Искусственные нейронные сети II

Сергей Николенко

Машинное обучение — ИТМО, осень 2006

Outline

Перцептрон без лимита активации

Здесь мы будем рассматривать перцептроны, у которых нет лимита активации. Они просто выдают линейную форму от своих входов:

$$o(x_0, \ldots, x_n) = \sum_0^n w_i x_i.$$

Т.е. у такого перцептрона не два, а континуально много возможных значений.

В остальном это всё те же линейные перцептроны.

Функция ошибки

Мы хотим найти перцептрон, который минимизирует ошибку.
Пусть есть m тестовых примеров x_i^j с верными ответами t^j, $j = 1..m$.
Ошибка — из статистики, среднеквадратичное отклонение:

$$E(w_0, \ldots, w_n) = \frac{1}{2} \sum_{j=1}^{m} (t_j - o(x_0^j, \ldots, x_n^j))^2.$$

Задача: минимизировать функцию E на пространстве возможных весов $\{w_i\}$.

Outline

Градиент

Как минимизировать нелинейную функцию от нескольких аргументов?

Градиент

Как минимизировать нелинейную функцию от нескольких аргументов?

Нужно двигаться в сторону, обратную *градиенту*. Градиент — направление, в котором достигается наибольший прирост значений:

$$\nabla E(w_0, \ldots, w_n) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n} \right].$$

Градиент

Как минимизировать нелинейную функцию от нескольких аргументов?
Нужно двигаться в сторону, обратную *градиенту*. Градиент — направление, в котором достигается наибольший прирост значений:

$$\nabla E(w_0, \ldots, w_n) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}\right].$$

Значит, веса нужно подправлять так:

$$w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}.$$

Градиент от функции ошибки

В нашем случае подсчитать градиент совсем просто:

$$\frac{\partial E}{\partial w_i} = \frac{1}{2}\sum_{j=1}^{m}\frac{\partial}{\partial w_i}\left(t^j - \sum_0^n w_i x_i^j\right)^2 =$$

$$= \sum_{j=1}^{m}\left(t^j - \sum_0^n w_i x_i^j\right)(-x_i^j).$$

Изменения весов примут вид:

$$w_i \leftarrow w_i + \eta\sum_j\left(t^j - \sum_0^n w_i x_i^j\right)x_i^j.$$

Алгоритм градиентного спуска: мелочи

- Алгоритм — в каждой эпохе подправлять веса на значения градиента.
- Непонятно, когда останавливаться; мы будем останавливаться после фиксированного количества шагов.
- Если η маленькая, мы не успеем дойти до минимума; если η большая — будем через него «перескакивать». Что делать?

Алгоритм градиентного спуска: мелочи

- Алгоритм — в каждой эпохе подправлять веса на значения градиента.
- Непонятно, когда останавливаться; мы будем останавливаться после фиксированного количества шагов.
- Если η маленькая, мы не успеем дойти до минимума; если η большая — будем через него «перескакивать». Что делать?
- Решение: начать с достаточно большой η, а потом её постепенно уменьшать.

Алгоритм градиентного спуска

$\texttt{GradientDescent}(\eta, \{x_i^j, t^j\}_{i=1,j=1}^{n,m})$

- Инициализировать $\{w_i\}_{i=0}^n$ маленькими случайными значениями.
- Повторить `NUMBER_OF_STEPS` раз:
  - Для всех i от 1 до n $\Delta w_i = 0$.
  - Для всех j от 1 до m:
    - Для всех i от 1 до n

$$\Delta w_i = \Delta w_i + \eta \left(t^j - \sum_0^n w_i x_i^j \right) x_i^j.$$

  - $w_i = w_i + \Delta w_i$.
- Выдать значения $w_0, w_1, \ldots, w_n$.

Алгоритм градиентного спуска на Python

```
def PerceptronGradientDescent(eta0,x,NUMBER_OF_STEPS):
    import random
    eta,w,deltaw=eta0,[],[]
    for i in xrange(len(x[0])):
        w.append((random.randrange(-5,5))/50.0)
        deltaw.append(0)
    for i in xrange(NUMBER_OF_STEPS):
        if ((NUMBER_OF_STEPS > 100) and (i % (NUMBER_OF_STEPS/5))==0):
            eta = eta/2.0
        for i in xrange(len(w)): deltaw[i]=0
        for xj in x:
            t,o,curx=xj[0],0,[1]+xj[1:len(xj)]
            for i in xrange(len(w)): o+=w[i]*curx[i]
            for i in xrange(len(w)): deltaw[i]+=eta*(t-o)*curx[i]
        for i in xrange(len(w)): w[i]+=deltaw[i]
    return w
```

Стохастический градиентный спуск

- Предыдущий алгоритм не мог преодолевать локальные минимумы.
  - Почему это пока не проблема?

