DEEP LEARNING I: ВВЕДЕНИЕ

Сергей Николенко

НИУ ВШЭ — Санкт-Петербург
13 октября 2017 г.

Random facts:

- 13 октября 1881 г. между Элиэзером бен-Иегудой и его друзьями произошел первый сохранившийся разговор на современном иврите
- 13 октября 1957 г. на Государственном совещании Мао Цзэдун заявил: «Не надо читать много книг»...
- ...и ровно через год, 13 октября 1958 г., вышла первая книга о медвежонке Паддингтоне

TEASER

TEASER

- 10 лет назад в машинном обучении началась революция.
- Нейронные сети были давно, но обучать глубокие сети не умели.
- А теперь — научились.
- И это перевернуло весь мир машинного обучения!
- Теперь практически во всех областях лучшие результаты получаются глубокими нейронными сетями.

- Обработка изображений:

- DeepFace: Closing the Gap to Human-Level Performance in Face Verification

TEASER

- В том числе в реальном времени:

- DeepTesla: End-to-End Learning from Human and Autopilot Driving

- Распознавание речи:

- Deep Speech: Accurate Speech Recognition with GPU-Accelerated Deep Learning

- Обработка естественного языка:

- Google NMT – но тут всё не так радужно…

TEASER

- Достижения, ранее казавшиеся немыслимыми:

ПЛАН

- Что же такое глубокие нейронные сети, как они работают и почему так хороши?
- Наш план:
  (1) начнём с того, что такое вообще машинное обучение, какие бывают задачи;
  (2) затем перейдём к нейронным сетям, обсудим, как они работают,
  (3) изучим массу разных архитектур нейронных сетей, обычно глубоких или вообще рекуррентных,
  (4) а также массу разных приёмов о том, как лучше обучать нейронные сети.
- Но начать придётся не сразу с нейронов.
- Сначала — о том, что мы вообще делаем.

ЗАДАЧИ МАШИННОГО ОБУЧЕНИЯ

ИСТОРИЯ

- Нейронные сети появились даже раньше, чем AI и ML как наука.
- AI – тест Тьюринга (1950), Дартмутский семинар (1956). Затем:

1950-60-е	время больших надежд и логического вывода;
1970-е	knowledge-based systems, основанные на комбинировании правил;
1980-е	возвращение нейронных сетей, второй bubble;
1990-2000-е	machine learning: байесовские методы, вероятностное обучение;
2010-е	deep learning.

ОСНОВНЫЕ ЗАДАЧИ И ПОНЯТИЯ МАШИННОГО ОБУЧЕНИЯ

ОСНОВНЫЕ ЗАДАЧИ И ПОНЯТИЯ МАШИННОГО ОБУЧЕНИЯ

- *Обучение с учителем* (supervised learning) – обучение, в котором есть некоторое число примеров с правильными ответами:
  - *обучающая выборка* (training set) – набор примеров, каждый из которых состоит из *признаков* (features, attributes);
  - у примеров есть правильные ответы – переменная (response), которую мы предсказываем; она может быть категориальная (categorical), непрерывная или ординальная (ordinal);

ОСНОВНЫЕ ЗАДАЧИ И ПОНЯТИЯ МАШИННОГО ОБУЧЕНИЯ

- *Обучение с учителем* (supervised learning) – обучение, в котором есть некоторое число примеров с правильными ответами:
  - модель *обучается* на этой выборке (training phase, learning phase), затем может быть применена к новым примерам (test set);
  - главное – обучить модель, которая не только точки из обучающей выборки объясняет, но и на новые примеры хорошо *обобщается* (generalizes);
  - иначе – оверфиттинг (overfitting).

ОСНОВНЫЕ ЗАДАЧИ И ПОНЯТИЯ МАШИННОГО ОБУЧЕНИЯ

- *Обучение с учителем* (supervised learning) – обучение, в котором есть некоторое число примеров с правильными ответами:
  - обычно нам дают просто обучающую выборку – как тогда проверить, обобщаются ли модели?
  - кросс-валидация – разбиваем выборку на тренировочный и валидационный набор (validation set);
  - перед тем как подавать что-то на вход, обычно делают предобработку, стараясь выделить из входных данных самые содержательные аспекты (feature extraction).

