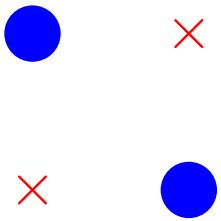
Введение

Алексей Андреевич Сорокин

МГУ им. М. В. Ломоносова
весенний семестр 2022–2023 учебного года
Межфакультетский курс “Введение в компьютерную лингвистику” 1 марта, занятие 3
Нейронные сети.

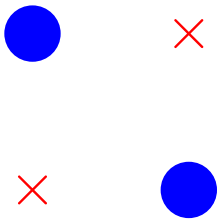
Недостатки линейных классификаторов

- Линейный классификатор не может проверить равенство двух координат:



Недостатки линейных классификаторов

- Линейный классификатор не может проверить равенство двух координат:



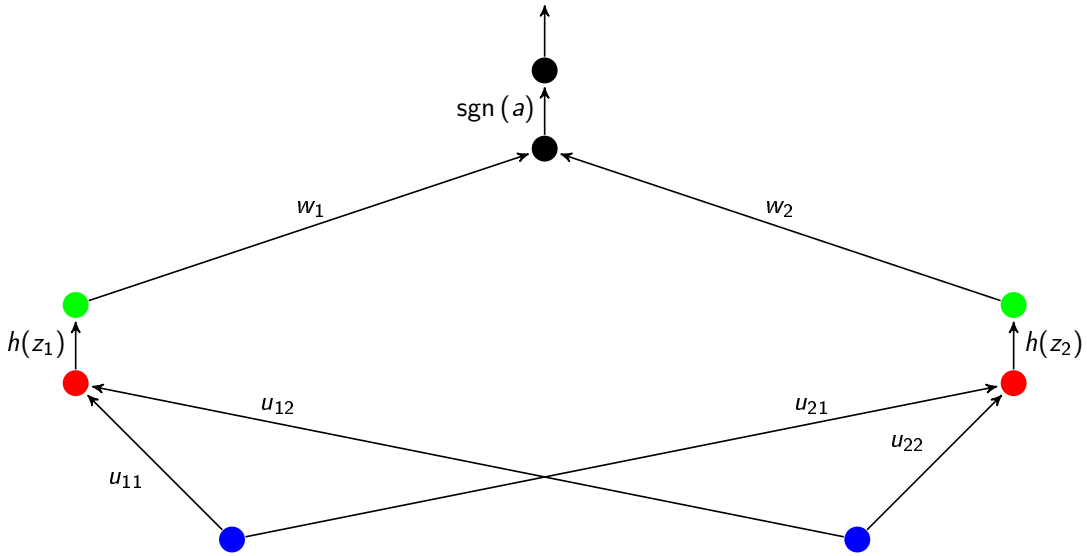
- Часто полезные признаки в задачах — комбинации исходных признаков:

$$w[-2] = \textit{в} \;\&\; \text{POS}(w[-1]) = \text{ADP} \;\&\; \text{TAG}(w[-1]).\text{case} = \text{LOC}$$

- Эти признаки приходится либо образовывать вручную, либо перебирать чрезмерно большое число комбинаций (SVM с полиномиальными ядрами).
- В результате модель не может выучить оптимальным образом большое число параметров.

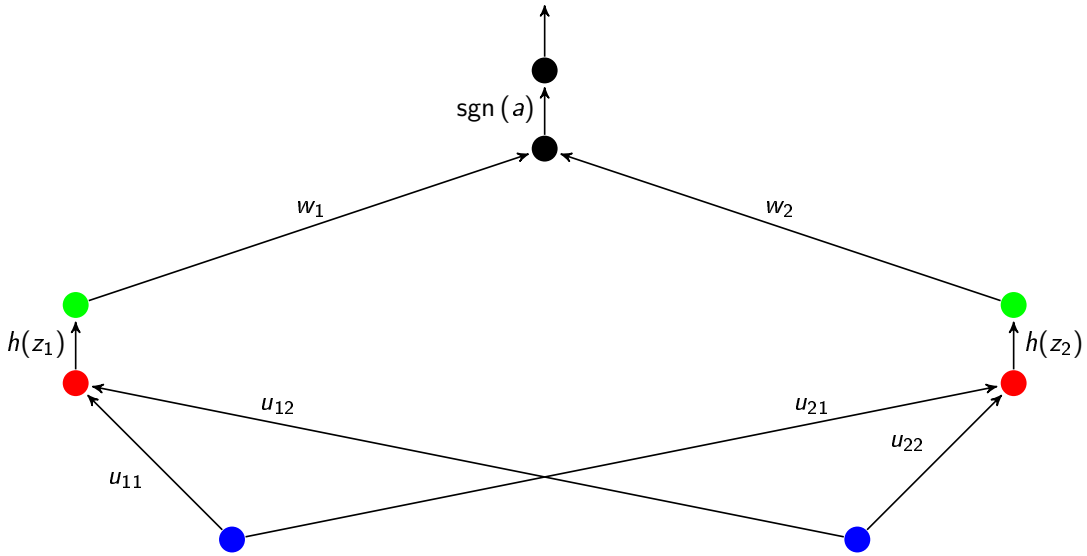
Двуслойная нейронная сеть для проверки равенства

- Схема нейронной сети:



Двуслойная нейронная сеть для проверки равенства

- Схема нейронной сети:



- Вычисление функции $(g(x) = \max(x, 0))$:

	$z_1 = x + y - 1$	$z_2 = 1 - (x + y)$	$h_1 = g(z_1)$	$h_2 = g(z_2)$	$2(h_1 + h_2) - 1$
$(0, 0)$	-1	1	0	1	1
$(0, 1)$	0	0	0	0	-1
$(1, 0)$	0	0	0	0	-1
$(1, 1)$	1	-1	1	0	1

Нейронная сеть: распознающая способность

- Уже двуслойная нейронная сеть распознаёт больше, чем линейный классификатор.
- Для этого между слоями была добавлена ReLU-активация:

$$ReLU(x) = \theta(x) = \max(x, 0)$$

- Без функции активации ничего бы не получилось.

