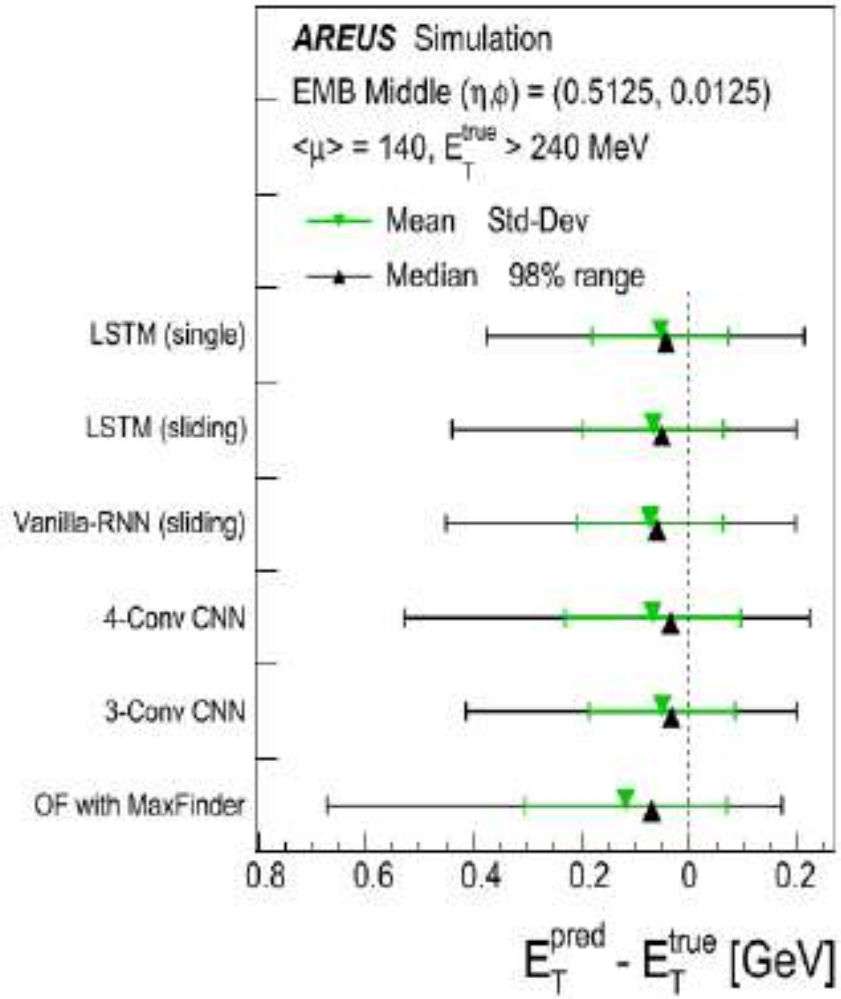
$$\frac{\partial \mathcal{L}_t}{\partial W} = \frac{\partial \mathcal{L}_t}{\partial y_t}\frac{\partial y_t}{\partial h_t}\sum_{k=0}^{t}\left(\prod_{i=k+1}^{t}\frac{\partial h_i}{\partial h_{i-1}}\right)\frac{\partial h_k}{\partial W}$$

- Для этого ограничивают предысторию (упрощают временной ряд до цепей Маркова). Для предотвращения затухания (взрыва) градиента выбирают функцию активации $\sigma(x)$ так, чтобы $\partial h_i / \partial h_{i-1} \to 1$.
- Двунаправленные RNN - сначала проходят временной ряд в одном направлении, потом в обратном.
- LSTM - попытка запомнить помимо краткосрочного контекста h_t долгий контекст C_t



