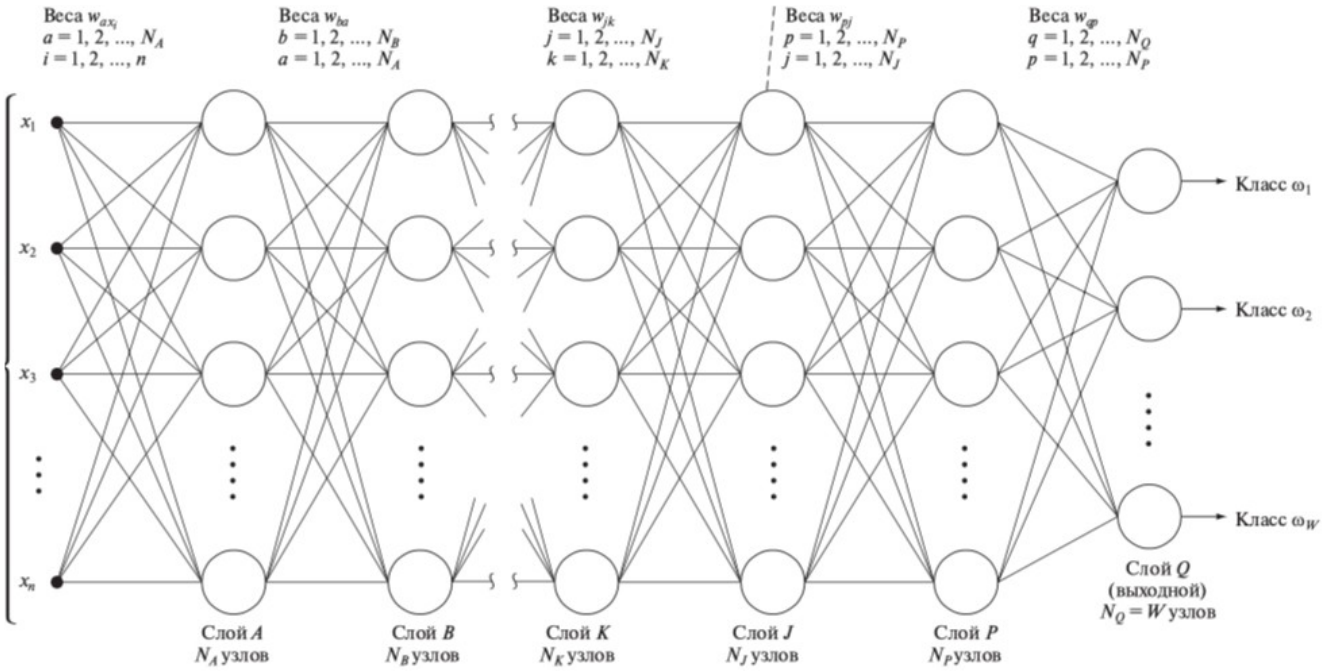
ЛР8: Распознавание рукописных цифр

Цель: Использование искусственных нейронных сетей для распознавая рукописных цифр.

Теоретическая часть

Нейронные сети состоят из большого числа простейших нелинейных вычислительных элементов (нейронов), которые организованы в виде сетей, напоминающих предположительный способ соединения нейронов в мозге человека. Архитектура модели нейронной сети представлена на рисунке.



Она состоит из слоев, в которых находятся идентичные по структуре вычислительные узлы (нейроны), организованные таким образом, что выход каждого нейрона одного слоя соединяется со входом каждого нейрона следующего слоя.