ОСНОВНЫЕ ЗАДАЧИ И ПОНЯТИЯ МАШИННОГО ОБУЧЕНИЯ

- *Обучение с учителем* (supervised learning) – обучение, в котором есть некоторое число примеров с правильными ответами:
  - *классификация*: есть некоторый дискретный набор категорий (классов), и надо новые примеры определить в какой-нибудь класс;
    - классификация текстов по темам, спам-фильтр;
    - распознавание лиц/объектов/текста;

- *Обучение с учителем* (supervised learning) – обучение, в котором есть некоторое число примеров с правильными ответами:
  - *регрессия*: есть некоторая неизвестная функция, и надо предсказать её значения на новых примерах:
    - инженерные приложения (предсказать температуру, положение робота, whatever);
    - финансы – предсказать цену акций;
  - то же плюс время – например, распознавание речи.

ОСНОВНЫЕ ЗАДАЧИ И ПОНЯТИЯ МАШИННОГО ОБУЧЕНИЯ

- *Обучение без учителя* (unsupervised learning) – обучение, в котором нет правильных ответов, только данные:
  - *кластеризация* (clustering): надо разбить данные на заранее неизвестные классы по некоторой мере похожести:
    - выделить семейства генов из последовательностей нуклеотидов;
    - кластеризовать пользователей и персонализовать под них приложение;
    - кластеризовать масс-спектрометрическое изображение на части с разным составом;

ОСНОВНЫЕ ЗАДАЧИ И ПОНЯТИЯ МАШИННОГО ОБУЧЕНИЯ

- *Обучение без учителя* (unsupervised learning) – обучение, в котором нет правильных ответов, только данные:
  - *снижение размерности* (dimensionality reduction): данные имеют огромную размерность (очень много признаков), нужно уменьшить её, выделить самые информативные признаки, чтобы все вышеописанные алгоритмы смогли работать;
  - *дополнение матриц* (matrix completion): есть разреженная матрица, надо предсказать, что на недостающих позициях.
  - Часто даны правильные ответы для небольшой части данных – semi-supervised learning.

- *Обучение с подкреплением* (reinforcement learning) – обучение, в котором агент учится из собственных проб и ошибок:
  - *многорукие бандиты*: есть некоторый набор действий, каждое из которых ведёт к случайным результатам;
  - *exploration vs. exploitation*: как и когда от исследования нового переходить к использованию того, что уже изучил;
  - *credit assignment*: конфетку дают в самом конце (выиграл партию), и надо как-то распределить эту конфетку.

ОСНОВНЫЕ ЗАДАЧИ И ПОНЯТИЯ МАШИННОГО ОБУЧЕНИЯ

- *активное обучение* (active learning) – как выбрать следующий (относительно дорогой) тест;
- *обучение ранжированию* (learning to rank) – ординальная регрессия, как породить упорядоченный список (интернет-поиск);
- *бустинг* (boosting) – как скомбинировать несколько слабых классификаторов так, чтобы получился хороший;
- *выбор модели* (model selection) – где провести черту между моделями с многими параметрами и с немногими.

ВЕРОЯТНОСТЬ В МАШИННОМ ОБУЧЕНИИ

- Во всех методах и подходах очень пригодится метод, который мог бы не просто выдавать ответ, а ещё оценивать, насколько модель уверена в этом ответе, насколько модель хорошо описывает данные, как изменятся эти величины при дальнейших экспериментах и т.д.
- Поэтому центральную роль в машинном обучении играет теория вероятностей.

ТЕОРЕМА БАЙЕСА

- Дискретные и непрерывные случайные величины.
- *Совместная вероятность* – вероятность одновременного наступления двух событий, $p(x, y)$; маргинализация:

$$p(x) = \sum_y p(x, y).$$

- *Условная вероятность* – вероятность наступления одного события, если известно, что произошло другое, $p(x \mid y)$:

$$p(x, y) = p(x \mid y)p(y) = p(y \mid x)p(x).$$

- *Теорема Байеса* – из предыдущей формулы:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{p(x|y)p(y)}{\sum_{y'} p(x|y')p(y')}.$$

- *Независимость*: x и y независимы, если $p(x, y) = p(x)p(y)$.

