1.1) $B = \{ x \in \mathbb{R}^d \mid \|x\|_2^2 \leq 1 \}$

Let $\Pi_B(x) = \min_{z \in B} \|z - x\|_2$

if $\|x\|_2 \leq 1$ i.e. $x \in B$

then, ~~$\|z\| \leq \|z - x\|_2 \leq$~~

$\|z - x\|_2 \geq 0$

and the minimum is achieved

$\|z - x\| = 0$

when $\boxed{z = x}$ $\Rightarrow$ $\boxed{\Pi_B(x) = x}$

if $\|x\|_2 > 1$

Let $\Pi_B(x) = z \neq \frac{x}{\|x\|_2}$

using the property,

$\langle y' - \Pi_D(y'), \Pi_D(y') - y \rangle \geq 0$

where D is a convex set,

$y' \notin D$, $\forall y \in D$

Let ~~$y' = x$ s.t. $\|x\|_2 > 1$ and $D = B$~~
and ~~Π_B~~ $\Pi_D(x) = \Pi_B$

Let $D = B$, $y' = x \in B$

and $\Pi_D(y') = \Pi_B(x) = z \neq \frac{x}{\|x\|_2}$

$\Rightarrow$ If $\Pi_B(x) = z$ is correct projection then

$\langle x - z, z - y \rangle \geq 0$

or $\langle x - z, y - z \rangle \leq 0 \quad \forall y \in B$

LHS $\rightarrow$ $\langle x, y \rangle - \langle x - z \rangle - \langle z - y \rangle + \langle z, z \rangle$

Let $y = \frac{x}{\|x\|_2}$

$\Rightarrow \langle x, y \rangle - \langle x, z \rangle - \langle z, y \rangle + \|z\|_2^2$

$= \langle x, \frac{x}{\|x\|_2} \rangle - \langle x, z \rangle - \frac{1}{\|x\|_2} \langle z, x \rangle + \|z\|_2^2$

$= \|x\|_2 + \langle x, z \rangle \left(\frac{1}{\|x\|_2} - 1 \right) + \|z\|_2^2$

$= \|x\|_2 + \|z\|_2^2 + \langle x, z \rangle \left(\frac{1}{\|x\|_2} - 1 \right) - (1)$

Now given $\|x\|_2 \geq 1$

$$\langle x, z \rangle \left(\frac{1}{\|x\|_2} - 1 \right) \geq \|x\|_2 \|z\|_2 \left(\frac{1}{\|x\|_2} - 1 \right)$$

~~$\leq \|x\|_2$~~

Substituting in (1)

$$\Rightarrow LHS \geq \|x\|_2 + \|z\|_2^2 + \|x\|_2 \|z\|_2 \left(\frac{1}{\|x\|_2} - 1 \right)$$

$$\geq \|x\|_2 + \|z\|_2^2 + \|z\|_2 - \|x\|_2 \|z\|_2$$

given $z \in B \Rightarrow \|z\|_2 \leq 1$

$$\Rightarrow LHS \geq \|x\|_2 + \|z\|_2^2 + \|z\|_2 - \|x\|_2$$

$$LHS \geq \|z\|_2^2 + \|z\|_2 \geq 0$$

$\Rightarrow$ Our assumption that $y, z \neq \frac{x}{\|x\|_2}$ is not true.

~~for at least $y = \frac{x}{\|x\|_2}$~~

as it violates, ~~$\langle u, z \rangle$~~

$$\langle x - z, y - z \rangle \geq 0$$

or $\langle x - z, z - y \rangle \leq 0$ for at least $y = \frac{x}{\|x\|_2}$

But does $z = \frac{x}{\|x\|_2}$ satisfies this. $\forall y \in B$

$$\langle x - z, z - y \rangle = \langle x - \frac{x}{\|x\|_2^2}, \frac{x}{\|x\|_2} - y \rangle$$

$$= \langle x, \frac{x}{\|x\|_2} \rangle - \langle x, y \rangle - \langle \frac{x}{\|x\|_2}, \frac{x}{\|x\|_2} \rangle + \langle \frac{x}{\|x\|_2}, y \rangle$$

$$= \|x\|_2 + \langle x, y \rangle \left(\frac{1}{\|x\|_2} - 1 \right) - 1$$

$$= (\|x\|_2 - 1) + \langle x, y \rangle \left(\frac{1}{\|x\|_2} - 1 \right)$$

~~given $\|x\|_2 > 1$~~

and $\langle x, y \rangle \left(\frac{1}{\|x\|_2} - 1 \right) \geq \|x\|_2 \|y\|_2 \left(\frac{1}{\|x\|_2} - 1 \right)$

we can get

~~$\langle x - z, z - y \rangle \geq \|y\|_2 (1 - \|x\|_2)$~~

$$\langle x - z, z - y \rangle \geq (\|x\|_2 - 1) + \|y\|_2 (1 - \|x\|_2)$$

$$\geq (\|x\|_2 - 1)(1 - \|y\|_2)$$

given $\|x\|_2 > 1$ and $\|y\|_2 \leq 1$ as $y \in B$

$\Rightarrow \quad \langle x - z, z - y \rangle \geq (\|x\|_2 - 1)(1 - \|y\|_2) \geq 0$

$\Rightarrow \quad \Pi_B(x) = z = \frac{x}{\|x\|_2}$ is the only projection which satisfies the given condition.

$$\Rightarrow \quad \Pi_B(x) = \begin{cases} x, & \text{if } \|x\|_2^2 \leq 1 \\ \frac{x}{\|x\|_2}, & \text{if } \|x\|_2^2 > 1 \end{cases}$$

1.2) $S = \{ x \in \mathbb{R}^d \mid U^T x = b \}$

s.t. $U^T U = I$

$b \in \mathbb{R}^k$

$$\Pi_S(y) = \underset{x \in S}{\operatorname{argmin}} \|x - y\|_2 = \underset{x \in S}{\operatorname{argmin}} \frac{1}{2} \|x - y\|_2^2$$

Lagrangian for the constraint minimization,

$$f(x) = \frac{1}{2} \|x - y\|_2^2$$
$$U^T x - b = 0$$

$$L(x, \alpha) = f(x) + \alpha^T (U^T x - b) \quad \{ \alpha \in \mathbb{R}^{k \times 1} \}$$

Using KKT conditions:-

- Stationarity $\nabla_x L(x, \alpha) = (x - y) + U\alpha = 0$

$$\Rightarrow x = y - U\alpha \quad - (1)$$

- Primal feasibility

$$U^T x - b = 0$$
$$\Rightarrow U^T (y - U\alpha) - b = 0$$
$$\Rightarrow U^T y - U^T U \alpha - b = 0$$

using $U^T U = I$

$\alpha = U^T y - b \quad - (2)$

Put (2) in (1)

$$x = y - U(-b + U^T y)$$

$$x = y + Ub - UU^T y$$

~~using $U^T U = I$~~

$$\boxed{x = Ub}$$

$$\Rightarrow \boxed{\Pi_S(y) = Ub}$$

$$\Rightarrow \quad x = (I - UU^T)y + Ub$$

$$\Rightarrow \boxed{\Pi_S(y) = (I - UU^T)y + Ub}$$

$$\Delta := \{ x \in \mathbb{R}^d \mid \forall i \in [d], x_i \geq 0, \sum_{i \in [d]} x_i = 1 \}$$

$$\Pi_\Delta(y) = \underset{x \in \Delta}{\operatorname{argmin}} \|x - y\|_2^2 = \underset{x \in \Delta}{\operatorname{argmin}} \frac{1}{2} \|x - y\|_2^2$$

$$f(x) = \frac{1}{2} \|x - y\|_2^2$$

$$\forall i \in [d] \quad -x_i \leq 0$$

$$\sum_{i=1}^{d} x_i = 1$$

Assume the dimensions of y are arranged in sorted increasing order.

Lagrangian

$$L(x, u, v) = \frac{1}{2} \|x - y\|_2^2 - \sum_{i=1}^{d} u_i x_i + v \left(\sum_{i=1}^{d} x_i - 1 \right)$$

Using KKT conditions

~~$\forall i \in [d]$~~

stationarity:

$$\forall i \in [d] \quad \nabla_{x_i} L(x, u, v) = (x_i - y_i) - u_i + v = 0 \quad - (1)$$

complimentry slackness

$$\forall i \in [d] \quad -u_i x_i = 0 \quad - (2)$$

Dual fecasibility $\forall i \in [d] \quad u_i \geq 0 \quad - (3)$

Primal feasibility :-

$$\sum_{i=1}^{d} x_i = 1 \quad -(4); \qquad x_i \geq 0 \;\; \forall i \in [d] \quad -(5)$$

from (1)

$$x_i = y_i + u_i - v$$

using (5) we get 2 cases,

case 1 $x_i > 0 \Rightarrow$ from (2) $\rightarrow u_i = 0$

(Let the be ρ such i) $\Rightarrow \quad x_i = y_i - v$

case 2 $x_i = 0 \quad \Rightarrow \quad u_i > 0$

Now using (4)

$$\sum_{i=1}^{d} x_i = \sum_{i=d-\rho+1}^{d} y_i - \rho v = 1$$

$$\Rightarrow v = \frac{1}{\rho}\left(\sum_{i=d-\rho+1}^{d} y_i - 1\right)$$

$$\Rightarrow x_i = y_i - \frac{1}{\rho}\left(\sum_{i=d-\rho}^{d} y_i - 1\right) \quad \forall i \in \{d-\rho, \ldots d\}$$

$\Rightarrow$ ~~$\forall i \in \{1, d-\rho-1$~~

$$\Rightarrow \begin{cases} \forall i \in \{1, \dots d-\rho\} & x_i = 0 \\ \forall i \in \{d-\rho+1, \dots, d\} & x_i = y_i - \frac{1}{\rho}\left(\sum_{i=d-\rho}^{d} y_i - 1\right) \end{cases}$$

Bonus:- ~~For every~~

The function $\frac{\sum_{i=d-\rho}^{d} y_i - 1}{\rho}$ is monotonically increasing ~~for~~ wrt. ρ.

- Sort x_i with merge sort - $O(d \log d)$
- For every $\rho \in \{1, 2 \dots, d\}$ - $O(d)$

~~Calculate~~ $\sum_{i=d-\rho}^{d} y_i = \sum_{i=d-(\rho-1)}^{d} y_i + y_{d-\rho}$

$$S_\rho = \sum_{i=d-\rho}^{d} y_i = S_{\rho-1} + y_{d-\rho}$$

- Do binary search on $\rho \leftarrow O(\log d)$

#(i) Calculate x_i - $O(d)$ time as $\sum_{i=d-\rho}^{d} y_i$ is known

Check $x \in \Delta$, $\sum_{i=1}^{d} x_i = 1$ and $x_i \geq 0 \ \forall i$ - $O(d)$

if $x \in \Delta$; return x.

$\Rightarrow$ Total $O(d \log d)$ time complexity.

