Learning with previously unseen features

Advisor- Naresh Manwani
Mentor-Kulin Shah
Team No.-11

Nakul Vaidya 201501108
Omkar Miniar 201530217
Ritvick Gupta 201530057
Akshay Vyas 201530032

References :

Learning with with Previously Unseen Features by YUAN SHI and CRAIG A. KNOBLOCK (IJCAI - 17):

https://www.ijcai.org/proceedings/2017/0379.pdf

GITHUB LINK :

https://github.com/akshay25vyas/learning_from_unseen_features

Problem Statement :

- **Improving** a machine learning model by identifying and using features that are **not in training set.**
- We propose a novel approach that learns a model over **both original** and **new features.**
- Author’s result of LUF shows significant improvement over baselines.

What is different ?

- Our problem is a special case of **semi-supervised learning.**
- Existing semi-supervised learning algorithms focus on using unlabeled data from the same feature space of labeled data .
- But we are building a new machine learning model that models over both original and new-features .

Ex : Consider a model that predicts a job applicant’s quality . Now qualities will be modeled using both qualities on resume and say applicant’s social media such as facebook.

Approach :

Given :

N Labeled samples $\{(X_s, Y_s)\}_s^N$ and M Unlabeled samples $\{(X_t, Z_t)\}_t^M$

Predict:

We want to learn a model $f\theta(x, z)$ where θ represents the model parameters, and the predicted label as $\boldsymbol{\hat{y} = f\theta(X, Z)}$

Challenges :

- For each source-domain sample (X_s , Y_s), if we could estimate its $\hat{z}_s$ reliably, we can simply train $f\theta(x,z)$ on the source domain. However, estimating z from x can be challenging when their dependency is weak.

- Training on the target domain is very challenging as there are no labels available.

Solutions:

- If our model predicts target-domain labels $\{\hat{y}_t\}$ well, then $\{\hat{y}_t\}$ should be consistent with the training labels. Consistency is expressed through the joint distribution of (x, y) such that $\{(x_s , y_s)\}$ and $\{(x_t , \hat{y}_t)\}$ are mixed as much as possible.

- When this happens, each source-domain sample (x_s, y_s) becomes close to its k-nearest neighbors in the target domain, and vise-versa. Therefore, we propose the following objective function to minimize the cross-domain k-nearest neighbor distances in the joint space of (x, y)

Problem Formulation :

$$\min_{\theta} \sum_{s} \sum_{t \in \mathcal{N}_{\mathcal{T}}^{k}(s)} \text{dist}[(\mathbf{x}_s, y_s), (\mathbf{x}_t, \hat{y}_t)] + \sum_{t} \sum_{s \in \mathcal{N}_{\mathcal{S}}^{k}(t)} \text{dist}[(\mathbf{x}_t, \hat{y}_t), (\mathbf{x}_s, y_s)] + \lambda \|\theta\|_2^2$$

Where ,

$$\text{dist}[(\mathbf{x}_s, y_s), (\mathbf{x}_t, \hat{y}_t)] = \|\mathbf{x}_s - \mathbf{x}_t\|_2^2 + \gamma \Delta(y_s, \hat{y}_t)$$

Regularized Linear Regression :

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m} h_\theta(x^{(i)}) - y^{(i)})^2 \right.$$

$$\left. + \lambda \sum_{j=1}^{n} \theta_j^2 \right]$$

Now,

$$\frac{\partial}{\partial\theta_j}\left(h_\theta(x^{(i)} - y^{(i)})\right)^2$$

$$\Rightarrow \quad 2\left[(h_\theta(x^{(i)} - y^{(i)})\frac{\partial}{\partial\theta_j}\{h_\theta(x^{(i)}\}\right]$$

For linear regression model

$$\frac{\partial}{\partial\theta_j}(h_\theta(x^{(i)})) = [x^{(i)}]_j$$

$$\frac{\partial}{\partial\theta_j}\lambda\sum_{j=1}^{n}\theta^2 = 2\lambda\theta_j$$