ТЕОРЕМА БАЙЕСА

- Главная формула машинного обучения – теорема Байеса:

$$p(\theta|D) = \frac{p(\theta)p(D|\theta)}{p(D)}.$$

- $p(\theta)$ — *априорная вероятность* (prior probability);
- $p(D|\theta)$ — *правдоподобие* (likelihood);
- $p(\theta|D)$ — *апостериорная вероятность* (posterior probability);
- $p(D) = \int p(D \mid \theta)p(\theta)\mathrm{d}\theta$ — вероятность данных (evidence).

I am
My Lord
Your Lordship's
most obedient
humble Servant
T. Bayes.

ТЕОРЕМА БАЙЕСА И МОНЕТКА

- Пример – монетка, о которой мы ничего не знаем:

- Умножаем $p(\theta) = 1$ на $[0, 1]$ на $p(s \mid \theta) = \theta$, получаем $p(\theta \mid s) = 2\theta$ на $[0, 1]$.

ТЕОРЕМА БАЙЕСА И МОНЕТКА

- Пример – монетка, которую мы достали из кармана:

- Умножаем $p(\theta) = \mathrm{Beta}(\theta; 10, 10)$ на $[0, 1]$ на $p(s \mid \theta) = \theta$, получаем $p(\theta \mid s) = \mathrm{Beta}(\theta; 10, 11)$ на $[0, 1]$.

ЛИНЕЙНАЯ РЕГРЕССИЯ

- Например, *линейная регрессия*.
- Линейная модель: рассмотрим линейную функцию

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{p} x_j w_j = \mathbf{x}^\top \mathbf{w}, \quad \mathbf{x} = (1, x_1, \dots, x_p).$$

- Как найти оптимальные параметры $\hat{\mathbf{w}}$ по тренировочным данным вида $(\mathbf{x}_i, y_i)_{i=1}^N$?

ЛИНЕЙНАЯ РЕГРЕССИЯ

- Как найти оптимальные параметры $\hat{\mathbf{w}}$ по тренировочным данным вида $(\mathbf{x}_i, y_i)_{i=1}^N$?
- Метод наименьших квадратов: будем минимизировать

$$\mathrm{RSS}(\mathbf{w}) = \sum_{i=1}^{N} (y_i - \mathbf{x}_i^\top \mathbf{w})^2.$$

- Можно даже точно решить, но сейчас это не важно.

- А интересно вот что: пусть шум (ошибка в данных) распределён нормально, т.е. переменная t, которую мы наблюдаем, получается как

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2), \text{ то есть}$$

$$p(t \mid \mathbf{x}, \mathbf{w}, \sigma^2) = \mathcal{N}(t \mid y(\mathbf{x}, \mathbf{w}), \sigma^2).$$

- Кстати, о нормальном распределении:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

- Почему оно так важно?

ЛИНЕЙНАЯ РЕГРЕССИЯ

- Рассмотрим набор данных $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ со значениями $\mathbf{t} = \{t_1, \dots, t_N\}$.
- Будем предполагать, что данные взяты независимо по одному и тому же распределению:

$$p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}\left(t_n \mid \mathbf{w}^\top \phi(\mathbf{x}_n), \sigma^2\right).$$

- Прологарифмируем (опустим $\mathbf{X}$, т.к. по нему всегда условная вероятность будет):

$$\ln p(\mathbf{t} \mid \mathbf{w}, \sigma^2) = -\frac{N}{2} \ln(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{n=1}^{N} \left(t_n - \mathbf{w}^\top \phi(\mathbf{x}_n)\right)^2.$$

- И вот мы получили, что для максимизации правдоподобия по $\mathbf{w}$ нам нужно как раз минимизировать среднеквадратичную ошибку!

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t} \mid \mathbf{w}, \sigma^2) = \frac{1}{\sigma^2} \sum_{n=1}^{N} \left(t_n - \mathbf{w}^\top \phi(\mathbf{x}_n)\right) \phi(\mathbf{x}_n).$$

- Можно и апостериорное распределение получить, введя априорные распределения – они тоже будут нормальными.
- А потом и предсказательное распределение

$$p(y \mid \mathbf{x}, D) = \int_{\mathbf{w}} p(y \mid \mathbf{x}, \mathbf{w}) p(\mathbf{w} \mid D) \mathrm{d}\mathbf{w}$$

...но мы увлеклись.

