Tutorial: ML debug, Optimize in numpy/scipy

Given by: Shikai Fang
For cs6350 machine learning fall 2022
Roadmap

- ML debug
  - Why so hard?
  - Procedure and tools
  - Tips

- Numpy: self-check points

- Scipy: optimize solver
Why debug in ML is so hard?

• 2 cases of debugging in ML
  • Your code doesn’t work. (error report by compiler, stupid output etc.)
  • Your code doesn’t work well enough.

• Debug for the case 1 is more general

Ref of this part: http://ai.stanford.edu/~zayd/why-is-machine-learning-hard.html
Why debug in ML is so hard?

- An example of 1\textsuperscript{st} case of a recursion func
  (you may use it in the decision-tree HW!)

```python
def recursion(input):
    if input is endCase:
        return transform(input)
    else:
        return recursion(transform(input))
```
Why debug in ML is so hard?

- Woop..I got bugs!
- 2-dims of possible error:
  - 1. Algorithm correctness
  - 2. Implementation bugs

- Identify your error state of these two-dim!
Why debug in ML is so hard?

• An example of 1st case of a recursion func
  (you may use it in the decision-tree HW!)

```
def recursion(input):
    if input is endCase:
        return transform(input)
    else:
        return recursion(transform(input))
```

Dataset, previous nodes, list of used features...

All left data are with same label / all features have been used ...

Compute the best split feature, add new node...
Why debug in ML is so hard?

• 2 cases of debugging in ML
  • Your code doesn’t work. (error report by complier, stupid output etc.)
  • Your code doesn’t work well enough.

• What about the case2?
  • Bad performance of model
  • Extremely slow
  • Unreasonable results..
Why debug in ML is so hard?

- Example: SGD in Linear regression: seems not converge.

```python
def optimize_SGD(self, x, y):
    dim = x.shape[1]
    n = x.shape[0]
    # update difference
    diff = 1
    # init w
    w = np.zeros([dim,1])
    fv = 1
    fun_val = []
    it = 0
    while fv > self.threshold:
        it = it + 1
        idx = np.random.randint(n,size=1)
        x1 = x[idx]
        y1 = y[idx]
        g = np.sum(np.matmul(x1,w) - y1)*x1, axis=1
        delta = -self.lr * np.reshape(g,(-1,1))
        w_new = w + delta
        diff = np.sqrt(np.sum(np.square(delta)))
        w = w_new
        tmp = np.reshape(np.squeeze(np.matmul(x,w)) - y, (-1,1))
        fv = 0.5 * np.sum(np.square(tmp))
        fun_val.append(fv)
```
Why debug in ML is so hard?

- Exponential debugging space: 4-dims!

- Algorithm
- Implementation
- Model
- Data
Why debug in ML is so hard?

- Exponential debugging space: 4-dims!

- Algorithm
- Implementation
- Model
- Data
Why debug in ML is so hard?

- Example: SGD in Linear regression: seems not converge..

```python
def optimize_SGD(self, x, y):
    dim = x.shape[1]
    n = x.shape[0]
    # update difference
    diff = 1
    # init w
    w = np.zeros([dim, 1])
    fv = 1
    fun_val = []
    it = 0
    while fv > self.threshold:
        it = it + 1
        idx = np.random.randint(n, size=1)
        x1 = x[idx]
        y1 = y[idx]
        g = np.sum(np.transpose((np.matmul(x1, w) - y1)*x1), axis=1)
        delta = -self.lr * np.reshape(g, (-1, 1))
        w_new = w + delta
        diff = np.sqrt(np.sum(np.square(delta)))
        w = w_new
        tmp = np.reshape(np.squeeze(np.matmul(x, w)) - y, (-1, 1))
        fv = 0.5 * np.sum(np.square(tmp))
        fun_val.append(fv)
```
Why debug in ML is so hard?

Thus..

- Be patient, no panic! – always hard (as life)..
- Identify your error state!
- Find your debug direction!

Often here for model finetuning..
Procedure and tools

More details/tools on “identify error state” “debug direction”

• Be patient, no panic!
• Identify your error state!
• Find your debug direction!
Procedure and tools

• Most important tool...

