Intro. Comp. for Data Science (FMI08)

Dr. Nono Saha

May 10, 2023

Max Planck Institute for Mathematics in the Sciences
University of Leipzig/ScaDS.AI

Spring 2023

Course plan

1. `NumPy` numerics
2. `NumPy` - Advanced indexing
3. `NumPy` - Broadcasting
4. `NumPy` - Basic file I/O
5. Structure of a Data Science (ML) project
6. Homework 3

NumPy numerics

NumPy numerics: basic operations

All basic mathematical operators in `Python` are implemented for arrays. They are applied element-wise to the array values.

```
np.arange(3) + np.arange(3)
## array([0, 2, 4])

np.arange(3) - np.arange(3)
## array([0, 0, 0])

np.arange(3) + 2
## array([2, 3, 4])

```

```
np.arange(3) * np.arange(3)
## array([0, 1, 4])

np.arange(1,4)/np.arange(1,4)
## array([1., 1., 1.])

np.arange(3) * 3
## array([0, 3, 6])

```

```
np.full((2,2), 2) ** np.arange(4).reshape((2,2))
np.full((2,2), 2) ** np.arange(4)
## Which of the two instructions will work?

```

NumPy numerics: mathematical functions

The package provides a wide variety of basic mathematical functions that are vectorized. In general, they will be faster than their base equivalents (e.g. `np.sum()` vs `sum()`).

```
np.sum(np.arange(1000))
## 499500

np.cumsum(np.arange(10))
## array([ 0,  1,  3,  6, 10, 15, 21, 28, 36, 45])

np.log10(np.arange(1,11))
## array([0., 0.30103, 0.47712125, 0.60205999, 0.69897,
## 0.77815125, 0.84509804, 0.90308999, 0.95424251, 1. ])

np.median(np.arange(10))
## 4.5

```

NumPy numerics: matrix multiplication

It is supported using the `matmul()` function or the `@` operator,

```
x = np.arange(6).reshape(3,2)
y = np.tri(2,2)
x @ y
## array([[1., 1.], [5., 3.], [9., 5.]])

y.T @ y
## array([[2., 1.], [1., 1.]])

np.matmul(x.T, x)
## array([[20, 26], [26, 35]])

y @ x
##  Can this work?

```

NumPy numerics: other linear algebra functions

The standard linear algebra functions are (mostly) implemented in the `linalg` submodule. See here for more details.

```
np.linalg.det(y)
## 1.0

np.linalg.eig(x.T @ x)
## (array([ 0.43988174, 54.56011826]), array([[-0.79911221,
    -0.6011819 ], [ 0.6011819 , -0.79911221]]))

np.linalg.inv(x.T @ x)
## array([[ 1.45833333, -1.08333333], [-1.08333333,
    0.83333333]])

np.linalg.cholesky(x.T @ x)
## array([[4.47213595, 0.],[5.81377674, 1.09544512]])

```

NumPy numerics: random values

`NumPy` has another submodule called random for functions used to generate random values,

To use this, you should construct a generator via `default_rng()`, with or without a seed, and then use the generator's methods to obtain your desired random values.

```
rng = np.random.default_rng(seed = 1234)
rng.random(3) # ~ Uniform [0,1)
## array([0.97669977, 0.38019574, 0.92324623])

rng.normal(0, 2, size = (2,2))
## array([[ 0.30523839,  1.72748778],[ 5.82619845,
    -2.95764672]])

rng.binomial(n=5, p=0.5, size = 10)
## array([2, 4, 2, 2, 3, 4, 4, 3, 3, 3])

```

NumPy - Advanced indexing

From last time: subsetting with tuples

Unlike lists, a ndarray can be subset by a tuple containing integers

```
x = np.arange(6)
x
## array([0, 1, 2, 3, 4, 5])

x[(0,1,3),]

## array([0, 1, 3])

x[(0,1,3)]

## Traceback (most recent call last):
File "<stdin>", line 1, in <module>
IndexError: too many indices for array: array is 1-
dimensional, but three were indexed
```

Question

What if we use the list instead?