- Главный вывод: во многих задачах регрессии разумно минимизировать сумму квадратов отклонений, это соответствует нормально распределённому шуму.

ЗАДАЧИ КЛАССИФИКАЦИИ

- Ещё более ясно получается для задач классификации: определить вектор $\mathbf{x}$ в один из K классов $\mathcal{C}_k$.
- Давайте каждому классу сопоставим плотность $p(\mathbf{x} \mid \mathcal{C}_k)$, найдём априорные распределения $p(\mathcal{C}_k)$, будем искать $p(\mathcal{C}_k \mid \mathbf{x})$ по теореме Байеса.
- Для двух классов:

$$p(\mathcal{C}_1 \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x} \mid \mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x} \mid \mathcal{C}_2)p(\mathcal{C}_2)}.$$

- Перепишем:

$$p(\mathcal{C}_1 \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x} \mid \mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x} \mid \mathcal{C}_2)p(\mathcal{C}_2)} = \frac{1}{1 + e^{-a}} = \sigma(a),$$

где

$$a = \ln \frac{p(\mathbf{x} \mid \mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x} \mid \mathcal{C}_2)p(\mathcal{C}_2)}, \qquad \sigma(a) = \frac{1}{1 + e^{-a}}.$$

ЗАДАЧИ КЛАССИФИКАЦИИ

- $\sigma(a)$ – *логистический сигмоид*:

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

- $\sigma(-a) = 1 - \sigma(a)$.
- $a = \ln\left(\frac{\sigma}{1-\sigma}\right)$ – *логит-функция*.

ЗАДАЧИ КЛАССИФИКАЦИИ

- Отсюда, например, *логистическая регрессия*: будем напрямую оптимизировать $\mathbf{w}$.
- Для датасета $\{\phi_n, t_n\}$, $t_n \in \{0, 1\}$, $\phi_n = \phi(\mathbf{x}_n)$:

$$p(\mathbf{t} \mid \mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n} (1 - y_n)^{1-t_n}, \quad y_n = p(\mathcal{C}_1 \mid \phi_n).$$

- Ищем параметры максимального правдоподобия, минимизируя $-\ln p(\mathbf{t} \mid \mathbf{w})$:

$$E(\mathbf{w}) = -\ln p(\mathbf{t} \mid \mathbf{w}) = -\sum_{n=1}^{N} \left[t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right].$$

- И получается сигмоид, оптимально разделяющий данные и моделирующий даже вероятности:

НЕСКОЛЬКО КЛАССОВ

- В случае нескольких классов получится

$$p(\mathcal{C}_k \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x} \mid \mathcal{C}_j)p(\mathcal{C}_j)} = \frac{e^{a_k}}{\sum_j e^{a_j}}.$$

- Здесь $a_k = \ln p(\mathbf{x} \mid \mathcal{C}_k)p(\mathcal{C}_k)$.
- $\frac{e^{a_k}}{\sum_j e^{a_j}}$ – нормализованная экспонента (softmax).
- Вывод: для задачи классификации разумно оптимизировать перекрёстную энтропию $\sum_{n=1}^{N} [t_n \ln y_n + (1 - t_n) \ln(1 - y_n)]$ и softmax (а не ошибку классификации – с ней проблемы).
- Но остаётся вопрос: как это всё оптимизировать?
- Как вообще оптимизировать сложные функции?

ГРАДИЕНТНЫЙ СПУСК

- Градиентный спуск — главный и фактически единственный способ оптимизации очень сложных функций.

- Берём градиент $\nabla E(\mathbf{w})$, сдвигаемся немножко, повторяем.

ГРАДИЕНТНЫЙ СПУСК

- Например, для логистической регрессии нужно оптимизировать

$$E(\mathbf{w}) = -\ln p(\mathbf{t} \mid \mathbf{w}) = -\sum_{n=1}^{N} \left[t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right].$$

- Воспользуемся тем, что $\sigma' = \sigma(1 - \sigma)$.
- Берём градиент:

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n) \phi_n.$$

- И дальше можно градиентным спуском.
- Можно и быстрее, но об этом позже.