Рекуррентные сети для быстрого восстановления энергии в калориметре ATLAS

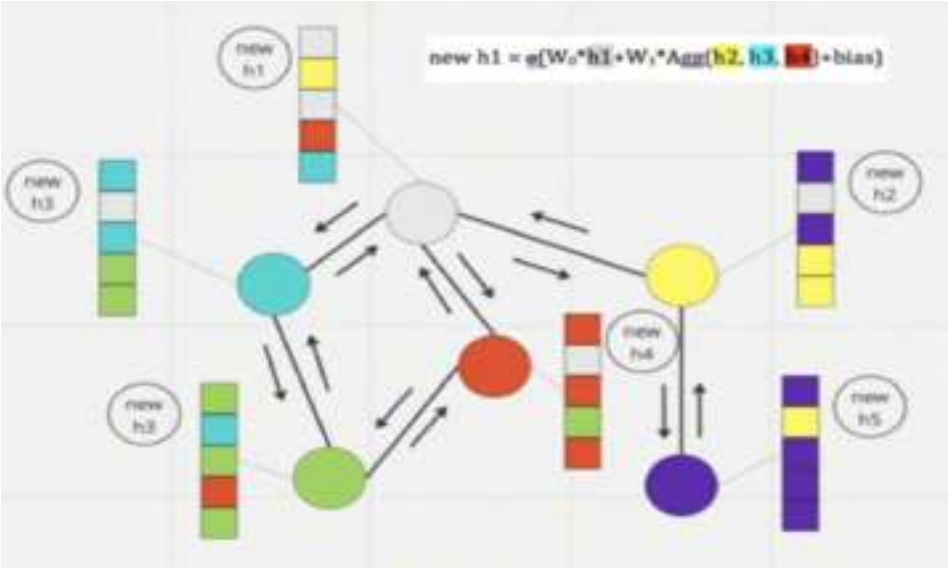
ATLAS LAr калориметр измеряет энергию фотонов, электронов и позитронов по их ионизационному сигналу. Из-за высокой светимости с частотой bunch crossing 40 МГц среднее число pile-up событий на одно зарегистрированное событие может достигать более 25 событий ($\sim$ 100 для HL-LHC). Нужно восстанавливать энергию в условиях больших фонов для всех каналов ($\sim$ $182k$). В работе https://inspirehep.net/literature/1938550 предлагается использовать RNN-сети вместо традиционного алгоритма, которые тренируются на моделировании и загружаются в прошивку FPGA. Энергетическое разрешение улучшается на 30%.



Графовый слой GNN

- Графовый слой на вход получает целый граф $\{V, E\}$, $V = \{v_i\}_{i=1,N_v}$ - множество эмбеддингов узлов, $E = \{(e_k, s_k, r_k)\}_{k=1,N_e}$ - множество эмбеддингов ребер и индексы узлов s_k и r_k, связанных с данным ребром.
- Message passing - механизм обмена сообщениями между эмбеддингами (узлов и/или ребер), когда эмбеддинг данного узла (ребра) обновляется в соответствии со значениями эмбеддингов соседних узлов (ребер). k - номер итерации, ν - номер узла (ребра).

$$h_\nu^k = UPDATE^{(k)}\left(h_\nu^{(k-1)}, AGG(\{h_u^{(k-1)}\}_{u \in N(\nu)})\right)$$