Нейронная сеть: распознающая способность

- Уже двуслойная нейронная сеть распознаёт больше, чем линейный классификатор.
- Для этого между слоями была добавлена ReLU-активация:

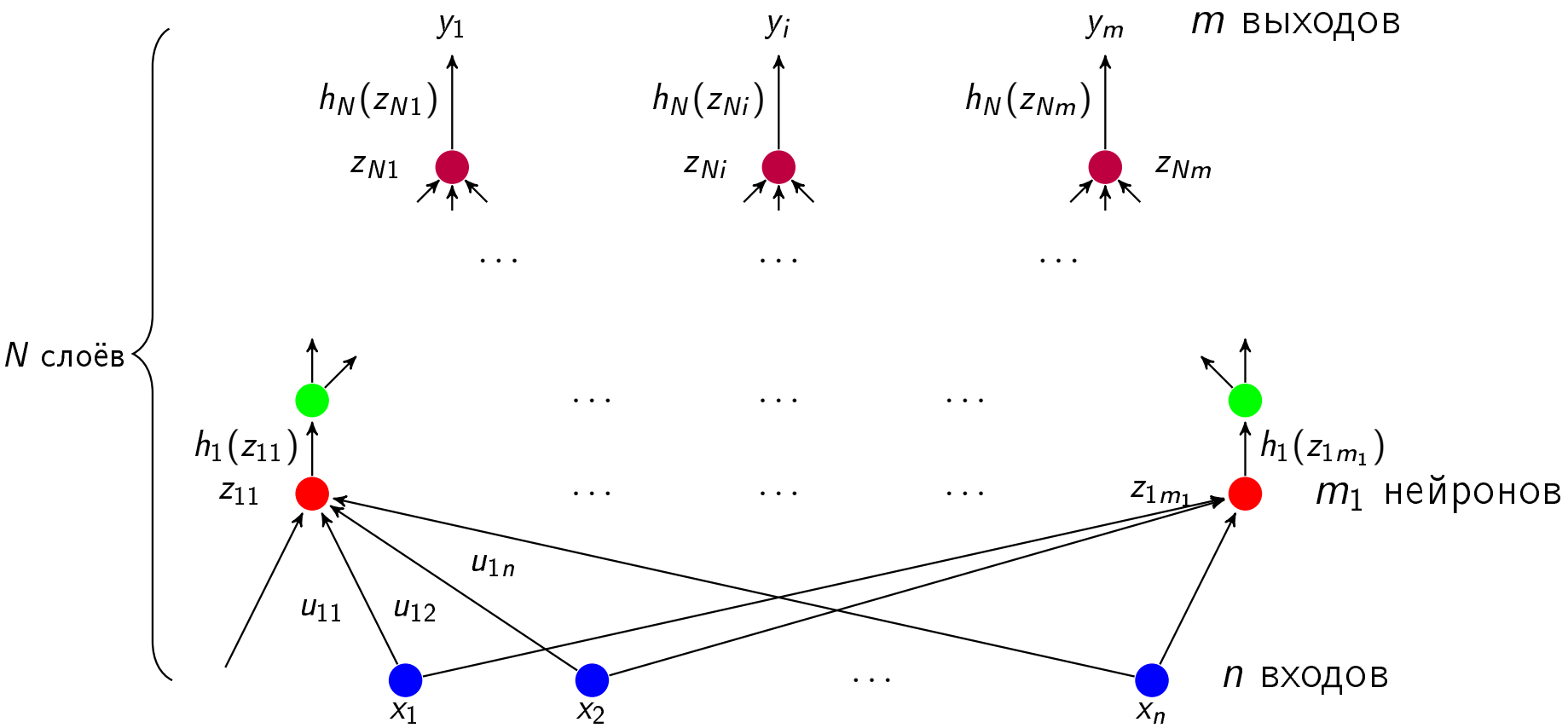
$$ReLU(x) = \theta(x) = \max(x, 0)$$

- Без функции активации ничего бы не получилось.

Theorem (Cybenko, Hornik)

Любую непрерывную функцию $f : \mathbb{R} \to \mathbb{R}$ можно приблизить двуслойной нейронной сетью. В качестве функции активации можно взять любую функцию, не являющуюся полиномом.

Общий вид нейронной сети



Нейронные сети: векторная запись

- Каждый нейрон сети вычитает взвешенную сумму нейронов с предыдущего слоя:

$$z_j^{(i)} = \sigma^{(i)}\left(\sum_r w_{jr}^{(i)} z_r^{(i-1)} + b_j^{(i)}\right),\ j = 1, \ldots, m_i$$

- $\sigma^{(i)}$ – функция активации на i-ом слое.
- Простейшие функции активации:

$$\begin{aligned} \sigma(x) &= \max(x, 0) && \text{(функция Хэвисайда, ReLU)} \\ \sigma(x) &= \frac{e^x}{e^x + 1} && \text{(сигмоида).} \end{aligned}$$

Нейронные сети: векторная запись

- Каждый нейрон сети вычитает взвешенную сумму нейронов с предыдущего слоя:

$$z_j^{(i)} = \sigma^{(i)}\left(\sum_r w_{jr}^{(i)} z_r^{(i-1)} + b_j^{(i)}\right),\ j = 1, \ldots, m_i$$

- $\sigma^{(i)}$ – функция активации на i-ом слое.
- Простейшие функции активации:

$$\begin{aligned} \sigma(x) &= \max(x, 0) && \text{(функция Хэвисайда, ReLU)} \\ \sigma(x) &= \frac{e^x}{e^x + 1} && \text{(сигмоида).} \end{aligned}$$

- Эту запись можно перевести на матричный язык:

$$\mathbf{z}^{(i)} = \sigma^{(i)}\left(W^{(i)}\mathbf{z}^{(i-1)} + b^{(i)}\right)$$

- Это *полносвязный слой*.

