Проектирование интеллектуальных систем

Лекция № 6. Особенности обучения многослойных нейронных сетей

Содержание

Проблема переобучения

Обучение нейронной сети – это более сложная задача, чем просто задача оптимизации. Цель обучения – построить адекватную модель окружающего мира для решения поставленной задачи [7]. Это подразумевает извлечение из данных, на которых обучается нейронная сеть, закономерностей, обусловленных существующими законами природы. Если нейронная сеть обучена правильно, то есть если она извлекла из данных истинные закономерности, то она будет корректно работать и на новых данных, которые не были использованы в процессе обучения. Это свойство нейронных сетей называется **обобщающей способностью** (англ. generalization ability).

Чтобы была возможность проверить, обладает ли сеть обобщающей способностью, имеющиеся в распоряжении данные перед обучением нейронной сети, как правило, делятся на три части. Первая часть – *обучающая выборка* – это те данные, которые используются для обучения нейронной сети. Вторая часть – *тестовая выборка* – это те данные, которые не используются в процессе обучения, а используются для последующий проверки того, насколько хорошо работает обученная нейронная сеть на новых данных. Проверка осуществляется при помощи вычисления специальной метрики (или функции потерь), выбор которой зависит от решаемой задачи. Например, для задачи классификации часто используется точность (англ. accuracy) – доля правильных ответов нейронной сети.

Объём обучающей и тестовой выборок зависит от решаемой задачи. Часто можно встретить рекомендацию поместить 70% имеющихся данных в обучающую выборку, и 30% – в тестовую. Однако, нужно также учитывать объём исходных данных, и с какой точностью нужно вычислить значение метрики. Например, если в качестве метрики используется accuracy и её нуж-

но вычислить с точностью до сотых долей процента, то в тестовой выборке должно быть не менее 10000 объектов.

Чтобы значение метрики на обучающей и тестовой выборках можно было сравнивать, выборки должны быть правильно сформированы. Это означает, что в них должно быть одинаковое распределение данных. Например, если речь идёт о задаче классификации, то в выборках должны быть представлены все имеющиеся классы в одинаковых пропорциях. Если нейронная сеть работает с изображениями, то они должны быть одного качества. Например, если для обучения нейронной сети были использованы фотографии, сделанные при помощи профессиональной аппаратуры, а для тестирования были использованы фотографии, сделанные на телефон, то в таком случае результаты тестирования можно считать недействительными.

Чтобы результаты обучения можно было считать приемлемыми, необходимо, чтобы они удовлетворяли двум основным критериям. Во-первых, значение метрики для тестовой выборки должно быть не ниже некоторого заранее установленного порогового значения. Например, для точности можно установить порог 80%. Во-вторых, не должно наблюдаться большой разницы в значениях метрики на обучающей и тестовой выборках.

Если не удаётся добиться удовлетворительного значения метрики на тестовой выборке, такое явление называют **недообучением** (англ. underfitting). Если же наблюдается большой разрыв в значении метрики на обучающей и тестовой выборках, то есть если нейронная сеть значительно лучше работает на обучающей выборке, чем на тестовой, – такое явление называют **переобучением** (англ. overfitting).

Явления недообучения и переобучения связаны с понятием **ёмкости** (англ. capacity) модели, которую можно понимать как способность модели представлять определённый набор функций. Если ёмкость модели небольшая,

модель может быть неспособна аппроксимировать искомую зависимость, что выражается в явлении недообучения. В противном случае, если ёмкость модели большая, она может излишне точно подстроиться под обучающую выборку, и в результате будет плохо работать на новых данных, то есть произойдёт переобучение.

Ёмкость нейронной сети определяется её архитектурой, в которую входит количество слоёв, количество нейронов в каждом слое, связи между нейронами, функции активации, и т. д. То есть архитектура нейронной сети задаёт то параметрическое семейство функций (*пространство гипотез*), среди которого ищется решение задачи. Например, для одного нейрона с линейной функцией активации пространство гипотез – это пространство всех возможных линейных функций, определённых на пространстве признаков.

Ёмкость модели можно регулировать, изменяя количество признаков или количество параметров модели. Наглядным примером изменения емкости модели является полиномиальная регрессия. Так, например, полином первой степени представляет собой линейную функцию

$$\hat{y} = w_1 x + b.$$

Если добавить ещё один признак x^2 и дополнительный параметр w_2, получим полином второй степени

$$\hat{y} = w_2 x^2 + w_1 x + b.$$

Обратите внимание, что при добавлении дополнительного параметра значительно расширилось пространство функций, которые может представлять модель. Теперь оно представляет собой пространство всех возможных квадратичных функций, которое включает в себя все возможные линейные функции (при $w_2 = 0$).

