Нейронные сети

Зинина Анастасия

Нейронные сети как универсальная модель аппроксимации

Нейронные сети принято считать универсальным методом решения задач регрессии и классификации. Такое восприятие связано со следующим утверждением:
Колмогоров,1957 Каждая непрерывная функция $a(x)$, заданная на единичном кубе n-мерного пространства,представима в виде

$$a(x) = \sum_{i=1}^{2d+1} \sigma_i (\sum_{j=1}^{d} f_{ij}(x_{ij})),$$

где $x = [x_1, ..., x_{x\,d}]^T$-вектор описания объекта, функции σ_i и $f_{ij}(\cdot)$ являются непрерывными функциями.

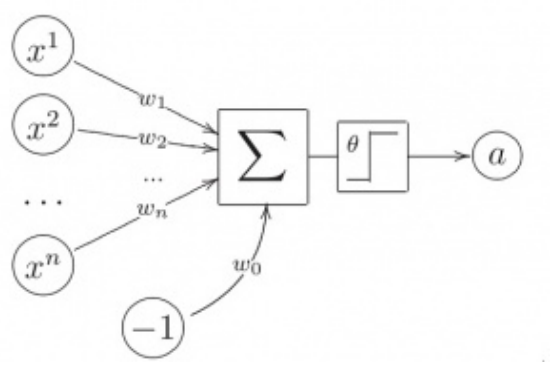
Однослойная нейронная сеть

Модель линейного порогового классификатора МакКаллока-Питтса

- Пусть все признаки $f_i(x)$ бинарные.
- Значения признаков-величины импульсов, поступающих на вход нейрона через n синапсов.
- Поступающие импульсы складываются с весами w_i
- Если суммарный импульс превышает порог активации, то нейрон выдаёт на выходе 1, иначе 0:

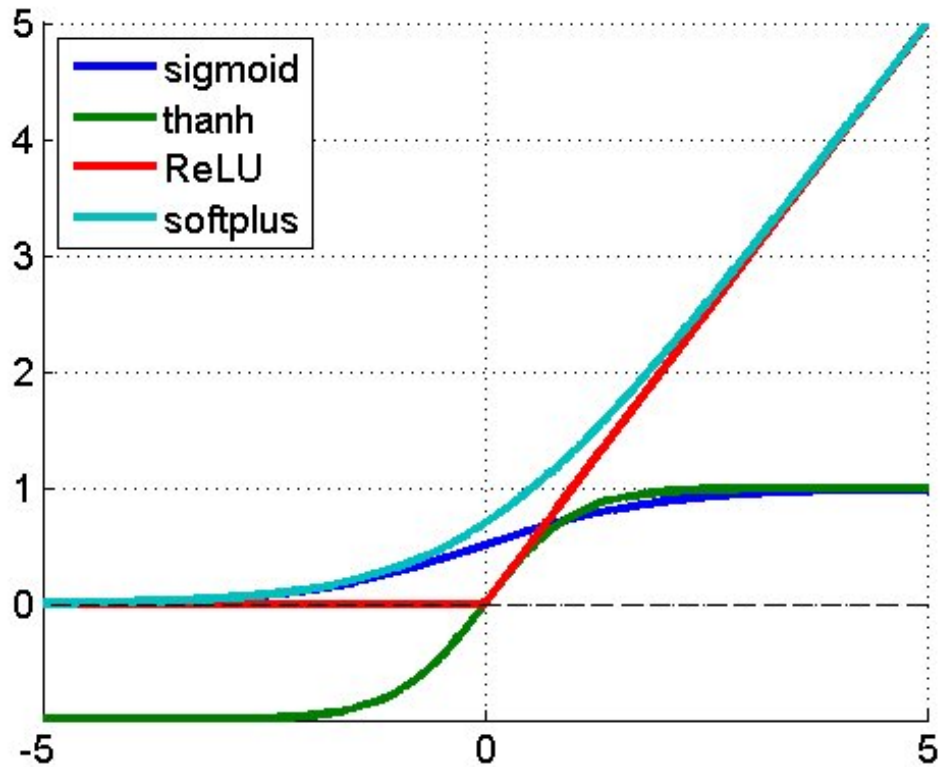
$$a(x) = \varphi(\sum_{j=1}^{n} w_j x^j - w_0),$$

где $\varphi(z) = I_{\{z \geqslant 0\}}$-функция активации.