Нейронные сети: векторная запись

- Многослойная нейронная сеть вычисляет функцию $f : \mathbb{R}^n \to \mathbb{R}^m$:

$$\begin{aligned}
\mathbf{z}^{(1)} &= \sigma^{(1)}(W^{(1)}\mathbf{x} + b^{(1)}), \\
\mathbf{z}^{(2)} &= \sigma^{(2)}(W^{(2)}\mathbf{z}^{(1)} + b^{(2)}), \\
&\dots \\
\mathbf{z}^{(N)} &= \sigma^{(N)}(W^{(N)}\mathbf{z}^{(N-1)} + b^{(N)}), \\
y &= \sigma(W\mathbf{z}^{(N)} + b)
\end{aligned}$$

Нейронные сети: векторная запись

- Многослойная нейронная сеть вычисляет функцию $f : \mathbb{R}^n \to \mathbb{R}^m$:

$$\begin{aligned}
\mathbf{z}^{(1)} &= \sigma^{(1)}(W^{(1)}\mathbf{x} + b^{(1)}), \\
\mathbf{z}^{(2)} &= \sigma^{(2)}(W^{(2)}\mathbf{z}^{(1)} + b^{(2)}), \\
&\dots \\
\mathbf{z}^{(N)} &= \sigma^{(N)}(W^{(N)}\mathbf{z}^{(N-1)} + b^{(N)}), \\
y &= \sigma(W\mathbf{z}^{(N)} + b)
\end{aligned}$$

- Обозначения:

$\mathbf{x}$	–	входной вектор,	$x \in \mathbb{R}^n$,
$\mathbf{y}$	–	выходной вектор,	$x \in \mathbb{R}^m$,
$\mathbf{W}^{(i)}$	–	обучаемая матрица i-го слоя,	$\mathbf{W}^{(i)} \in \mathbb{R}^{(m_i \times n)}$,
$\mathbf{b}^{(i)}$	–	свободный член i-го слоя,	$\mathbf{b}^{(i)} \in \mathbb{R}^{m_i}$,
m_i	–	число нейронов i-го слоя.	
N	–	число скрытых слоёв.	

Функции активации

- Функции активации позволяют нейронным сетям вычислять нелинейные функции.
- Стандартные функции активации:

$$
\begin{aligned}
g(x) &= x \text{ (тождественная)} \\
g(x) &= \max(x, 0) \text{ (Rectified linear unit)} \\
g(x) &= \frac{e^x}{e^x + 1} \text{ сигмоида} \\
g(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} \text{ гиперболический тангенс} \\
g(x) &= [\frac{e^{x_1}}{\sum e^{x_i}}, \ldots, \frac{e^{x_n}}{\sum e^{x_i}}] \text{ (softmax)}
\end{aligned}
$$

Функции активации

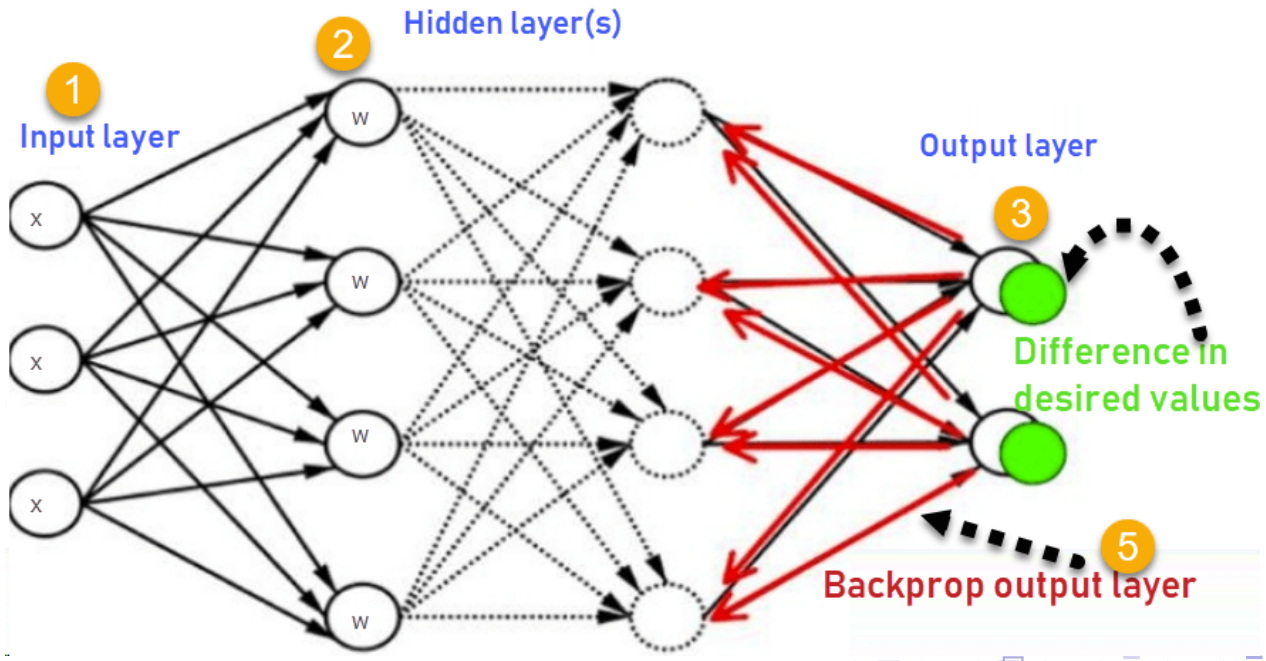
- Применение функций активации:
  - ReLU — сохраняет только положительные активации,
  - $\sigma(x)$ — преобразует $(-\infty; \infty)$ в $[0; 1]$.