На рис. 11 видно, как меняются возможности модели при увеличении степени полинома. При использовании полинома 1-й степени наблюдается недообучение, так как в данных явно присутствует нелинейная зависимость, которую нельзя аппроксимировать линейной функцией. С увеличением степени полином всё лучше описывает имеющиеся данные, однако наблюдаются явно выраженные осцилляции между точками обучающей выборки, что говорит о низкой обобщающей способности и возникновении переобучения. Можно сказать, что наилучшим образом данные описывает полином 3-й степени.

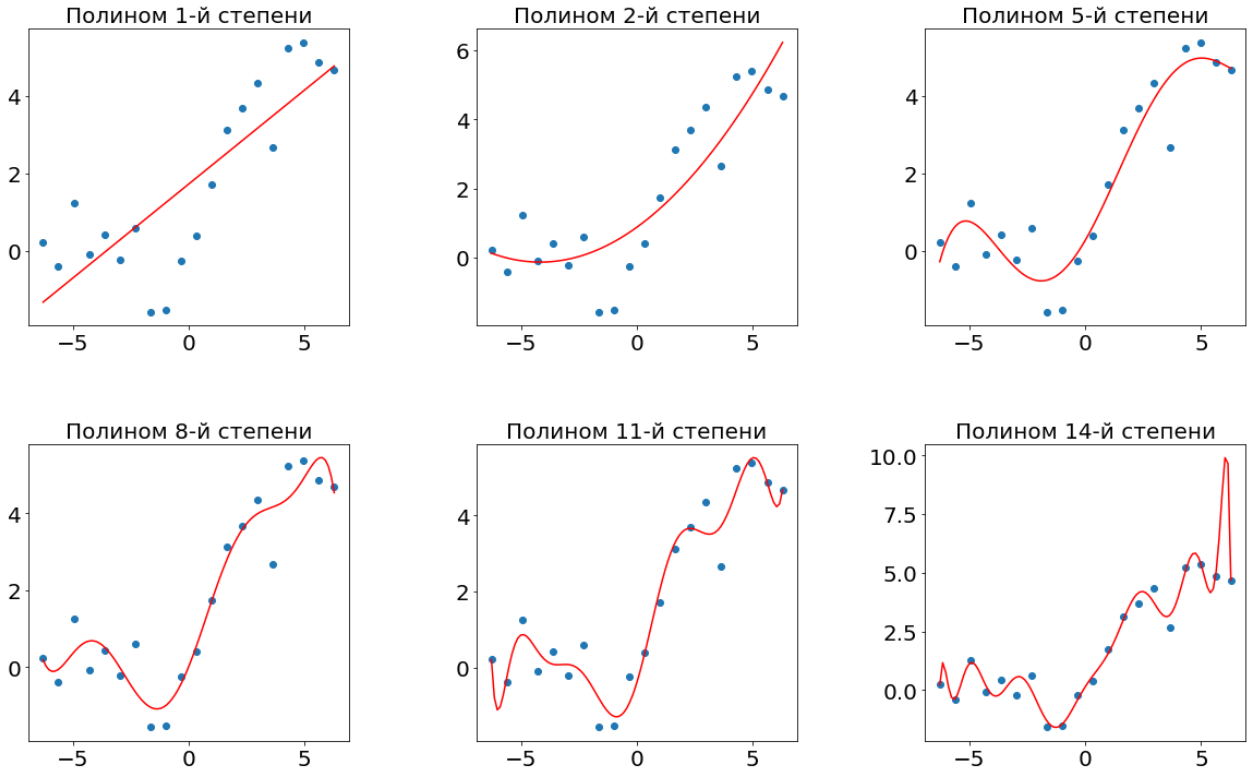

Рис. 1. Демонстрация эффекта переобучения на примере полиномиальной регрессии.

