Basic RL.6

Judy Tutorial

Softmax (Multiclass Logistic) Regression

&

Neural Networks

MNIST dataset

[LeCun 1998]

Multiple Labels
$y \in \{0,1,2, \dots, 9\}$

Hypothesis function

inputs

$$h_k(\boldsymbol{\theta}) = g(\boldsymbol{\theta}_k^{\mathrm{T}} X)$$

parameters

Activation function

$$g(z_k) = \frac{e^{z_k}}{\sum_{l=1}^{K} e^{z_l}}$$

softmax

Objective function

$$J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{K} y_l^{(i)} \log h_l(\theta)^{(i)}$$

Cross entropy loss

Hypothesis function

$$h(\boldsymbol{\theta}) = \begin{bmatrix} g(\boldsymbol{\theta}_{k=1}^{\mathrm{T}} X) \\ g(\boldsymbol{\theta}_{k=2}^{\mathrm{T}} X) \\ g(\boldsymbol{\theta}_{k=3}^{\mathrm{T}} X) \\ \vdots \\ g(\boldsymbol{\theta}_{k=K}^{\mathrm{T}} X) \end{bmatrix} = \begin{bmatrix} \frac{\exp \boldsymbol{\theta}_{k=1}^{\mathrm{T}} X}{\sum_{l=1}^{K} \exp \boldsymbol{\theta}_{l}^{\mathrm{T}} X} \\ \frac{\exp \boldsymbol{\theta}_{k=2}^{\mathrm{T}} X}{\sum_{l=1}^{K} \exp \boldsymbol{\theta}_{l}^{\mathrm{T}} X} \\ \frac{\exp \boldsymbol{\theta}_{k=3}^{\mathrm{T}} X}{\sum_{l=1}^{K} \exp \boldsymbol{\theta}_{l}^{\mathrm{T}} X} \\ \vdots \\ \frac{\exp \boldsymbol{\theta}_{k=K}^{\mathrm{T}} X}{\sum_{l=1}^{K} \exp \boldsymbol{\theta}_{l}^{\mathrm{T}} X} \end{bmatrix}$$

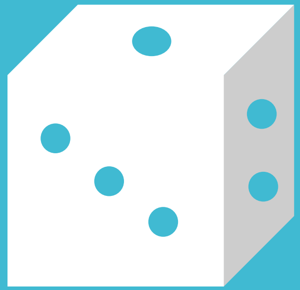
Objective function

$$\boxed{J(\boldsymbol{\theta})} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{K} y_l^{(i)} \log h_l(\boldsymbol{\theta})^{(i)}$$

Negative Log-Likelihood of

Multinomial distribution

Multinomial Distribution



Wiki:

- the Multinomial distribution gives the probability of any particular combination of numbers of successes for the various categories

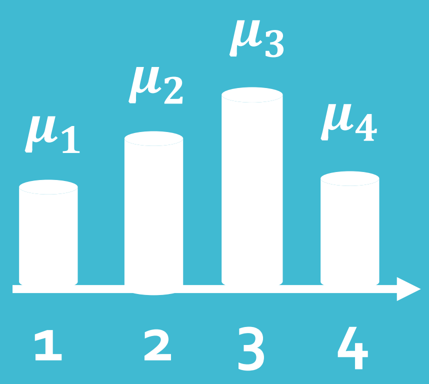
Example:

- Repeat rolling a dice for many times
- Suppose we roll a dice 5 times, what's the probability of the appearances of two 3s and three 4s?
- $p(34344|\boldsymbol{\mu}) = \frac{5!}{2!3!}\left(\frac{1}{6}\right)^2\left(\frac{1}{6}\right)^3$

Parameter vector for dice probability

where $\mu_k = \frac{1}{6}, K = 6$

Multinomial Distribution



$\mu_4 = 1 - \mu_1 - \mu_2 - \mu_3$

- x_k - the number of repeatition of a specific outcome when conducting some events
- μ_k - the probability of each event

#events

$$Mult(x_1, x_2, \ldots, x_K | \boldsymbol{\mu}, N) = C_N^{x_1 x_2 \ldots x_K} \prod_{k=1}^{K} \mu_k^{x_k}$$

distribution parameter vector

- $0 \leq \mu_k \leq 1, \sum_{k=1}^{K} \mu_k = 1$
- $\sum_{k=1}^{K} x_k = N$
- $C_N^{x_1 x_2 \ldots x_K} = \dfrac{N!}{x_1! x_2! \ldots x_K!}$

$$Mult(x_1, x_2, \dots, x_K | \boldsymbol{\mu}) \propto \prod_{k=1}^{K} \mu_k^{x_k}$$