ГРАДИЕНТНЫЙ СПУСК

- Градиентный спуск – локальный метод оптимизации.

- Но глобальных и нету... есть только методы, которые позволяют лучше находить хорошие локальные минимумы (и мы о них поговорим), но гарантий не даст никто.

КРАТКОЕ РЕЗЮМЕ

- Итак, в машинном обучении много разных задач, с учителем и без.
- Их обычно решают по теореме Байеса, пересчитывая наши представления о параметрах в зависимости от полученных данных.
- Для этого обычно надо оптимизировать какую-нибудь сложную функцию (например, апостериорное распределение).
- И для невыпуклых функций это обычно делают тем или иным вариантом градиентного спуска.
- А теперь – о том, при чём тут нейроны и сети.

ЧЕЛОВЕЧЕСКИЙ МОЗГ И ИСТОРИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

КОМПЬЮТЕР У НАС В ГОЛОВЕ

- Компьютер считает быстрее человека, но гораздо хуже
  - понимает естественный язык,
  - распознаёт изображения, узнаёт людей (это уже не совсем так),
  - обучается в широком смысле этого слова и т.д...
- Почему так? Как человек всего этого добивается?

КОМПЬЮТЕР У НАС В ГОЛОВЕ

- В мозге много нейронов; каждый нейрон:
  - через дендриты получает сигнал от других нейронов;
  - время от времени запускает сигнал по аксону;
  - через синапсы сигнал аксона доходит до дендритов других нейронов.

КОМПЬЮТЕР У НАС В ГОЛОВЕ

- Нейрон время от времени, стохастически, выдаёт сигналы.
- Всего нейронов очень много: 10^{11} нейронов, в среднем 7000 связей у каждого, т.е. 10^{15} синапсов.
- Но firing rate от 10 до 200 герц.
- А распознаём мы лицо за пару сотен миллисекунд.
- Это значит, что цепочки короткие, и вычисления скорее параллельные, чем последовательные.

КОМПЬЮТЕР У НАС В ГОЛОВЕ

- Вот как мы обрабатываем картинку:

ОБУЧЕНИЕ ПРИЗНАКОВ

- Другая сторона вопроса – обучение признаков.
- Мозг очень хорошо умеет обучаться на очень-очень маленькой выборке данных.
- И может адаптироваться к новым источникам информации.
- Как он это делает? Можем ли мы сделать так же?

ОБУЧЕНИЕ ПРИЗНАКОВ

- Системы обработки неструктурированной информации выглядят обычно так:

$$\text{вход} \to \text{признаки} \to \text{классификатор}$$

- Люди много десятилетий пытались придумать хорошие признаки:
  - MFCC для распознавания речи;
  - SIFT для обработки изображений;
  - ...
- Задача feature engineering: как сделать такие признаки?
- Но, может быть, можно найти признаки автоматически? Мозг ведь это как-то делает...
- Но это сложно!

ПОЧЕМУ ЭТО СЛОЖНО: РАСПОЗНАВАНИЕ ИЗОБРАЖЕНИЙ

ПЛАСТИЧНОСТЬ

- Третий аспект — пластичность мозга.

- Она показывает, что есть единый алгоритм обучения, который можно применять к самым разным ситуациям.

ПРЕДОСТЕРЕЖЕНИЕ

- Мы ещё не раз вернёмся к тому, как работает настоящий мозг и настоящие нейроны.
- Но лучше не слишком увлекаться:
  - 2004 – может ли биолог починить радиоприёмник?

  - 2016 – может ли нейробиолог понять микропроцессор?

ПРЕДОСТЕРЕЖЕНИЕ

- Мало что понимаем в целых нервных системах:
  - Human Brain Project Генри Маркрама провалился;

  - а получается пока OpenWorm (нематода *C. elegans*, 302 нейрона).

- Так что будем вдохновляться, но без фанатизма.