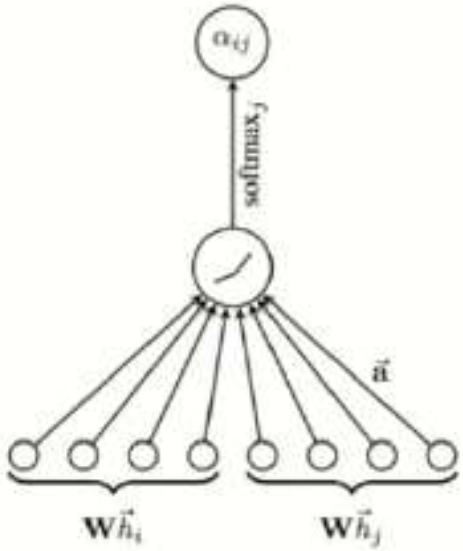
Механизм внимания в графовом слое

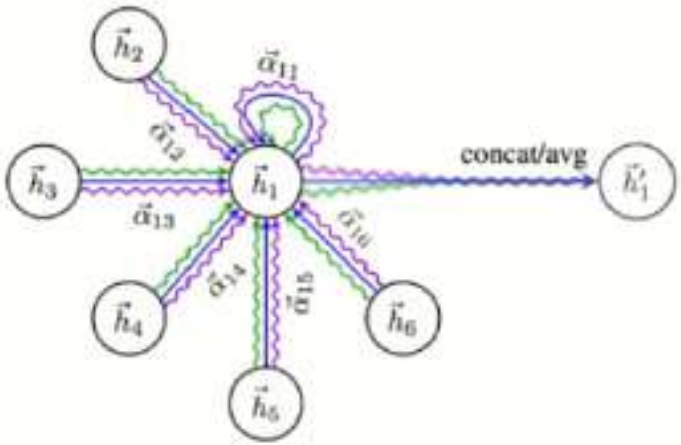
- Иногда не все ребра данного узла эквиваленты: какие-то связи предпочтительнее остальных. Механизм внимания позволяет узлам динамически взвешивать вклады соседей при агрегации в процессе обучения.

$$AGG = \sum_{u \in \{N(\nu) \cup \{\nu\}\}} \alpha_{u,\nu}^{k-1} h_\nu^{k-1}, \quad h_\nu^k = UPDATE(AGG)$$

$$\alpha_{u,\nu}^{(k-1)} = \frac{exp(f(a^T[Wh_u^{(k-1)}, Wh_\nu^{(k-1)}])}{\sum_{u \in N(\nu)} exp(f(a^T[Wh_u^{(k-1)}, Wh_\nu^{(k-1)}])}$$

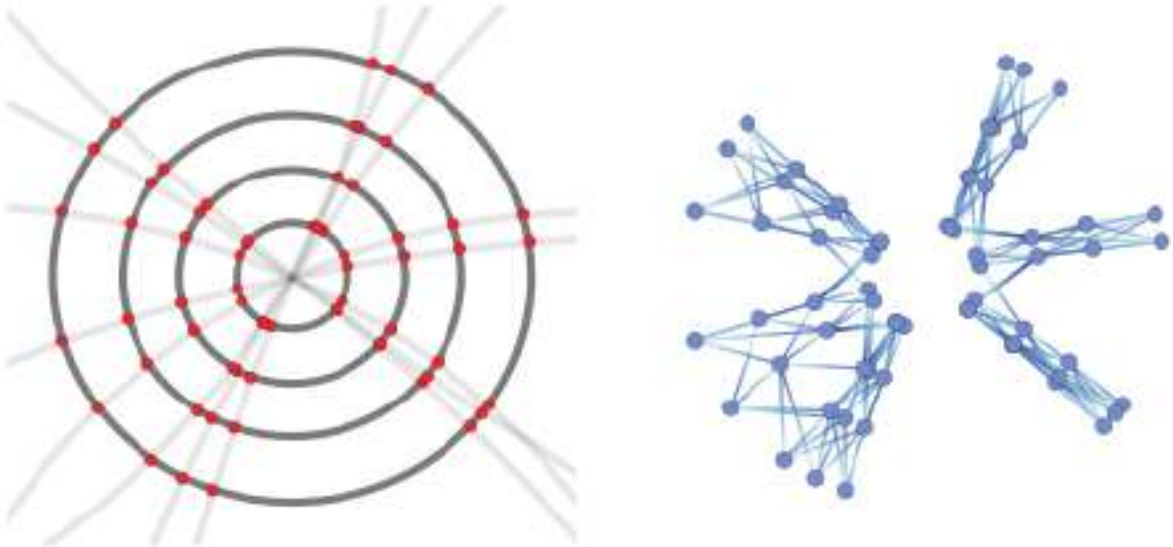
- $(f(a^T[Wh_u^{(k-1)}, Wh_\nu^{(k-1)}])$ - полносвязный слой, где $f = LeakyReLU$.