Multinomial Likelihood

#data

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{N} \prod_{l=1}^{K} h_l(\boldsymbol{\theta})^{(i)\, y_l^{(i)}}$$

$$\boldsymbol{y}_l^{(i)} = \mathbf{1}_{y=k} = \begin{cases} \mathbf{1}, \textbf{if } \boldsymbol{y} = \boldsymbol{k} \\ \mathbf{0}, \textbf{if } \boldsymbol{y} \neq \boldsymbol{k} \end{cases}$$

One hot classification label

$$g\left(\boldsymbol{\theta}_k^T \boldsymbol{X}\right)$$

$$0 \leq h_k(\boldsymbol{\theta}) \leq 1$$

$$\sum_l^K h_l(\boldsymbol{\theta}) = 1$$

MLE for Softmax Regression

$$\max_{\theta} \log \mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \sum_{l=1}^{K} y_l^{(i)} \log h_l(\boldsymbol{\theta})^{(i)}$$

$$\Updownarrow$$

$$\min_{\theta} J(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{l=1}^{K} y_l^{(i)} \log h_l(\boldsymbol{\theta})^{(i)}$$

Cross entropy loss

Objective function

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{K} y_l^{(i)} \log h_l(\boldsymbol{\theta})^{(i)}$$

Batch Gradient

$$\nabla_{\theta_j} J(\theta_j) = \nabla_{\theta_j} -\frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{K} y_l^{(i)} \log h_l(\boldsymbol{\theta})^{(i)}$$

$$= -\frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{K} \frac{y_l^{(i)}}{h_l(\theta_j)^{(i)}} \nabla_{\theta_j} h(\theta_j)^{(i)}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{K} \left[h(\theta_j)^{(i)} - y_l^{(i)}\right] x^{(i)}$$

Element-wise

Matrix-wise

$$= \frac{1}{N}\boldsymbol{X}[\boldsymbol{h}(\boldsymbol{\theta}) - \boldsymbol{Y}]^T$$

same as linear/logistic regression related to Exponential Family and Generalized Linear Model

Gradient Descent

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Batch

Mini Batch

&

Stochastic Gradient Descent

Require a batch/entire dataset

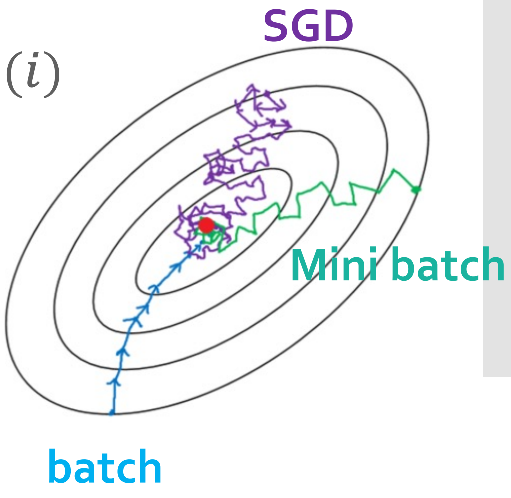
$$\nabla_{\theta_j} J(\theta_j) = \frac{1}{N}\sum_{i=1}^{N}\left[h(\theta_j)^{(i)} - y^{(i)}\right]x^{(i)}$$

Sample random mini batch from the entire dataset

$$\nabla_{\theta_j} J(\theta_j) = \frac{1}{N_{mini}}\sum_{i=1}^{N_{mini}}\left[h(\theta_j)^{(i)} - y^{(i)}\right]x^{(i)}$$

Require only one data point

$$\nabla_{\theta_j} J(\theta_j) = \left[h(\theta_j)^{(i)} - y^{(i)}\right]x^{(i)}$$