Суть происходящего

- Вообще говоря, всё, что мы делаем в машинном обучении, — это аппроксимация и оптимизация функций.
- Например:
  - есть функция из картинок в то, что на них изображено;
  - ещё проще: бинарная функция из пикселей картинки в «котик или не котик»;
  - функция, мягко скажем, довольно сложная; задана в виде датасета; как нам её реализовать?
- Обычный подход в машинном обучении:
  - строим какой-то класс моделей, обычно параметрический;
  - и подгоняем параметры так, чтобы модель хорошо описывала данные;
  - то есть оптимизируем какую-то функцию ошибки или функцию качества (правдоподобие, апостериорную вероятность).
- Нейронные сети — это очень мощный и гибкий класс таких моделей.
- Сложность в том, как же обучить их параметры.

ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- Давайте попробуем как в природе: есть связанные между собой нейроны, которые передают друг другу сигналы.
- McCulloch, Pitts, 1943: идея.

- Идею искусственных сетей, похожих на современные (даже с несколькими уровнями), предлагал ещё Алан Тьюринг (1948).
- Rosenblatt, 1958: перцептрон (один искусственный нейрон).
- Тогда же появился алгоритм обучения градиентным спуском.

ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- Один нейрон обычно моделируется вот так:

- Линейная комбинация входов, потом нелинейность:

$$y = h(\mathbf{w}^\top \mathbf{x}) = h\left(\sum_i w_i x_i\right).$$

- Как обучить перцептрон, и каков будет результат?

ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- Обучение *градиентным спуском*.
- Для исходного перцептрона Розенблатта ($h = \mathrm{id}$):

$$E_P(\mathbf{w}) = -\sum_{\mathbf{x}\in\mathcal{M}} y(\mathbf{x})\left(\mathbf{w}^\top\mathbf{x}\right),$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla_{\mathbf{w}}E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta t_n\mathbf{x}_n.$$

- Или, к примеру, можно делать классификацию, подставляя в качестве функции активации логистический сигмоид $h(x) = \sigma(x) = \frac{1}{1+e^{-x}}$:

$$E(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i\log\sigma(\mathbf{w}^\top\mathbf{x}_i) + (1-y_i)\log\left(1-\sigma(\mathbf{w}^\top\mathbf{x}_i)\right)\right).$$

- По сути это просто логистическая регрессия.

ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- Первый перцептрон (Rosenblatt, 1958) распознавал цифры.

ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- И люди в него верили:

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human beings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

Without Human Controls

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

1958 New York Times...

In today's demonstration, the "704" was fed two cards, one with squares marked on the left side and the other with squares on the right side.

Learns by Doing

In the first fifty trials, the machine made no distinction between them. It then started registering a "Q" for the left squares and "O" for the right squares.

Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the computer had undergone a "self-induced change in the wiring diagram."

The first Perceptron will have about 1,000 electronic "association cells" receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.

ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- 1960-е годы: изучали перцептроны.
- (Minsky, Papert, 1969): даже с нелинейностью один перцептрон задаёт только линейную разделяющую поверхность, XOR нельзя моделировать перцептроном...

- Это почему-то восприняли как большую проблему, которая ставит крест на нейронных сетях.

ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- ...хм, ну конечно! Для нелинейных функций нужна сеть перцептронов:

ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- (Brison, Hoback, 1969): предложили алгоритм обратного распространения ошибки (backpropagation).
- (Hinton, 1974): переоткрыл backpropagation, с тех пор он стал популярным.
- Во второй половине 1970-х появились многоуровневые ANN, была разработана современная теория.
- Глубокие модели появились в первой половине 1980-х гг.! Это вообще не очень хитрая идея сама по себе.

ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- Но к началу 1990-х решили, что нейронные сети использовать смысла нет, и началась «вторая зима».
- Это было потому, что тогда ни математически, ни вычислительно не могли нормально обучить большие модели.
- Нужно было обучать сети неделями, а качество, например, на распознавании речи проигрывало HMM, в других задачах проигрывало другим методам.
- John Denker, 1994: «neural networks are the second best way of doing just about anything».
- К концу 90-х на нейросетях остались только обработка изображений (Yann LeCun) и несколько других групп (Geoffrey Hinton, Yoshua Bengio).

ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- Революция deep learning начинается в 2006: в группе Джеффри Хинтона изобрели Deep Belief Networks (DBN).
- Основная идея: научились делать *обучение без учителя*, выделять признаки (при помощи машин Больцмана, RBM; появились в 1983), использовать это как предобучение.
- Компьютеры стали мощнее, появились способы передать вычисления на GPU, и вычислительно мы смогли обрабатывать гораздо более объёмные датасеты, которые как раз стали доступными.
- А потом появились новые методы регуляризации (дропаут, затем batch normalization), и RBM с предобучением тоже стали не очень нужны.
- Вот основные компоненты революции глубокого обучения.
- Осталось только понять, что всё это значит. :)
- Но сначала давайте до конца разберёмся с одним перцептроном.

ФУНКЦИИ АКТИВАЦИИ

- Есть много разных нелинейных функций, которые можно использовать в перцептроне.
- Логистический сигмоид:

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

ФУНКЦИИ АКТИВАЦИИ

- Гиперболический тангенс:

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

- Очень похоже, но ноль не является точкой насыщения.

ФУНКЦИИ АКТИВАЦИИ

- Ступенька (функция Хевисайда):

$$\mathrm{step}(x) = \begin{cases} 0, & \text{if } x < 0, \\ 1, & \text{if } x > 0. \end{cases}$$

ФУНКЦИИ АКТИВАЦИИ

- Rectified linear unit (ReLU):

$$\mathrm{ReLU}(x) = \begin{cases} 0, & \text{if } x < 0, \\ x, & \text{if } x \geq 0. \end{cases}$$

ФУНКЦИИ АКТИВАЦИИ

- Математическая мотивация для ReLU:

$$\sigma\left(x+\frac{1}{2}\right)+\sigma\left(x-\frac{1}{2}\right)+\sigma\left(x-\frac{3}{2}\right)+\sigma\left(x-\frac{5}{2}\right)+\ldots.$$

ФУНКЦИИ АКТИВАЦИИ

- Биологическая мотивация для ReLU:

$$f(I) = \begin{cases} \left(\tau \log \frac{E+RI-V_{\text{reset}}}{E+RI-V_{th}}\right)^{-1}, & \text{if } E + RI > V_{th}, \\ 0, & \text{if } E + RI \leq V_{th}, \end{cases}$$

ФУНКЦИИ АКТИВАЦИИ

- Несколько вариантов ReLU — Leaky ReLU и Parametric ReLU:

$$\mathrm{LReLU}(x) = \begin{cases} ax, & \text{if } x < 0, \\ x, & \text{if } x > 0. \end{cases}.$$

ФУНКЦИИ АКТИВАЦИИ

- Exponential LU:

$$\mathrm{ELU}(x) = \begin{cases} \alpha\left(e^x - 1\right), & x < 0, \\ x, & x \geq 0. \end{cases}$$

- Обычно ReLU хорошо работает, но последнего слова ещё не сказано.

НАСТОЯЩИЕ НЕЙРОНЫ

- Есть ли градиентный спуск в настоящих нейронах?
- Скорее всего нет (увидим позже почему).
- Хотя у Хинтона есть интересный доклад об этом… но всё-таки вряд ли.
- *Обучение по Хеббу* – neurons that fire together, wire together:

$$\Delta w_i = \eta x_i x_j.$$

- В ML привело к *сетям Хопфилда*, которые вроде как реализуют ассоциативную память, но плохо.
- Новая тема: *spike-timing-dependent plasticity* (STDP), т.е. насколько спайки зависят от временных свойств сигналов.

ОБЪЕДИНЯЕМ ПЕРЦЕПТРОНЫ В СЕТЬ

- Сеть перцептронов; выходы одних – входы других.
- Hornik, 1990: двух уровней достаточно для приближения любой функции.
- Но глубокие сети эффективнее — distributed representations:

ОБЪЕДИНЯЕМ ПЕРЦЕПТРОНЫ В СЕТЬ

- Обычно нейронные сети организованы в *слои*.
- Это очень естественно и легко параллелизуется.

ОБЪЕДИНЯЕМ ПЕРЦЕПТРОНЫ В СЕТЬ

- Но по сути мы приближаем очень сложную функцию большой композицией простых функций.
- Как теперь её обучить?
- Ответ прост: градиентный спуск. Но есть и сложности.

СПАСИБО!

Спасибо за внимание!