NumPy - Advanced indexing: exercise

Given the following matrix,

```
x = np.arange(16).reshape((4,4))
x
## array([[ 0,  1,  2,  3], [ 4,  5,  6,  7], [ 8,  9, 10,
11], [12, 13, 14, 15]])

```

Write an expression to obtain the centre 2x2 values (i.e. 5, 6, 9, 10 as a new matrix).

NumPy - Advanced indexing: boolean indexing

Lists or ndarrays of boolean values can also be used to subset, positions with True are kept, and False are discarded.

```
x = np.arange(6)
## array([0, 1, 2, 3, 4, 5])

x[[True, False, True, False, True, False]]
## array([0, 2, 4])

x[np.array([True, True, False, False, True, False])]
## array([0, 1, 4])

```

The utility comes from vectorized comparison operations,

```
x > 3
## array([False, False, False, False,  True,  True])
x[x>3]
## array([4, 5])
x % 2 == 1
## array([False,  True, False,  True, False,  True])

```

NumPy - Advanced indexing: boolean operators

If we want to use a boolean operator on an array, we need to use &, |, and ~ instead of **and**, **or**, **and not** respectively.

```
x = np.arange(6)
x
## array([0, 1, 2, 3, 4, 5])

y = x % 2 == 0
y
## array([ True, False,  True, False,  True, False])

~y
## array([False,  True, False,  True, False,  True])

y & (x > 3)
## array([False, False, False, False,  True, False])

y | (x > 3)
## array([ True, False,  True, False,  True,  True])

```

NumPy - meshgrid

One other useful function in **NumPy** is meshgrid(), which generates all possible combinations between the input vectors,

```
pts = np.arange(3)
x, y = np.meshgrid(pts, pts)
x
## array([[0, 1, 2], [0, 1, 2], [0, 1, 2]])

y
## array([[0, 0, 0], [1, 1, 1], [2, 2, 2]])

np.sqrt(x**2 + y**2)

## array([[0.        , 1.        , 2.        ],
##        [1.        , 1.41421356, 2.23606798],
##        [2.        , 2.23606798, 2.82842712]])

```

NumPy - meshgrid: exercise

We will now use this to attempt a simple brute force approach to numerical optimization, define a grid of points using `meshgrid()` to approximate the minima of the following function:

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

Considering values of $x, y \in (-1, 3)$, which values of x, y minimize this function?

NumPy - Broadcasting

NumPy - Broadcasting: general broadcasting

When operating on two arrays, NumPy compares their shapes element-wise. It starts with the trailing (i.e. rightmost) dimensions and works its way left. Two dimensions are compatible when

- they are equal, or
- one of them is 1

If these conditions are not met, a ValueError: operands could not be broadcast together exception is thrown, indicating that the arrays have incompatible shapes.

```
x = np.arange(12).reshape((4,3))
x
## array([[ 0,  1,  2], [ 3,  4,
       5], [ 6,  7,  8], [ 9, 10,
      11]])

x + np.array([1,2,3])

```

```
x = np.arange(12).reshape((3,4))
x
## array([[ 0,  1,  2,  3], [ 4,
       5,  6,  7], [ 8,  9, 10,
     11]])

x + np.array([1,2,3])

```

NumPy - Broadcasting: mechanism

```
x = np.arange(12).reshape((4,3))
y = 1
x+y

x    (2d array): 4 x 3
y    (1d array):     1
----------------------
x+y  (2d array): 4 x 3


x = np.arange(12).reshape((4,3))
y = np.array([1,2,3])
x+y

x    (2d array): 4 x 3
y    (1d array):     3
----------------------
x+y  (2d array): 4 x 3
```

```
x = np.arange(12).reshape((3,4))
y = np.array([1,2,3])
x+y

x    (2d array): 3 x 4
y    (1d array):     3
----------------------
x+y  (2d array): Error


x = np.arange(12).reshape((3,4))
y = np.array([1,2,3]).reshape
    ((3,1))
x+y

x    (2d array): 3 x 4
y    (1d array): 3 x 1
----------------------
x+y  (2d array): 3 x 4
```