Feedforward Neural Networks

$$f(X) = f^{[M]} \ldots (f^{[3]}(f^{[2]}(f^{[1]}(X))))$$

First layer

Second layer

Mth layer

Third layer

$$h(\boldsymbol{\theta}) = g(\boldsymbol{\theta}^T X)$$

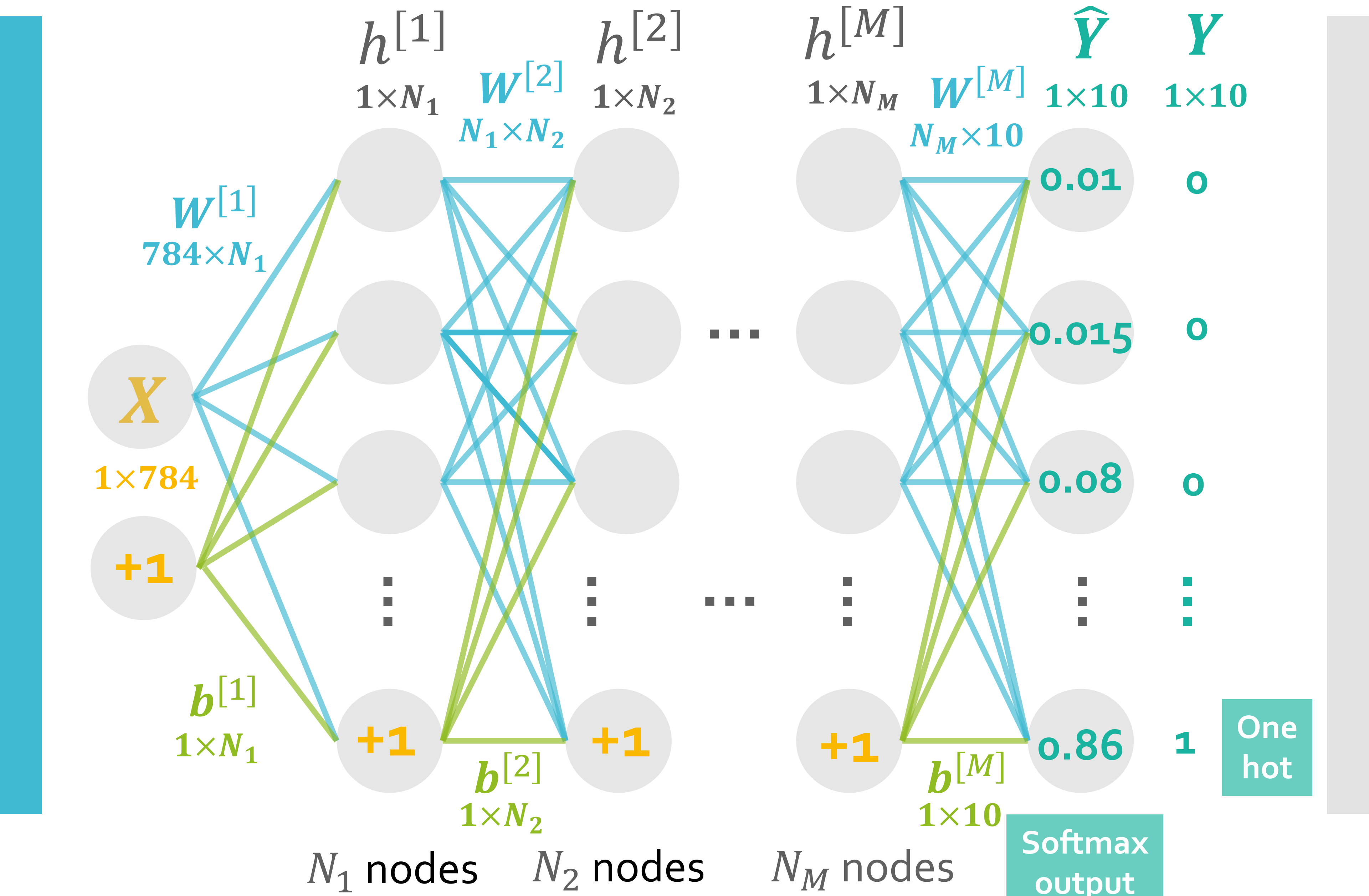
$$h(\boldsymbol{W}, \boldsymbol{b}) = g(\underbrace{X}_{\text{inputs}}\underbrace{\boldsymbol{W} + \boldsymbol{b}}_{\text{parameters}})$$

$$h^{[1]}\left(\boldsymbol{W}^{[1]}, \boldsymbol{b}^{[1]}\right) = g^{[1]}(X\boldsymbol{W}^{[1]} + \boldsymbol{b}^{[1]})$$

$$h^{[2]}\left(\boldsymbol{W}^{[2]}, \boldsymbol{b}^{[2]}\right) = g^{[2]}(h^{[1]}\boldsymbol{W}^{[2]} + \boldsymbol{b}^{[2]})$$

$$h^{[3]}\left(\boldsymbol{W}^{[3]}, \boldsymbol{b}^{[3]}\right) = g^{[3]}(h^{[2]}\boldsymbol{W}^{[3]} + \boldsymbol{b}^{[3]})$$

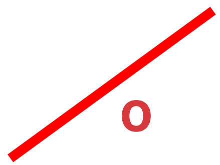
$$\vdots$$



$h^{[1]}$
$1 \times N_1$
$W^{[2]}$
$N_1 \times N_2$
$h^{[2]}$
$1 \times N_2$
$h^{[M]}$
$1 \times N_M$
$W^{[M]}$
$N_M \times 10$
$\hat{Y}$
1×10
Y
1×10
$W^{[1]}$
$784 \times N_1$
X
1×784
+1
0.01
0.015
0.08
0.86
0
0
0
1
$b^{[1]}$
$1 \times N_1$
+1
+1
+1
$b^{[2]}$
$1 \times N_2$
$b^{[M]}$
1×10
One hot
Softmax output
N_1 nodes
N_2 nodes
N_M nodes