Графовые сети для реконструкции заряженных треков

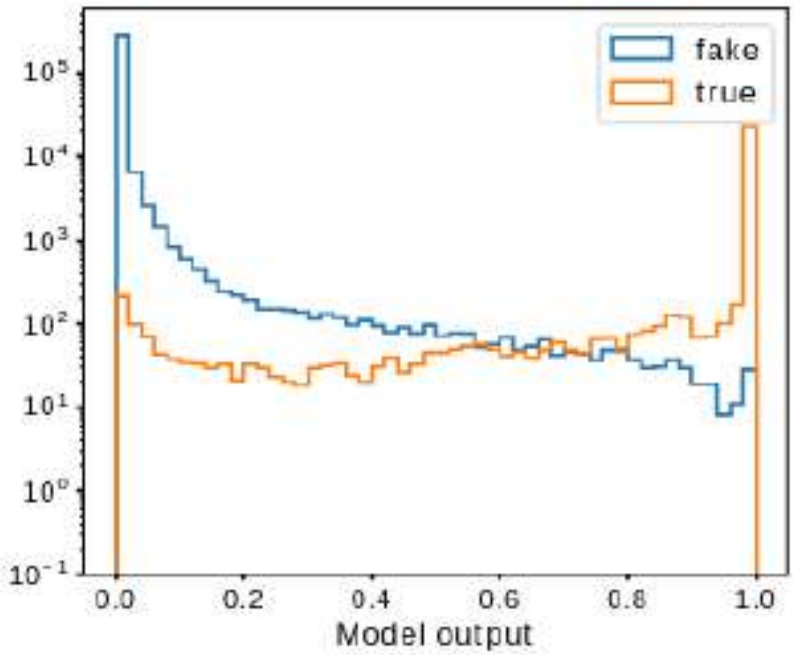
В работе NeurIPS Workshop ”Machine Learning and the Physical Sciences” тестируются GNN-сети для реконструкции заряженных треков на TrackML-датасете, моделирующем условия HL-LHC (в среднем 200 взаимодействий / bunch crossing). Узлы графа - цилиндрические координаты хитов, ребра графа - пара хитов на соседних слоях треккера, которые прошли предварительную комбинаторную обработку.

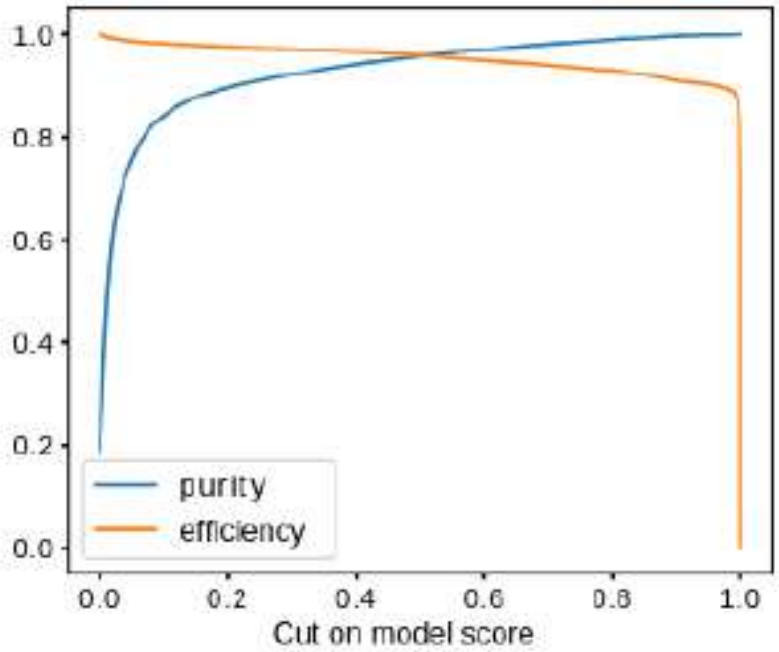


Графовое представление сегментов хитов в треккере.