So for regularized linear case,

$$\frac{\partial}{\partial\theta_j} J(\theta) = \frac{1}{m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \lambda\theta_j\right]$$

$$\min_\theta \sum_s \sum_{t \in \mathcal{N}_\mathcal{T}^k(s)} \left[v_{st}^2 + \gamma \Delta\left(y_s, f_\theta(\mathbf{x}_t, \mathbf{z}_t)\right)\right]$$

$$+ \sum_t \sum_{s \in \mathcal{N}_\mathcal{S}^k(t)} \left[v_{ts}^2 + \gamma \Delta(y_s, f_\theta(\mathbf{x}_t, \mathbf{z}_t))\right] + \lambda ||\theta||_2^2$$

For regression tasks, we simply set :

$$\Delta(y_s, \hat{y}_t) = ||y_s - \hat{y}_t||^2.$$

For classification tasks with C classes, we use probabilistic classification models and set :

$$\Delta(y_s, \hat{y}_t) = 1 - \sum_{c=1}^{C} y_s(c)\hat{y}_t(c).$$

Where $\hat{y}_t$ is a C-dimensional vector representing the probability in each class, and y_s is a C-dimensional binary vector.

Alternating Optimization :

Let $V^k{}_T(s)$ index (X_s, Y_s)'s any (not necessarily the nearest) k neighbors in the target domain, and $V^k{}_s(t)$ index (x_t, y_t)'s any k neighbors in the source domain.

$$\sum_{t \in \mathcal{N}_{\mathcal{T}}^{k}(s)} \left[v_{st}^2 + \Delta(y_s, f_\theta(\mathbf{x}_t, \mathbf{z}_t))\right]$$
$$= \min_{\mathcal{V}_{\mathcal{T}}^{k}(s)} \sum_{t \in \mathcal{V}_{\mathcal{T}}^{k}(s)} \left[v_{st}^2 + \Delta(y_s, f_\theta(\mathbf{x}_t, \mathbf{z}_t))\right]$$

This is equivalent to

$$\min_{\theta, \{\mathcal{V}_{\mathcal{T}}^{k}(s)\}, \{\mathcal{V}_{\mathcal{S}}^{k}(t)\}} \sum_{s} \sum_{t \in \mathcal{V}_{\mathcal{T}}^{k}(s)} \left[v_{st}^2 + \Delta(y_s, f_\theta(\mathbf{x}_t, \mathbf{z}_t))\right]$$
$$+ \sum_{t} \sum_{s \in \mathcal{V}_{\mathcal{S}}^{k}(t)} \left[v_{ts}^2 + \Delta(y_s, f_\theta(\mathbf{x}_t, \mathbf{z}_t))\right] + \lambda \|\theta\|_2^2$$

When θ is fixed, we update $\{V^k{}_T(s)\}$ and $\{V^k{}_s(t)\}$ based on nearest neighbor search.
When $\{V^k{}_T(s)\}$ and $\{V^k{}_s(t)\}$ are fixed, we optimize θ by solving

$$\min_{\theta} \sum_{s} \sum_{t \in \mathcal{V}_{\mathcal{T}}^{k}(s)} \Delta(y_s, f_\theta(\mathbf{x}_t, \mathbf{z}_t)) + \sum_{t} \sum_{s \in \mathcal{V}_{\mathcal{S}}^{k}(t)} \Delta(y_s, f_\theta(\mathbf{x}_t, \mathbf{z}_t)) + \lambda \|\theta\|_2^2$$

This is easier to optimize than earlier equation when fθ is smooth in θ.

Algorithm 1 Optimization algorithm for **LUF**

Input: source-domain samples $\{(\mathbf{x}_s, y_s)\}_{s=1}^{\mathsf{N}}$ and target-domain samples $\{(\mathbf{x}_t, \mathbf{z}_t)\}_{t=1}^{\mathsf{M}}$, neighbor size k, weight parameter γ, and regularization parameter λ.