PRINT function (check the middle states)

• All other debugging tools should help more convenient “print”
  • Jupiter notebook (module-execution, data-visualization, )
  • Variable explorer in pycharm/vscode/spyder etc.
  • Setting breakpoints
Key points: Start from the simple and reduce your debugging search space

ML Model Development Process

If you follow best practices for developing your ML model, then debugging your ML model will be simpler. These best practices are as follows:

1. Start with a simple model that uses one or two features. Starting with a simple, easily debuggable model helps you narrow down the many possible causes for poor model performance.

2. Get your model working by trying different features and hyperparameter values. Keep your model as simple as possible to simplify debugging.

3. Optimize your model by iteratively trying these changes:
   - adding features
   - tuning hyperparameters
   - increasing model capacity

4. After each change to your model, revisit your metrics and check whether model quality increases. If not, then debug your model as described in this course.

5. As you iterate, ensure you add complexity to your model slowly and incrementally.

Ref of this part: https://developers.google.com/machine-learning/testing-debugging/summary
Tips and common cases

Some tips on “reduce the prob of bugs” “debug efficiently”..
Tips and common cases

• **Tip 1.** Comment /document your code well
Tips and common cases

• Tip 1. Comment /document your code well
• Example: for vector/matrix size-check

```plaintext
alpha = W_post_mean; %size: R_prod*1
eta = W_post_var; %size: R_prod*1

W_vec_var_post = inv(diag(1.0./eta) + tau*sum(bi2,3)); %size: R_prod * R_prod : prior+llk
W_vec_mean_post = W_vec_var_post*(alpha./eta + bi'*y*tau); %size: R_prod * 1 : prior+llk

W_vec_fac_var_post = inv(tau*sum(bi2,3)); %size: R_prod * R_prod : llk factor
W_vec_fac_mean_post = W_vec_fac_var_post*(bi'*y*tau); %size: R_prod * 1 : llk factor
```
Tips and common cases

- Tip 2. **Module your code & unit test**
- Write your data_loader /process /model/train /test ... into different functions/methods/classes
Tips and common cases

• Tip 2. **Module your code & unit test**
• Example: decision tree task

```python
# decision tree node, catogorical data values
class TreeNode:
    def __init__(self):
        self.feature = None
        self.children = None
        self.depth = -1
        self.isLeaf = False
        self.label = None

    def set_feature(self, feature):
        self.feature = feature

class ID3:
    # constructor
    # feature_selection: 0 information gain; 1 majority error; 2 gini index
    # max_depth maximum depth of decision tree
    # feature_selection, max_depth
    def __init__(self, feature_selection = 0, max_depth = 10):
        self.feature_selection = feature_selection
        self.max_depth = max_depth

    def set_feature_selection(self, feature_selection):
        self.feature_selection = feature_selection

    def set_max_depth(self, max_depth):
        self.max_depth = max_depth
```
Tips and common cases

• Tip 2. **Module your code & unit test**
• Example: decision tree task

```python
# generate decision tree
# dataset 2d matrix, each row is an instance, pandas dataframe
# features feature column names and values {'column': [values]}
# label label column name {'label':[values]}

def generate_decision_tree(self, dataset, features, label):
    Q = []
    dtRoot = TreeNode()
    dtRoot.set_depth(0)
    # processing node root
    root = {'dataset': dataset,'features': features, 'label': label, 'dtNode': dtRoot}
    Q.append(root)
    while len(Q) > 0:
        cur = Q.pop(0)
        nodes = self.split_dataset(cur)
        for node in nodes:
            Q.append(node)
    return dtRoot
```
Tips and common cases

• Tip 3. Search your error/questions on internet
• “Google / Stack-overflow oriented programming...”
Overall steps of debug in ML

Develop process: to

simple(data/model) → complex(data/module)

Bugs happens at any step

Locate your error state in 4-dim: Algo, Model, Data, Implement?

Debug & Fine-tunning

Jupyter-notebook/
Variable explorer/
Breakpoint helps..