Графовые сети для реконструкции заряженных треков

До графового слоя данные пропускаются через энкодер, который формирует эмбеддинг. Решается задача классификации на ребрах: часть трека / не часть трека. Efficiency - отношение числа ребер-треков, прошедших условие на классификатор, к общему числу всех ребер-треков. Purity - отношение числа ребер-треков, прошедших условие на классификатор, к общему числу всех ребер, прошедших отбор.

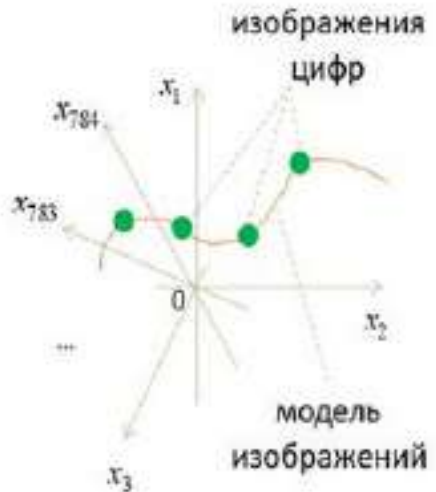


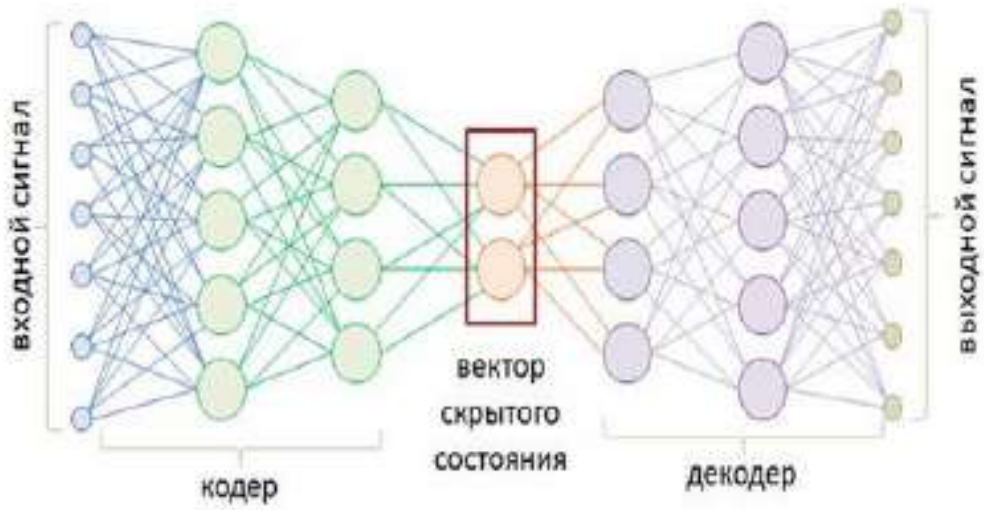


Автоэнкодер

- Автоэнкодер - тип сети, используемый для эффективного представления данных. В процессе обучения автоэнкодер пытается уловить такое многообразие точек в исходном векторном пространстве, которое наилучшим образом передает особенности входных данных.
- Энкодер кодирует входной сигнал в некотором латентном пространстве с меньшей размерностью. Декодер разворачивает данные в новое состояние.
- Автоэнкодер обучается на разности входных и выходных данных (обучение без учителя).