2.1) $\mathbb{E}[\tilde{\nabla} f(x_t)] = \nabla f(x)$ and $\mathbb{E}[\|\tilde{\nabla} f(x)\|_2^2] \le G$

update rule :- $x_{t+1} = x_t - \eta_t \tilde{\nabla} f(x_t)$

Starting from the basic mirror descent lemma for SGD.

$$f(x_t) \le f(x) + \mathbb{E}\left[\frac{1}{2\eta}\left(\|x_t - x\|_2^2 - \|x_{t+1} - x\|_2^2 + \|x_t - x_{t+1}\|_2^2\right)\right]$$

replace η with η_t

$$f(x_t) \le f(x) + \mathbb{E}\left[\frac{1}{2\eta_t}\left(\|x_t - x\|_2^2 - \|x_{t+1} - x\|_2^2 + \|x_t - x_{t+1}\|_2^2\right)\right]$$

$$\Rightarrow \eta_t f(x_t) \le \eta_t f(x) + \mathbb{E}\left[\frac{1}{2}\left(\|x_t - x\|_2^2 - \|x_{t+1} - x\|_2^2 + \|x_t - x_{t+1}\|_2^2\right)\right]$$

Taking telescopic sum

$$\sum_{t=0}^{T-1} \eta_t \mathbb{E}[f(x_t)] \le f(x) \sum_{t=0}^{T-1} \eta_t + \mathbb{E}\left[\frac{1}{2}\left(\|x_0 - x^*\|_2^2 + \sum_{t=0}^{T-1} \|x_t - x_{t+1}\|_2^2\right)\right]$$

$\sum_{t=0}^{T-1}$

$$\frac{1}{\sum_{s=0}^{T-1}\eta_s}\sum_{t=0}^{T-1}\mathbb{E}[f(x_t)] \le f(x) + \frac{\|x_0 - x\|_2^2}{2\eta} + \mathbb{E}$$

$$\left(\frac{1}{\sum_{s=0}^{T-1}\eta_s}\right)\left(\sum_{t=0}^{T-1}\eta_t \mathbb{E}[f(x_t)]\right) \le f(x) + \frac{\|x_0 - x\|_2^2 + \mathbb{E}\left[\sum_{t=0}^{T-1}\|x_t - x_{t+1}\|_2^2\right]}{2\sum_{t=0}^{T-1}\eta_t}$$

using the update rule

$$x_t - x_{t+1} = -\eta_t \tilde{\nabla} f(x_t)$$

$$\Rightarrow \left(\frac{1}{\sum_{s=0}^{T-1}\eta_s}\right)\left(\sum_{t=0}^{T-1}\eta_t \mathbb{E}[f(x_t)]\right) \le f(x_s) + \frac{\|x_0 - x\|_2^2 + \sum_{t=0}^{T-1}\mathbb{E}\left[\|\tilde{\nabla} f(x_t)\|_2^2\right]\eta_t^2}{2\sum_{t=0}^{T-1}\eta_t}$$

given $\mathbb{E}\left[\|\tilde{\nabla} f(x_t)\|_2^2\right] \le G$

$$\Rightarrow \left(\frac{1}{\sum_{s=0}^{T-1}\eta_s}\right)\left(\sum_{t=0}^{T-1}\eta_t \mathbb{E}[f(x_t)]\right) \le f(x) + \frac{\|x_0 - x\|_2^2 + G\sum_{t=0}^{T-1}\eta_t^2}{2\sum_{t=0}^{T-1}\eta_t}$$

2.2) Starting from the second last step of Q.(2.1)

and setting $\eta_t = \eta \Rightarrow \sum_{t=0}^{T-1} \eta_t = \eta_s T$

$$\frac{1}{\eta T}\eta\left(\sum_{t=0}^{T-1}\mathbb{E}[f(x_t)]\right) \le f(x) + \frac{\|x_0 - x\|_2^2 + \eta^2 \mathbb{E}\left[\sum_{t=0}^{T-1}\|\bar{\nabla} f(x_t)\|_2^2\right]}{2\eta T}$$

$$\frac{1}{T}\left(\mathbb{E}\left[\sum_{t=0}^{T-1} f(x_t)\right]\right) \le f(x) + \frac{\left(\|x_0 - x\|_2^2 + \cancel{2\eta \|\nabla f(x} + 2\eta^2 \sum_{t=0}^{T-1}\|\nabla f(x_t)\|_2^2\right)}{2\eta T}$$

Now if being a convex function,

$$\mathbb{E}\left[\frac{\sum_{t=0}^{T-1} f(x_t)}{T}\right] \ge f(\bar{x}_T)$$

where $\bar{x}_T = \sum_{t=0}^{T-1} \frac{x_t}{T}$

$$\Rightarrow \mathbb{E}\left[f(\bar{x}_T)\right] \le f(x) + \frac{\|x_0 - x\|_2^2}{2\eta T} + \frac{2\eta}{T}\sum_{t=0}^{T-1}\|\nabla f(x_t)\|_2^2 \quad -(1)$$

using ~~taiwan~~ quadratic upper bound

$$f(x_{t+1}) \leq f(x_t) + \langle \nabla f(x_t), (x_{t+1} - x_t) \rangle + \frac{L}{2} \|x_{t+1} - x_t\|_2^2$$

using the update rule

$$x_{t+1} = x_t - \eta \tilde{\nabla} f(x_t)$$

$$f(x_{t+1}) \leq f(x_t) - \eta \langle \nabla f(x_t), \tilde{\nabla} f(x_t) \rangle + \frac{L}{2} \eta^2 \|\tilde{\nabla} f(x_t)\|_2^2$$

Taking expectation

$$\mathbb{E}[f(x_{t+1})] \leq \mathbb{E}[f(x_t)] - \eta (\nabla f(x_t))^T \mathbb{E}[\tilde{\nabla} f(x_t)] + \frac{L}{2} \eta^2 \mathbb{E}\left[\|\tilde{\nabla} f(x_t)\|_2^2\right]$$

given,

$$\mathbb{E}[\tilde{\nabla} f(x_t)] = \nabla f(x_t)$$

and $$\mathbb{E}[\|\tilde{\nabla} f(x_t)\|_2^2] \leq 2 \|\nabla f(x_t)\|_2^2$$

$$\Rightarrow \mathbb{E}[f(x_{t+1})] \leq \mathbb{E}[f(x_t)] - \eta \|\nabla f(x_t)\|_2^2 + L\eta^2 \|\nabla f(x_t)\|_2^2$$

for $\eta \leq \frac{1}{2L} \Rightarrow L \leq \frac{1}{2\eta}$

$$\mathbb{E}[f(x_{t+1})] \leq \mathbb{E}[f(x_t)] - \frac{\eta}{2} \|\nabla f(x_t)\|_2^2 \quad -(2)$$

Taking telescopic sum,

$$\sum_{t=0}^{T-1} \mathbb{E}[f(x_{t+1})] \leq \sum_{t=0}^{T-1} \mathbb{E}[f(x_t)] - \frac{\eta}{2} \sum_{t=0}^{T-1} \|\nabla f(x_t)\|_2^2$$

$$\mathbb{E}[f(x_T)] - \mathbb{E}[f(x_0)] \leq -\frac{\eta}{2} \sum_{t=0}^{T-1} \|\nabla f(x_t)\|_2^2$$

$$\Rightarrow \sum_{t=0}^{T-1} \|\nabla f(x_t)\|_2^2 \leq \left(\frac{2}{\eta}\right)\left(f(x_0) - \mathbb{E}[f(x_T)]\right)$$

now, $f(x^*) \leq f(x_T)$ { x^* is the minimizer }

$$\Rightarrow \mathbb{E}[f(x^*)] = f(x^*) \leq \mathbb{E}[f(x_T)]$$

$$\Rightarrow \sum_{t=0}^{T-1} \|\nabla f(x_t)\|_2^2 \leq \frac{2}{\eta}\left(f(x_0) - f(x^*)\right) \quad -(3)$$

~~Now using L-Lipschitz smooth~~

Now using upper quadratic bound on x_0 and x^*

$$f(x_0) \leq f(x^*) + \langle \nabla f(x^*), x_0 - x^* \rangle + \frac{L}{2}\|x_0 - x^*\|_2^2$$

given x^* is the minimizer

$$\Rightarrow \nabla f(x^*) = 0$$

$$\Rightarrow f(x_0) - f(x^*) \leq \frac{L}{2}\|x_0 - x^*\|_2^2$$

$$\Rightarrow \sum_{t=0}^{T-1} \|\nabla f(x_t)\|_2^2 \leq \frac{L\eta}{2}\|x_0 - x^*\|_2^2 \quad -(4)$$

using (4) in (1)

~~$\Rightarrow \frac{1}{T}\mathbb{E}[f(x_T)]$~~

$$\mathbb{E}[f(\bar{x}_T)] \leq f(x) + \frac{\|x_0 - x\|_2^2}{2\eta T} + \frac{2\eta}{T}\frac{L}{\eta}\|x_0 - x^*\|_2^2$$

given $\eta \leq 1/2L$

$$\leq f(x) + \frac{\|x_0 - x\|_2^2}{2\eta T} + \frac{\|x_0 - x^*\|_2^2}{\eta T}$$

if ~~x_0~~ is $x \in \mathbb{R}^d$ is farther away from x_0 than x^*

$$\Rightarrow \|x_0 - x\|_2^2 \geq \|x_0 - x^*\|_2^2$$

$$\Rightarrow \boxed{\mathbb{E}[f(\bar{x}_T)] \leq f(x) + \frac{3}{2}\frac{\|x_0 - x\|_2^2}{\eta T}}$$

~~$\Rightarrow$ The convergence~~

$$\Rightarrow \boxed{\mathbb{E}[f(\bar{x}_T)] \leq f(x) + O\left(\frac{\|x_0 - x\|_2^2}{\eta T}\right)}$$

2.3)

$$f(w) = \frac{1}{N} \sum_{i=1}^{N} \log\left(1 + \exp(-y_i \langle w, x_i \rangle)\right)$$

$$\nabla_w f(w) = \frac{1}{N} \sum_{i=1}^{N} \frac{\exp(-y_i \langle w, x_i \rangle)\, \nabla_w (-y_i \langle w, x_i \rangle)}{1 + \exp(-y_i \langle w, x_i \rangle)}$$

$$\boxed{\nabla_w f(w) = \frac{1}{N} \sum_{i=1}^{N} \frac{\exp(-y_i \langle w, x_i \rangle)\, (-y_i x_i)}{1 + \exp(-y_i \langle w, x_i \rangle)}}$$

Observations :-

- Lower batch size gives a less smooth graph as the gradients have high variance for lower batch size
- For a given batch size as we increase the learning rate the loss drops more quickly compared to lower ones.
- As the batch size increases, the member of operations required to calculate gradients will increase and thus there is an increase in training time for higher batch sizes.

sgd_mnist

March 6, 2020

[31]:
```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import os
import argparse
import pdb
import time
```

[32]:
```
def parse_args():
    parser = argparse.ArgumentParser()

    parser.add_argument('--lr', dest='lr', type=float, default=1e-3,␣
 ↪help="learning rate")
    parser.add_argument('--batch_size', dest='momentum', type=float, default=0.
 ↪9, help="batch_size")

    return parser.parse_args()
```