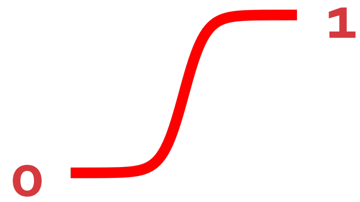
Activation functions

$$g(z) = z$$



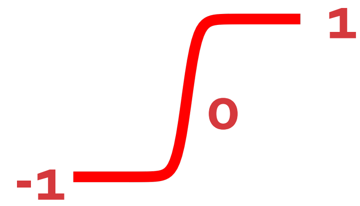
linear

$$g(z) = \frac{1}{1+e^{-z}}$$



Sigmoid

$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$



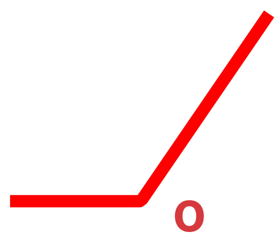
tanh

$$g(z_k) = \frac{e^{z_k}}{\sum_{l=1}^{K} e^{z_l}}$$

A generalized sigmoid

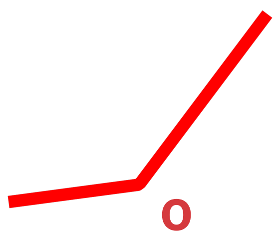
softmax

$$g(z) = \max(z, 0)$$



ReLU

$$g(z) = \max(z, 0.1z)$$



Leaky ReLU

Objective functions

$$\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2$$

MSE

$$-[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)]$$

Binary Cross entropy

$$\frac{1}{N}\sum_{i=1}^{N}\left|y_i - \hat{y}_i\right|$$

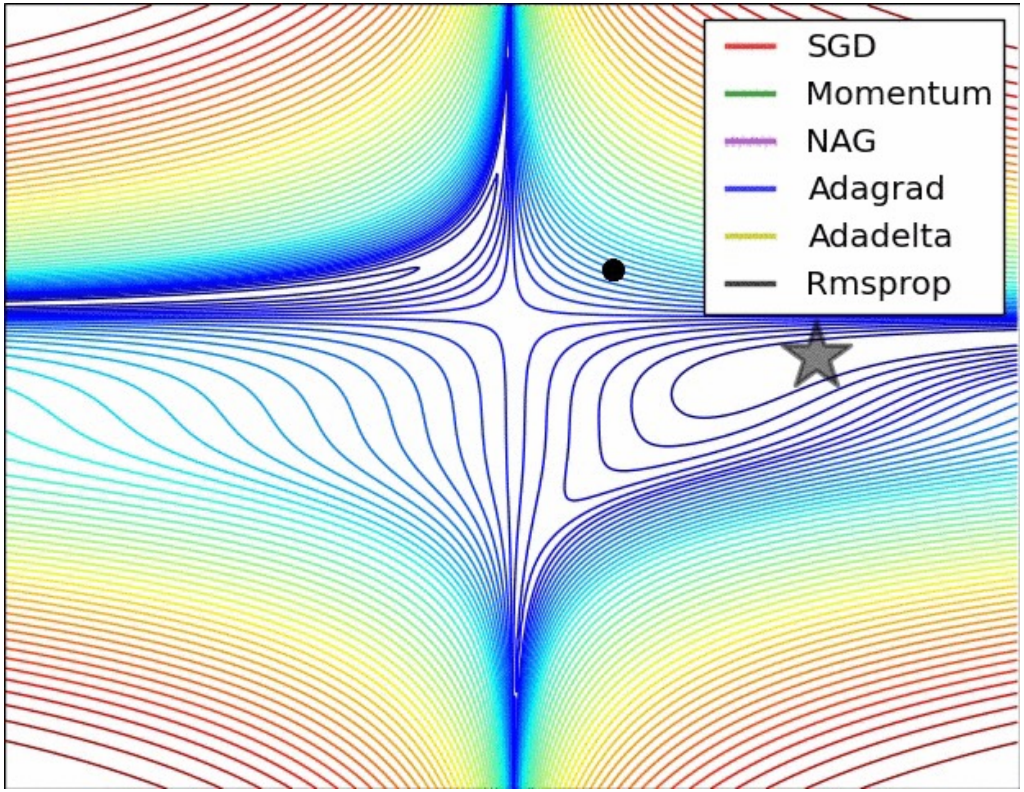
MAE

$$-\frac{1}{N}\sum_{i=1}^{N} y_i \log \hat{y}_i$$

Multiclass Cross entropy

Optimizers

- Batch Gradient
- MiniBatch Gradient
- SGD
- SGD+momentum
- Adam
- Adagrad
- Rmsprop
- ...