Ёмкость модели, определяемая пространством гипотез, среди которых модель ищет решение, также называют *«представительная ёмкость»* (англ. representational capacity) [2]. Поиск подходящей функции из пространства гипотез является сложной задачей оптимизации, и практически никогда не удаётся найти наилучшее решение этой задачи, так как все используемые алгоритмы оптимизации не гарантируют нахождение глобального оптимума. В связи с этим вводится понятие *«эффективная ёмкость»* (англ. effective capacity) [2], которая ограничивает представительную ёмкость модели, учитывая ограничения используемого алгоритма оптимизации.

В процессе разработки нейронной сети помимо выбора архитектуры и настройки параметров (синаптических весов и смещений), также необходимо подбирать значения **гиперпараметров**. К гиперпараметрам относятся все те параметры, которые не настраиваются в процессе обучения алгоритмом оптимизации. В основном это параметры самого алгоритма оптимизации (например, скорость обучения α, максимальное количество эпох обучения и т. д.). Если настраивать гиперпараметры, руководствуясь значеним метрики на тестовой выборке, оно перестаёт быть объективным показателем обобщающей способности. Поэтому для этих целей выделяют ещё одну (третью) выборку – *валидационную* (отладочную) выборку. При помощи валидационной выборки подбирают значения гиперпараметров, а также контролируют эффект переобучения.

Чтобы контролировать эффект переобучения строятся так называемые *кривые обучения*, которые представляют собой графики зависимости значения функции потерь или метрики на обучающей и на валидационной выборках.

При возникновении переобучения в какой-то момент времени ошибка на валидационной выборке перестанет уменьшаться, и может даже начать расти.

Этот момент можно отследить по кривым обучения (рис. 12). Самый простой способ борьбы с переобучением – это ранняя остановка (англ. early stopping). Этот приём заключается в том, что нужно остановить процесс обучения как только на графиках появляются признаки переобучения. Таким образом можно подобрать оптимальное значение одного из гиперпараметров – максимальное количество эпох обучения.

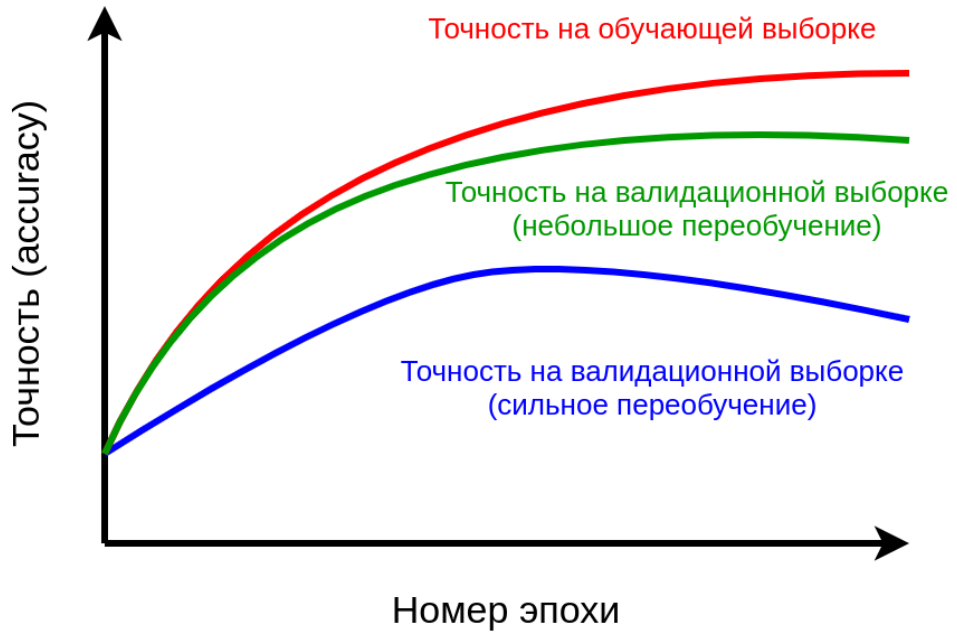

Рис. 2. Кривые обучения.

Расстояние Кульбака-Лейблера

Для задачи бинарной классификации подходящей функцией активации выходного нейрона является сигмоида, так как её значения принадлежат интервалу (0, 1) и их можно интерпретировать как вероятность «положительного» класса. Однако, если классов более двух, сигмоида может не подойти. Задачи классификации со многими классами можно разделить на задачи с пере-

секающими классами (англ. multilabel) и задачи многоклассовой классификации (англ. multiclass). В первом случае один объект может принадлежать одновременно нескольким классам, а во втором случае – только одному. Для задачи классификации с пересекающимися классами сигмоида является подходящим выбором, так как вероятности принадлежности к каждому классу могут быть оценены независимо друг от друга. В случае многоклассовой классификации на оценки вероятностей принадлежности объекта к каждому из классов накладывается дополнительное ограничение – для них должно выполняться условие нормировки, то есть в сумме они должны давать 1, так как согласно постановке задачи классифицируемый объект гарантированно принадлежит одному из рассматриваемых классов.