[33]:
```
def load_dataset(data_path="./"):
    image_size = 28 # width and length of mnist image
    num_labels = 10 #  i.e. 0, 1, 2, 3, ..., 9
    image_pixels = image_size * image_size
    train_data = np.loadtxt(os.path.join(data_path,"mnist_train.csv"),␣
 ↪delimiter=",")
    test_data = np.loadtxt(os.path.join(data_path,"mnist_train.csv"),␣
 ↪delimiter=",")
    return {"train_data":train_data,
            "test_data": test_data,
           }
```

[12]:
```
def process_data(raw_data, labels_req=[0,1]):
    train_data = raw_data["train_data"]
    test_data = raw_data["test_data"]
```

```
    # rescale image from 0-255 to 0-1
    fac = 1.0 / 255
    train_imgs = np.asfarray(train_data[:, 1:])
    test_imgs = np.asfarray(test_data[:, 1:])
    train_labels = np.asfarray(train_data[:, :1])
    test_labels = np.asfarray(test_data[:, :1])

    train_imgs = np.divide(train_imgs, np.linalg.norm(train_imgs, axis=1,␣
↪keepdims=True))
    test_imgs = np.divide(test_imgs, np.linalg.norm(test_imgs, axis=1,␣
↪keepdims=True))

    train_mask = np.isin(train_labels[:,0],labels_req)
    test_mask = np.isin(test_labels[:,0],labels_req)

    dataset = { "X_train": train_imgs[train_mask],
                "Y_train": train_labels[train_mask]*2.0 - 1.0,
                "X_test": test_imgs[test_mask],
                "Y_test": test_labels[test_mask]*2.0 - 1.0,
            }

    return dataset
```

[34]:
```
raw_data = load_dataset(data_path="./")
```

[35]:
```
dataset = process_data(raw_data.copy(),labels_req=[0,1])
```

[36]:
```
def plot_props(data_arr,prop_names, figname, xlabel, x_data=None):
    fig = plt.figure(figsize=(16,9))

    for i in range(len(data_arr)):
        if(x_data):
#             print(x_data.shape, data_arr.shape)
            plt.plot(x_data[i], data_arr[i], label=prop_names[i])
        else:
            plt.plot(data_arr[i], label=prop_names[i])
        plt.ylabel("train_losses")
        plt.xlabel(xlabel)
        plt.legend()
    plt.title(figname)
    plt.savefig("./{}.pdf".format(figname))
#     plt.show()
```

[37]:
```
def get_loss_grad(W, X, y_true, require_grad=True):
    '''
    W: weight vector (n,)
```

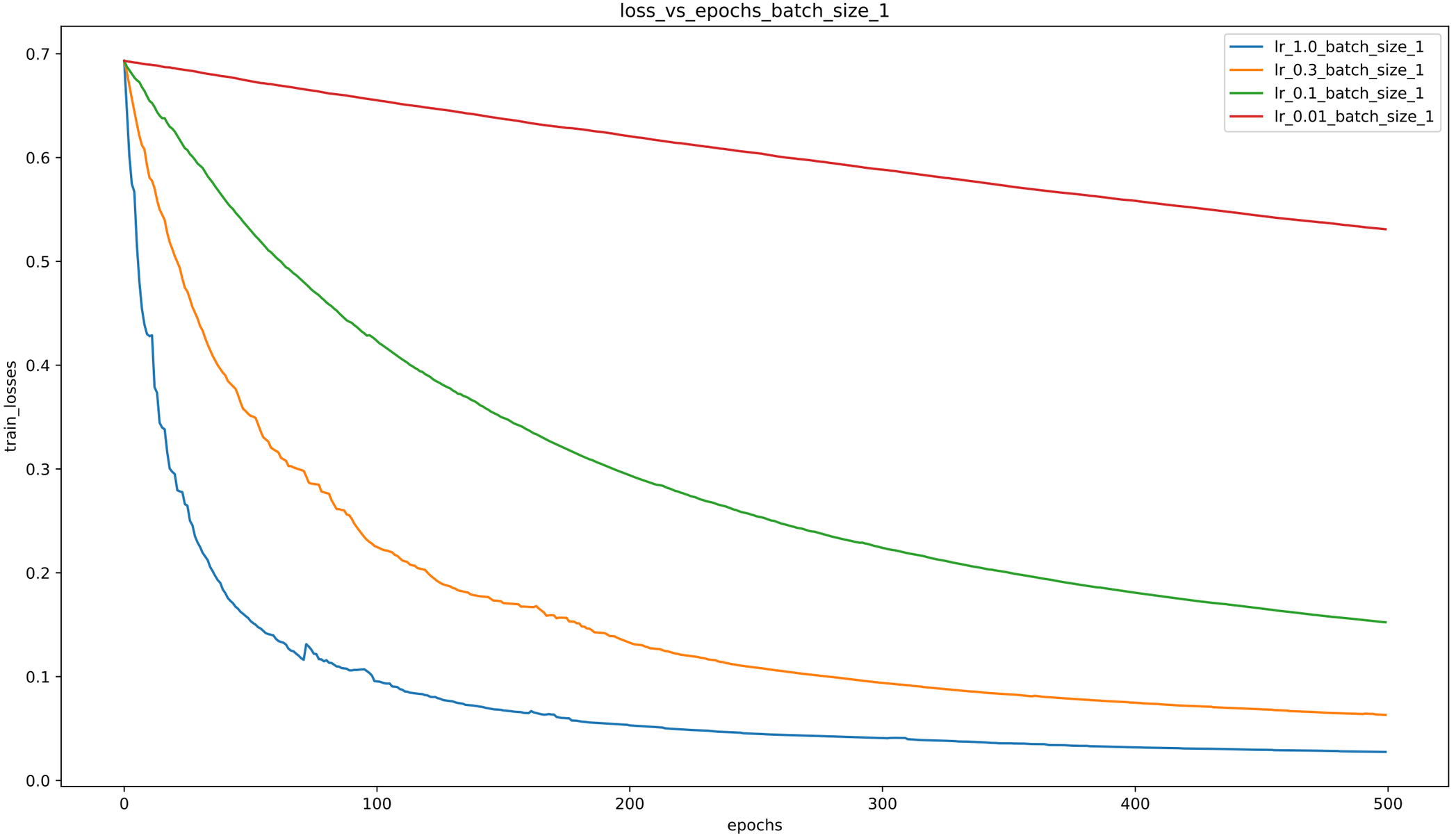
```
    X: input batch (batch_size, n)
    '''
    dot_prod = np.matmul(X,W)
    expo = np.exp(-np.multiply(y_true, dot_prod))
    loss = np.mean(np.log(1 + expo))
    if require_grad:
        grad = np.divide(expo , (1+expo))
        grad = np.multiply(grad, -1.0*np.multiply(y_true,X))
        grad = np.mean(grad, 0)
        return loss, grad
    return loss
```

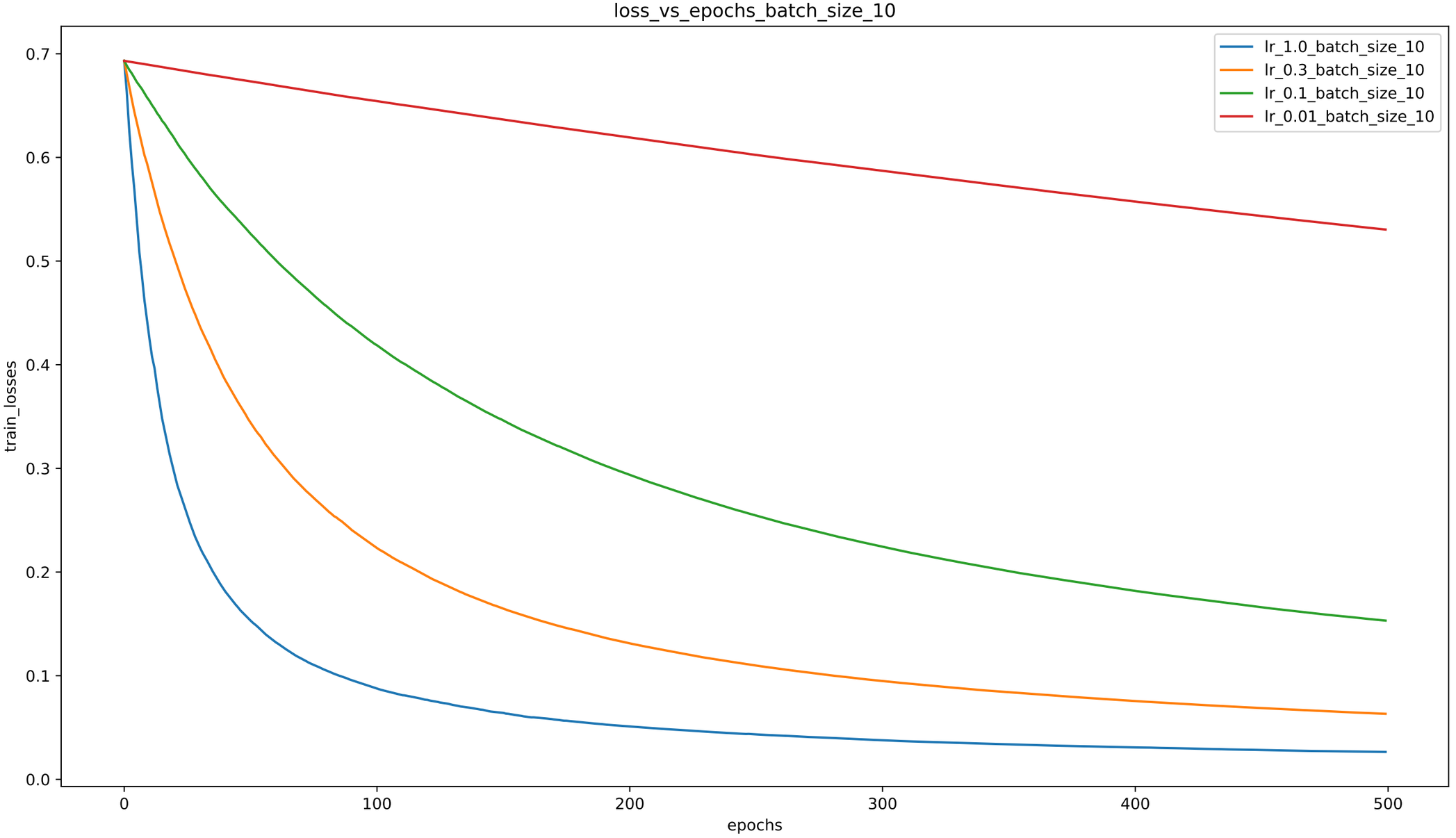
[38]:
```
def test_train_data(W, train_data, train_labels):
    train_loss = get_loss_grad(W, train_data, train_labels, require_grad=False)
    return train_loss
```