Стохастический градиентный спуск

- Предыдущий алгоритм не мог преодолевать локальные минимумы.
  - Почему это пока не проблема?
  - Потому что у параболоида локальных минимумов не бывает. Но в более сложных случаях...

Стохастический градиентный спуск

- Предыдущий алгоритм не мог преодолевать локальные минимумы.
  - Почему это пока не проблема?
  - Потому что у параболоида локальных минимумов не бывает. Но в более сложных случаях...
- Модификация — изменять веса после каждого тестового примера, а не накапливать Δw_j.

Алгоритм стохастического градиентного спуска

$\texttt{GradientDescent}(\eta, \{x_i^j, t^j\}_{i=1,j=1}^{n,m})$

- Инициализировать $\{w_i\}_{i=0}^n$ маленькими случайными значениями.
- Повторить NUMBER_OF_STEPS раз:
  - Для всех j от 1 до m:
    - Для всех i от 1 до n

$$w_i = w_i + \eta \left(t^j - \sum_0^n w_i x_i^j \right) x_i^j.$$

- Выдать значения $w_0, w_1, \ldots, w_n$.

Упражнение

Реализовать алгоритм стохастического градиентного спуска.

Главная проблема

- Какое главное ограничение у перцептронов без лимита активации?

Главная проблема

- Какое главное ограничение у перцептронов без лимита активации?
- Они реализуют только линейные функции.
- А какая проблема у перцептронов с лимитом активации?

Главная проблема

- Какое главное ограничение у перцептронов без лимита активации?
- Они реализуют только линейные функции.
- А какая проблема у перцептронов с лимитом активации?
- У них негладкий выход, и градиент от него не подсчитать. Соответственно, и алгоритм не сработает.
- Что же делать?

Главная проблема

- Какое главное ограничение у перцептронов без лимита активации?
- Они реализуют только линейные функции.
- А какая проблема у перцептронов с лимитом активации?
- У них негладкий выход, и градиент от него не подсчитать. Соответственно, и алгоритм не сработает.
- Что же делать?
- Строить гладкие, но не линейные перцептроны.

Outline

Нелинейные перцептроны

- Всё то же самое, но функция выхода уже не линейная.
- Но от линейной формы всё же не отказываемся, а берём её результат и сглаживаем.
- Популярная функция — *сигмоид*:

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

- То есть выход перцептрона подсчитывается по формуле:

$$o(x_1, \ldots, x_n) = \frac{1}{1 + e^{-\sum_i w_i x_i}}.$$

Градиент

- Почему сигмоид? Потому что легко считать производную:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)).$$

- Соответственно, правило модификации веса для одного перцептрона выглядит как

$$w_i \leftarrow w_i + \eta o(x)(1 - o(x))(t(x) - o(x))x_i,$$

где $t(x)$ — целевое значение функции.