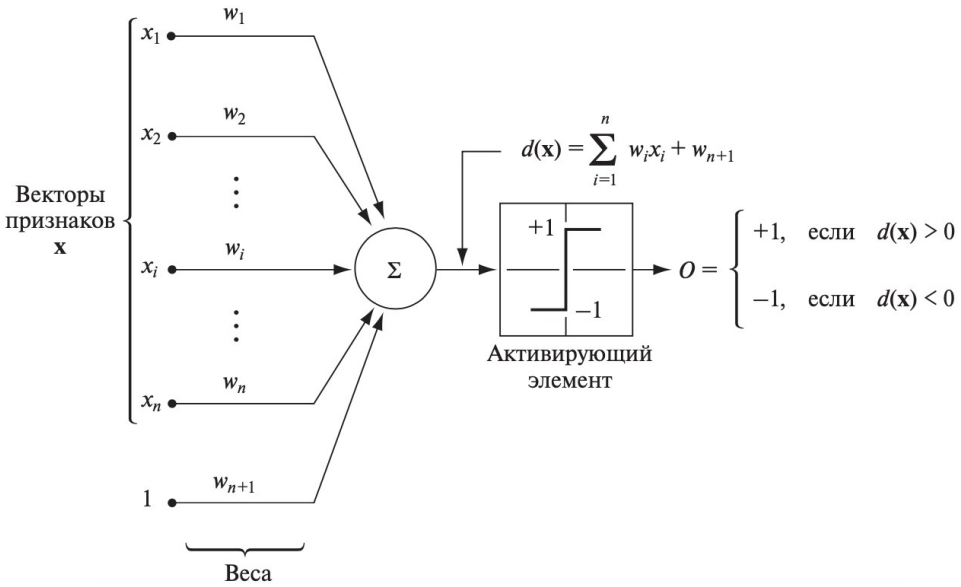
Число нейронов в первом слое, называемом слоем A, равно N_A; оно часто выбирается равным размерности входных векторов-образов: $N_A = n$. Число нейронов выходного слоя, называемого слоем Q, обозначается N_Q. Это число N_Q равно W — числу классов, образы которых данная нейронная сеть обучена распознавать.

Сеть распознает объект с вектором признаков **x** как принадлежащий классу ω_i, если на i-м выходе сети присутствует «высокий» уровень, а на остальных выходах — «низкий».

Выходной сигнал (реакция) нейрона базируется на взвешенной сумме его входных сигналов:

$$d(\mathbf{x}) = \sum_{i=1}^{n} w_i\, x_i + w_{n+1} \qquad (1.1)$$

Коэффициенты w_i, называемые весами, масштабируют входные сигналы перед тем, как они суммируются и подаются на пороговое устройство. В этом смысле веса аналогичны синапсам в нервной системе человека. Функцию, которая



отображает результат суммирования в конечный выходной сигнал устройства, называют *функцией активации*. В качестве функции активации используется непрерывная S-образная функция (сигмоида), поскольку для обучения сети необходима дифференцируемость вдоль всех путей в нейронной сети:

$$h_j(I_j) = \frac{1}{1 + \exp(-I_j)} \qquad (1.2)$$

где I_j, j = 1, 2,..., N_J — значение на входе активирующего элемента каждого узла слоя J нейронной сети.

В качестве функции активации на последнем слое сети часто используется функция softmax. Softmax — это обобщение логистической функции для многомерного случая. Функция преобразует вектор z размерности K в вектор σ той же размерности, где каждая координата σi полученного вектора представлена вещественным числом в интервале [0,1] и сумма координат равна 1:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}} \quad (1.3)$$

где i — номер строки вектора z; K — размер ветора.

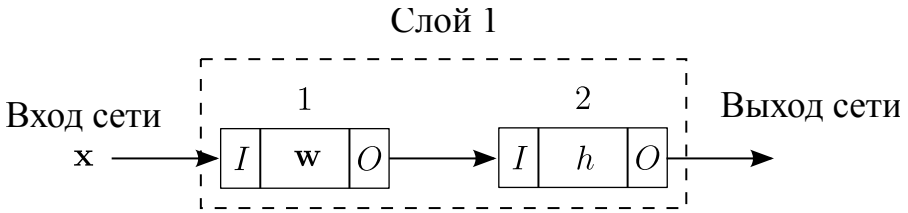
Учтем коэффициент w_{n+1} (смещение или *bias*) в векторе **y** и перепишем уравнение с помощью матричной нотации

$$d(\mathbf{x}) = \sum_{i=1}^{n} w_i\, x_i + w_{n+1} = \mathbf{w} * \mathbf{y} \quad (1.4)$$

где $y = (x_1, x_2,..., x_n, 1)^T$ — расширенный вектор признаков, а $w = (w_1, w_2,..., w_n, w_{n+1})^T$ называется весовым вектором; * - операция матричного умножения.