Comment/
Module program/
Unit-test/stackoverflow helps..
Some self-check points of NumPy

1. Basic

What is np.array?
How to build an all-zero/all-one/random array given size?
How to get it from data file(cvs,txt,etc.)/ save it as data file?
How to assign value?
How to modify value?
How to indexing/slicing array?
How to build array from list/build list from array

2. For-loop/vectorize

How to do above operations in a for loop (by iteration) ?
How to do above operations for group of eles by once (vectorize/broadcasting) ?
Some self-check points of NumPy

3. Linear-algebra
How to **get the size** of an array?
How to **reshape** an array?
How to **transpose** an array?
How to do **matrix-mut** of two arrays?
How to do **element-wise mut/div/add/sub** of two arrays?
How to **expand a new-dim** of an array?
How to **concatenate/stack** of multi-arrays?
How to **get the mean/max/min value** of an array (on given dim)?
...
If you are not clear... google + check doc!
SciPy: a fast & strong Sci-computing library

• Based on NumPy: computing value should be np.ndarray

• Literally, almost everything you need in machine learning computing step: linear algebra, statistic, calculus, optimization, signal process..

• We will focus on the optimization module in the implementation of SVM

• Specifically, scipy.optimize.minimize()
Revisit the SVM problem:

Soft SVM

Variables \( \min \limits_w \)

\[
\begin{align*}
\text{Object functions} \\
\frac{1}{2} w^T w + C \sum_i \xi_i \\
\text{s.t.} \forall i, \\
y_i (w^T x_i + b) \geq 1 - \xi_i \\
\xi_i \geq 0
\end{align*}
\]

Constrains: ineq/eq

Some cases, we have “bound” like: \(0 < w_i < 1\) – special case of constrains!
What dose its doc says? ([link](#))

```python
scipy.optimize.minimize
```

Minimization of scalar function of one or more variables.

For the input things : seems trivial...

(not such trivial essentially )
Returns: \texttt{res : OptimizeResult}

The optimization result represented as a \texttt{OptimizeResult} object. Important attributes are: \texttt{x} the solution array, \texttt{success} a Boolean flag indicating if the optimizer exited successfully and \texttt{message} which describes the cause of the termination. See \texttt{OptimizeResult} for a description of other attributes.

What’s the output?: A strange class...->check doc! Everything needed
Example: Box design

- Design optimal L, W, H of a box.
- To maximize the volume of a box with limited surface area
- $H \leq 1.2$, $W \leq 2$, $L \leq 3$

- Volume $= L \times W \times H$
- Surface area $= 2 \times (L \times W + W \times H + L \times H)$

Ref: https://www.youtube.com/watch?v=iSnTkV6b0Gw
**Tips/Challenge for you in HW..**

- Use setting: “method = SLSOP” - fast, stable

- What if: I want to add **multi / equal** constrains

- What if: I want to add **bounds** for many variable

- What if: my **constrains/object function has other fix parameters**

_Hints-possible solutions:_

1. check doc carefully for solution
2. Lambda function
3. Class_method ...
• Thanks!
```
import numpy as np
from scipy.optimize import minimize

# define the function of volume
def calVolume(x):
    L = x[0]
    W = x[1]
    H = x[2]
    Volume = L*H*W
    return Volume

# define the function of surface
def calSurface(x):
    L = x[0]
    W = x[1]
    H = x[2]
    Surface = 2*(L*H + H*W + L*W)
    return Surface

# define the objective function: maximize vol <-> minimize -vol
def objective(x):
    return -calVolume(x)

# define the constrain for optimization
def constrain(x):
    return 10 - calSurface(x)

# load constrains into dictionary
cons = ([{'type': 'ineq', 'fun': constrain}])

# init the variables
L_init = 1
W_init = 1
H_init = 1

x0 = np.array([L_init, W_init, H_init])

# def the bounds of variables
bnds = Industries(0, 3), (0, 2), (0, 1)

sol = minimize(fun=objective, x0=x0, method='SLSQP', constraints=cons, bounds=bnds)

print(sol)
```

# load constrains into dictionary
cons = ([{'type': 'ineq', 'fun': constrain}])

# init the variables
L_init = 1
W_init = 1
H_init = 1

x0 = np.array([L_init, W_init, H_init])

# def the bounds of variables
bnds = Industries(0, 3), (0, 2), (0, 1)

sol = minimize(fun=objective, x0=x0, method='SLSQP', constraints=cons, bounds=bnds)

print(sol)