Для задачи многоклассовой классификации существует специальная функция активации для нейронов выходного слоя, которая получила название Softmax:

$$Softmax(v_q) = \frac{e^{v_q}}{\sum_{j=1}^{k} e^{v_j}},$$

где q – некоторый нейрон выходного слоя, k – количество нейронов в выходном слое (равное количеству классов), v_q – взвешенная сумма входных сигналов.

Таким образом при использовании Softmax на выходе из нейронной сети получается вектор

$$\mathbf{q} = (q_1, \ldots, q_k),$$

где q_i – вероятность i-го класса ($q_i = \hat{y}_i$), причём выполняется условие $\sum_{i=1}^{k} q_i = 1$. Этот вектор задаёт **категориальное распределение** (англ. categorical distribution), которое также называют обобщённым распределением Бернулли. Обозначим указанное выше распределение как Q.

Метка класса объекта, закодированная в унитарном коде (англ. one-hot encoding), также задаёт категориальное распределение с вектором параметров

$$\mathbf{p} = (p_1, \dots, p_k),$$

$$p_i = \begin{cases} 1, & \text{если объект принадлежит } i-\text{му классу}, \\ 0, & \text{иначе.} \end{cases}$$

Обозначим это распределение за P. Тогда получается, что цель обучения нейронной сети состоит в том, чтобы для каждого объекта обучающей выборки распределение Q, котрое выдаёт нейронная сеть, и распределение P, которое определяется данными, были одинаковыми. Иными словами, распределение Q является некоторой аппросимацией распределения P.

Для оценки степени близости двух вероятностных распределений используется **расстояние Кульбака-Лейблера** (англ. Kullback–Leibler divergence, KL-Divergence) [3]. Для дискретных распределений (к которым относится и категориальное распределение) KL-Divergence определяется как

$$KL(P||Q) = \sum_{i=1}^{k} p_i \log \frac{p_i}{q_i}.$$

Это выражение можно использовать в функции потерь для обучения нейронной сети. Однако, его можно упростить

$$\begin{aligned} KL(P||Q) &= \sum_{i=1}^{k} p_i \log \frac{p_i}{q_i} = \\ &= -\sum_{i=1}^{k} p_i \log q_i - \left(-\sum_{i=1}^{k} p_i \log p_i \right) = \\ &= H(P,Q) - H(P), \end{aligned}$$

где $H(P,Q)$ – перекрёстная энтропия между распределениями P и Q, $H(P)$ – энтропия распределения P.

Так как величина $H(P)$ определяется исключительно данными и не зависит от параметров нейронной сети, это слагаемое можно не учитывать при её обучении. Таким образом, в данном случае минимизация расстояния Кульбака-Лейблера эквивалентна минимизации перекрёстной энтропии

$$H(P,Q) = -\sum_{i=1}^{k} p_i \log q_i = -\sum_{i=1}^{k} y_i \log \hat{y}_i.$$

Это же выражение можно получить воспользовавшись методом максимального правдоподобия [1]. Значение $\hat{y}_q$ для некоторого нейрона q выходного слоя есть ничто иное как оценка вероятности $P(y = q \mid \mathbf{x}, \mathbf{w})$. Тогда функцию правдоподобия можно записать следующим образом

$$l(\mathbf{w}) = \prod_{i=1}^{n} P\left(\mathbf{y_i} | \mathbf{x_i}, \mathbf{w}\right) = \prod_{i=1}^{n} \prod_{j=1}^{k} \hat{y}_{ij}^{y_{ij}},$$

где $\hat{y}_{ij}$ – фактическое выходное значение нейрона j для объекта i, y_{ij} – желаемое выходное значение нейрона j для объекта i, k – количество классов, n – количество объектов в обучающей выборке, $\mathbf{y_i}$ – метка класса объекта i, закодированная в унитарном коде (англ. one-hot encoding), $\mathbf{x_i}$ – вектор параметров для объекта i, $\mathbf{w}$ – вектор параметров нейронной сети.