Используемые обозначения

Сигнал на входе элементов i=1,2 для J-го слоя сети обозначается I_{ij}, $j = 1,...,N_J$, т.е. I_{11} есть сигнал на входе взвешивающего элемента 1 слоя 1, I_{21}— сигнал на входе активирующего элемента 2 слоя 1 и т.д.



У каждого элемента w в слое J имеется N_K входов, но каждый отдельный вход умножается на свой собственный весовой коэффициент. Так, N_K входов слое J взвешиваются с коэффициентами w_{kj}, $k = 1,...,N_K$. Следовательно, для преобразования выходных сигналов требуется в общей сложности $N_J \times N_K$ коэффициентов. Чтобы полностью описать элементы w в слое J, необходимы еще дополнительные N_J коэффициентов — смещений w_{n+1}.

$$O_{k+1} = L_k(I_k) = h_k(\mathbf{w}_k * I_k)$$	(1.5)

где J — номер слоя, а w — весовые коэффициенты; h — функция активации; L — J слой нейронной сети.

Обучение путем обратного распространения ошибки

Что мы понимаем под обучением? В работе Mitchell (1997) дано такое лаконичное определение: «Говорят, что компьютерная программа обучается, если качество на задачах из класса T, измеренное с помощью меры качества P, возрастает с ростом опыта E». В большинстве алгоритмов машинного обучения опытом является просто набор данных, который можно описать разными способами. Но в любом случае набор данных – это совокупность примеров, каждый из которых является совокупностью признаков.

Чтобы оценить возможности алгоритма необходимо иметь меру качества. Иногда трудно решить, что же нужно измерять. Например, в задаче транскрипции нужно измерять верность при транскрипции всей последовательности или лучше использовать более детальную меру, которая поощряет за правильную транскрипцию отдельных элементов последовательности? В задаче регрессии за что штрафовать систему сильнее: за частые ошибки средней серьезности или за редкие, но очень серьезные ошибки? Ответ на такие вопросы зависит от приложения. Классификация, транскрипция измеряется точность (accuracy) — долю примеров, для которых модель правильный результат.

Алгоритмы машинного обучения можно разделить на два больших класса, **без учителя** и **с учителем**, в зависимости от опыта на котором они могут обучаться. Алгоритму обучения без учителя в качестве опыта предъявляется набор данных, содержащий много признаков, а алгоритм должен выявить полезные структурные свойства этого набора (например, разделить набор данных на кластеры). Алгоритму обучения с учителем предъявляется набор данных, содержащий признаки, в котором каждый пример снабжен **меткой**, или **целевым классом**.

Грубо говоря, обучение без учителя означает наблюдение нескольких примеров случайного вектора **x** с последующей попыткой вывести, явно или неявно, распределение вероятности p(**x**) или некоторые интересные свойства этого распределения. А обучение с учителем сводится к наблюдению нескольких примеров случайного вектора **x** и ассоциированного с ним значения или вектора **y** с последующей попыткой предсказать **y** по **x**, обычно в виде оценки p(**y** | **x**). Мы считаем, что метка y предоставлена неким учителем, который объясняет системе машинного обучения, что делать, – отсюда и название «обучение с учителем». В

обучении без учителя никакого учителя не существует, и алгоритм должен извлечь из данных смысл без подсказки.