Initialize θ by solving Eq. (8)

for $iter = 1, 2, \cdots, T$ **do**

  for $s = 1, 2, \cdots, N$ **do**

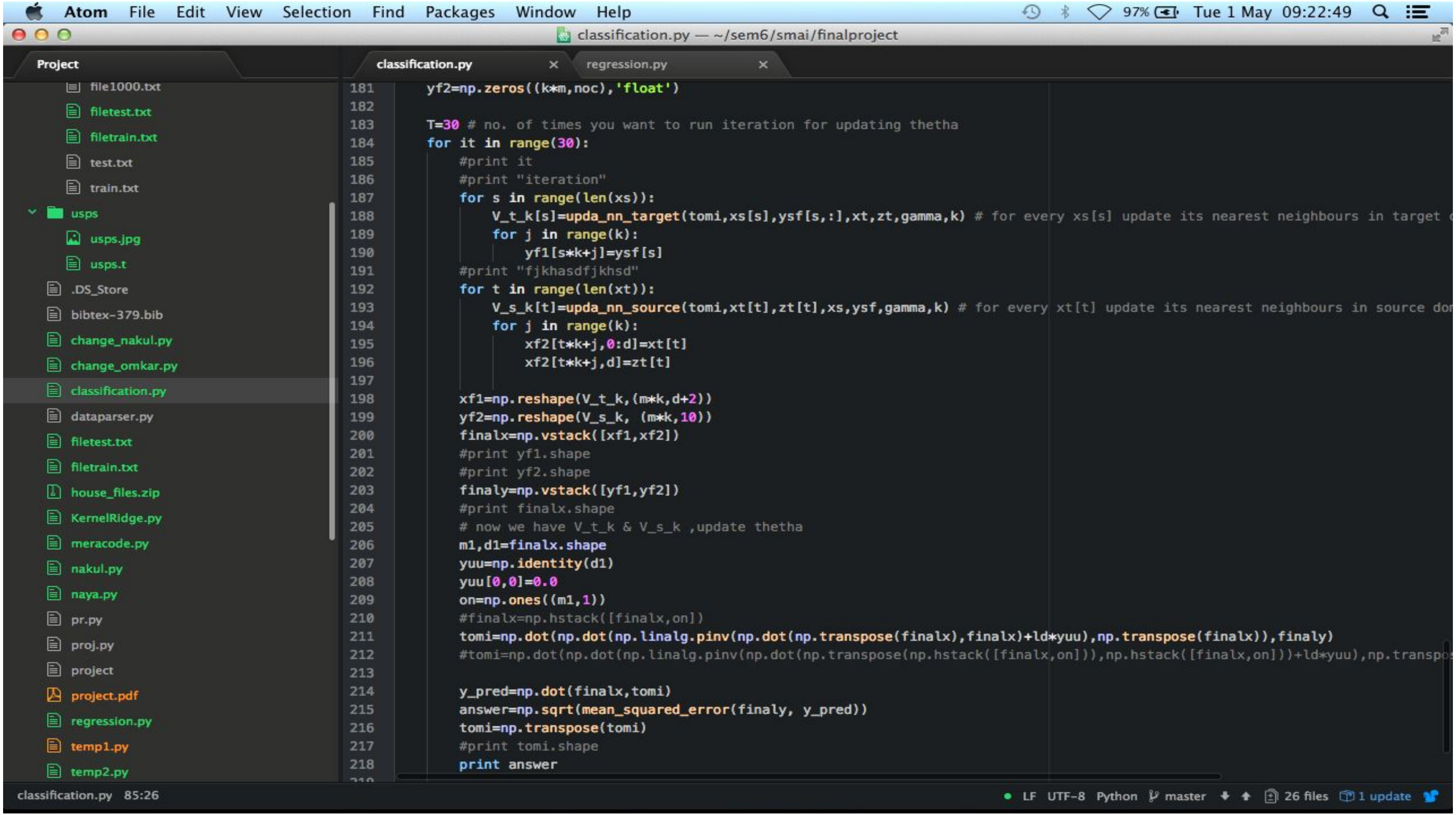
    Fix θ, update $\mathcal{V}_{\mathcal{T}}^{k}(s)$

