Московский государственный университет имени М. В. Ломоносова

Факультет Вычислительной математики и Кибернетики
Кафедра Математических Методов Прогнозирования

Отчет по заданию

«Нейросетевой разреженный автокодировщик»

Выполнил:
студент 3 курса 317 группы
Таскынов Ануар

Москва, 2016

Содержание.

1 Введение.

В данном задании необходимо было реализовать нейросетевой разреженный автокодировщик. Базовая часть была выполнена полностью. Была выполнена бонусная часть на 0.5 баллов. В отчете расписано вычисление градиентов для автокодировщика. В связи с тем, что оперативная память на компьютере недостаточная для полного исследования (пункты 8 и 9 по заданию), не было проверено качество классификаторов при маленьких шагах.

Все функции были оптимизированы путем матричных вычислений. В Ipython Notebook'е выполнены все эксперименты и показаны результаты работы классификаторов и разреженного автокодировщика.

2 Предварительные вычисления.

Минимизируемый функционал:

$$J(W,b) = [\frac{1}{m}\sum_{k=1}^{m}(\frac{1}{2}||h_{W,b}(x_k) - y_k||^2)] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{j=1}^{s_l}\sum_{i=1}^{s_{l+1}}(W_{ij}^{(l)}) + \beta KL(\rho||\hat{\rho}),$$

где

$$KL(\rho||\hat{\rho}) = \sum_{j=1}^{s_{l^*}}(\rho\log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{(1-\rho)}{(1-\hat{\rho}_j)})$$

$$\hat{\rho} = \frac{1}{m}\sum_{k=1}^{m}(a_j^{(l^*)}(x_k))$$

l^* - номер среднего скрытого слоя, $a^{(l)}(x)$ - значение функции активации на l-ом слое, λ - коэффициент контроля весов, β - коэффициент разреженности.

$$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)}$$

$$a^{(l+1)} = f(z^{(l+1)})$$

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

1. Прямой проход.

2.
$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}}(\frac{1}{2}||y - h_{W,b}(x)||^2) = -(y_i - a_i^{(n_l)})f'(z_i^{(n_l)})$$
, где n_l - последний слой.

3.
$$\delta_i^{(n_l-1)} = \frac{\partial}{\partial z_i^{(n_l-1)}}(\frac{1}{2}||y-h_{W,b}(x)||^2) = \frac{\partial}{\partial z_i^{(n_l)}}(\frac{1}{2}||y-h_{W,b}(x)||^2)\frac{\partial z_i^{(n_l)}}{\partial z_i^{(n_l-1)}} =$$
$$= -(y_i - a_i^{(n_l)})f'(z_i^{(n_l)})\frac{\partial z_i^{(n_l)}}{\partial a_i^{n_l-1}}\frac{\partial a_i^{n_l-1}}{\partial z_i^{(n_l-1)}} = (W_i^{(n_l-1)}\delta_i^{n_l})f'(z_i^{(n_l)})$$

4. ...

5. - Если $l \neq l^*$:
$$\delta_i^{(l)} = (\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)}\delta_j^{(l+1)})f'(z_i^{(l)})$$
   - Если $l = l^*$:
$$\delta_i^{(l)} = \frac{\partial}{\partial z_i^{(l)}}[\frac{1}{2}||y-h_{W,b}(x)||^2+\beta\sum_{j=1}^{s_l}(\rho\log\frac{\rho}{\hat{\rho}_j}+(1-\rho)\log\frac{(1-\rho)}{(1-\hat{\rho}_i)})] =$$
$$= (\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)}\delta_j^{(l+1)})f'(z_i^{(l)}) + \beta[\rho\frac{\partial\log\frac{\rho}{\hat{\rho}_i}}{\partial z_i^{(l)}} + (1-\rho)\frac{\partial log\frac{(1-\rho)}{(1-\hat{\rho}_i)}}{\partial z_i^{(l)}}] =$$
$$= (\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)}\delta_j^{(l+1)})f'(z_i^{(l)})+\beta[\rho\frac{\partial log\frac{\rho}{\hat{\rho}_i}}{\partial\frac{\rho}{\hat{\rho}_i}}\frac{\partial\frac{\rho}{\hat{\rho}_i}}{\partial\hat{\rho}_i}\frac{\partial\hat{\rho}_i}{\partial z_i^{(l)}}+(1-\rho)\frac{\partial\log\frac{(1-\rho)}{(1-\hat{\rho}_i)}}{\partial\frac{(1-\rho)}{(1-\hat{\rho}_i)}}\frac{\partial\frac{(1-\rho)}{(1-\hat{\rho}_i)}}{\partial\hat{\rho}_i}\frac{\partial\hat{\rho}_i}{\partial z_i^{(l)}}] =$$
$$= (\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)}\delta_j^{(l+1)})f'(z_i^{(l)})+\beta[\rho\frac{\hat{\rho}_i}{\rho}(\frac{-\rho}{\hat{\rho}_i^2})+(1-\rho)\frac{(1-\hat{\rho}_i)}{(1-\rho)}\frac{(1-\rho)}{(1-\hat{\rho}_i)^2}]\frac{\partial\hat{\rho}_i}{\partial z_i^{(l)}} =$$
$$= (\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)}\delta_j^{(l+1)})f'(z_i^{(l)}) + \beta[-\frac{\rho}{\hat{\rho}_i} + \frac{(1-\rho)}{(1-\hat{\rho}_i)}]\frac{\partial\hat{\rho}_i}{\partial z_i^{(l)}} =$$
$$= [\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)}\delta_j^{(l+1)} + \beta(-\frac{\rho}{\hat{\rho}_i} + \frac{(1-\rho)}{(1-\hat{\rho}_i)})]f'(z_i^{(l)})$$