Главная проблема машинного обучения состоит в том, что алгоритм должен хорошо работать на новых данных, которых он раньше не видел, а не только на тех, что использовались для обучения модели. Эта способность правильной работы на ранее не предъявленных данных называется **обобщением**.

В процессе обучения мы минимизирует **ошибку обучения**. Но в машинном обучения необходимо также уменьшить ошибку обобщения (тестирования). Таким образом возникает вопрос: Как можно повлиять на качество работы на тестовом наборе, если для наблюдения доступен только обучающий набор? Можно математически проследить связь если выполнить условия:

1. примеры к каждом наборе независимы;
2. **одинаково распределены** (выбираются из одного и того же распределения вероятности).

Мы выбираем обучающий набор, используем его для минимизации ошибки обучения, а затем выбираем тестовый набор. При таком процессе ожидаемая ошибка тестирования больше или равна ожидаемой ошибке обучения. Факторов, определяющих качество работы алгоритма машинного обучения, два:

1. сделать ошибку обучения как можно меньше;
2. сократить разрыв между ошибками обучения и тестирования.

Эти факторы соответствуют двум центральным проблемам машинного обучения: **недообучению** и **переобучению**. Недообучение имеет место, когда модель не позволяет получить достаточно малую ошибку на обучающем наборе, а переобучение – когда разрыв между ошибками обучения и тестирования слишком велик.

Управлять склонностью модели к переобучению или недообучению позволяет ее **емкость** (**capacity**). Модели малой емкости испытывают сложности в аппроксимации обучающего набора. Модели большой емкости склонны к переобучению, поскольку запоминают свойства обучающего набора, не присущие тестовому.

Расхождение между **ошибкой обучения** и **ошибкой обобщения** растет с ростом емкости модели, но убывает по мере увеличения количества обучающих примеров. Ожидаемая ошибка обобщения никогда не может увеличиться с ростом количества обучающих примеров.

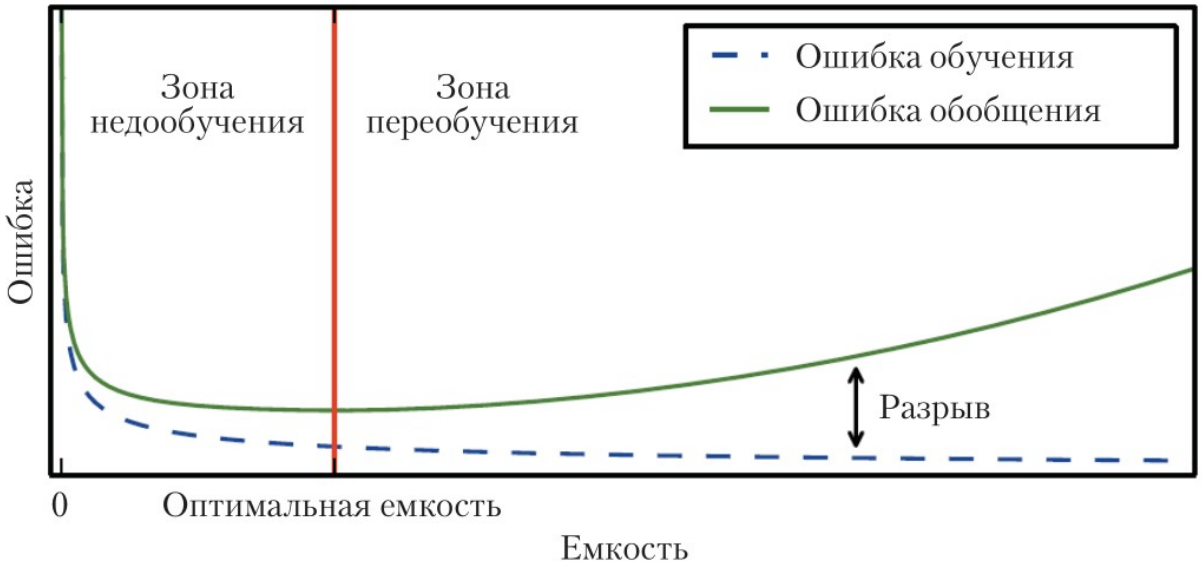


Рис. 1.1 Типичная связь между емкостью и ошибкой.