https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

Backpropagation – chain rule

$$h^{[1]} = g^{[1]}(XW^{[1]} + b^{[1]})$$
$$h^{[2]} = g^{[2]}(h^{[1]}W^{[2]} + b^{[2]})$$
$$\hat{Y} = g^{[3]}(h^{[2]}W^{[3]} + b^{[3]})$$
$$J = -\sum_{i=1}^{N} y_i \log \hat{y}_i \rightarrow -\log \hat{y}_i$$

ReLU

ReLU

softmax

Multiclass Cross entropy

$$\frac{\partial J}{\partial W^{[3]}} = \frac{\partial J}{\partial \hat{Y}} \frac{\partial \hat{Y}}{\partial g^{[3]}} \underbrace{\frac{\partial g^{[3]}}{\partial W^{[3]}}}_{h^{[2]}}$$

$$\frac{\partial J}{\partial W^{[2]}} = \frac{\partial J}{\partial \hat{Y}} \frac{\partial \hat{Y}}{\partial g^{[3]}} \frac{\partial g^{[3]}}{\partial h^{[2]}} \frac{\partial h^{[2]}}{\partial g^{[2]}} \underbrace{\frac{\partial g^{[2]}}{\partial W^{[2]}}}_{h^{[1]}}$$

$$\frac{\partial J}{\partial W^{[1]}} = \underbrace{\frac{\partial J}{\partial \hat{Y}} \frac{\partial \hat{Y}}{\partial g^{[3]}}}_{(\hat{Y} - Y)} \underbrace{\frac{\partial g^{[3]}}{\partial h^{[2]}}}_{W^{[3]}} \underbrace{\frac{\partial h^{[2]}}{\partial g^{[2]}}}_{0,1} \underbrace{\frac{\partial g^{[2]}}{\partial h^{[1]}}}_{W^{[2]}} \underbrace{\frac{\partial h^{[1]}}{\partial g^{[1]}}}_{0,1} \underbrace{\frac{\partial g^{[1]}}{\partial W^{[1]}}}_{X}$$

$$\frac{\partial J}{\partial b^{[3]}} = \frac{\partial J}{\partial \hat{Y}} \frac{\partial \hat{Y}}{\partial g^{[3]}} \underbrace{\frac{\partial g^{[3]}}{\partial b^{[3]}}}_{1}$$

$$\frac{\partial J}{\partial b^{[2]}} = \frac{\partial J}{\partial \hat{Y}} \frac{\partial \hat{Y}}{\partial g^{[3]}} \frac{\partial g^{[3]}}{\partial h^{[2]}} \frac{\partial h^{[2]}}{\partial g^{[2]}} \underbrace{\frac{\partial g^{[2]}}{\partial b^{[2]}}}_{1}$$

$$\frac{\partial J}{\partial b^{[1]}} = \underbrace{\frac{\partial J}{\partial \hat{Y}} \frac{\partial \hat{Y}}{\partial g^{[3]}}}_{(\hat{Y} - Y)} \underbrace{\frac{\partial g^{[3]}}{\partial h^{[2]}}}_{W^{[3]}} \underbrace{\frac{\partial h^{[2]}}{\partial g^{[2]}}}_{0,1} \underbrace{\frac{\partial g^{[2]}}{\partial h^{[1]}}}_{W^{[2]}} \underbrace{\frac{\partial h^{[1]}}{\partial g^{[1]}}}_{0,1} \underbrace{\frac{\partial g^{[1]}}{\partial b^{[1]}}}_{1}$$

Learning Flow

Given a training dataset $(\boldsymbol{X}, \boldsymbol{Y})$

1. Initialize $\boldsymbol{W}, \boldsymbol{b}$
2. Forward: calculate $\boldsymbol{h}^{[1]}, \boldsymbol{h}^{[2]}, \dots, \widehat{\boldsymbol{Y}}, \boldsymbol{J}$
3. Backward: calculate $\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{W}}, \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{b}}$
4. Optimize: e.g. SGD
5. Verify with test dataset

in python keras

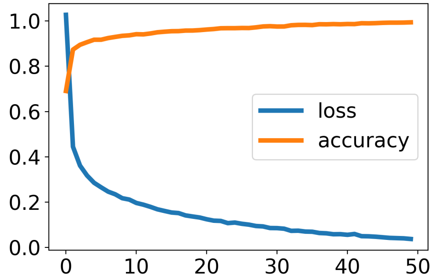
```
x_train,y_train,x_test,y_test=process_data()

model=Sequential()
model.add(Dense(10,input_dim=784,activation='relu'))
model.add(Dense(10,activation='softmax'))
model.compile(optimizer=SGD(lr=0.1),
              loss='categorical_crossentropy',
              metrics=['categorical_accuracy'])

traj=model.fit(x_train,
               y_train,
               epochs=50,
               batch_size=32,
               validation_data=(x_test, y_test),
               shuffle=True,verbose=0)

plt.plot(traj.history['loss'],linewidth=4,label='loss')
plt.plot(traj.history['categorical_accuracy'],linewidth=4,label='accuracy')
plt.legend()
```