Если взять логарифм от функции правдоподобия и умножить результат на $-1/n$, получим следующую функцию потерь

$$L(\mathbf{w}) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} y_{ij} \log \hat{y}_{ij}.$$

Решение задачи распознавания рукописных цифр

Процесс решения задачи распознавания рукописных цифр при помощи многослойной нейронной сети во многом похож на процесс решения этой задачи с помощью однослойного персептрона. Основное отличие состоит в том, что для обучения многослойной нейронной сети нужно реализовать алгоритм обратного распространения ошибки. В данном разделе описано, как это можно сделать.

Будем использовать те же обозначения, что и в предыдущей главе, для обозначения матрицы входных значений $\mathbf{X}$ и вектора желаемых откликов $\mathbf{y}$. Процесс предобработки входных значений (нормализация) и желаемых откликов (one-hot encoding) остаётся без изменений. В качестве данных для обучения нейронной сети будем так же использовать набор данных MNIST, то есть размер обучающей выборки составит $n = 60000$, а рамерность вектора признаков для каждого примера $\mathbf{x_i}$ составит $m = 784$.

Для решения задачи распознавания рукописных цифр будем использовать полносвязную нейронную сеть прямого распространения с одним скрытым слоем. В качестве функции активации скрытых слоёв используем сигмоиду, а качестве функции активации нейронов выходного слоя – softmax.

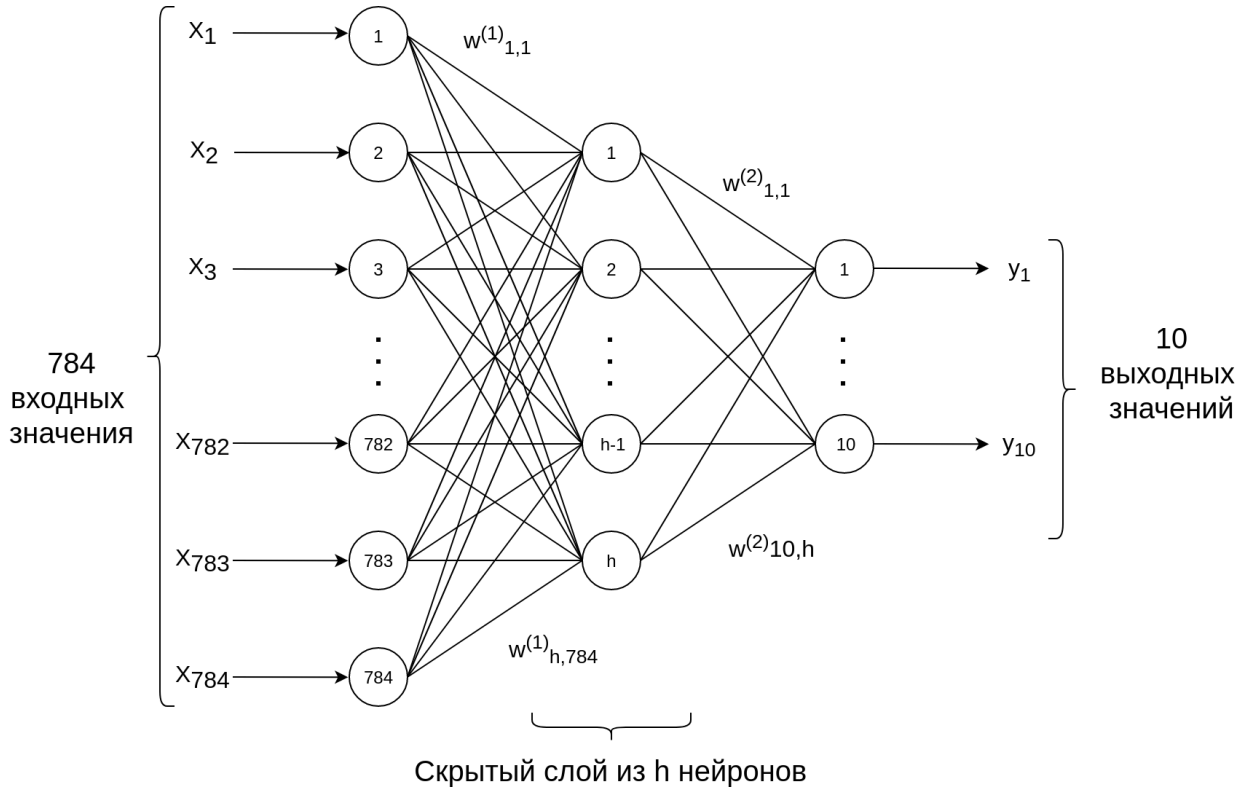

Рис. 3. Архитектура используемой нейронной сети.