У большинства алгоритмов машинного обучения имеются гиперпараметры, управляющие поведением алгоритма. К гиперпараметрам также относят параметры управляющие емкостью модели. Подбор гиперпараметров в процессе обучения - бессмысленна. При попытке обучить их на обучающем наборе всегда выбиралась бы максимально возможная емкость модели, что приводило бы к переобучению.

Метод обучения нейронных сетей на каждом шаге обучения минимизирует ошибку между фактической **O** и желаемой реакциями **r**. В качестве функции ошибки будем использовать кросс-энтропию:

$$y = N(\mathbf{w}, \mathbf{x})$$
$$E(\mathbf{w}) = -\sum_{i=1}^{q}\sum_{j=1}^{l} \mathbf{r}_j \log(\mathbf{y}_{ji}) \quad (1.6)$$

где **r** — желаемая реакция (метки); N — нейронная сеть; **w** — веса нейронной сети; q – число элементов в выборке; l – число классов; $\mathbf{y}_{ji}$ – ответ нейронной сети на i-м объекте на вопрос принадлежности его к j-му классу, $\mathbf{y}_{ji}$=1 если i-й объект принадлежит j-му классу, в противном случае $\mathbf{y}_{ji}$=0.

Задача состоит в том, чтобы путем последовательных приращений корректировать весовой вектор w в обратном направлении по отношению к вектору градиента функции E(w), чтобы найти минимум этой функции, который достигается при r = N(w,x); т.е. минимум соответствует безошибочной классификации.

Если обозначить через w(k) весовой вектор на k-м шаге итерации, то в обобщенном виде можно записать следующий алгоритм градиентного спуска:

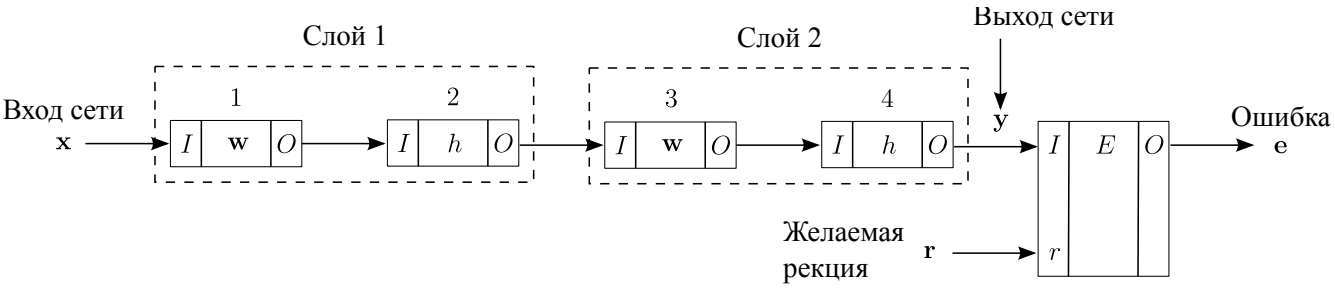
$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu(\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}) \quad (1.7)$$

где w(k+1) — новое значение вектора w, а параметр μ > 0 задает коэффициент скорости сходимости алгоритма. От выбора параметра μ зависит устойчивость и скорость сходимости алгоритма.