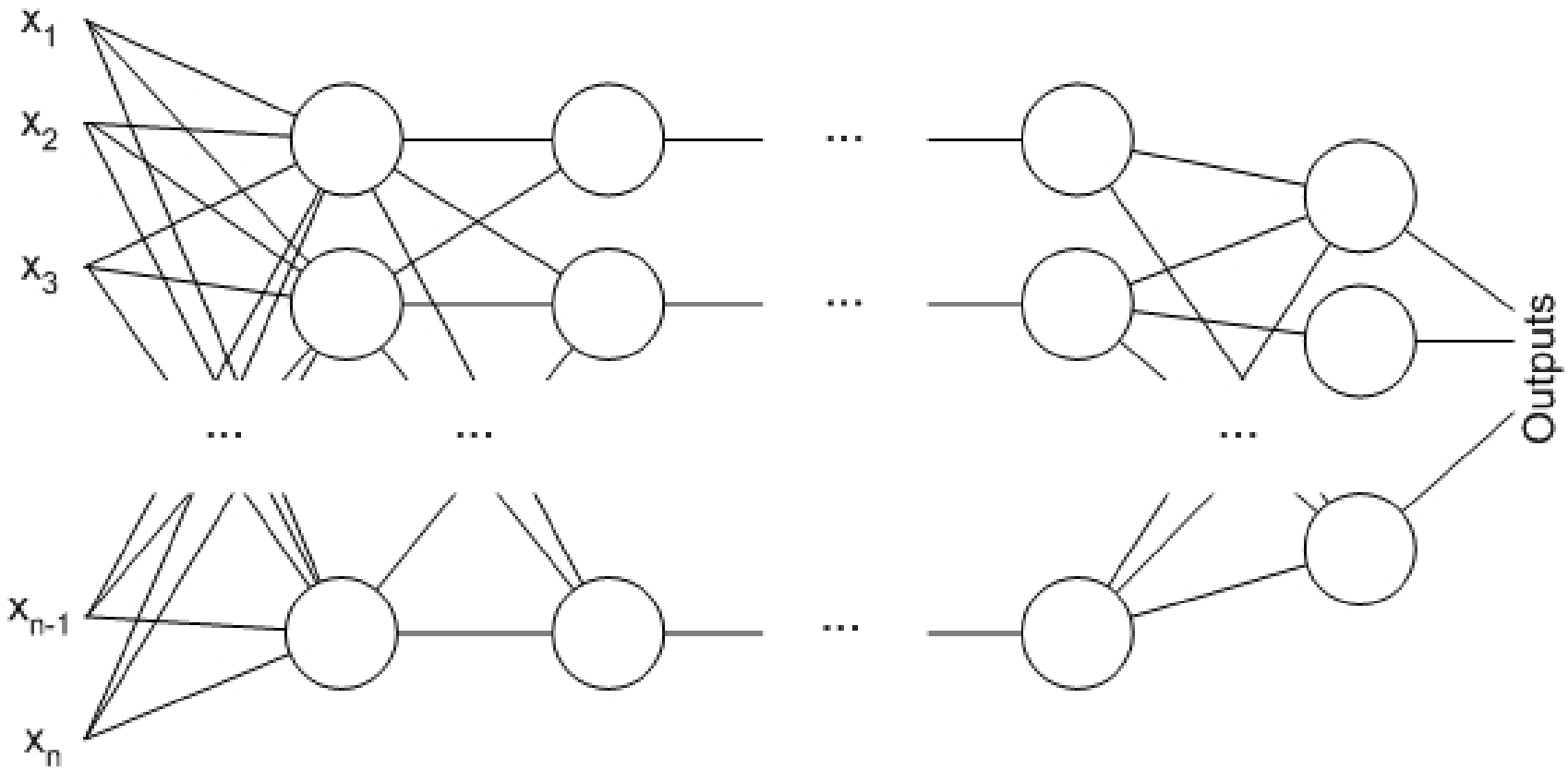
Упражнение

Реализовать обучение одного нелинейного перцептрона с сигмоидным выходом.

Outline

Нейронная сеть

Теперь у нас не один перцептрон, а целая их сеть.



Обозначения

- У сети есть входы $x_1, \ldots, x_n$, выходы Outputs и внутренние узлы.
- Перенумеруем все узлы (включая входы и выходы) числами от 1 до N.
- w_{ij} — вес, стоящий на ребре (i, j).
- o_i — выход i-го узла.
- Даны m тестовых примеров с целевыми значениями выходов $\{t_k^d\}, d=1..m, k \in \text{Outputs}$.

Функция ошибки:

$$E(\{w_{ij}\}) = \frac{1}{2} \sum_{d=1}^{m} \sum_{k \in \text{Outputs}} \left(t_k^d - o_k(x_1^d, \ldots, x_n^d) \right)^2.$$

Градиент

Как подсчитать градиент функции ошибки для каждого тестового примера?

$$E^d(\{w_{ij}\}) = \frac{1}{2} \sum_{k \in \text{Outputs}} \left(t_k^d - o_k^d\right)^2.$$

Случай 1. Считаем $\Delta w_{ij} = -\eta \frac{\partial E^d}{\partial w_{ij}}$, где $j \in \text{Outputs}$.