Функции активации

- Применение функций активации:
  - ReLU — сохраняет только положительные активации,
  - $\sigma(x)$ — преобразует $(-\infty; \infty)$ в $[0; 1]$.
  - $\tanh(x) = 2\sigma(\frac{x}{2}) - 1$ — преобразует $(-\infty; \infty)$ в $[-1; 1]$.
  - softmax — применяется, когда нужно вернуть вероятности (преобразует вектор из n чисел в распределение вероятностей $p_1, \ldots, p_n$).
- Функции активации должны быть дифференцируемы, чтобы сеть можно было обучать градиентными методами.

Обучение нейронной сети.

- Нейронная сеть задаёт функцию из некоторого параметрического семейства.
- Обучение сети: подбор этих параметров с целью наилучшего приближения значения на обучающей выборке.
- Обучение осуществляется градиентным спуском, другое название – обратное распространение ошибок:



Обучение нейронной сети.

- Нейронная сеть задаёт функцию из некоторого параметрического семейства.

$$y' = F_\theta(x),\ \theta \in \mathbb{R}^n$$

- Качество ответа измеряется значением функции потерь:

$$L(y, y') \to \min_\theta$$

- Параметры оптимизируются шагом градиентного спуска:

$$\begin{aligned} \theta &\leftarrow \theta - \eta \frac{\partial F(y, F_\theta(x))}{\partial \theta}, \\ \theta > 0 &- \text{темп обучения.} \end{aligned}$$

Обучение нейронной сети.

- Нейронная сеть задаёт функцию из некоторого параметрического семейства.
$$y' = F_\theta(x),\ \theta \in \mathbb{R}^n$$
- Качество ответа измеряется значением функции потерь:
$$L(y, y') \to \min_\theta$$
- Параметры оптимизируются шагом градиентного спуска:
$$\begin{aligned} \theta &\leftarrow \theta - \eta \frac{\partial F(y, F_\theta(x))}{\partial \theta}, \\ \theta > 0 &- \text{темп обучения.} \end{aligned}$$
- Функция потерь считается:
  - Для одного объекта выборки (стохастический градиентный спуск).
  - Для группы объектов выборки (батча).

Функции потерь

- Пусть $C(y, y')$ — функция потерь, где y — эталонный ответ, а y' — ответ нейронного классификатора
- Возможные функции потерь:
  - Квадратичная: $(y-y')^2$. Применяется, когда классификатор возвращает $y' \in \mathbb{R}$.

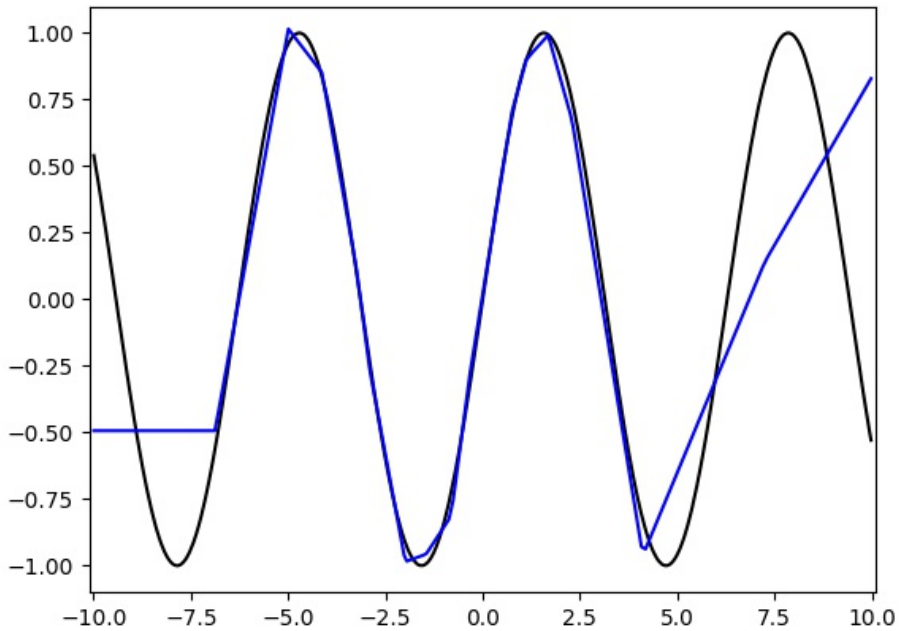
Функции потерь

- Пусть $C(y, y')$ — функция потерь, где y — эталонный ответ, а y' — ответ нейронного классификатора
- Возможные функции потерь:
  - Квадратичная: $(y - y')^2$. Применяется, когда классификатор возвращает $y' \in \mathbb{R}$.
  - Бинарная кросс-энтропия (cross-entropy): $-(y \log y' + (1 - y) \log (1 - y'))$. Измеряет, насколько далеки две вероятности $y, y' \in [0, 1]$.
  - В случае $y \in \{0, 1\}$ бинарная кросс-энтропия равна $-\log p(y)$.