$$MSE = \sum_{i=1}^{N} (x_i - y_i)^2$$





Совместное обучение автоэнкодера и классификатора

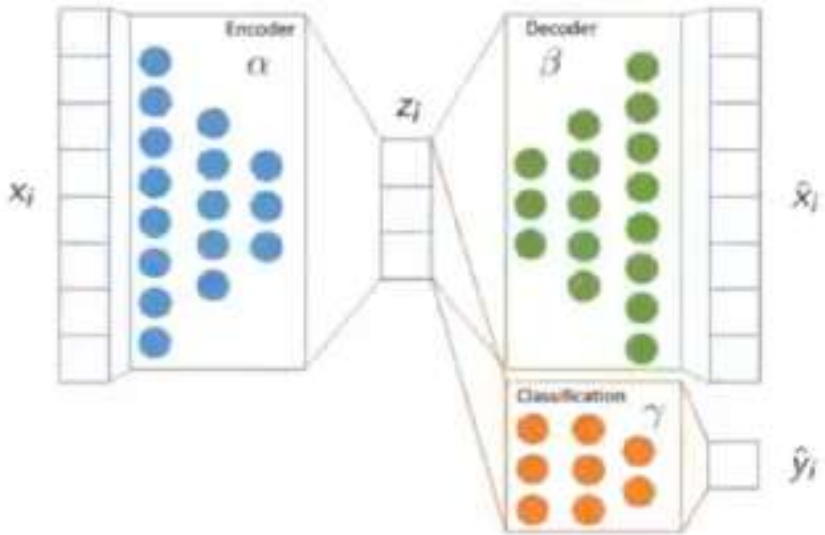
Данные: размеченные $(x_i, y_i)_{i=1}^{k}$, неразмеченные $(x_i)_{i=k+1}^{\ell}$

Найти:

$z_i = f(x_i, \alpha)$ — кодировщик

$\hat{x}_i = g(z_i, \beta)$ — декодировщик

$\hat{y}_i = \hat{y}(z_i, \gamma)$ — предиктор



Задаются функции потерь:

$\mathscr{L}(\hat{x}_i, x_i)$ — реконструкция

$\tilde{\mathscr{L}}(\hat{y}_i, y_i)$ — предсказание

Критерий: совместное обучение автокодировщика и предсказательной модели (классификации, регрессии или др.):

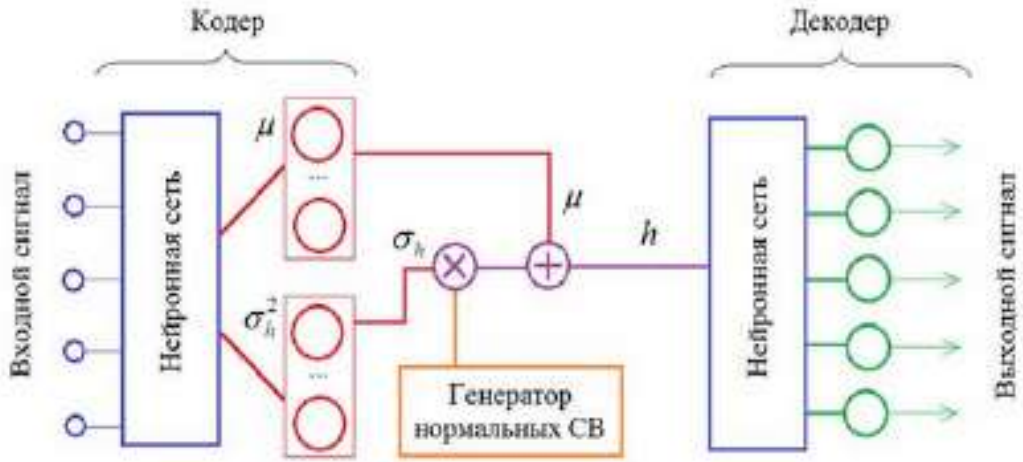
$$\sum_{i=1}^{\ell} \mathscr{L}\big(g(f(x_i, \alpha), \beta), x_i\big) + \lambda \sum_{i=1}^{k} \tilde{\mathscr{L}}\big(\hat{y}(f(x_i, \alpha), \gamma), y_i\big) \to \min_{\alpha, \beta, \gamma}$$

Вариационный автоэнкодер VAE

- VAE учится не просто сжимать данные, а моделирует распределение кодируемых переменных, что позволяет генерировать новые данные, похожие на обучающие.
- VAE пытается формировать область точек в соответствии с заданным распределением, $\mathcal{N}(0, 1)$. Это делается путем трансформации латентного пространства в компактное многообразие предопределенной формы.
- VAE не только обучается на входных данных, но минимизирует расхождение между формируемым распределением кодера и приором $\mathcal{N}(0, 1)$. В качестве критерия минимизации выступает расстояние Кульбака-Лейблера.

$$D_{KL}(p, q) = \int_{-\infty}^{\infty} p(x)\ log\left(\frac{p(x)}{q(x)}\right) dx$$

- Итоговая функция потерь становится двухкомпонентной: $MSE + \alpha D_{KL}$.