Различные функции активации

Позже модель была обобщена на случай произвольных вещественных входов и выходов, а также произвольных функций активации.



Многослойные нейронные сети

- Однослойные сети применимы только для линейно разделимых выборок.
- Двухслойная сеть в R^n позволяет отделить произвольный выпуклый многогранник.
- Трёхслойная сеть в R^n позволяет отделить произвольную многогранную (необязательно выпуклую,необязательно связную) область.
- **Универсальная теорема аппроксимации, Hornik, 1991**
  Для любой непрерывной функции $f(x)$ найдётся нейронная сеть $a(x)$ $a(x, W) = \sigma^{(M)}(x)$, где

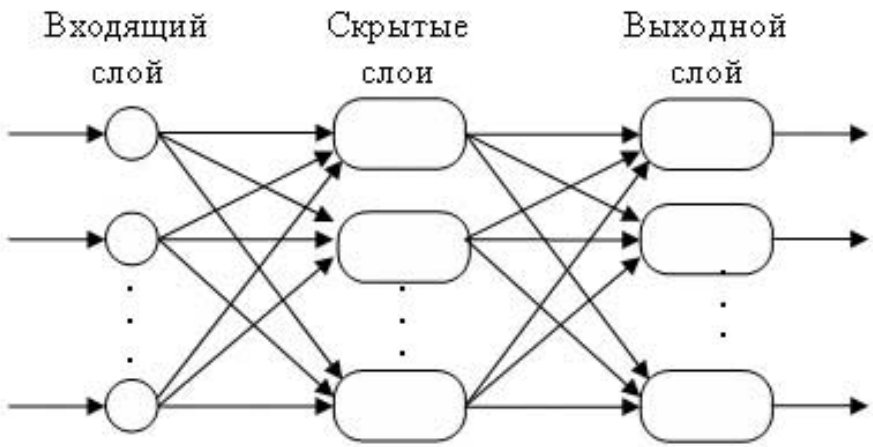
$$f^{(k)}(x) = w_0^{(k)} + W^{(k)} z^{(k-1)}(x), z^{(k)}(x) = \sigma^{(k)}(f^{(k-1)}(x)),$$

  аппроксимирующая $f(x)$ с заданной точностью.

Многослойные нейронные сети

Двухслойная нейронная сеть определяется как линейная комбинация D нейронов:

$$a(x,w) = \sigma^{(2)}(\sum_{i=1}^{D} w_i^{(2)} \sigma^{(1)}(\sum_{j=1}^{d} w_{ji}^{(1)} x_j^{(1)} + w_{0i}^{(1)}) + w_0^{(2)})$$



Аналогично определяются сети с большим числом слоёв.

Оптимизация параметров нейронной сети

Оптимальные значения параметров определяются как:

$$w^* = \underset{w}{argmin} Q(w),$$

где Q-функция ошибки.
Методы оптимизации:

- стохастическая оптимизация(генетические алгоритмы, метод отжига, метод Нелдера-Мида);
- градиентные методы

Метод обратного распространения ошибок

Рассматриваем полносвязную сеть, $X = R^n, Y = R^M$.

- Выходной слой: M нейронов, функции активации σ_m, выходы a^m, m=1,...,M.
- Скрытый слой: H нейронов, функции активации σ_h, выходы u^h, h=1,...,H.
- Синаптические связи между h-м нейроном скрытого слоя и m-м нейроном выходного слоя обозначим w_{hm}
- Перед эти скрытым слоем находится либо распределительный, либо ещё один скрытый слой с выходами $v_j, j = 1, ..., J$ и синаптическими весами $w_j h$.

