

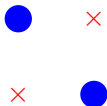
Введение

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.

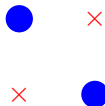
Недостатки линейных классификаторов

- Линейный классификатор не может проверить равенство двух координат:



Недостатки линейных классификаторов

- Линейный классификатор не может проверить равенство двух координат:



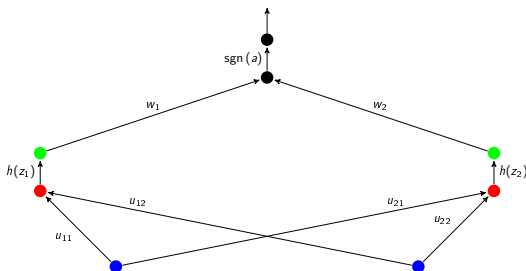
- Часто полезные признаки в задачах — комбинации исходных признаков:

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

- 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.

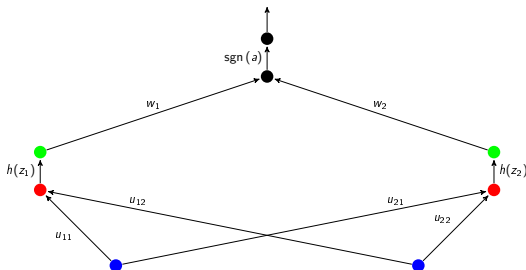
Двуслойная нейронная сеть для проверки равенства

- 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:

$$\text{ReLU}(x) = \theta(x) = \max(x, 0)$$

- 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:

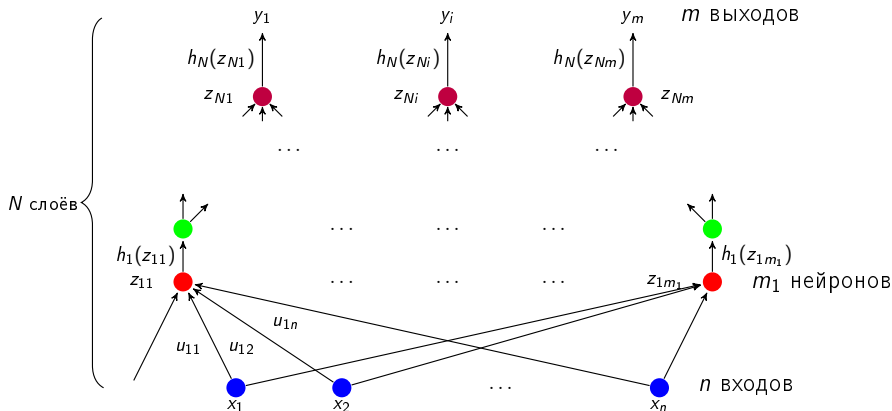
$$\text{ReLU}(x) = \theta(x) = \max(x, 0)$$

- Без функции активации ничего бы не получилось.

Theorem (Cybenko, Hornik)

Любую непрерывную функцию $f: \mathbb{R} \rightarrow \mathbb{R}$ 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:

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

- $\sigma^{(i)}$ – 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:

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

- $\sigma^{(i)}$ – 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:

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

- Это *полносвязный слой*.

Нейронные сети: векторная запись

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

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

Нейронные сети: векторная запись

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

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

- Обозначения:

\mathbf{x}	— входной вектор,	$\mathbf{x} \in \mathbb{R}^n$,
\mathbf{y}	— выходной вектор,	$\mathbf{y} \in \mathbb{R}^m$,
$W^{(i)}$	— обучаемая матрица i -го слоя,	$W^{(i)} \in \mathbb{R}^{(m_i \times n)}$,
$\mathbf{b}^{(i)}$	— 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:

$$g(x) = x \text{ (тождественная)}$$

$$g(x) = \max(x, 0) \text{ (Rectified linear unit)}$$

$$g(x) = \frac{e^x}{e^x + 1} \text{ сигмоида}$$

$$g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \text{ гиперболический тангенс}$$

$$g(x) = \left[\frac{e^{x_1}}{\sum e^{x_i}}, \dots, \frac{e^{x_n}}{\sum e^{x_i}} \right] \text{ (softmax)}$$

Функции активации

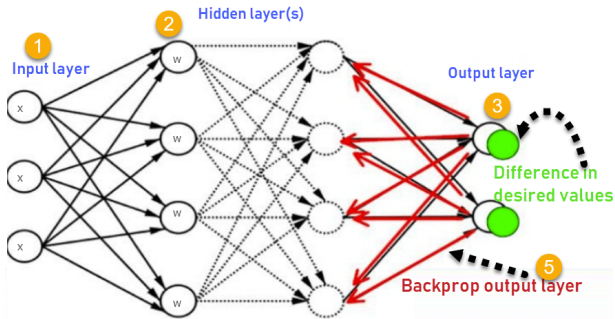
- Применение функций активации:
 - ReLU — сохраняет только положительные активации,
 - $\sigma(x)$ — преобразует $(-\infty; \infty)$ в $[0; 1]$.