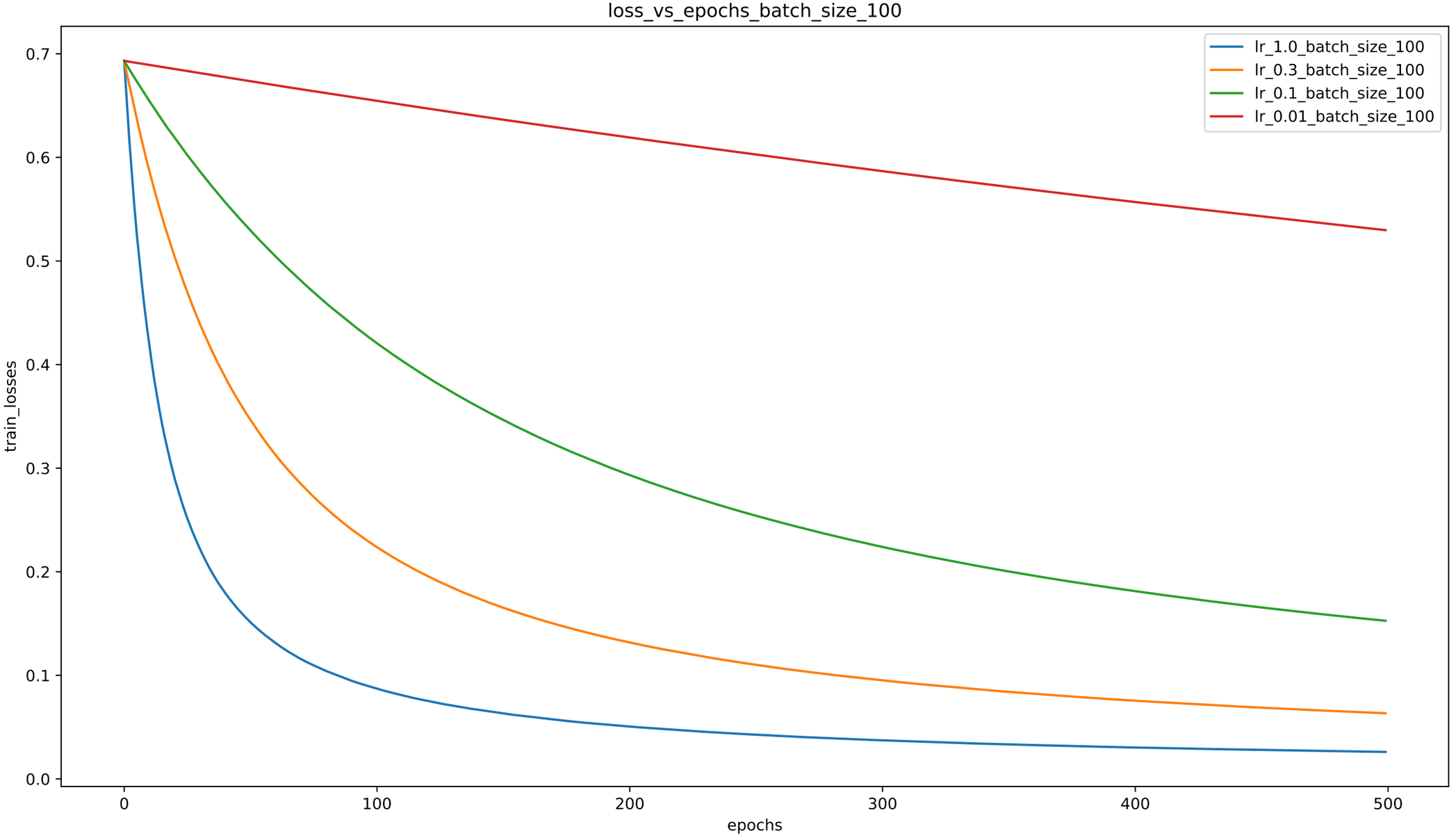
[39]:
```
def main(lr,batch_size):
    data_path = "./"
    num_iters = 500
    X_train, Y_train = dataset["X_train"], dataset["Y_train"]
    X_test, Y_test = dataset["X_test"], dataset["Y_test"]

    in_dim = X_train.shape[1]
    W = np.zeros(shape=(in_dim,1))

    train_data_loss_arr = []
    train_time_arr = []
    start_time = time.time()
    eval_time = 0.
    loss_calc_time = 0.
    for i in range(num_iters):
        train_data_loss = 0.0
        idxs = np.random.choice(X_train.shape[0], batch_size, replace=True)
        X = X_train[idxs]
        y_true = Y_train[idxs]

        loss, grad = get_loss_grad(W, X, y_true)
        train_time_arr.append(time.time() - start_time - loss_calc_time)

        temp = time.time()
        train_data_loss_arr.append(test_train_data(W, X_train, Y_train)*1.0)
        loss_calc_time += time.time() - temp
        W[:,0] -= lr*grad



    return np.array(train_data_loss_arr), np.array(train_time_arr)
```

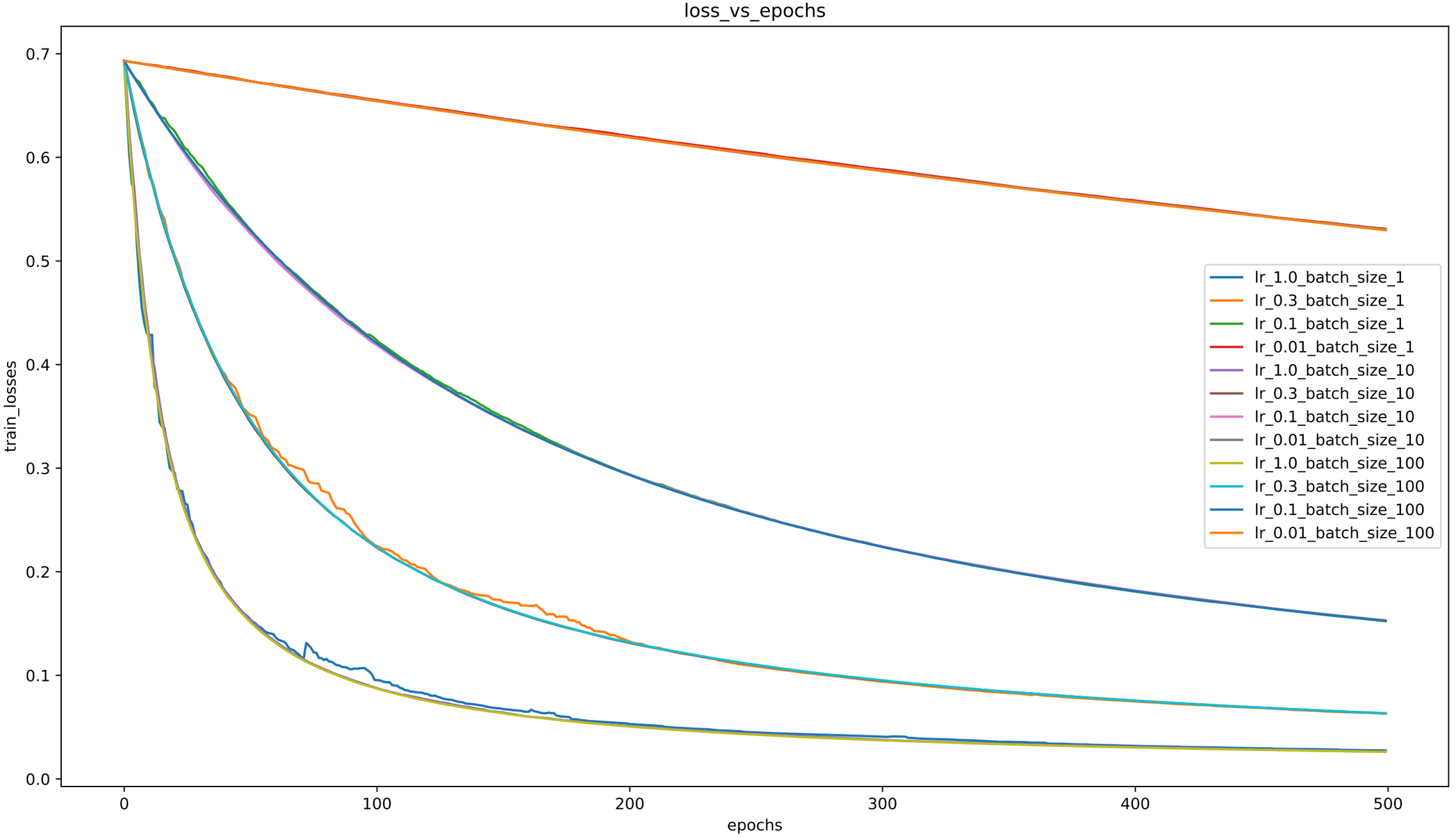
```
#     plot_props(train_data_loss_arr, "train_data_loss_lr_{}_batch_size_{}".
 ↪format(lr, batch_size))
```

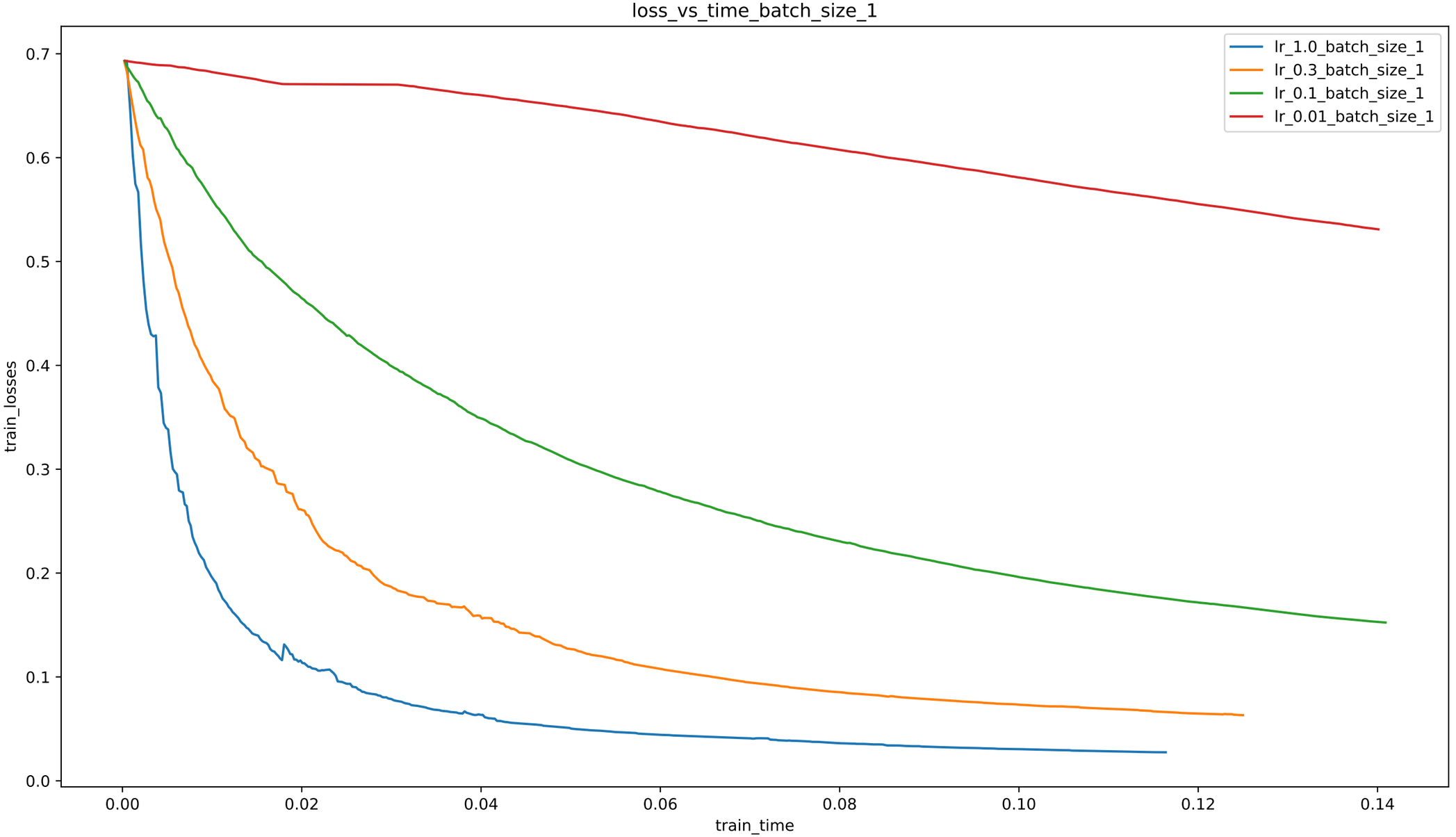
[ ]:
```
lrs = [1.0,0.3,0.1,0.01]
batch_sizes = [1,10,100]
# lrs = [1.0]
# batch_sizes = [10]
TRAIN_LOSSES = []
TRAIN_TIMES = []
PROP_NAMES = []
for batch_size in batch_sizes:
    print("batch_size", batch_size)
    prop_names = []
    train_losses = []
    train_times = []
    for lr in lrs:
        print("learning_rate", lr)
        start_time = time.time()
        train_loss, train_time = main(lr, batch_size)
        train_losses.append(train_loss*1.0)
        train_times.append(train_time*1.0)
#         print("Time_taken: {:1f}".format(time.time() - start_time))
        prop_names.append("lr_{}_batch_size_{}".format(lr, batch_size))
    plot_props(train_losses, prop_names, "loss_vs_epochs_batch_size_{}".
 ↪format(batch_size),"epochs")
    plot_props(train_losses, prop_names, "loss_vs_time_batch_size_{}".
 ↪format(batch_size), "train_time", train_times)
    TRAIN_LOSSES.extend(train_losses)
    TRAIN_TIMES.extend(train_times)
    PROP_NAMES.extend(prop_names)
plot_props(TRAIN_LOSSES, PROP_NAMES, "loss_vs_epochs", "epochs")
plot_props(TRAIN_LOSSES, PROP_NAMES, "loss_vs_time", "train_time", TRAIN_TIMES)
```