Выходные значения сети на объекте x_i определяются как суперпозиция:

$$a^m(x_i) = \sigma_m(\sum_{h=0}^{H} w_{hm} u^h(x_i)),$$

$$u^h(x_i) = \sigma_h(\sum_{j=0}^{J} w_{jh} v^j(x_i)).$$

Функционал среднеквадратичной ошибки для объекта x_i:

$$Q(w) = 0.5 \sum_{m=1}^{M} (a^m(x_i) - y_i^m)^2.$$

в дальнейшем для вычисления градиента нам понадобятся частные производные:

$$\frac{\partial Q(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m - \text{ошибка на выходном слое,}$$

$$\frac{\partial Q(w)}{\partial u^h} = \sum_{m=1}^{M} (a^m(x_i) - y_i^m)\sigma_m^{'} w_{hm} = \sum_{m=1}^{M} \varepsilon_i^m \sigma_m^{'} w_{hm} = \varepsilon_i^h - \text{ошибка на скрытом слое}$$

$\varepsilon_i^h \varepsilon_i^m$, "".
Можем записать градиент Q:

$$\frac{\partial Q(w)}{\partial w_{hm}} = \frac{\partial Q(w)}{\partial a^m} \frac{\partial a^m}{\partial w_{hm}} = \varepsilon_i^m \sigma_m^{'} u^h,$$

$$\frac{\partial Q(w)}{\partial w_{jh}} = \frac{\partial Q(w)}{\partial u^h} \frac{\partial u^h}{\partial w_{jh}} = \varepsilon_i^h \sigma_h^{'} v^j,$$

Прореживание сети

Метод оптимального учечения сети(Optimal Brain Damage) удаляет те связи, к изменению которых функционал Q наименее чувствителен.
В локальном минимуме аппроксимируем Q квадратичной формой:

$$Q(w+\delta) = Q(w) + 0.5\delta^T H(w)\delta + o(||\ \delta\ ||^2)$$

NB Предполагается, что гессиан диагонален:

$$\delta^T H(w)\delta = \sum_{j=0}^{J}\sum_{h=1}^{H}\delta_{jh}^2 \frac{\partial^2 Q(w)}{\partial w_{jh}^2}$$

Исключаем параметр, если $w_{jh} + \delta_{jh} = 0$ и корректируем остальные веса, чтобы $\Delta Q(w, \delta)$ было минимальным. Т.е. определяем исключаемый признак следующим образом:

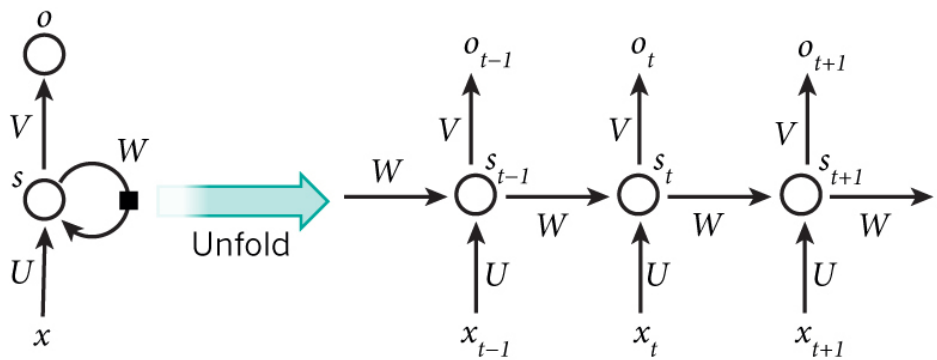
$$j = \underset{j}{argmin}(\min_{\delta}(\delta^T H\delta \mid w_{jh} + \delta_{jh} = 0))$$

Можно показать, что для этого параметра w_{jh} будем минимальным значение функции выпуклости,называемой величеной значимости :

$$S_{jh} = w_{jh}^2 \frac{\partial^2 Q(w)}{\partial w_{jh}^2}$$

Рекуррентные сети: архитектура

- Трациционная сеть прямого распространения: входные векторы (и векторы, получаемые на выходе) независимы.
- RNN: выход зависит не только от входных нейронов и нейронов скрытого слоя, но и от вычислений на предыдущем шаге.
- Теоретически могут использовать произвольное число предыдущих шагов