Build model

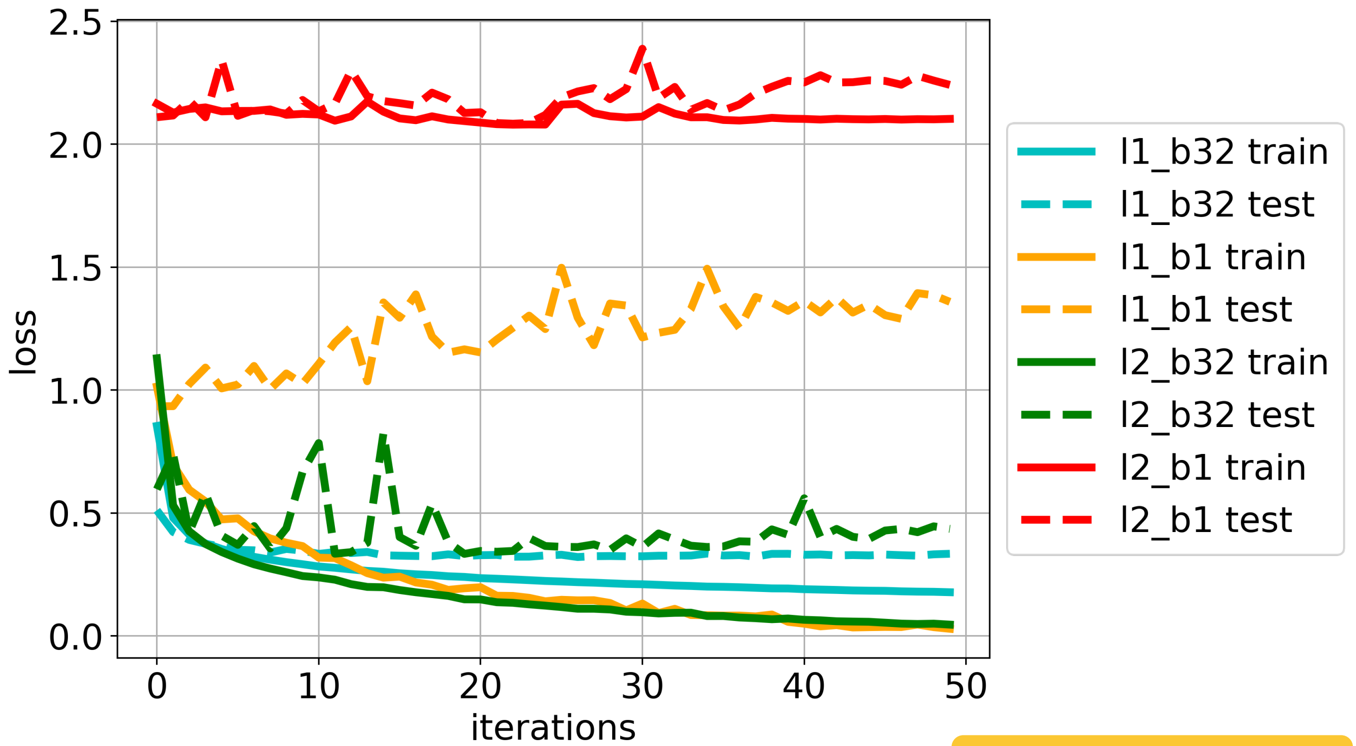
Training

Plot results

x_train: (5000,784)
y_train: (5000,10) - one hot
x_test: (1000,784)
y_test: (1000,10) – one hot