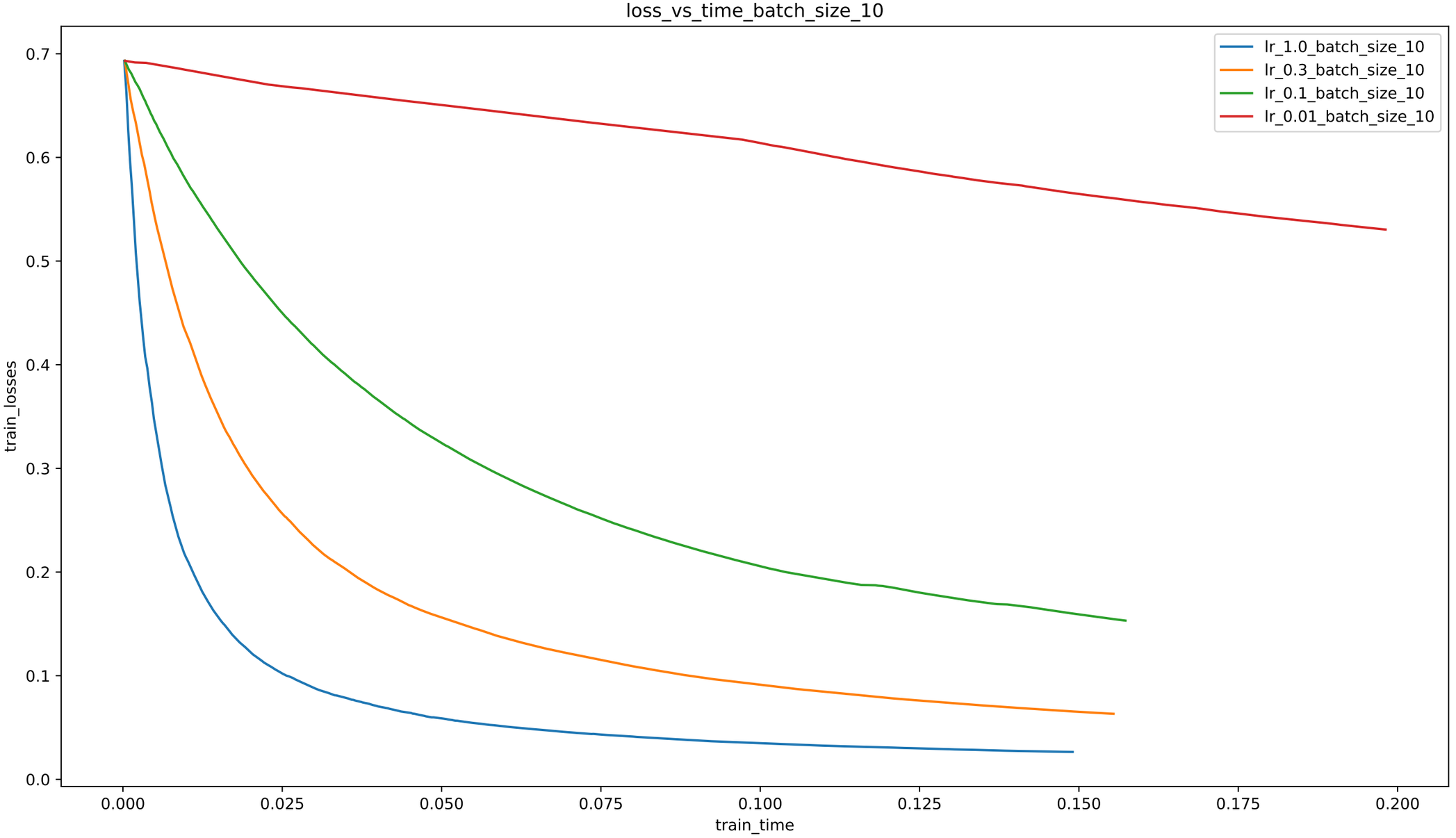

loss_vs_epochs_batch_size_1
lr_1.0_batch_size_1
lr_0.3_batch_size_1
lr_0.1_batch_size_1
lr_0.01_batch_size_1
train_losses
epochs
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
100
200
300
400
500

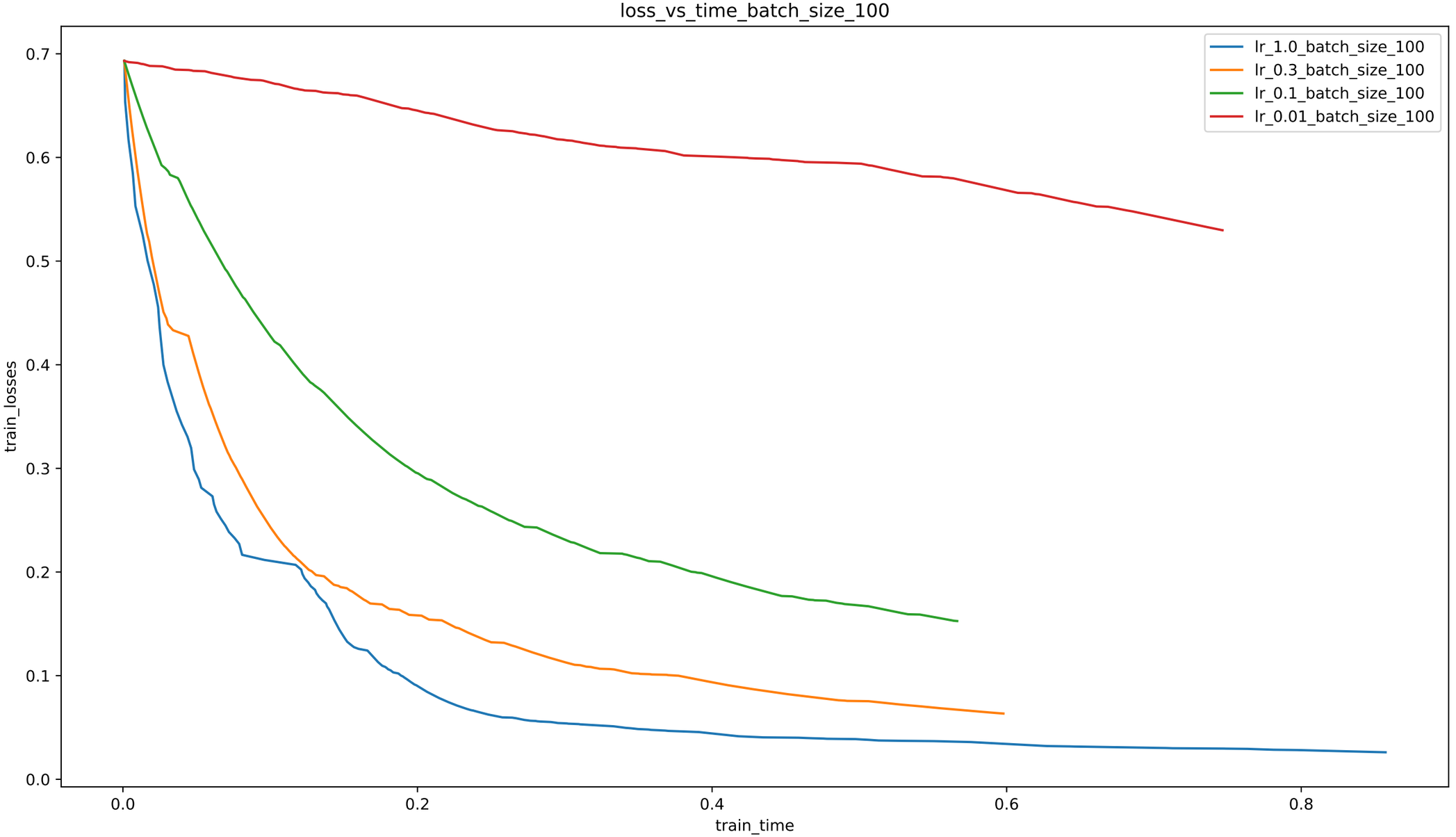

loss_vs_epochs_batch_size_10
lr_1.0_batch_size_10
lr_0.3_batch_size_10
lr_0.1_batch_size_10
lr_0.01_batch_size_10
train_losses
epochs
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
100
200
300
400
500