Рассмотрим обучение двухслойной нейронной сети с функцией активации softmax на выходе сети:

$$\mathbf{y} = N(\mathbf{w}_1, \mathbf{w}_2) = \sigma(\mathbf{w}_2\, h(\mathbf{w}_1 \mathbf{x})) \quad (1.8)$$

Сеть состоит из двух слоев, следовательно необходимо пересчитывать два набора коэффициентов w1 и w2. Схема обратного распространения ошибки для сети заданной уравнением (1.7) показана на рисунке



Распишем уравнения частной производной для E(w1,w2), с помощью уравнения производной сложной функции:

$$\frac{\partial E(\mathbf{w}_1, \mathbf{w}_2)}{\partial \mathbf{w}_2} = \frac{\partial E(\mathbf{w}_1, \mathbf{w}_2)}{\partial \mathbf{O}_4} \frac{\partial \mathbf{O}_4}{\partial \mathbf{O}_3} \frac{\partial \mathbf{O}_3}{\partial \mathbf{w}_2}$$

$$\frac{\partial E(\mathbf{w}_1, \mathbf{w}_2)}{\partial \mathbf{w}_1} = \frac{\partial E(\mathbf{w}_1, \mathbf{w}_2)}{\partial \mathbf{O}_4} \frac{\partial \mathbf{O}_4}{\partial \mathbf{O}_3} \frac{\partial \mathbf{O}_3}{\partial \mathbf{O}_2} \frac{\partial \mathbf{O}_2}{\partial \mathbf{O}_1} \frac{\partial \mathbf{O}_1}{\partial \mathbf{w}_1} \quad (1.9)$$

Уравнения обновления коэффициентов состоят из частных производных выходов O_k по входу $I_k=O_{k-1}$. Для обновления коэффицинтов нужно умножить результат вычисления на прошлых этапах на частную производную выхода O_k по весовым коэффициентам w.

Частная производная функции ошибки E(w1,w2) в случае, если функция активации на последнем слое равна softmax:

$$\frac{\partial E(\mathbf{w}_1, \mathbf{w}_2)}{\partial \mathbf{O}_4} \frac{\partial \mathbf{O}_4}{\partial \mathbf{O}_3} = \mathbf{O}_4 - \mathbf{r} \tag{1.10}$$

Частная производная функции h(I) равна:

$$\frac{\partial \mathbf{O}_2}{\partial \mathbf{O}_1} = \frac{\partial (h(\mathbf{O}_1))}{\partial \mathbf{O}_1} = \mathbf{O}_2(1 - \mathbf{O}_2) \tag{1.11}$$

Частная производная функции wx равна:

$$\frac{\partial \mathbf{O}_3}{\partial \mathbf{w}_2} = \frac{\partial (\mathbf{w}_2 * \mathbf{I}_3)}{\partial \mathbf{w}_2} = \mathbf{I}_3$$

$$\frac{\partial \mathbf{O}_1}{\partial \mathbf{w}_1} = \frac{\partial (\mathbf{w}_1 * \mathbf{x})}{\partial \mathbf{w}_1} = \mathbf{x} \tag{1.12}$$

$$\frac{\partial \mathbf{O}_1}{\partial \mathbf{O}_2} = \frac{\partial (\mathbf{w}_2 * \mathbf{I}_3)}{\partial \mathbf{O}_2} = \frac{\partial (\mathbf{w}_2 * \mathbf{O}_2)}{\partial \mathbf{O}_2} = \mathbf{w}_2$$

Процесс обучения начинается с произвольного набора весов в узлах сети (но не всех одинаковых). После этого процесс обучения на любом шаге итерации складывается из двух основных этапов:

1) На первом этапе на вход сети предъявляется обучающий вектор признаков **x**, и сигналы распространяются по слоям сети. Затем выход сети сравнивается с желаемыми выходными сигналами **r**, в результате чего строятся составляющие ошибок **e**.

2) На втором этапе осуществляется обратный проход по сети, при котором сигнал ошибки передается о обратной направлении по сети, что позволяет нужным образом откорректировать его веса. Эта процедура применяется и к смещениям, которые, как говорилось выше, рассматриваются в качестве дополнительных весовых коэффициентов, на которые умножается единичный сигнал, подаваемый на вход сумматора каждого узла нейронной сети.