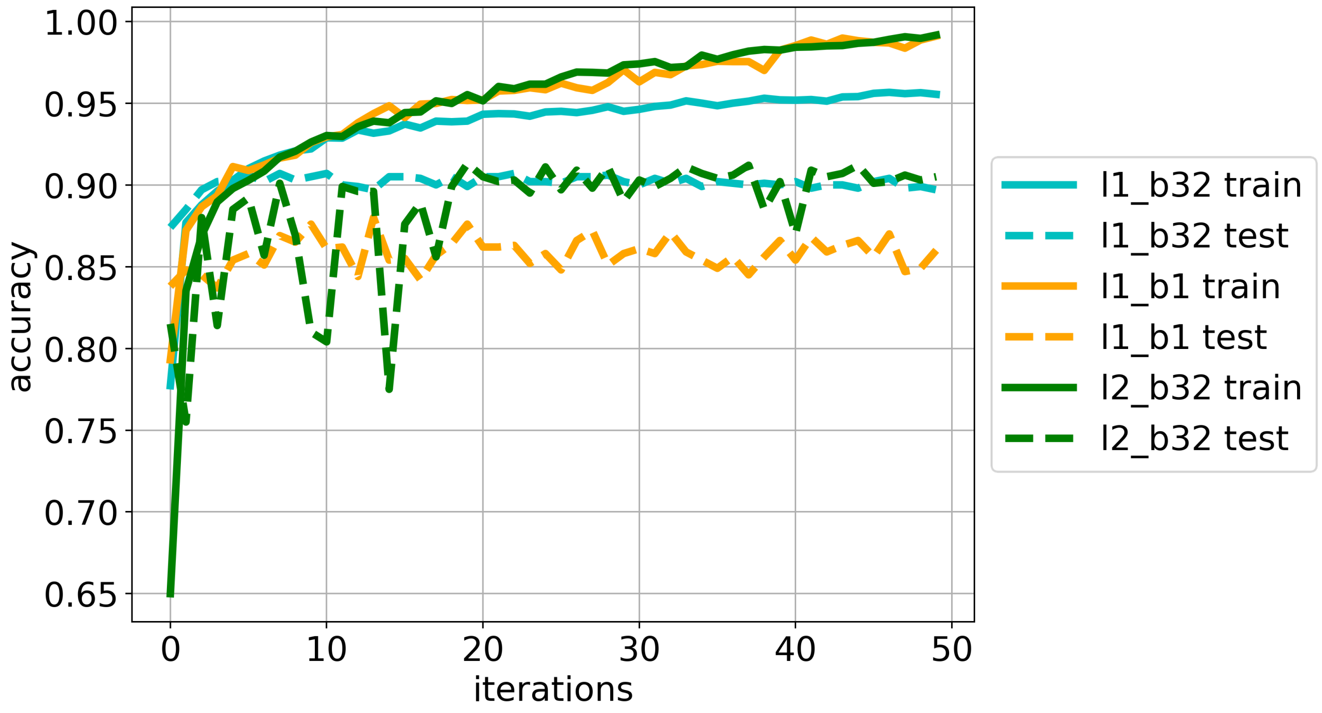
Learning behaviors
[layers,batchsize]