Генеративные модели GAN

- VAE генерирует ”смазанные”по батчу данные за счет MSE, качество генерации становится менее четким (слабо-выраженная индивидуальность)
- Идея GAN - использовать в качестве критерия качества новую сеть (дискриминатор $D(x, \beta)$), которая решает задачу бинарной классификации (отличить генерируемое изображение $G(z_i, \alpha)$ от реального x_i).

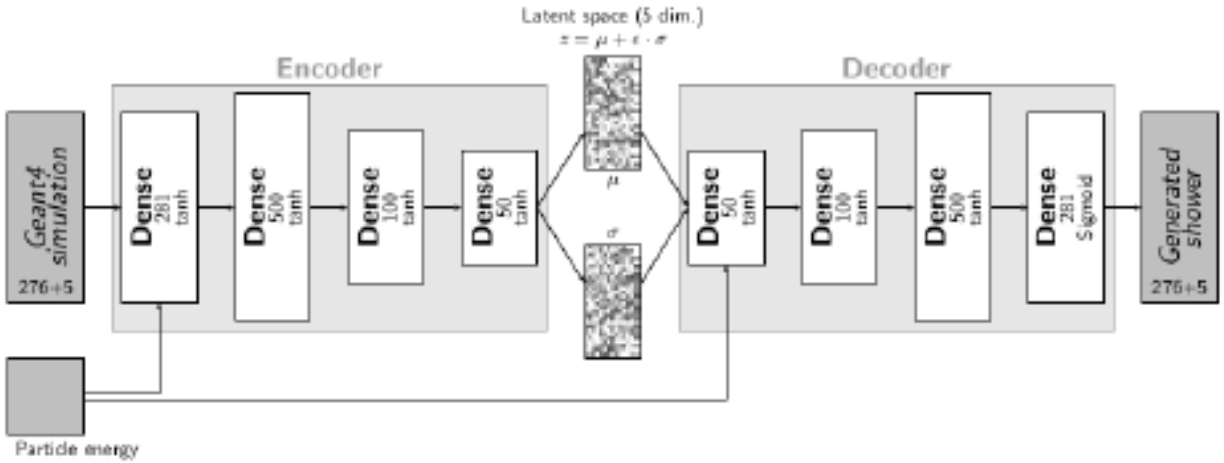
$$\sum_{i=1}^{m} logD(x_i, \beta) + log(1 - D(G(z_i, \alpha), \beta)) \rightarrow max(\beta)$$