- w_{ij} влияет на выход o^d только как часть суммы $S_j = \sum_i w_{ij} x_{ij}$, поэтому

$$\frac{\partial E^d}{\partial w_{ij}} = \frac{\partial E^d}{\partial S_j} \frac{\partial S_j}{\partial w_{ij}} = x_{ij} \frac{\partial E^d}{\partial S_j}.$$

- S_j влияет на общую ошибку только в рамках выхода j-го узла o_j (это выход всей сети). Поэтому:

Градиент

Как подсчитать градиент функции ошибки для каждого тестового примера?

$$E^d(\{w_{ij}\}) = \frac{1}{2} \sum_{k \in \text{Outputs}} \left(t_k^d - o_k^d\right)^2.$$

$$\frac{\partial E^d}{\partial S_j} = \frac{\partial E^d}{\partial o_j} \frac{\partial o_j}{\partial S_j} = \left(\frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in \text{Outputs}} (t_k - o_k)^2 \right) \left(\frac{\partial \sigma(S_j)}{\partial S_j} \right) =$$

$$= \left(\frac{1}{2} \frac{\partial}{\partial o_j} (t_j - o_j)^2 \right) (o_j(1 - o_j)) = -o_j(1 - o_j)(t_j - o_j).$$

Градиент

Как подсчитать градиент функции ошибки для каждого тестового примера?

$$E^d(\{w_{ij}\}) = \frac{1}{2} \sum_{k \in \text{Outputs}} \left(t_k^d - o_k^d\right)^2.$$

Случай 2. Считаем $\Delta w_{ij} = -\eta \frac{\partial E^d}{\partial w_{ij}}$, где $j \notin \text{Outputs}$.

- У узла j есть потомки $\text{Children}(j)$. Тогда

$$\frac{\partial E^d}{\partial S_j} = \sum_{k \in \text{Children}(j)} \frac{\partial E^d}{\partial S_k} \frac{\partial S_k}{\partial S_j}, \text{ и } \frac{\partial S_k}{\partial S_j} = \frac{\partial S_k}{\partial o_j} \frac{\partial o_j}{\partial S_j} = w_{jk} o_j (1 - o_j).$$

- $\frac{\partial E^d}{\partial S_k}$ — это в точности аналогичная поправка, но вычисленная для узла следующего уровня. Так мы и дойдём от выходов ко входам.

Резюме подсчёта градиента

- Для узла последнего уровня

$$\delta_j = -o_j(1 - o_j)(t_j - o_j).$$

- Для внутреннего узла сети

$$\delta_j = -o_j(1 - o_j) \sum_{\text{Outputs}(j)} \delta_k w_{jk}.$$

- Для любого узла

$$\Delta w_{ij} = -\eta \delta_j x_{ij}.$$

Алгоритм обратного распространения ошибки

$\texttt{BackPropagation}(\eta, \{x_i^d, t^d\}_{i=1,d=1}^{n,m}, \texttt{NUMBER\_OF\_STEPS})$

- Инициализировать $\{w_{ij}\}_{i,j}$ случайными значениями.
- Повторить NUMBER_OF_STEPS раз:
  - Для всех d от 1 до m:
    - Подать $\{x_i^d\}$ на вход и подсчитать выходы o_i каждого узла.
    - Для всех $k \in \text{Outputs}$ $\delta_k = o_k(1 - o_k)(t_k - o_k)$.
    - Для каждого уровня l, начиная с предпоследнего:
      — Для каждого узла j уровня l вычислить

$$\delta_j = o_j(1 - o_j) \sum_{k \in \text{Children}(j)} w_{jk}\delta_k.$$

    - Для каждого ребра сети $\{ij\}$

$$w_{ij} = w_{ij} + \eta\delta_j x_{ij}.$$

- Выдать значения w_{ij}.

Домашнее задание

Упражнение

Реализовать алгоритм обратного распространения ошибки хотя бы для сети с одним скрытым уровнем (т.е. входы поступают на нейроны скрытого уровня, те передают результат дальше, на нейроны, которые уже формируют выходные значения).

Резюме

Чему мы научились за две лекции:

- Понятие перцептрона: с лимитом активации и без, линейного и нелинейного.
- Как обучать перцептроны поодиночке, в том числе:
  - алгоритмом обучения перцептрона,
  - методом градиентного спуска.
- Как обучать сеть из перцептронов с гладким выходом алгоритмом обратного распространения ошибки.

Спасибо за внимание!

- Lecture notes, слайды и коды программ появятся на моей homepage:

  `http://logic.pdmi.ras.ru/~sergey/index.php?page=teaching`

- Присылайте любые замечания, коды программ на других языках, решения упражнений, новые численные примеры и прочее по адресам:
  `sergey@logic.pdmi.ras.ru`, `smartnik@inbox.ru`