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, \dots, 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:

$$L(y, y') \rightarrow \min_{\theta}$$

- Параметры оптимизируются шагом градиентного спуска:

$$\theta \leftarrow \theta - \eta \frac{\partial F(y, F_{\theta}(x))}{\partial \theta},$$

$\theta > 0$ – темп обучения.

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:

$$L(y, y') \rightarrow \min_{\theta}$$

- Параметры оптимизируются шагом градиентного спуска:

$$\theta \leftarrow \theta - \eta \frac{\partial F(y, F_{\theta}(x))}{\partial \theta},$$

$\theta > 0$ – темп обучения.

- 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 $y' \in \mathbb{R}$.

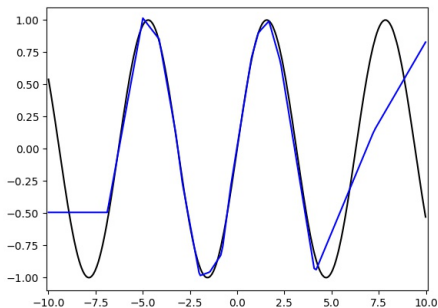
Функции потерь

- 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

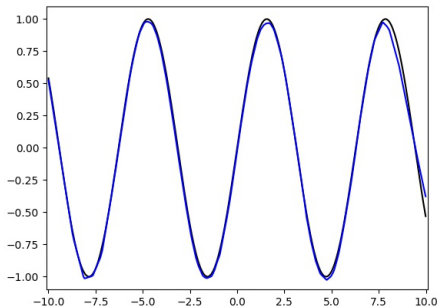
- Пусть $C(y, y')$ — функция потерь, где y — эталонный ответ, а y' — ответ нейронного классификатора
- Возможные функции потерь:
 - Квадратичная: $(y - y')^2$. Применяется, когда классификатор возвращает $y' \in \mathbb{R}$.
 - Бинарная кросс-энтропия (cross-entropy): $-(y \log y' + (1 - y) \log (1 - y'))$. 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 $y = [y_1, \dots, y_n]$ и предсказанным распределением $y' = [y'_1, \dots, y'_n]$.
- 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:

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

Базовая схема текстовой классификации

- Общая схема классификации:

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

- 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:

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

- 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 детерминированы, 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 детерминированы, 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 яблоко .
печёное

- 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 яблоко .
печёное

- 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 $p = [p_1, \dots, p_{|D|}]$:

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

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

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

Предсказание слова

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

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

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

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

- 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:

$$\begin{aligned} u_i &\leftarrow u_i - \frac{\partial Q(\hat{p}, p)}{\partial u_i}, \\ v_k &\leftarrow v_k - \frac{\partial Q(\hat{p}, p)}{\partial v_k}, k = 1, \dots, |D| \end{aligned}$$

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

Предсказание слова

- Штраф за различие можно минимизировать градиентным спуском:

$$\begin{aligned}u_i &\leftarrow u_i - \frac{\partial Q(\hat{p}, p)}{\partial u_i}, \\v_k &\leftarrow v_k - \frac{\partial Q(\hat{p}, p)}{\partial v_k}, k = 1, \dots, |D|\end{aligned}$$

- 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 :

$$p(w_j|w_i) = \frac{\exp((u_i, v_j))}{\sum_k \exp((u_i, v_k))} = \text{softmax}((u_i, v_1), \dots, (u_i, v_k))$$

Предсказание соседних слов

- По u_i предсказывается вероятностное распределение на множестве v_j :

$$p(w_j|w_i) = \frac{\exp((u_i, v_j))}{\sum_k \exp((u_i, v_k))} = \text{softmax}((u_i, v_1), \dots, (u_i, v_k))$$