Рекуррентные сети: как это работает

- x_t - входной вектор на шаге t
- s_t - скрытое состояние на шаге t, "память"сети

$$s_t = f(Ux_t + Ws_{t-1})$$

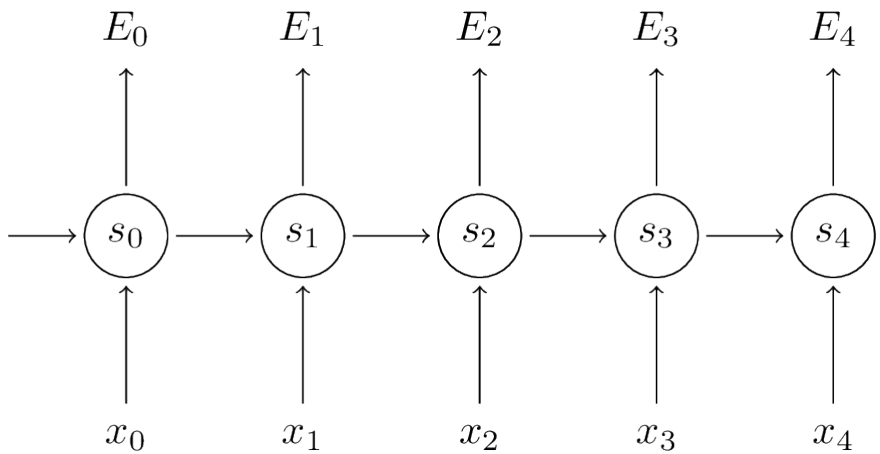
- o_t -выход на шаге t. Например, $o_t = softmax(Vs_t)$
- U, V, W - одинаковы на каждом шаге, тогда как в традиционных сетях на каждом слое свои матрицы весов.

Рекуррентные сети: обучение

- Небольшое изменение backprop'a - Backpropagation Through Time (BPTT)
- На каждом временном шаге параметры те же $\rightarrow$ градиент зависит не только от текущего шага, но и от предыдущих. Т.о. нам нужно вычислить градиент предыдущих шагов и просуммировать.
- $s_t = tanh(Ux_t + WS_{t-1})$
  $o_y = \hat{y}_t = softmax(Vs_t)$
- Функционал качества - кросс-энтропия:

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

$$E(y_t, \hat{y}_t) = \sum_t E_t(y_t, \hat{y}_t) = -\sum_t y_t \log \hat{y}_t$$



Рекуррентные сети: BPTT

- Находим градиент на каждом шаге и суммируем:

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$

- $\frac{\partial E_t}{\partial V}$ зависит только от значений на текущем шаге

$$\frac{\partial E_t}{\partial V} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial V} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial z_t}\frac{\partial \hat{z}_t}{\partial V} = (\hat{y}_t - y_t)_t,$$

где $z_t = V s_t$

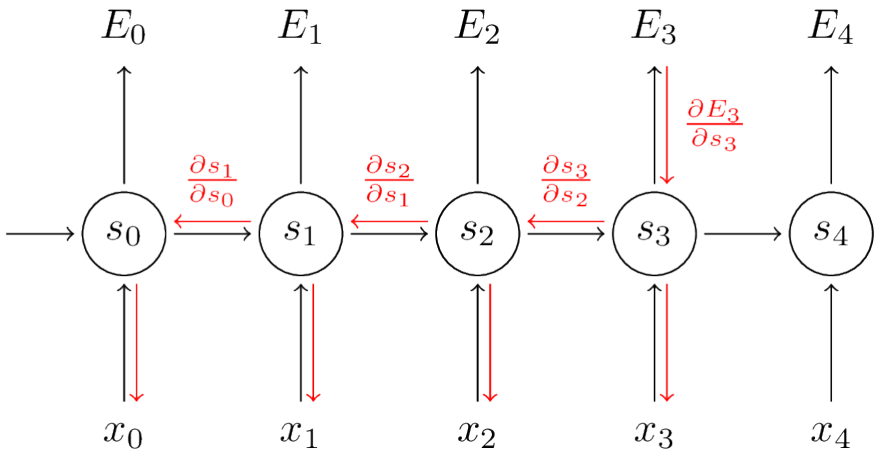
Рекуррентные сети: BPTT

- Для $\frac{\partial E_t}{\partial W}$ и $\frac{\partial E_t}{\partial U}$ ситуация иная