Так как в нейронной сети теперь больше одного слоя, необходимо ввести дополнительный индекс в обозначения весов и смещений, показывающий, какому слою принадлежит нейрон. Обозначим за l номер слоя. Тогда вес связи, соединяющей i-й нейрон слоя l с j-м нейроном слоя $l-1$ можно обозначить как $w_{ij}^{(l)}$. Тогда работу нейрона можно описать при помощи следующего набора уравнений

$$v_q^{(l)} = \sum_{i=1}^{m} w_{qi}^{(l)} x_i + b_q^{(l)}, \quad \hat{y}_q^{(l)} = \varphi(v_q^{(l)}),$$

$$\varphi(v_q^{(l)}) = \begin{cases} e^{v_q^{(l)}} / \sum_{j=1}^{k} e^{v_j^{(l)}}, & \text{если } l - \text{ выходной слой}, \\ 1/(1+e^{-v_q^{(l)}}), & \text{иначе.} \end{cases}$$

Параметры нейронной сети могут быть представлены в виде следующей пары матриц (смещения опустим для простоты изложения)

$$\mathbf{W^{(1)}} = \begin{bmatrix} w_{11}^{(1)} & w_{21}^{(1)} & \dots & w_{h1}^{(1)} \\ w_{12}^{(1)} & w_{22}^{(1)} & \dots & w_{h2}^{(1)} \\ \dots & \dots & \dots & \dots \\ w_{1m}^{(1)} & w_{2m}^{(1)} & \dots & w_{hm}^{(1)} \end{bmatrix}, \qquad \mathbf{W^{(2)}} = \begin{bmatrix} w_{11}^{(2)} & w_{21}^{(2)} & \dots & w_{k1}^{(2)} \\ w_{12}^{(2)} & w_{22}^{(2)} & \dots & w_{k2}^{(2)} \\ \dots & \dots & \dots & \dots \\ w_{1h}^{(2)} & w_{2h}^{(2)} & \dots & w_{kh}^{(2)} \end{bmatrix}.$$

Тогда выходное значение может быть вычислено путём вычисления двух матричных произведений с последующим применением функции активации к каждому элементу получившихся матриц:

$$\hat{\mathbf{Y}}^{(1)} = \varphi(\mathbf{X}\mathbf{W}^{(1)}),$$

$$\hat{\mathbf{Y}}^{(2)} = \varphi(\hat{\mathbf{Y}}^{(1)}\mathbf{W}^{(2)}) = \begin{bmatrix} \hat{y}_{11}^{(2)} & \hat{y}_{12}^{(2)} & \dots & \hat{y}_{1k}^{(2)} \\ \hat{y}_{21}^{(2)} & \hat{y}_{22}^{(2)} & \dots & \hat{y}_{2k}^{(2)} \\ \dots & \dots & \dots & \dots \\ \hat{y}_{n1}^{(2)} & \hat{y}_{n2}^{(2)} & \dots & \hat{y}_{nk}^{(2)} \end{bmatrix},$$

где $\hat{\mathbf{Y}}^{(1)}$ – матрица выходных значений нейронов скрытого слоя, $\hat{y}_{nk}^{(2)}$ – выходное значение нейрона k последнего (выходного) слоя для объекта n.

Обучение нейронной сети будет заключаться в минимизации целевой функции, то есть в нахождении таких значений параметров нейронной сети, при которых её ошибка на обучающей выборке минимальна. Так как решается задача многоклассовой классификации, в качестве целевой функции будем использовать перекрёстную энтропию

$$L(\mathbf{w}) = -\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij} \log(\hat{y}_{ij}^{(2)}),$$

где n – размер обучающей выборки, k – количество классов, y_{ij} – желаемый отклик k-го нейрона выходного слоя для объекта i, $\hat{y}_{ij}^{(2)}$ – фактический отклик k-го нейрона выходного слоя для объекта i.

В качестве алгоритма оптимизации будем использовать стохастический градиентный спуск.

Для обучения нейронной сети необходимо выполнить следующие шаги.