Функции потерь

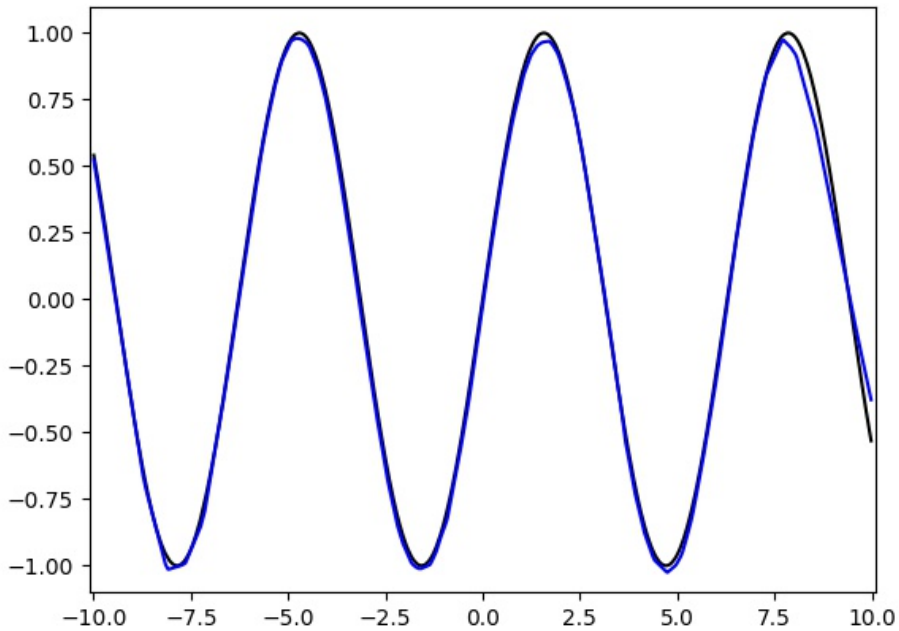
- Пусть $C(y, y')$ — функция потерь, где y — эталонный ответ, а y' — ответ нейронного классификатора
- Возможные функции потерь:
  - Квадратичная: $(y-y')^2$. Применяется, когда классификатор возвращает $y' \in \mathbb{R}$.
  - Бинарная кросс-энтропия (cross-entropy): $-(y \log y' + (1-y) \log (1-y'))$. Измеряет, насколько далеки две вероятности $y, y' \in [0, 1]$.
  - В случае $y \in \{0, 1\}$ бинарная кросс-энтропия равна $-\log p(y)$.
  - Категориальная кросс-энтропия: $-(\sum_i y_i \log y'_i)$. Измеряет расстояние между истинным вероятностным распределением $y = [y_1, \ldots, y_n]$ и предсказанным распределением $y' = [y'_1, \ldots, y'_n]$.
- Обе версии кросс-энтропии максимизируют вероятность обучающей выборки.

Приближение функций нейронными сетями



Приближение функции $y = \sin x$ на отрезке $[-10; 10]$. 3 полносвязных слоя, $n = 10$ нейронов на скрытых слоях.

Приближение функций нейронными сетями



Приближение функции $y = \sin x$ на отрезке $[-10; 10]$. 3 полносвязных слоя, $n = 100$ нейронов на скрытых слоях.

Базовая схема текстовой классификации

- Общая схема классификации:

$$
\begin{aligned}
X &= [x_1, \ldots, x_n], \\
x_i &\in \mathbb{R}^D, \\
s &= \sum(X) = [\sum_i \{x_{i1}\}, \ldots, \sum_i \{x_{iD}\}], \\
p = [p_1, \ldots, p_K] &= \mathrm{softmax}(Ws + b)
\end{aligned}
$$

Базовая схема текстовой классификации

- Общая схема классификации:

$$\begin{aligned} X &= [x_1, \ldots, x_n], \\ x_i &\in \mathbb{R}^D, \\ s &= \sum(X) = [\sum_i \{x_{i1}\}, \ldots, \sum_i \{x_{iD}\}], \\ p = [p_1, \ldots, p_K] &= \mathrm{softmax}(Ws + b) \end{aligned}$$

- Сеть состоит из трёх компонент:
  - Вычисление векторов слов x_i по слову w_i (0/1-вектора, $x_i = w_i$).

Базовая схема текстовой классификации

- Общая схема классификации:

$$\begin{aligned} X &= [x_1, \ldots, x_n], \\ x_i &\in \mathbb{R}^D, \\ s &= \sum(X) = [\sum_i \{x_{i1}\}, \ldots, \sum_i \{x_{iD}\}], \\ p = [p_1, \ldots, p_K] &= \mathrm{softmax}(Ws + b) \end{aligned}$$

- Сеть состоит из трёх компонент:
  - Вычисление векторов слов x_i по слову w_i (0/1-вектора, $x_i = w_i$).
  - Агрегация их в вектор предложения (покоординатный максимум, среднее или сумма).
  - Вычисления самой вероятной метки (однослойный персептрон)

Базовая схема текстовой классификации

- Сеть состоит из трёх компонент:
  - Вычисление векторов слов x_i по слову w_i (0/1-вектора, $x_i = w_i$).
  - Агрегация их в вектор предложения (покоординатный максимум, среднее или сумма).
  - Вычисления самой вероятной метки (однослойный персептрон)

Базовая схема текстовой классификации

- Сеть состоит из трёх компонент:
  - Вычисление векторов слов x_i по слову w_i (0/1-вектора, $x_i = w_i$).
  - Агрегация их в вектор предложения (покоординатный максимум, среднее или сумма).
  - Вычисления самой вероятной метки (однослойный персептрон)
- В линейном классификаторе первые две компоненты детерминиро
  обучается только последняя.
- То есть мы требуем, чтобы классы разделялись плоскостью в исходном пространстве признаков.