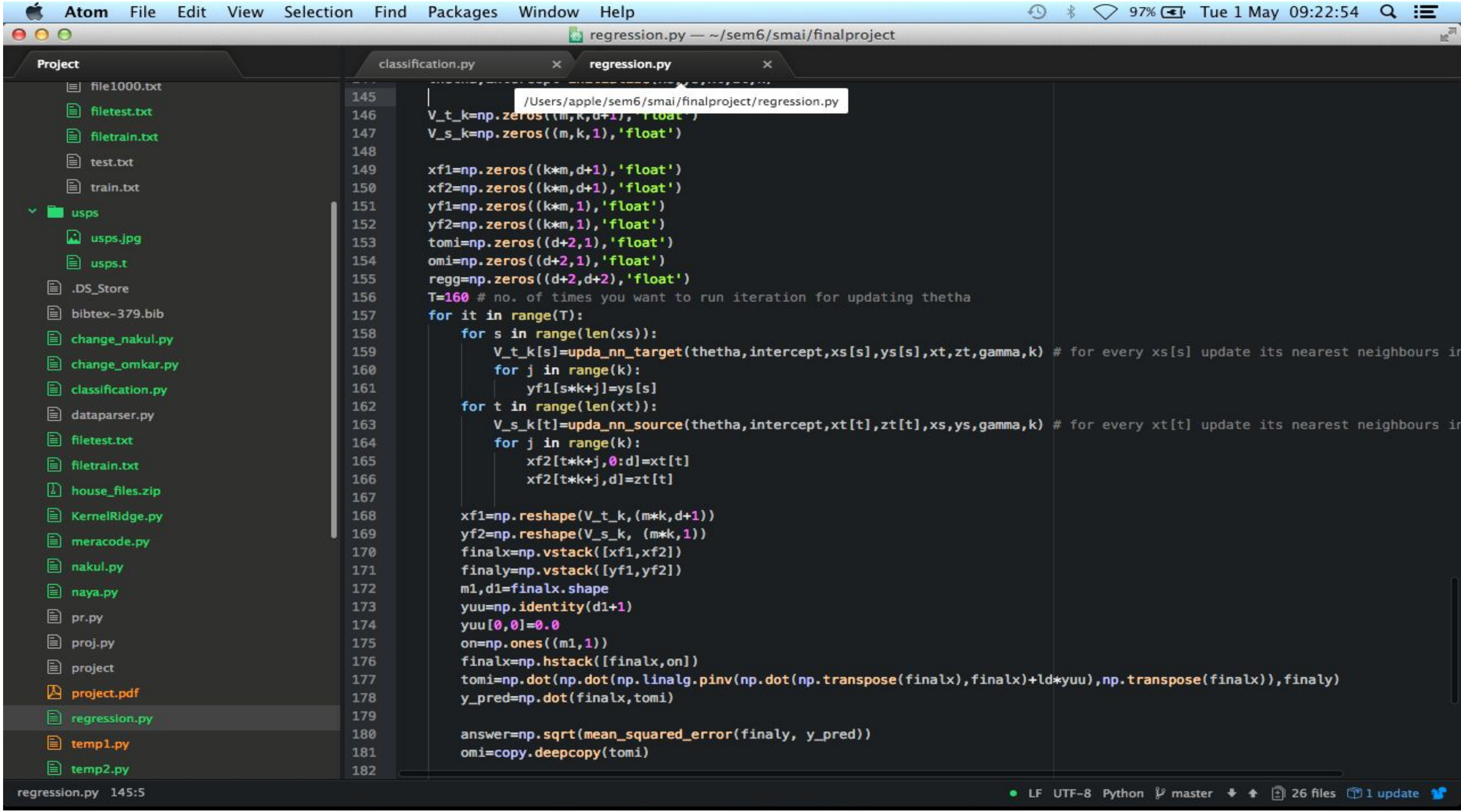
  for $t = 1, 2, \cdots, M$ **do**

    Fix θ, update $\mathcal{V}_{\mathcal{S}}^{k}(t)$

  Fix $\{\mathcal{V}_{\mathcal{T}}^{k}(s)\}, \{\mathcal{V}_{\mathcal{S}}^{k}(t)\}$, optimize θ by solving Eq. (6)

Output: a local optimal solution θ^*.