- Генератор (это может быть VAE) учится ”обмануть”дискриминатор

$$log(1 - D(G(z_i, \alpha), \beta)) \rightarrow min(\alpha)$$

Быстрое моделирование на основе VAE/GAN

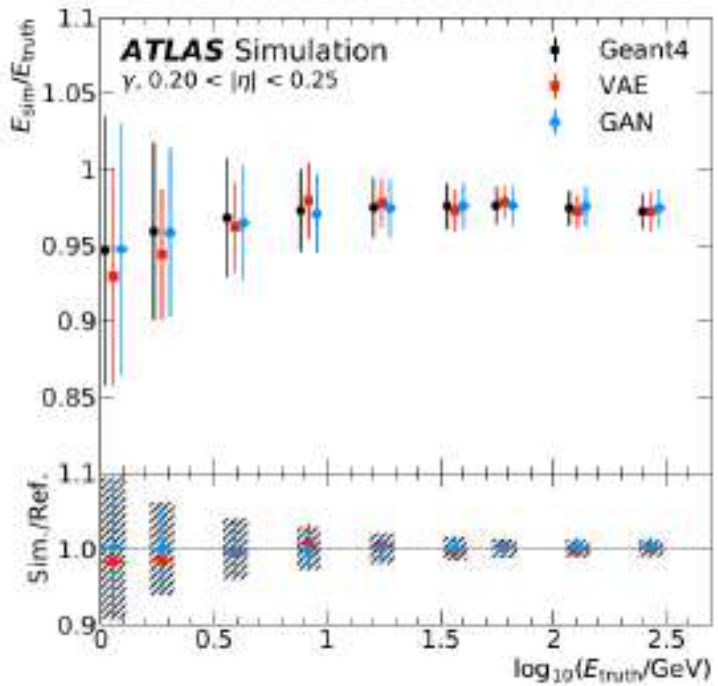
- ATLAS / CMS разрабатывают пакеты быстрого моделирования треккера / калориметра с использованием VAE / GAN-техники.
- В работе Deep generative models for fast photon shower simulation in ATLAS изучается возможность быстрого моделирования фотонных ливней в EM-калориметре ATLAS. Фотонные ливни генерируются в диапазоне энергий от 1 ГэВ до 262 ГэВ в некоторой области калориметра, которая обладает более однородной геометрией.
- Обучающая выборка генерируется GEANT4.



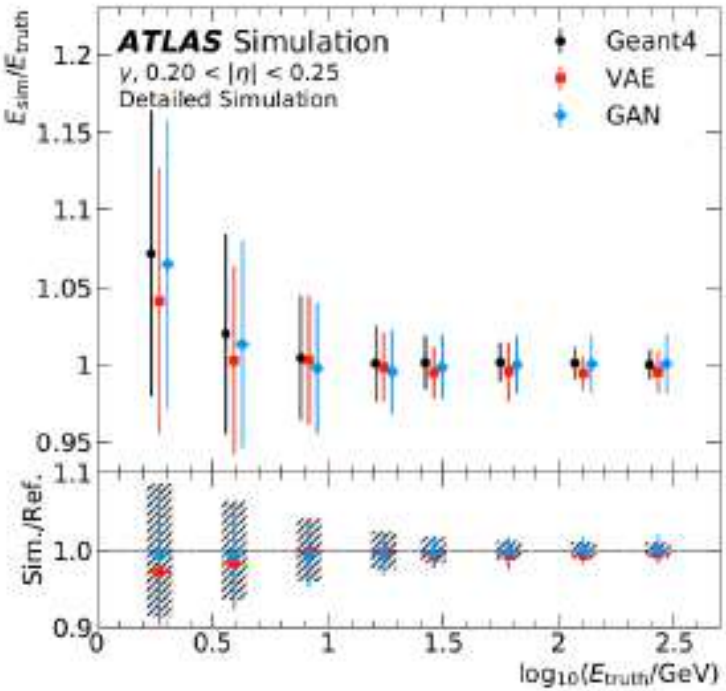
-

Быстрое моделирование на основе VAE/GAN

- Энерговыделение (энергетическое разрешение) в калориметре как функция энергии первичного фотона. (a) - обучающая выборка не учитывает crosstalk шумы. (b) - детальное моделирование с учетом шумов.



(a)



(b)

Заключение

- Машинное обучение - компьютерная автоматизация научного подхода. Почти все базовые подходы ML имеют под собой строгое математическое обоснование.
- Однако в глубоком обучении содержится огромное количество настраиваемых параметров, которые превращают модель в ”черный ящик”, который сложно интерпретировать. Нужно выбирать - либо интерпретируемая модель, либо модель, которая работает.

- Компактификация глубоких сетей без потери их предсказательной способности является актуальной математической задачей.
- Интерпретируемое машинное обучение позволяет анализировать, почему был получен такой ответ. Существует ряд приемов в этой области (SHAP - анализ вклада каждого признака в предсказание на базе теории игр, LIME - локальное упрощение модели), но полной картины еще нет.