- В матричном виде можно записать как

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

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

Обучение векторных представлений

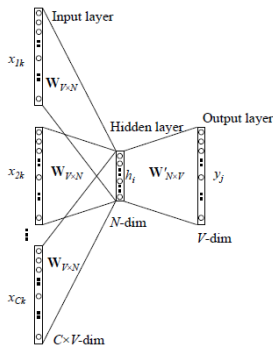
- Схема предсказания:

$$x_i = \underbrace{[0, \dots, 0, x_{ij} = 1, 0, \dots, 0]},$$

j – индекс i -го слова в словаре

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

$$p = \text{softmax}(W'h)$$



Обучение векторных представлений

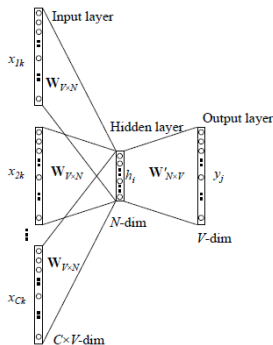
- Схема предсказания:

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

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

$$p = \text{softmax}(W'h)$$

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



Обучение векторных представлений

- Схема предсказания:

$$x_i = \underbrace{[0, \dots, 0, x_{ij} = 1, 0, \dots, 0]},$$

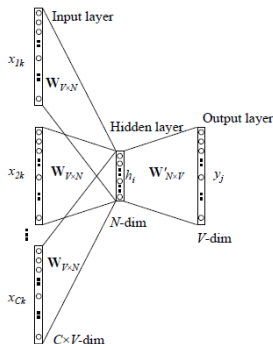
j – индекс i -го слова в словаре

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

$$p = \text{softmax}(W'h)$$

- 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:

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

- 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:

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

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

Мама мыла раму
Моя мама красивая

“положительными” словами будут *мыла, моя, красивая*, а “отрицательными” — все остальные.

Векторные представления слов

Векторные представления: пример

The screenshot shows a web browser window with the URL `rusvectors.org/ru/#`. The page has a blue header with navigation links: [RusVectores](#), [Похожие слова](#), [Визуализации](#), [Калькулятор](#), [Различные операции](#), [Модели](#), [О проекте](#), [Контакты](#), and [EN/RU](#). Below the header, a text box contains the word `разум_NOUN`, and a button below it says [Найти похожие слова!](#). The main content area is titled **Семантические ассоциаты для *разум* (вычисленные на модели НКРЯ и Wikipedia)**. Below this title, there is a filter section with three radio buttons: ☒ **Высокая**, ☐ **Средняя**, and ☐ **Низкая**. A list of 10 semantic associates is displayed, each with a number, the word, and a score. To the right of each word is a small circular icon. The list is as follows:

Rank	Word	Score
1.	рассудок	0.783
2.	ум	0.650
3.	разумение	0.620
4.	сознание	0.613
5.	истина	0.606
6.	познание	0.597
7.	мудрость	0.594
8.	мышление	0.591
9.	интуиция	0.590
10.	совесть	0.588

At the bottom of the page, there is a note: *Показаны только ассоциаты той же части речи, что и слово в запросе. Изменить этот фильтр можно на вкладке Похожие слова.*

Векторные представления слов

Векторные представления: пример

The screenshot shows the RusVectores website's semantic calculator interface. The browser tabs include 'tikz x-shape dot', 'tikzpgfmanual.pdf', and 'RusVectores: Семантический калькулятор'. The URL is 'rusvectors.org/ru/calculator/#'. The page title is 'Семантический калькулятор'. It explains that users can calculate relationships between words, such as finding a word related to 'собака' (dog) as 'кошка' (cat) is to 'кошка'.

Under the heading 'Семантический калькулятор', there are two input fields: 'собака_NOUN' and 'кошка_NOUN'. Below 'собака_NOUN' is a blue arrow pointing to 'лаять_VERB'. Below 'кошка_NOUN' is a blue arrow pointing to '???'. Below these fields, there is a section titled 'Частотность слова' (Word frequency) with a table showing the frequency of the word 'Тайга' (Taiga) in various contexts.

Слово	Частотность
1. млыкать	0.73
2. тывкать	0.64
3. гавкать	0.59
4. залаять	0.59
5. млынуть	0.59

Below the table, there is a note: 'Для каждого слова показана его часть речи (если в модели они размечены)'. There is also a section 'Выберите модель:' (Select model) with radio buttons for 'Анализ fastText', 'HMPR', 'Новостной корпус', 'Тайга', 'Тайга fastText', and 'HMPR и Wikipedia'. There is also a section 'Показывать только:' (Show only) with radio buttons for 'Наречия', 'Существительные', 'Имена собственные', 'Глаголы', 'Прилагательные', and 'Все части речи'. A 'Вычислить!' (Calculate!) button is at the bottom right.