То есть:

$$\delta_i^{(l)} = [\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} + \beta(-\frac{\rho}{\hat{\rho}_i} + \frac{(1-\rho)}{(1-\hat{\rho}_i)})] f'(z_i^{(l)})$$

6. Соответственно градиенты считаются таким образом:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b,x,y) = \delta_i^{(l+1)} \frac{\partial z_i^{(l+1)}}{\partial W_{ij}^{(l)}} + \lambda W_{ij}^{(l)} = \delta_i^{(l+1)} a_j^{(l)} + \lambda W_{ij}^{(l)}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W,b,x,y) = \delta_i^{(l+1)} \frac{\partial z_i^{(l+1)}}{\partial b_i^{(l)}} = \delta_i^{(l+1)}$$

3 Эксперименты.

3.1 Точность вычисления градиента.

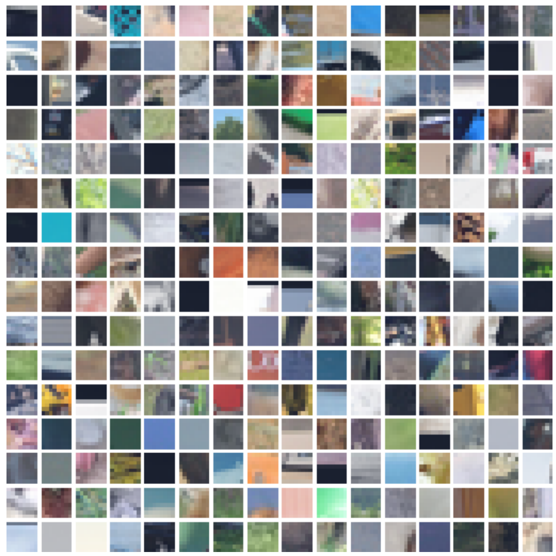
Проверкой на отдельной простой функцией дала точность $1.54 * 10^{-11}$, что говорит о правильности написанной функции compute gradient.

Проверка compute gradient на autoencoder loss дала точность $1.3 * 10^{-9}$, что в свою очередь говорит о правильности подсчитанных аналитически градиентов функции потерь автокодировщика.

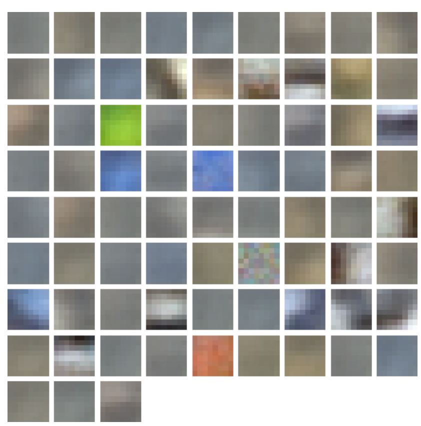
Минимизация происходила функцией scipy.optimize.minimize.

3.2 Сгенерированные патчи.