Atom File Edit View Selection Find Packages Window Help
97% Tue 1 May 09:22:49
classification.py — ~/sem6/smai/finalproject
Project
file1000.txt
filetest.txt
filetrain.txt
test.txt
train.txt
usps
usps.jpg
usps.t
.DS_Store
bibtex-379.bib
change_nakul.py
change_omkar.py
classification.py
dataparser.py
filetest.txt
filetrain.txt
house_files.zip
KernelRidge.py
meracode.py
nakul.py
naya.py
pr.py
proj.py
project
project.pdf
regression.py
temp1.py
temp2.py
classification.py
regression.py
yf2=np.zeros((k*m,noc),'float')
T=30 # no. of times you want to run iteration for updating thetha
for it in range(30):
#print it
#print "iteration"
for s in range(len(xs)):
V_t_k[s]=upda_nn_target(tomi,xs[s],ysf[s,:],xt,zt,gamma,k) # for every xs[s] update its nearest neighbours in target
for j in range(k):
yf1[s*k+j]=ysf[s]
#print "fjkhasdfjkhsd"
for t in range(len(xt)):
V_s_k[t]=upda_nn_source(tomi,xt[t],zt[t],xs,ysf,gamma,k) # for every xt[t] update its nearest neighbours in source do
for j in range(k):
xf2[t*k+j,0:d]=xt[t]
xf2[t*k+j,d]=zt[t]
xf1=np.reshape(V_t_k,(m*k,d+2))
yf2=np.reshape(V_s_k, (m*k,10))
finalx=np.vstack([xf1,xf2])
#print yf1.shape
#print yf2.shape
finaly=np.vstack([yf1,yf2])
#print finalx.shape
now we have V_t_k & V_s_k ,update thetha
m1,d1=finalx.shape
yuu=np.identity(d1)
yuu[0,0]=0.0
on=np.ones((m1,1))
#finalx=np.hstack([finalx,on])
tomi=np.dot(np.dot(np.linalg.pinv(np.dot(np.transpose(finalx),finalx)+ld*yuu),np.transpose(finalx)),finaly)
#tomi=np.dot(np.dot(np.linalg.pinv(np.dot(np.transpose(np.hstack([finalx,on])),np.hstack([finalx,on]))+ld*yuu),np.transpo
y_pred=np.dot(finalx,tomi)
answer=np.sqrt(mean_squared_error(finaly, y_pred))
tomi=np.transpose(tomi)
#print tomi.shape
print answer
classification.py 85:26
LF UTF-8 Python master 26 files 1 update




```
V_t_k=np.zeros((m,k,d+1),'float')
V_s_k=np.zeros((m,k,1),'float')

xf1=np.zeros((k*m,d+1),'float')
xf2=np.zeros((k*m,d+1),'float')
yf1=np.zeros((k*m,1),'float')
yf2=np.zeros((k*m,1),'float')
tomi=np.zeros((d+2,1),'float')
omi=np.zeros((d+2,1),'float')
regg=np.zeros((d+2,d+2),'float')
T=160 # no. of times you want to run iteration for updating thetha
for it in range(T):
    for s in range(len(xs)):
        V_t_k[s]=upda_nn_target(thetha,intercept,xs[s],ys[s],xt,zt,gamma,k) # for every xs[s] update its nearest neighbours in
        for j in range(k):
            yf1[s*k+j]=ys[s]
    for t in range(len(xt)):
        V_s_k[t]=upda_nn_source(thetha,intercept,xt[t],zt[t],xs,ys,gamma,k) # for every xt[t] update its nearest neighbours in
        for j in range(k):
            xf2[t*k+j,0:d]=xt[t]
            xf2[t*k+j,d]=zt[t]

    xf1=np.reshape(V_t_k,(m*k,d+1))
    yf2=np.reshape(V_s_k, (m*k,1))
    finalx=np.vstack([xf1,xf2])
    finaly=np.vstack([yf1,yf2])
    m1,d1=finalx.shape
    yuu=np.identity(d1+1)
    yuu[0,0]=0.0
    on=np.ones((m1,1))
    finalx=np.hstack([finalx,on])
    tomi=np.dot(np.dot(np.linalg.pinv(np.dot(np.transpose(finalx),finalx)+ld*yuu),np.transpose(finalx)),finaly)
    y_pred=np.dot(finalx,tomi)

    answer=np.sqrt(mean_squared_error(finaly, y_pred))
    omi=copy.deepcopy(tomi)
```