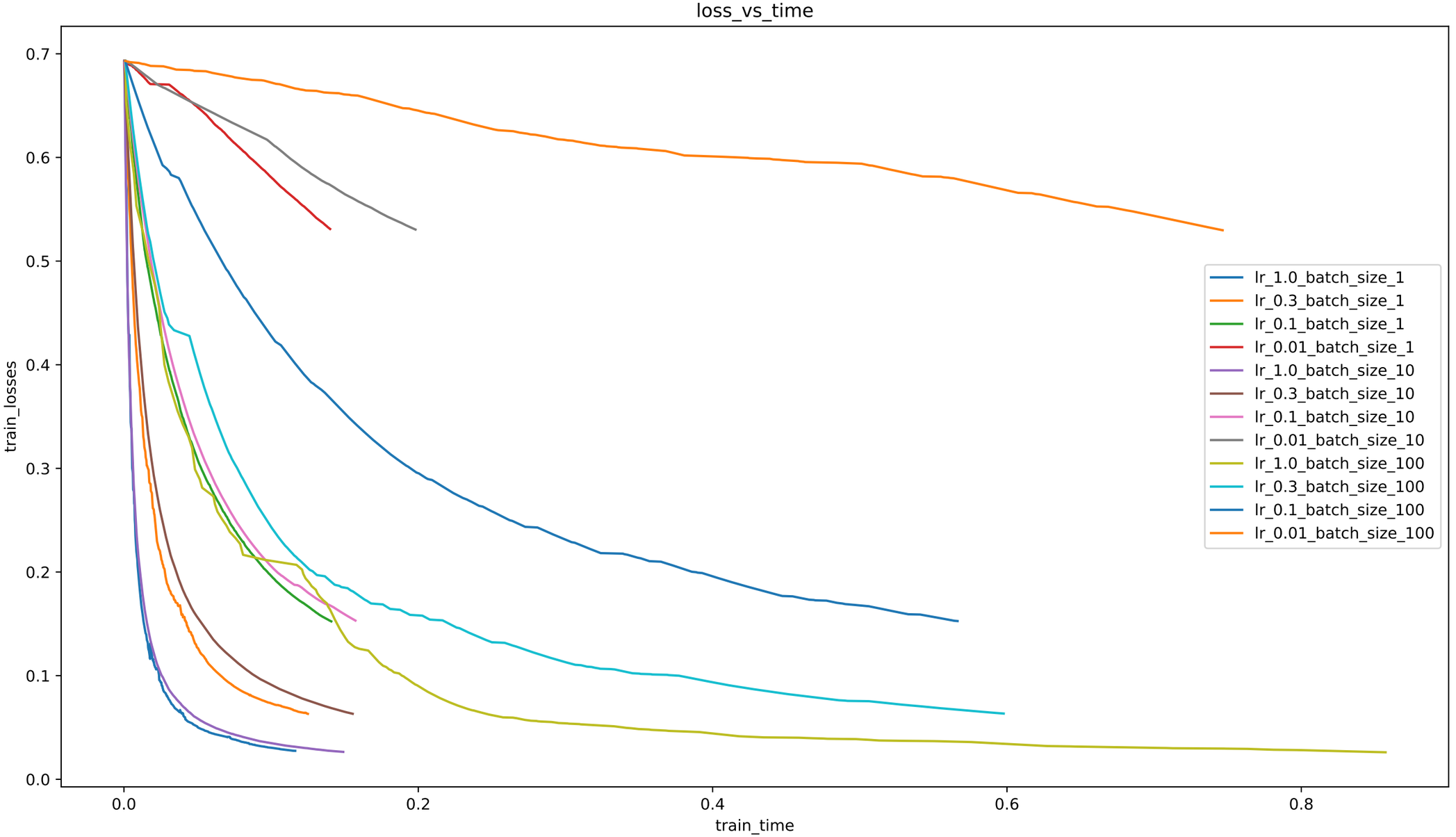

loss_vs_epochs_batch_size_100
lr_1.0_batch_size_100
lr_0.3_batch_size_100
lr_0.1_batch_size_100
lr_0.01_batch_size_100
train_losses
epochs
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
100
200
300
400
500


loss_vs_epochs
train_losses
epochs
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
100
200
300
400
500
lr_1.0_batch_size_1
lr_0.3_batch_size_1
lr_0.1_batch_size_1
lr_0.01_batch_size_1
lr_1.0_batch_size_10
lr_0.3_batch_size_10
lr_0.1_batch_size_10
lr_0.01_batch_size_10
lr_1.0_batch_size_100
lr_0.3_batch_size_100
lr_0.1_batch_size_100
lr_0.01_batch_size_100


loss_vs_time_batch_size_1
lr_1.0_batch_size_1
lr_0.3_batch_size_1
lr_0.1_batch_size_1
lr_0.01_batch_size_1
train_losses
train_time
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14


loss_vs_time_batch_size_10
lr_1.0_batch_size_10
lr_0.3_batch_size_10
lr_0.1_batch_size_10
lr_0.01_batch_size_10
train_losses
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
train_time


loss_vs_time_batch_size_100
lr_1.0_batch_size_100
lr_0.3_batch_size_100
lr_0.1_batch_size_100
lr_0.01_batch_size_100
train_losses
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.0
0.2
0.4
0.6
0.8
train_time


loss_vs_time
train_losses
train_time
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.0
0.2
0.4
0.6
0.8
lr_1.0_batch_size_1
lr_0.3_batch_size_1
lr_0.1_batch_size_1
lr_0.01_batch_size_1
lr_1.0_batch_size_10
lr_0.3_batch_size_10
lr_0.1_batch_size_10
lr_0.01_batch_size_10
lr_1.0_batch_size_100
lr_0.3_batch_size_100
lr_0.1_batch_size_100
lr_0.01_batch_size_100

3.1) $f(w) = f_1(w) + f_2(w)$;

$f_1(w) = 2(w-1)^2 - w^2$

$f_2(w) = 4(w+1)^2 - w^2$

Global minimum of f

$\nabla_w f(w) = \nabla f_1(w) + \nabla_w f_2(w) = 0$

$= 4(w-1) - 2w + 8(w+1) - 2w = 0$

$= 2w - 4 + 6w + 8 = 0$

$\Rightarrow \boxed{w = -\frac{4}{8} = -1/2}$

Naive ADMM :-

$$w_j^{(t+1)} = \underset{w}{\operatorname{argmin}} \left\{ f_j(w) + \lambda \| w - w^{(t)} \|_2^2 \right\}$$

iteration 1 :-

for j = 1

$$w_1^{(1)} = \underset{w_1}{\operatorname{argmin}} \left\{ 2(w_1-1)^2 - w_1^2 + \lambda \| w_1 - w^{(0)} \|_2^2 \right\}$$

$\nabla_{w_1} f_1$

iteration 1 : $w^{(0)} = 0$, $\lambda = 1$

$$w_1^{(1)} = \underset{w}{\operatorname{argmin}} \left(2(w-1)^2 + w^2 + \|w - w^{(0)}\|_2^2 \right)$$

Setting $\nabla_w = 4(w-1) - 2w + 2(w - w^{(0)}) = 0$

$$2w + 2w - 4 = 0$$

$$\Rightarrow w_1^{(1)} = 1$$

$$w_2^{(1)} = \underset{w}{\operatorname{argmin}} \left(4(w+1)^2 + w^2 + \|w - w^{(0)}\|_2^2 \right)$$

Setting $\nabla_w = 8(w+1) - 2w + 2(w - w^{(0)}) = 0$

$$\Rightarrow 8w + 8 = 0$$

$$\Rightarrow w = -1$$

$$\Rightarrow w_2^{(1)} = -1$$

$$\Rightarrow w^{(1)} = \frac{1}{2}\left(w_1^{(1)} + w_2^{(1)}\right) = \frac{1}{2}(0)$$

$$\Rightarrow \boxed{w^{(1)} = 0}$$

as $w^{(1)} = w^{(0)}$

$$\Rightarrow w_1^{(2)} = w_1^{(1)} = 1$$

and $w_2^{(2)} = w_2^{(1)} = -1$

$\Rightarrow w_2^{(2)} = 0$

$\Rightarrow w^{(i)} = 0 \quad \forall i \in \mathbb{N}$

i.e. w converges to $w = 0$

$\Rightarrow$ Clearly the naive ADMM does not converge to the global minimum.

3.2) $$\min_{\forall w_j, w} \frac{1}{m} \sum_{j=1}^{m} \left(f_j(w_j) + \lambda \|w_j - w\|_2^2 \right)$$

$$\text{S.t. } w_j = w \quad \forall j$$

Dual problem

$$L(\alpha, w_j, w) = \frac{1}{m} \sum_{j=1}^{m} \left[\left(f_j(w_j) + \lambda \|w_j - w\|_2^2 \right) + \alpha_j (w_j - w) \right]$$

$$\nabla_{w_1} L = 4w_1 - 4 - 2w_1 + 2(w_1 - w) + \alpha_1 = 0$$

$$\Rightarrow \alpha_1 = 2w - 4w_1 + 4 \quad -(1)$$

$$\nabla_{w_2} L = 8w_2 + 8 - 2w_2 + 2(w_2 - w) + \alpha_2 = 0$$

$$\Rightarrow \alpha_2 = 2w - 8w_2 - 8 \quad -(2)$$

$$\nabla_w L = 2(w_1 - w) + 2(w_2 - w) - \alpha_1 - \alpha_2 = 0 \quad -(3)$$

Put $w_1 = w_2 = w$ in (1), (2), and (3)

$$\Rightarrow \alpha_1 = -2w + 4 \quad -(4)$$

$$\alpha_2 = -6w + 8 \quad -(5)$$

$$\alpha_1 = -\alpha_2 \quad -(6)$$

using (4) and (5) in (6)

$$-2w + 4 = 6w + 8$$

$$\Rightarrow \quad 8w = -4$$

$$\Rightarrow \boxed{w = -1/2}$$

and $\boxed{\alpha_1 = 5}$ and $\boxed{\alpha_2 = -5}$

Clearly the optimal dual solution $\boxed{w = -1/2}$ is the same as the primal optimal solution obtained in (3.1) when $f(w)$ was directly minimized.

admm

March 6, 2020

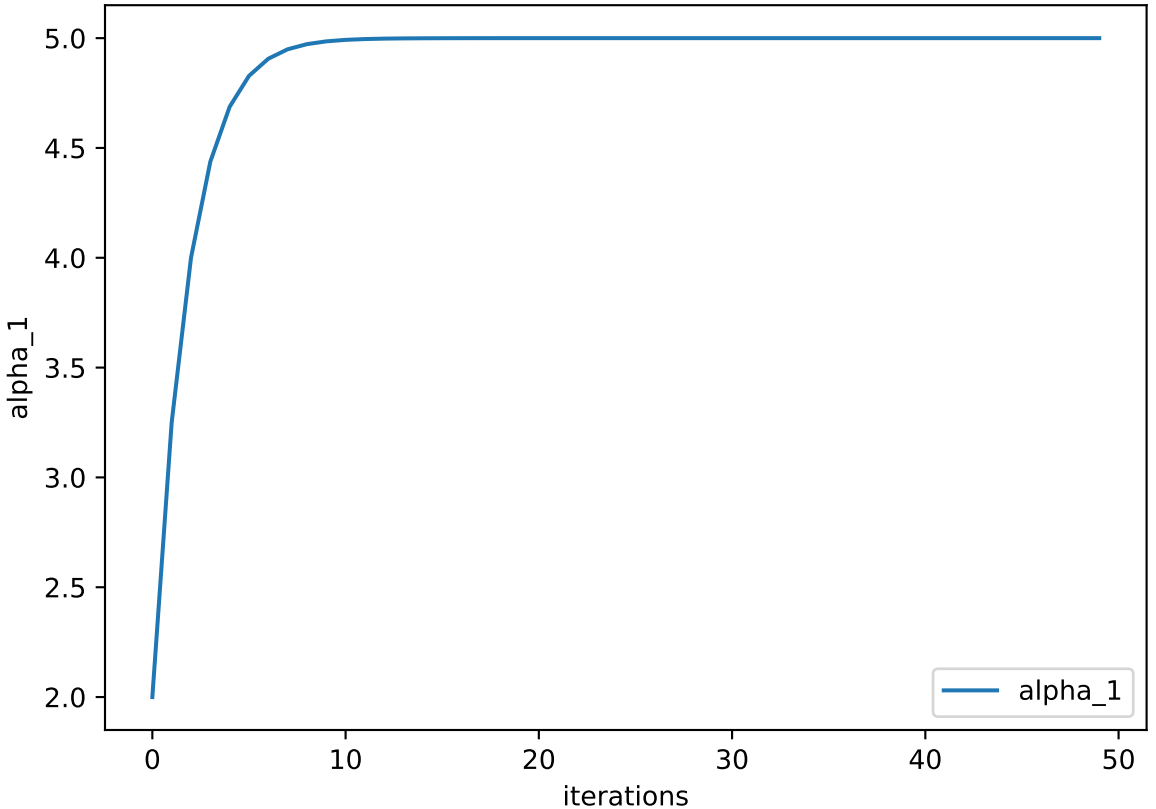
[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