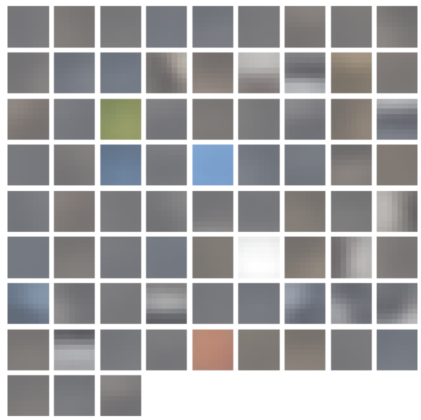
Сгенерированные патчи с unlabeled.pk



Выход после первого слоя разреженного автокодировщика ($\rho = 0.01$, $\lambda = 10^{-5}$, $\beta = 3$):



Выход после второго (разреженного) слоя ($\rho = 0.01$, $\lambda = 10^{-5}$, $\beta = 3$):

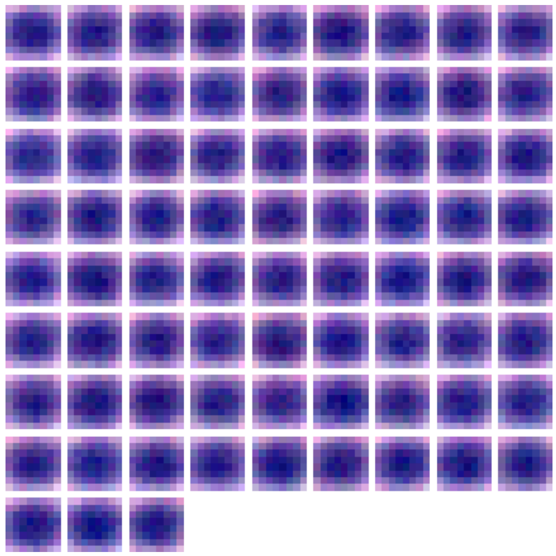


Таким образом становится ясным то, что наилучшее представление изображений – представление в виде ребер и цветовых переходов.

Что будет при изменении одного из гиперпараметров сети.

$\rho = 0.1$, остальные параметры не изменены

$\lambda = 0.001$, остальные параметры не изменены



Сеть с тремя скрытыми слоями, кол-во скрытых нейронов: 75, 48, 75

Как мы видим при изменении гиперпараметров сети ухудшается и визуализация весов на скрытых слоях.

3.3 Классификация

Обучаем нейронную сеть на unlabeled.pk и дальше применяем классификаторы RandomForest и LogisticRegression для test.pk по обученным train.pk.

Зависимость точности классификаторов от шага, по которому картинки разбиваются на патчи:

Сеть с одним скрытым слоем

Features (or step)	RandomForest	LogisticRegression
Pixels	0.363	0.292
step=32	0.305	0.335
step=28	0.335	0.375
step=24	0.329	0.369
step=20	0.341	0.404
step=16	0.34	0.417
step=12	0.347	0.435
step=8	0.358	0.465

Как мы видим, качество логистической регрессии гораздо улучшилось по сравнению с Random Forest. И еще, чем меньше шаг, тем лучше качество классификации, то есть теоретически step=6 качество классификаторов должно также улучшиться, но однако в данной работе не было проведено этого эксперимента в связи с огромной памятью, затрачиваемой для генерации патчей.

Необходимо было сравнить сеть с тремя скрытыми слоями с сетью с одним скрытым слоем.

Зависимость точности классификаторов от шага, по которому картинки разбиваются на патчи:

Сеть с тремя скрытыми слоями

Features (or step)	RandomForest	LogisticRegression
Pixels	0.363	0.292
step=32	0.292	0.184
step=28	0.32	0.195
step=24	0.302	0.191
step=20	0.33	0.216
step=16	0.332	0.225
step=12	0.339	0.239
step=8	0.346	0.25

-

Как мы видим сеть с тремя скрытыми слоями работает хуже, чем однослойная, возможно потому, что приходится подбирать больше параметров и параметры, которые были подобраны мной не совсем соответствуют истине. Необходимо было провести кросс-валидацию для поиска оптимальных параметров для сети с тремя скрытыми слоями, но это занимает достаточно много времени. Поэтому предпочтительнее сеть с одним скрытым слоем.

4 Бонус

Необходимо было проверить как количество патчей, сгенерированных с unlabeled.pk влияет на качество классификации. Была выбрана сеть с одним скрытым слоем и с step=12 для более эффективного использования оперативной памяти.

Зависимость точности классификаторов от количества патчей, сгенерированных с unlabeled.p

step = 12

Number of patches	RandomForest	LogisticRegression
Pixels	0.363	0.292
patches=100	0.336	0.41
patches=1000	0.343	0.438
patches=10000	0.347	0.435
patches=100000	0.345	0.439

То есть можно заметить, что при увеличении количества патчей, сгенерированных с unlabeled.pk, увеличивается качество классификации.

5 Выводы

Нейросетевой разреженный автокодировщик с одним скрытым слоем значительно улучшает качество классификации логистической регрессии и уменьшает размерность признакового пространства. Для случайного леса качество немного ухудшается.

Сеть с тремя скрытыми слоями ухудшает качество логистической регрессии и ненамного ухудшает качество случайных лесов.