Базовая схема текстовой классификации

- Сеть состоит из трёх компонент:
  - Вычисление векторов слов x_i по слову w_i (0/1-вектора, $x_i = w_i$).
  - Агрегация их в вектор предложения (покоординатный максимум, среднее или сумма).
  - Вычисления самой вероятной метки (однослойный персептрон)
- В линейном классификаторе первые две компоненты детерминиро
  обучается только последняя.
- То есть мы требуем, чтобы классы разделялись плоскостью в исходном пространстве признаков.
- Если первые две части будут обучаемые, то сеть “сама” научится так представлять тексты, чтобы классы можно было разделить плоскостью.

Дистрибутивные вектора

- Основная идея дистрибутивной семантики:

  You should know the word by the company it keeps.
- Похожие слова встречаются в похожих контекстах.
- Можно представлять слово как усреднённый вектор контекста (Latent Semantic Analysis).

Дистрибутивные вектора

- Основная идея дистрибутивной семантики:

  You should know the word by the company it keeps.

- Похожие слова встречаются в похожих контекстах.
- Можно представлять слово как усреднённый вектор контекста (Latent Semantic Analysis).
- Можно предсказывать контекст слова по его вектору (или наоборот).
- Тогда похожие слова будут приводить к похожим контекстам.

Предсказание слова

- Дистрибутивные вектора обучаются на задаче предсказания слова по контексту:

Я съел вкусное / зелёное / печёное яблоко .

- Будем для простоты считать, что контекст тоже состоит из одного слова (например, предсказывается следующее слово справа).

Предсказание слова

- Дистрибутивные вектора обучаются на задаче предсказания слова по контексту:

Я съел вкусное / зелёное / печёное яблоко .

- Будем для простоты считать, что контекст тоже состоит из одного слова (например, предсказывается следующее слово справа).
- Для каждого слова вводятся два вектора: u_i (входной) и v_i (выходной).
- По u_i предсказывается вероятностное распределение на множестве v_j:

$$p(w_j|w_i) = \frac{\exp((u_i, v_j))}{\sum_k \exp((u_i, v_k))}$$

Предсказание слова

- Контекст (слово w_i) порождает вероятностное распределение $p = [p_1, \ldots, p_{|D|}]$:

$$p_j = p(w_j|w_i) = \frac{\exp((u_i, v_j))}{\sum_k \exp((u_i, v_k))}$$

- Это распределение сравнивается с эталонным $\hat{p} = [0, \ldots, 1, \ldots, 0]$:

$$\hat{p}_r = 1 \leftrightarrow w_r \text{ находится в контексте с } w_i$$

Предсказание слова

- Контекст (слово w_i) порождает вероятностное распределение $p = [p_1, \ldots, p_{|D|}]$:

$$p_j = p(w_j|w_i) = \frac{\exp\left((u_i, v_j)\right)}{\sum_k \exp\left((u_i, v_k)\right)}$$

- Это распределение сравнивается с эталонным $\hat{p} = [0, \ldots, 1, \ldots, 0]$:

$$\hat{p}_r = 1 \leftrightarrow w_r \text{ находится в контексте с } w_i$$

- Штраф за различие – кросс-энтропия:

$$Q(\hat{p}, p) = -\sum_k \hat{p}_k \log p_k = -\log p_r$$

Предсказание слова

- Штраф за различие можно минимизировать градиентным спуском:

$$u_i \leftarrow u_i - \frac{\partial Q(\hat{p}, p)}{\partial u_i},$$
$$v_k \leftarrow u_i - \frac{\partial Q(\hat{p}, p)}{\partial v_k}, k = 1, \ldots, |D|$$

- Этот штраф изменяет один контекстный вектор и все выходные вектора.

Предсказание слова

- Штраф за различие можно минимизировать градиентным спуском:

$$u_i \leftarrow u_i - \frac{\partial Q(\hat{p}, p)}{\partial u_i},$$
$$v_k \leftarrow u_i - \frac{\partial Q(\hat{p}, p)}{\partial v_k}, k = 1, \ldots, |D|$$

- Этот штраф изменяет один контекстный вектор и все выходные вектора.
- На практике рассматривают контекст из нескольких слов $(5 - 10)$ и предсказывают центральное.
- Вектор контекста: среднее векторов входящих в него слов.

Предсказание соседних слов

- По u_i предсказывается вероятностное распределение на множестве v_j:

$$p(w_j|w_i) = \frac{\exp\left((u_i, v_j)\right)}{\sum_k \exp\left((u_i, v_k)\right)} = \mathrm{softmax}((u_i, v_1), \ldots, (u_i, v_k))$$

Предсказание соседних слов

- По u_i предсказывается вероятностное распределение на множестве v_j:

$$p(w_j|w_i) = \frac{\exp\left((u_i, v_j)\right)}{\sum_k \exp\left((u_i, v_k)\right)} = \mathrm{softmax}((u_i, v_1), \ldots, (u_i, v_k))$$

- В матричном виде можно записать как

$$p = \mathrm{softmax}(Vu)$$

- i-ая строка матрицы V — представление i-го слова в словаре.