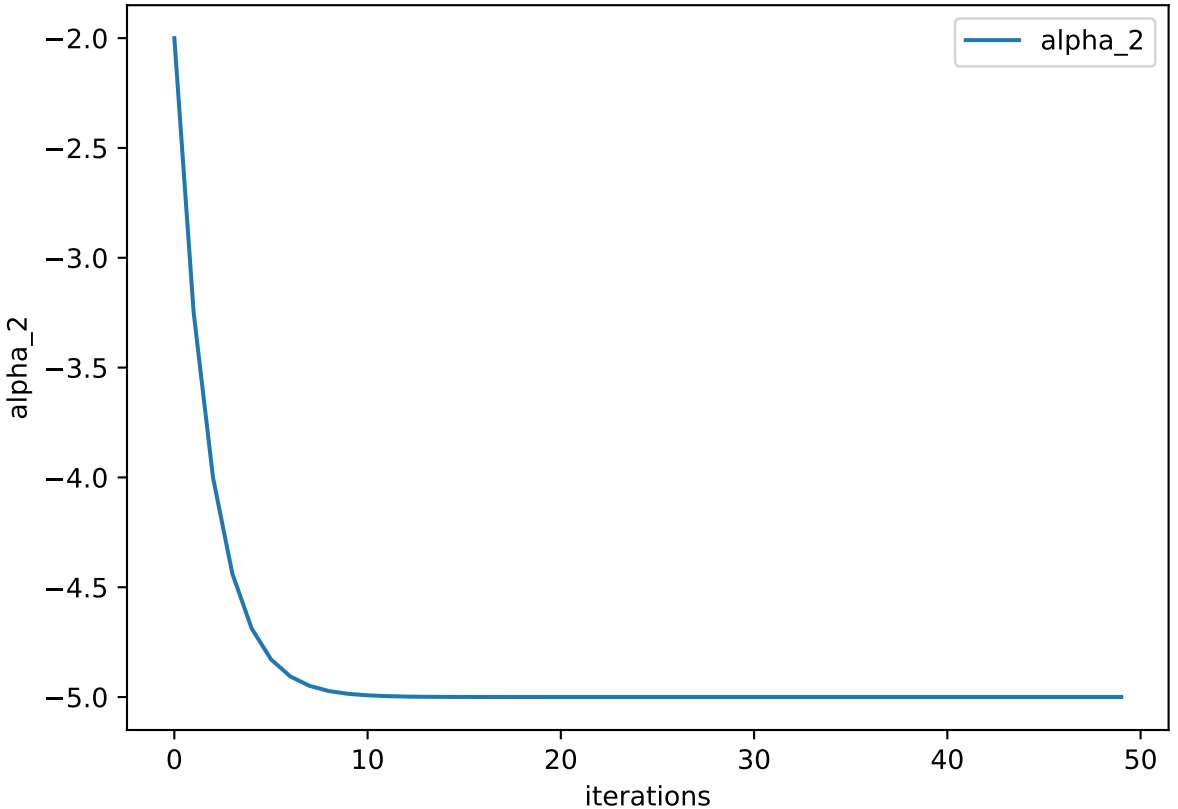
[17]:
```
def plot_props(data,prop_name, figname, xlabel):
    fig = plt.figure(figsize=(7,5))
    plt.plot(data, label=prop_name)
    plt.ylabel(prop_name)
    plt.xlabel(xlabel)
    plt.legend()
    plt.savefig("./{}.pdf".format(figname))
#     plt.show()
```

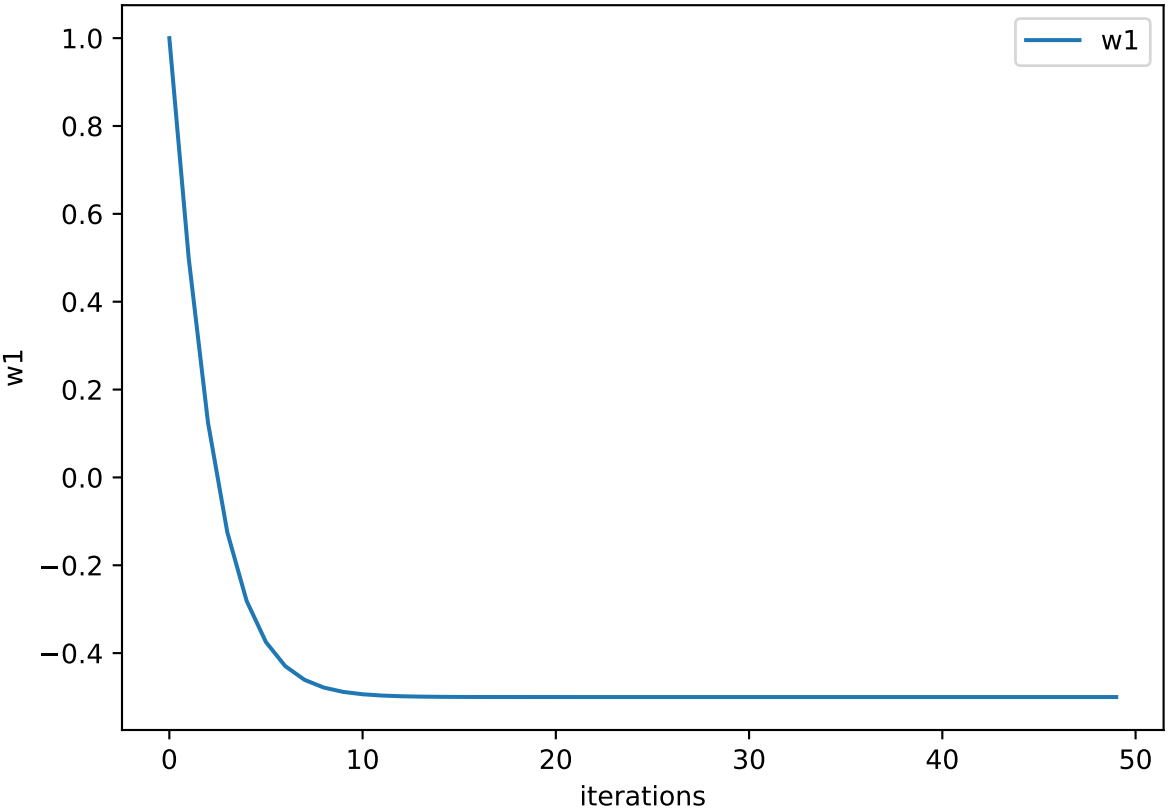
[19]:
```
def admm(lambda_, w1_init, w2_init, W_init, alpha_init):
    # learning rate
    lr = 2*lambda_
    # initialize weights
    w1, w2, W = w1_init, w2_init, W_init

    # initialize alpha
    alpha = alpha_init

    alpha1_arr = []
    alpha2_arr = []

    w1_arr = []
    w2_arr = []

    W_arr = []

    del_W = 1e5

    for i in range(50):
        # inner minimization
        w1 = (2*W +4 - alpha[0]) / 4.0
        w2 = (2*W -8 - alpha[1]) / 8.0
```

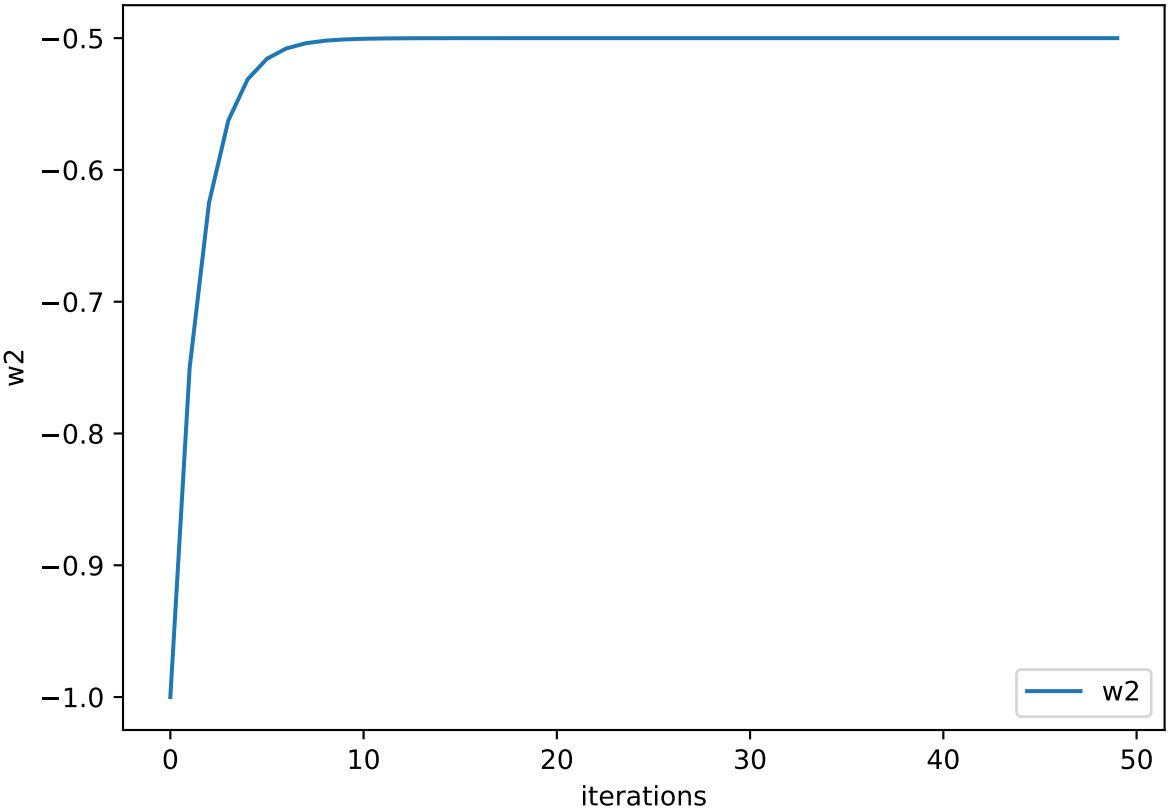
```
        # update W
        del_W = W*1.0
        W = (alpha.sum()/(4.0*lambda_)) + 0.5*(w1+w2)

        del_W = abs(del_W - W)
#         print(del_W)
        #update alpha
        alpha = alpha - lr*np.array([W-w1, W-w2])
        alpha1_arr.append(alpha[0]*1.0)
        alpha2_arr.append(alpha[1]*1.0)
        w1_arr.append(w1*1.0)
        w2_arr.append(w2*1.0)
        W_arr.append(W*1.0)

    ## plotting

    plot_props(alpha1_arr, "alpha_1", "alpha_1_vs_iters","iterations")
    plot_props(alpha2_arr, "alpha_2", "alpha_2_vs_iters","iterations")
    plot_props(w1_arr, "w1", "w1_vs_iters","iterations")
    plot_props(w2_arr, "w2", "w2_vs_iters","iterations")
    plot_props(W_arr, "W", "W_vs_iters","iterations")
```