Шаг 1 Инициализировать матрицы весов $\mathbf{W^{(1)}}$ и $\mathbf{W^{(2)}}$ небольшими случайными числами.

Шаг 2 Для каждого объекта i из обучающей выборки:

- Вычислить вектор выходных значений $\hat{\mathbf{Y}}_\mathbf{i}^{(\mathbf{2})}$.
- Вычислить вектор значений δ для параметров выходного слоя

$$\Delta^{(2)} = (\hat{\mathbf{Y}}_\mathbf{i}^{(\mathbf{2})} - \mathbf{Y_i}) = \begin{bmatrix} \delta_{i1}^{(2)} & \delta_{i2}^{(2)} & \dots & \delta_{ik}^{(2)} \end{bmatrix}.$$

- Вычислить $\frac{\partial L_i}{\partial \mathbf{W^{(2)}}}$

$$\frac{\partial L_i}{\partial \mathbf{W^{(2)}}} = \left[\hat{\mathbf{Y}}_\mathbf{i}^{(\mathbf{1})}\right]^T \Delta^{(2)} = \begin{bmatrix} \hat{y}_{i1}^{(1)} \\ \hat{y}_{i2}^{(1)} \\ \dots \\ \hat{y}_{ih}^{(1)} \end{bmatrix} \begin{bmatrix} \delta_{i1}^{(2)} & \delta_{i2}^{(2)} & \dots & \delta_{ik}^{(2)} \end{bmatrix}$$

- Вычислить вектор значений δ для параметров скрытого слоя

$$\Delta^{(1)} = (\hat{\mathbf{Y}}_\mathbf{i}^{(\mathbf{1})} \odot (\mathbf{I_h} - \hat{\mathbf{Y}}_\mathbf{i}^{(\mathbf{1})})) \odot \Delta^{(2)}\left[\mathbf{W^{(2)}}\right]^T =$$

$$= \left[\hat{y}_{i1}^{(1)}(1 - \hat{y}_{i1}^{(1)}) \sum_{j=1}^{k} \delta_{ij}^{(2)} w_{j1}^{(2)} \quad \dots \quad \hat{y}_{ih}^{(1)}(1 - \hat{y}_{ih}^{(1)}) \sum_{j=1}^{k} \delta_{ij}^{(2)} w_{jh}^{(2)}\right],$$

где $\mathbf{I_h}$ – вектор-строка из h единиц, $\odot$ – произведение Адамара (покомпонентное произведение).

- Обновить значения весов по следующим формулам

$$\mathbf{W^{(1)}} = \mathbf{W^{(1)}} - \alpha \frac{\partial L_i}{\partial \mathbf{W^{(1)}}};\ \mathbf{W^{(2)}} = \mathbf{W^{(2)}} - \alpha \frac{\partial L_i}{\partial \mathbf{W^{(2)}}}.$$

Шаг 3 Повторять шаг 2 до тех пор, пока значение ошибки не опустится ниже порогового значения, либо не будет достигнуто максимальное число эпох обучения.

Вопросы для самопроверки

1. Что такое обобщающая способность модели?
2. Что такое явления переобучения и недообучения?
3. Как связана ёмкость модели с её потенциальной склонностью к переобучению?
4. Что такое расстояние Кульбака-Лейблера? Для чего оно используется?
5. Является ли расстояние Кульбака-Лейблера коммутативным?

Список литературы

1. Bishop C. Pattern Recognition and Machine Learning. Springer, 2006.
2. Goodfellow I. J., Bengio Y., Courville A. Deep Learning. MIT Press, 2016.
3. Géron A. A Short Introduction to Entropy, Cross-Entropy and KL-Divergence // YouTube. 2018. URL: https://www.youtube.com/watch?v=ErfnhcEV1O8 (дата обращения: 17.09.2020)
4. Katanforoosh, Kunin et al. «Parameter optimization in neural networks», 2019. URL: http://deeplearning.ai/ai-notes/optimization/ (дата обращения: 17.09.2020)

5. Nielsen M. A. Neural Networks and Deep Learning. Determination Press, 2015.

6. Уоссермен Ф. Нейрокомпьютерная техника : Теория и практика. М.: Мир, 1992. 240 с.

7. Хайкин С. Нейронные сети: полный курс, 2-е издание.: Пер. с англ. М.: Издательский дом «Вильямс», 2006. 1104 с.