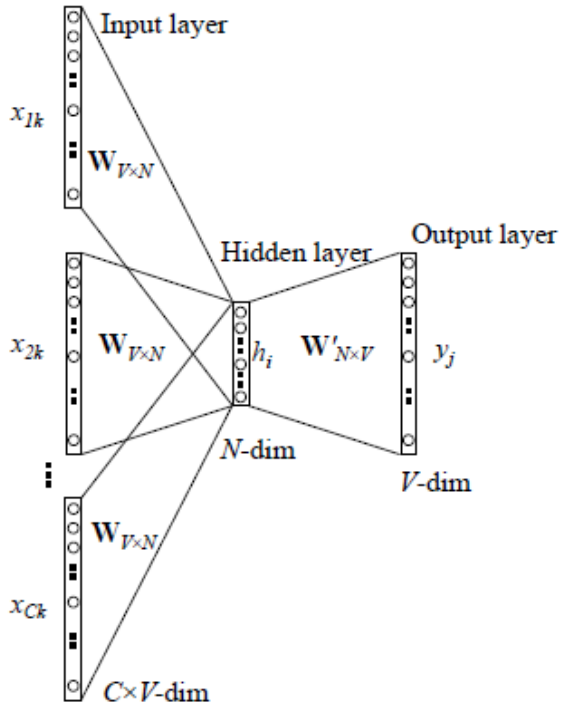
Обучение векторных представлений

- Схема предсказания:

$$x_i = \underbrace{[0, \ldots, 0, x_{ij} = 1, 0, \ldots, 0]}_{j\text{– индекс } i\text{-го слова в словаре}},$$

$$h = \frac{1}{C}(Wx_1 + \ldots + Wx_C),$$

$$p = softmax(W'h)$$

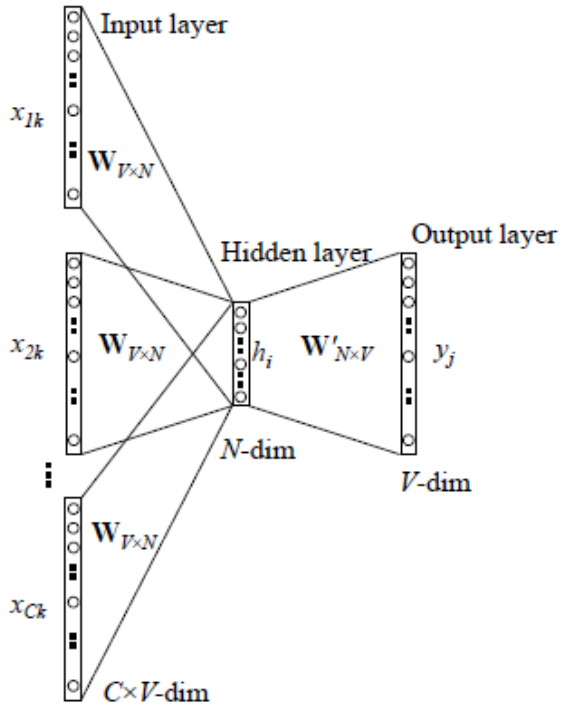


Обучение векторных представлений

- Схема предсказания:

$$
\begin{aligned}
x_i &= \underbrace{[0, \ldots, 0, x_{ij} = 1, 0, ..., 0]}_{j\text{– индекс } i\text{-го слова в словаре}}, \\
h &= \frac{1}{C}(Wx_1 + \ldots + Wx_C), \\
p &= softmax(W'h)
\end{aligned}
$$

- Столбцы матрицы W (ранее U) и строки матрицы W' (на предыдущем слайде V)— сжатые представления слов.



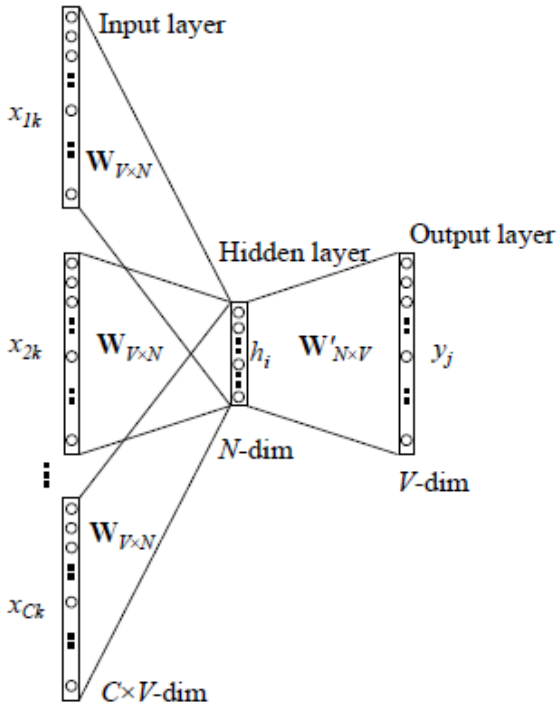
Обучение векторных представлений

- Схема предсказания:

$$x_i = \underbrace{[0, \ldots, 0, x_{ij} = 1, 0, ..., 0]}_{j\text{– индекс } i\text{-го слова в словаре}},$$

$$h = \frac{1}{C}(Wx_1 + \ldots + Wx_C),$$

$$p = softmax(W'h)$$



- Столбцы матрицы W (ранее U) и строки матрицы W' (на предыдущем слайде V)— сжатые представления слов.
- Семантически близкие слова переходят в близкие вектора.
- Вектора сохраняют семантические связи:

$$\vec{king} - \vec{queen} = \vec{man} - \vec{woman}$$

Негативная выборка

- Стандартный штраф — кросс-энтропия:

$$Q(\pi, \pi') = -\sum_i \pi_i \log \pi'_i$$

- Его можно минимизировать градиентным спуском.