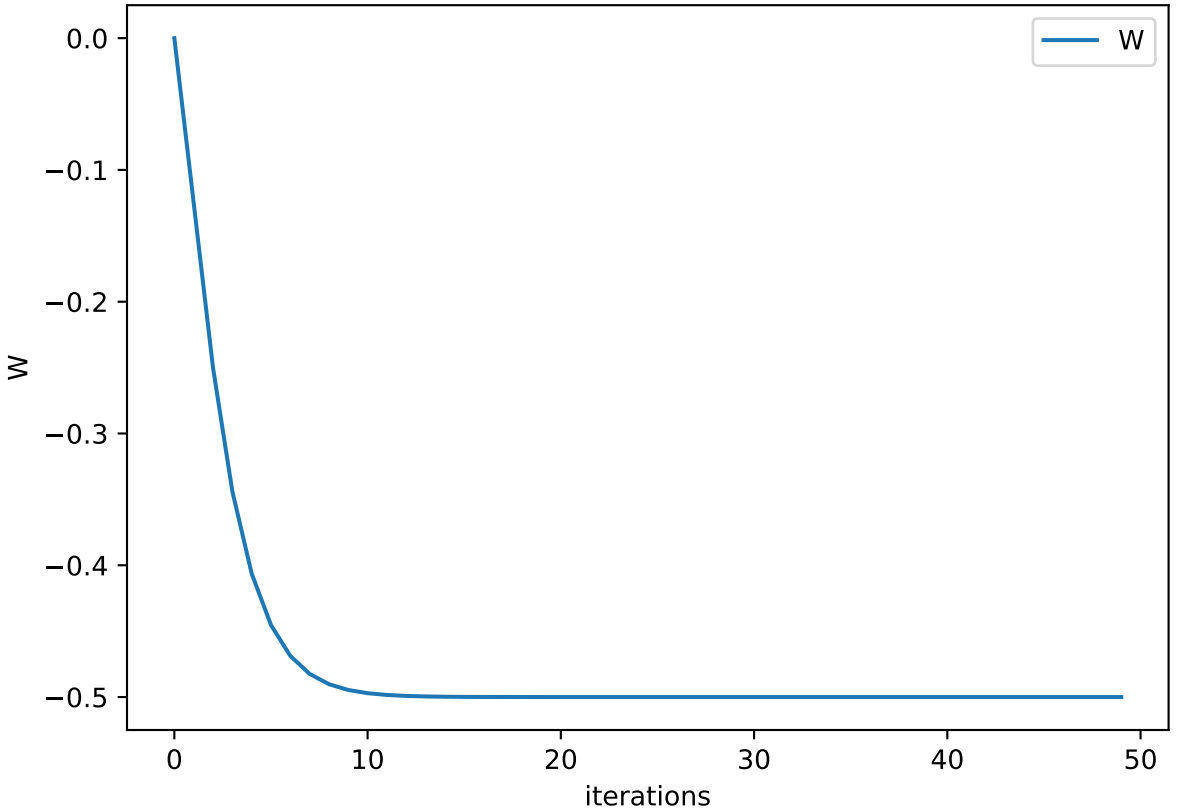
[20]:
```
lambda_ = 1.
w1_init = 0.0
w2_init = 0.0
W_init = 0.0
alpha_init = np.zeros(shape=(2,1))

admm(lambda_, w1_init, w2_init, W_init, alpha_init)
```


5.0
4.5
4.0
3.5
3.0
2.5
2.0
alpha_1
0
10
20
30
40
50
iterations
alpha_1


alpha_2
−2.0
−2.5
−3.0
−3.5
−4.0
−4.5
−5.0
alpha_2
0
10
20
30
40
50
iterations


w1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
w1
0
10
20
30
40
50
iterations


−0.5
−0.6
−0.7
−0.8
−0.9
−1.0
w2
0
10
20
30
40
50
iterations
w2


W
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
W
0
10
20
30
40
50
iterations

4.1) $h(x) = \lambda|x|$, $\lambda > 0$

$$\text{prox}_h(x) = \underset{z}{\text{argmin}} \left\{ h(z) + \frac{1}{2}\|z - x\|_2^2 \right\}$$

$$= \underset{z}{\text{argmin}} \left\{ \lambda|z| + \frac{1}{2}\|z - x\|_2^2 \right\}$$

Let $f(z) = \lambda|z| + \frac{1}{2}\|z - x\|_2^2$

~~$\Rightarrow \nabla_z f(z)$~~ for $z \neq 0$

$$\nabla_z f(z) = \lambda \text{sgn}(z) + (z - x) = 0$$

$$\Rightarrow z = x - \lambda \text{sgn}(z)$$

Case 1 $x > \lambda$

$$\Rightarrow z = x - \lambda \text{sgn}(z) > \lambda - \lambda \text{sgn}(z)$$

if $z > 0$

$$\Rightarrow z > \lambda - \lambda \Rightarrow \boxed{z > 0}$$

~~or $\text{sgn}(z) = \text{sgn}(x)$~~

if $z < 0$

$$\Rightarrow z > \lambda + \lambda \Rightarrow z > 2\lambda > 0$$

but z has to be < 0 (contradiction)

$\Rightarrow$ for $x > \lambda$, $z > 0$

~~$\Rightarrow \text{sgn}(x) = \text{sgn}(z)$~~

$\Rightarrow \text{sgn}(z) = \text{sgn}(x)$

Case 2 $x < -\lambda$

$\Rightarrow z = x - \text{sgn}(z) < -\lambda - \text{sgn}(z)$

if $z > 0$

$\Rightarrow z < -\lambda - \lambda \Rightarrow z < -2\lambda < 0$

contradiction

if $z < 0$

$\Rightarrow z < -\lambda + \lambda \Rightarrow z < 0$

$\Rightarrow$ for $x < -\lambda$, $z < 0$

$\Rightarrow \text{sgn}(z) = \text{sgn}(x)$

Case 3 when $-\lambda \leq x \leq \lambda$

$\Rightarrow -\lambda - \lambda\,\text{sgn}(z) \leq z = x - \lambda\,\text{sgn}(z) \leq \lambda - \lambda\,\text{sgn}(z)$

if $z > 0$

$\Rightarrow -2\lambda \leq z \leq 0$ (contradiction)

if $z < 0$

$\Rightarrow 0 \leq z \leq 2\lambda$ (contradiction)

$\Rightarrow$ There is no valid z which can be obtained when $x \in [-\lambda, \lambda]$ when we set $\nabla_2 f(z) = 0$

As both $z<0$ and $z>0$ can't be minimizer of $f(z)$ when $x \in [-\lambda, \lambda]$

$\Rightarrow$ $z = 0$ has to be the minimizer when $x \in [-\lambda, \lambda]$

$\Rightarrow$ $$\text{prox}_h(x) = \begin{cases} x - \lambda \, \text{sgn}(x) & , \; x > |\lambda| \\ 0 & , \; \text{otherwise} \end{cases}$$

4.2) $h(x) = \lambda \|x\|_1 \quad , \quad x \in \mathbb{R}^d$

update rule → $x_{t+1} = \text{prox}_{\eta h}(x_t)$

From Lecture 10, slide 15,

$$\forall i \in [d]; \quad [x_{t+1}]_i = \left[\text{prox}_{\eta h}(x_t)\right]_i = \begin{cases} [x_t]_i - \eta\lambda \, \text{sgn}([x_t]_i), & |[x_t]_i| > \eta\lambda \\ 0 & \text{, otherwise} \end{cases}$$

$\Rightarrow \forall i \in [d]$

if $|[x_t]_i| > \eta\lambda$, $\Rightarrow [x_{t+1}]_i = [x_t]_i - \eta\lambda \, \text{sgn}([x_t]_i)$

~~$\Rightarrow |[x_{t+1}]_i| = |[x_t]_i|$~~

and as soon as $|[x_t]_i| \leq \eta\lambda$

$|[x_{t+1}]_i| = 0$

So starting from some $|[x_0]_i|$, in m number of steps where $m = \left\lceil \frac{|[x_0]_i|}{\eta\lambda} \right\rceil$ and $\lceil y \rceil$ is the ceiling function,

$|[x_m]_i| \leq \eta\lambda$

$\Rightarrow |[x_{m+1}]_i| = 0$

~~and once $|[x_{t+1}]_i| = 0$~~

$\Rightarrow$ ~~the~~ It will ~~take~~ take $\left\lceil \frac{\max_i |[x_0]_i|}{\eta_1} - 1 \right\rceil$ steps to ~~reduce~~ all dimensions to ~~0~~ exact 0.

$\Rightarrow$ It will take $O\left(\frac{\max_i |[x_0]_i|}{\eta_1}\right)$ steps.

Now, $\frac{\max_i |[x_0]_i|}{\eta_1} = \|x_0\|_\infty \leq \|x_0\|_2$

$\Rightarrow$ It will take $O\left(\frac{\|x_0\|_2}{\eta_1}\right)$ ~~steps~~ steps

to converge to exact 0.

Using gradient descent

In this case ~~when~~ the ~~gradient~~ update is still the same for $|[x_t]_i| > \eta_1$ ~~or~~ but when $|[x_t]_i| \leq \eta_1$ the ~~or~~ next $|[x_{t+1}]_i|$ is not set to 0 but rather follows the same update.

In this case the dimension can overshoot the exact minimizer and keep on oscillating.

eg:- Let $x \in \mathbb{R}^2$, $x_0 = [2, -4]$ and Let $\lambda = 5/3$

$\Rightarrow$ $i = 1$

$$[x_1]_1 = 2 - 5/3 = \frac{1}{3}$$

$$[x_{12}]_2 = -4 + 5/3 = -7/3$$

$i = 2$ $\quad [x_2]_1 = 1/3 - 5/3 = -4/3$

$$[x_2]_2 = -7/3 + 5/3 = -2/3$$

$i = 3$ $\quad [x_3]_1 = -4/3 + 5/3 = 1/3$

$$[x_3]_2 = -2/3 + 5/3 = 1$$

$i = 4$ $\quad [x_3]_1 = 1/3 - 5/3 = -4/3$

$$[x_3]_2 = 1 - 5/3 = -2/3$$

Clearly both the dimensions have started oscillating around 0 without converging.