$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial s_t} \frac{\partial s_t}{\partial W}$$

Вспомним, что $s_t = tanh(Ux_t + Ws_{t-1})$, а s_{t-1} в свою очередь снова выражается через W и s_{t-2}.
Получаем

$$\frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial s_t} \frac{\partial s_t}{\partial s_k} \frac{\partial s_k}{\partial W}$$



- Для U аналогично

Рекуррентные сети: проблема затухающего градиента

- Использование BPTT через большое количество слоёв затруднительно
- Посмотрим на $\frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial s_t} \frac{\partial s_t}{\partial s_k} \frac{\partial s_k}{\partial W}$

  Здесь $\frac{\partial s_t}{\partial s_{k-1}} = \frac{\partial s_t}{\partial s_k} \frac{\partial s_k}{\partial s_{k-1}}$

  Т.о.

$$\frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial s_t} \left(\prod_{j=k+1}^{t} \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial W}$$

- Евклидова норма матрицы Якоби, о которой говорится выше, не превышает 1. Здесь играет роль тот факт, что функции активации и их производные по модулю не превышает 1.
- Т.о. маленькие значения в матрицах и $t - k$ уматричных умножений приводят к затуханию градиента.
- Решения проблемы затухающего градиента:
  - регуляризация;
  - использование ReLU;
  - Long Short-Term Memory (LSTM) и Gated Recurrent Unit (GRU)
- Обратная ситуация - exploding gradient. Решение - установление порога до обучения.

Рекуррентные сети: LSTM

- Борьба с затуханием градиента - gating mechanism
- Вычисление скрытых состояний ($\circ$ - покомпонентное умножение):

$$i = \sigma(x_t U^i + s_{t-1} W^i)$$

$$f = \sigma(x_t U^f + s_{t-1} W^f)$$

$$o = \sigma(x_t U^o + s_{t-1} W^o)$$

$$g = \tanh(x_t U^g + s_{t-1} W^g)$$

$$c_t = c_{t-1} \circ f + g \circ i$$

$$s_t = \tanh(c_t) \circ o$$

LSTM: интерпретация

- i,g,o - input, forget, output gates
  - input gate: какую часть вновь вычисленного состояния для данного входного вектора следует пропустить;
  - forget gate: какую часть предыдущего состояния следует пропустить;
  - output gate: какую часть внутреннего состояния следует пропустить дальше - в верхние слои и следующий временной шаг.
- g - "кандидатное"внутреннее состояние, вычисленное по входному вектору и предыдущему внутреннему состоянию. Берём $g \circ i$
- c_t внутренняя память на шаге t - комбинация части предыдущего состояния памяти $c_{t-1} \circ f$ и части "кандидатного"внутреннего состояния $g \circ i$
- Наконец, получаем скрытое состояние, определяя, какую часть пропускаем дальше

Рекуррентные сети: GRU

- GRU - упрощение LSTM

$$z = \sigma(x_t U^z + s_{t-1} W^z)$$

$$r = \sigma(x_t U^r + s_{t-1} W^r)$$

$$s_t = (1 - z) \circ h + z \circ s_{t-1}$$

- r - reset gate: как объединить новый входной вектор с предыдущим состоянием памяти;
  z - update gate: какую часть предыдущего состояния памяти следует сохранить
- Отличия от LSTM:
  - GRU не имеет внутренней памяти c_t, не имеют output gate
  - input gate и forget gate соединены в reset gate
  - для вычисления выхода не используем ещё одну нелинейную функцию активации