Негативная выборка

- Стандартный штраф — кросс-энтропия:

$$Q(\pi, \pi') = -\sum_i \pi_i \log \pi'_i$$

- Его можно минимизировать градиентным спуском.
- Обычно делают по-другому: на каждом шаге выбирают "положительное" слово u и отрицательное слово v и стараются максимизировать разницу между их вероятностями:

$$\log p(u|w_i) - \log p(v|w_i) \to \max$$

- Положительные — слова, встречавшиеся в данном контексте в корпусе, отрицательные — все остальные.

Негативная выборка

- Стандартный штраф — кросс-энтропия:

$$Q(\pi, \pi') = -\sum_i \pi_i \log \pi'_i$$

- Его можно минимизировать градиентным спуском.
- Обычно делают по-другому: на каждом шаге выбирают "положительное" слово u и отрицательное слово v и стараются максимизировать разницу между их вероятностями:

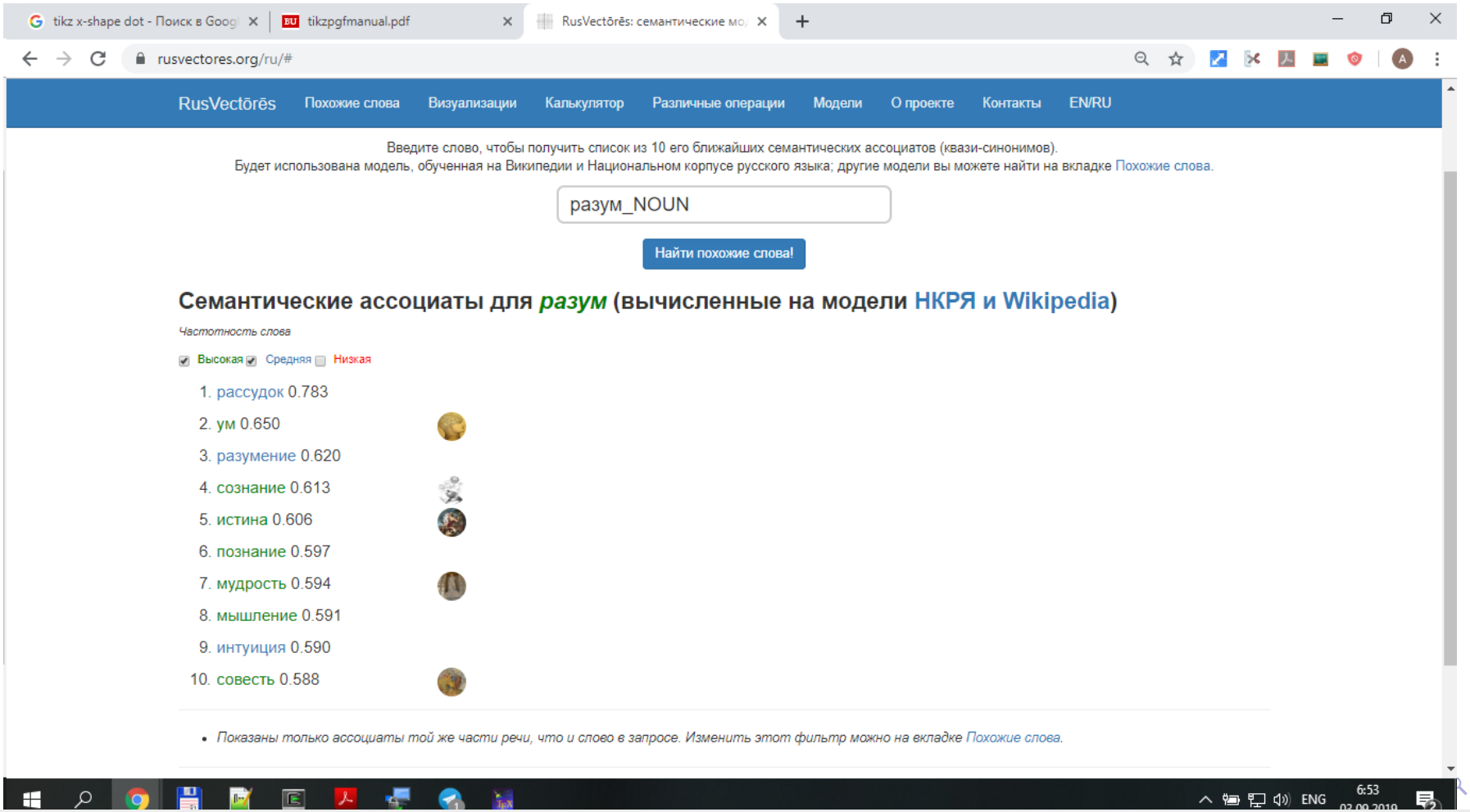
$$\log p(u|w_i) - \log p(v|w_i) \to \max$$

- Положительные — слова, встречавшиеся в данном контексте в корпусе, отрицательные — все остальные.
- Например, для "корпуса" и $w_i = \textit{мама}$,

Мама мыла раму
Моя мама красивая

"положительными" словами будут *мыла*, *моя*, *красивая*, а "отрицательными" – все остальные.

Векторные представления: пример



Векторные представления: пример