Обычная практика состоит в том, чтобы прослеживать ошибки сети **e**. При успешном сеансе обучения величина ошибки сети уменьшается по мере роста числа итераций, и процедура обучения сходится к устойчивому набору весовых векторов, в котором в случае дополнительного обучения наблюдаются лишь небольшие флуктуации. Для выяснения того, что в процессе обучения некоторый

образ классифицируется правильно, необходимо проверить, что реакция узла выходного слоя, сопоставляемого тому классу, к которому принадлежит данный образ, имеет высокий уровень, а сигналы остальных узлов выходного слоя имеют низкий уровень.

После того, как обучение системы закончено, она применяется для классификации неизвестных образов с использованием тех значений параметров **w**, которые были установлены в процессе обучения. Сигналы любого поступающего образа свободно распространяются по всем слоям, и образ классифицируется как принадлежащий классу, который соответствует узлу с высоким уровнем на выходе. Если высокий уровень отмечается сразу на нескольких узлах выходного слоя сети или не обнаруживается ни на одном из узлов, то либо объявляется об отказе от классификации, либо принимается решение отнести объект к тому классу, на выходном узле которого присутствует сигнал максимальной амплитуды.

Учебный набор данных

Набор данных dataset.mat содержит примеры рукописных цифр от 0 до 9. Набор данных состоит из двух массивов inputs и targets. Inputs — матрица размерами 256x11000 примеров, столбец матрицы представляется собой преобразованное в вектор изображение. Для отображения использовать команду imagesc(reshape(data.training.inputs(:, 1), 16, 16)). Targets — матрица размерами 1x11000 содержит ответы показанных цифр на изображениях. На картинке inputs(:, 1) изображена цифра 0, targets(:, 1) равен 0. Для преобразования в onehot-код использовать команду labels = targets == (0:9)'.

Используемые функции

main	Главный файл tr_data_ext - расширенный набор данных для обучения tr_targets - метки классов набора данных для обучения test_data_ext - расширенный набор данныхдля теста test_targets - метки классов набора данных для теста
Dense	Полносвязный слой
sigmoid	Функция активации сигмоида
softmax	Функция активации softmax
E_logloss	Функция ошибки кросс-энтропия
forward	Функция вычисляет результат работы сети

training	Функция обучения сети
testing	Функция тестирования сети

Ход работы

1. **Подготовка сети:**
   a) Загрузить набор данных (dataset) с помощью команды load('dataset.mat').
   b) Сформировать тестовый набор путем разделения <<датасета>> на две части в соотношении 80/20: большая часть используется для обучения — training набор, меньшая как тестовый набор - test.
   c) Расширить обучающий и тестовый набор данных по формуле (1.4).
   d) Составить уравнение многослоной сети с помощью уравнения (1.5) на выходе последнего слоя обязательно должна быть функция активации *softmax*.
2. **Заполнение недостающих частей в классах**
   a) Заполнить функции нужно в местах отмеченных *%fill*

```
function dx = backward( self, x )
    % fill
    dx = ...
end
```

   Вместо … нужно записать уравнение. X — входное значение, dx — выходное.
   b) Класс Dense
      1. Инициализировать веса случайными значениями в конструкторе класса

         w=2*randn(10,256)/sqrt(10+256).

      2. Заполнить функции forward и backward. Обновить веса в соотвествии с уравнением (1.12).
   c) Класс sigmoid. Заполнить функции forward и backward.
   d) Класс softmax. Заполнить функции forward.
   e) Класс E_logloss. Заполнить функции forward и backward с учетом того, что последний слой использует функцию активации softmax.
3. **Обучение сети**
   a) Обучить сеть на 100 эпохах. Использовать для обучения набор данных tr_data_ext.
4. **Тест сети**
   a) Выполнить проверку обученной сети на тестом наборе данных test_data_ext.
5. **Подобрать размер сети обеспечивающий максимальное качество распознавания цифр**