Initialization :

The quality of the solution depends on how we initialize θ. If we could get a good estimate of zˆs based on each source-domain sample xs, we can initialize θ by solving

$$\min_{\theta} \sum_{s} \Delta(y_s, f_\theta(\mathbf{x}_s, \hat{\mathbf{z}}_s)) \qquad (7)$$

However, estimating zˆs can be very challenging when the dependency between x and z is weak. for each xs, we find a set of its nearest neighbors in {xt}, and use the corresponding zt to form a candidate set Zs. We then minimize the model error by optimizing both θ and {zˆs}:

$$\min_{\theta} \sum_{s} \min_{\hat{\mathbf{z}}_s \in Z_s} \Delta(y_s, f_\theta(\mathbf{x}_s, \hat{\mathbf{z}}_s)) \qquad (8)$$

Complexity Analysis:

- In our Algorithm,each iteration involves N updates on $V^k_T(s)$ and M updates on $V^k_s(t)$.

- Each update on $V^k_T(s)$ takes $O(MD^2)$ and each update on $V^k_s(t)$ takes $O(ND^2)$, where D is the dimensionality of original features.Therefore the complexity of updating $\{V^k_T(s)\}$ and $\{V^k_s(t)\}$ at each iteration is $O(NMD^2)$.

- Additionally, each iteration involves learning a linear regression or logistic regression function, whose complexity is $O((D + W)^2(N + M))$ where W is the dimensionality of new features and we assume $N > D + W$.

- Further assuming $W = O(D)$ and $M = O(N)$, we have the overall complexity as $O(D^2N^2)$.

Results:

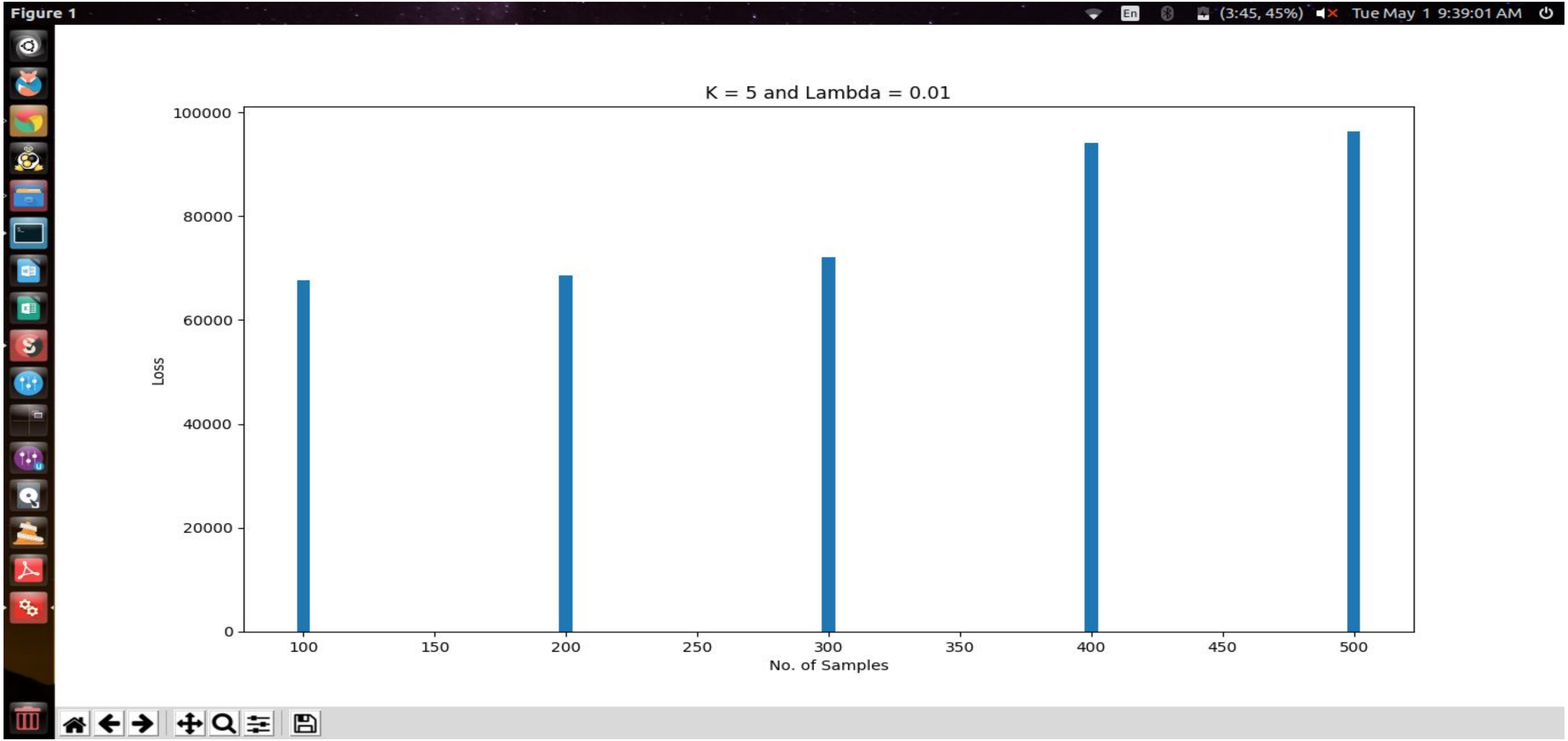


Figure 1
En
(3:45, 45%)
Tue May 1 9:39:01 AM
K = 5 and Lambda = 0.01
100000
80000
60000
40000
20000
0
Loss
100
150
200
250
300
350
400
450
500
No. of Samples

Results:

For Regression phase:

% of increase as compared to diff model:-

We used kernel Regression: 164071.171593 was its rmse error.

Our model Rmse Error : 135234.050503 (for 500 samples)

% increase is : (164071.171593 - 135234.050503)/164071.171593 * 100

I.e - 17.57% of improvement.

Results:

For Classification phase:

% of increase as compared to diff model:-

We used Logistic Regression: 13.256% was its classification error rate.

Our model classification error rate : 12.586% (for 500 samples)

% increase is : (13.256 - 12.586)/13.256 * 100

I.e - 5.05% of improvement.

Work Done:

- Implemented complete Model with Alternating optimization.
- Implemented both regression and classification model.
- Dataset used for calculating prediction error (RMSE) on
    - Regression : House, 8 for housing price prediction, contains 20,640 samples with 9 features.
    - Classification: USPS, which recognizes handwriting digits from images, contains 9,298 samples from 10 classes.

Conclusion

- We presented a novel machine learning approach that leverages previously unseen features in the training set. The approach is applicable to both classification and regression tasks. Supported by our empirical results, the approach can be used to improve a learning model when new features are accessible.
- We also plan to develop algorithms that can automatically determine when to explore new features and which features to select from a large pool of features in an open environment.

Thank You