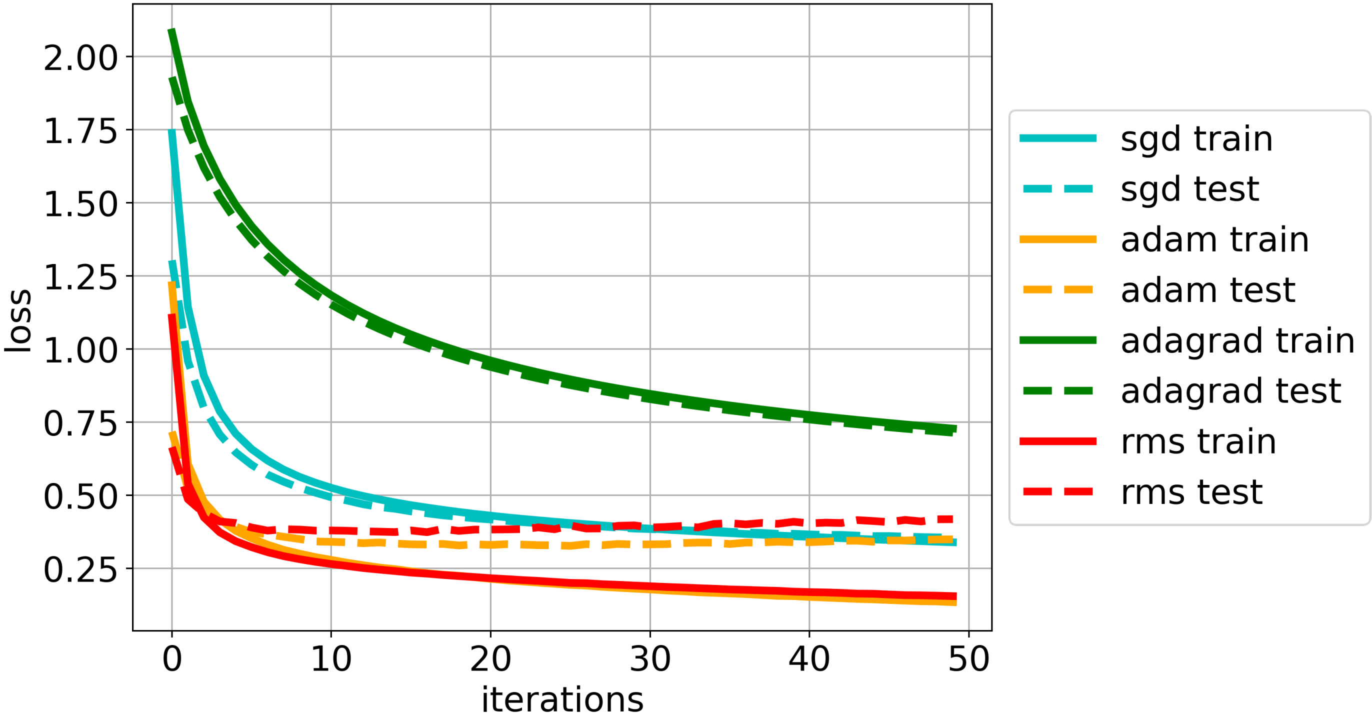
Param size 7850

- l1_b32: one layer, batch size 32 <=> Softmax regression
- l1_b1: one layer, batch size 1 <=> Softmax regression SGD
- L2_b32: two layers, batch size 32 Param size 7960
- L2_b1: two layers, batch size 1 SGD

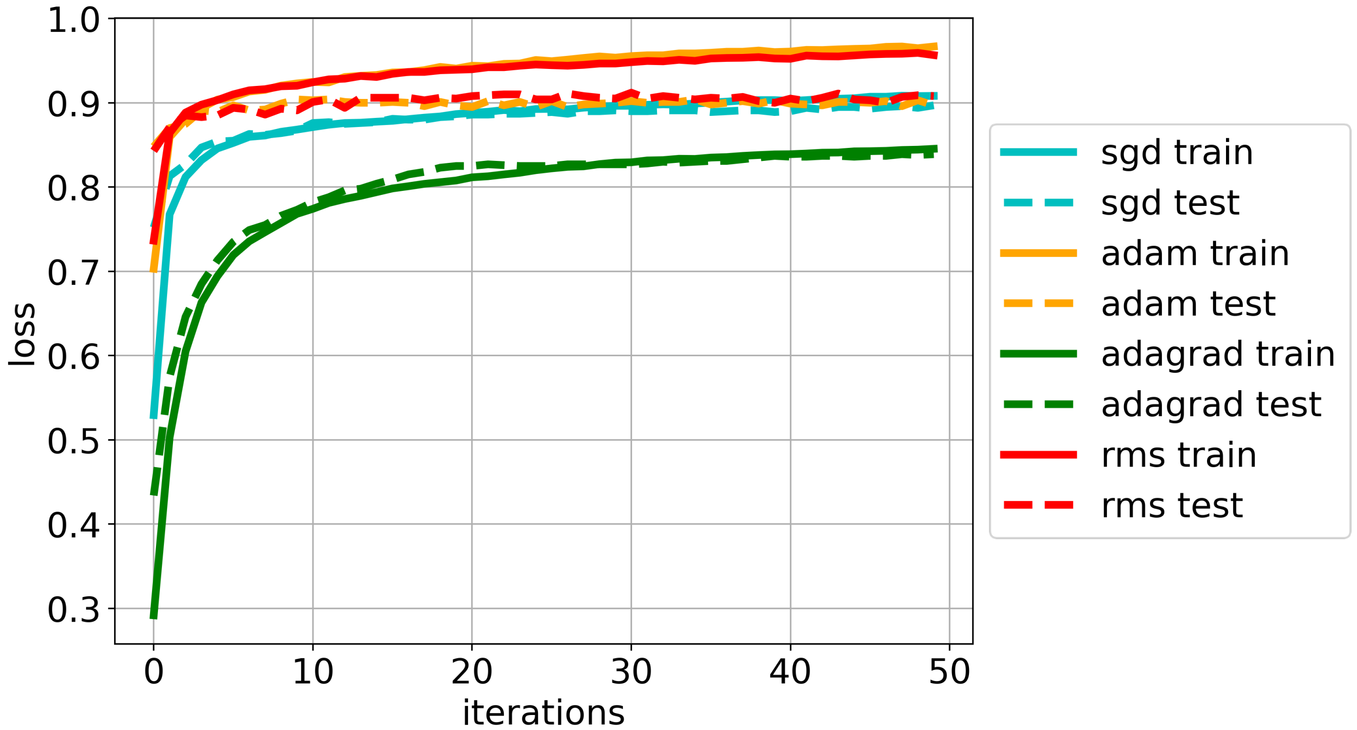
Learning behaviors
[layers,batchsize]



Behaviors of different optimizers
[layer : 1, batch: 32]



Behaviors of different optimizers
[layer : 1, batch: 32]



Weights visualization for softmax regression (one-layer mlp)