Please, check the official **NumPy** user guide - Broadcasting

NumPy - Broadcasting: example for data standardizing

Below we generate a data set with 3 columns of random normal values. Each column has a different mean and standard deviation which we can check with `mean()` and `std()`.

```
rng = np.random.default_rng(1234)
d = rng.normal(loc=[-1,0,1], scale=[1,2,3], size=(1000,3))
d.mean(axis=0)
## array([-1.0294382 , -0.01396257,  1.01241784])

d.std(axis=0)
## array([0.99674719, 2.03222595, 3.10625219])

```

Use broadcasting to standardize all three columns to have a mean of 0 and a standard deviation of 1.

Check the new data set using `mean()` and `std()`.

NumPy - Broadcasting: exercises

For each of the following combinations, determine what the resulting dimension will be:

- $A(128 \times 128 \times 3) + B(3)$
- $A(8 \times 1 \times 6 \times 1) + B(7 \times 1 \times 5)$
- $A(2 \times 1) + B(8 \times 4 \times 3)$
- $A(3 \times 1) + B(15 \times 3 \times 5)$
- $A(3) + B(4)$

NumPy - Basic file I/O

NumPy - Basic file I/O: reading and writing arrays

We will not spend much time on this as most data you will encounter is more likely to be in a tabular format (e.g. data frame), and tools like `Pandas` are more appropriate.

For basic saving and loading of `NumPy` arrays, there are the `save()` and `load()` functions, which use a built-in binary format.

```
    x = np.arange(1e5)
    np.save("data/x.npy", x)
    new_x = np.load("data/x.npy")
    np.all(x == new_x)

    ## True

```

Additional functions for saving (`savez()`, `savez_compressed()`, `savetxt()`) exist for saving multiple arrays or saving a text representation of an array.

If you need to read delimited (CSV, tsv, etc.) data into a `NumPy` array, you can use `genfromtxt()`.

Structuring an ML project

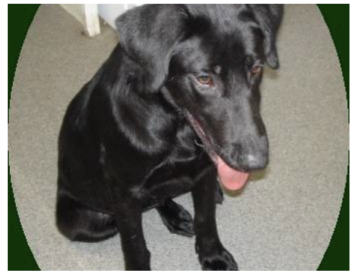
Structuring an ML project

1. Introduction to ML strategy
2. Settting your project goal
3. Comparing your model to human-level
4. Carrying out the error analysis
5. Mismatched training and dev/test sets
6. Homework 5

Why ML strategy?

Motivating example: cat classifier



Ideas

- Collect more data
- Collect a more diverse training set
- Train algorithm longer with GD
- Try Adam instead of GD
- Try dropout
- L_2 regulisation
- Change the network architecture
- Try bigger network
- Try smaller network

Setting up your goal

Using a single number evaluation metric

- Idea $\rightarrow$ Code $\rightarrow$ Experiments

Classifier	Precision	Recall	F1 Score
A	95%	90%	92.4%
B	98%	85%	91.0%

- With two evaluation metrics is difficult to choose which model performs better
- Dev set and a single number of evaluation metrics can speed up your iterative process

Example of single evaluation metrics

- Harmonic mean (F1 score)
- Geometric mean
- Median
- etc...

Setting up your goal

Satisficing and optimizing metrics

- Another cat classification example

Classifier	Accuracy	Running time
A	90%	80ms
B	92%	95ms
C	95%	1.5ms

- Cost = accuracy - 0.5$\times$ running time
- Maximize the accuracy subject to running time $\leq 100ms$: accuracy is an optimizing metric, and the running time is a satisficing metric.
- With N metrics: 1- optimizing and $N-1$ satisficing metrics

Another example: trigger words/ Wakewords

- Maximizing accuracy (optimizing metric)
- Number of false positives (satisficing metric ≤ 1)

Setting up your goal: training/dev/test data sets

Cat classification dev/test sets

- USA
- Germany
- China
- Cameroon
- Congo
- France
- Russia
- Others

Dev set:

- USA
- Germany
- China, Cameroon

Test set:

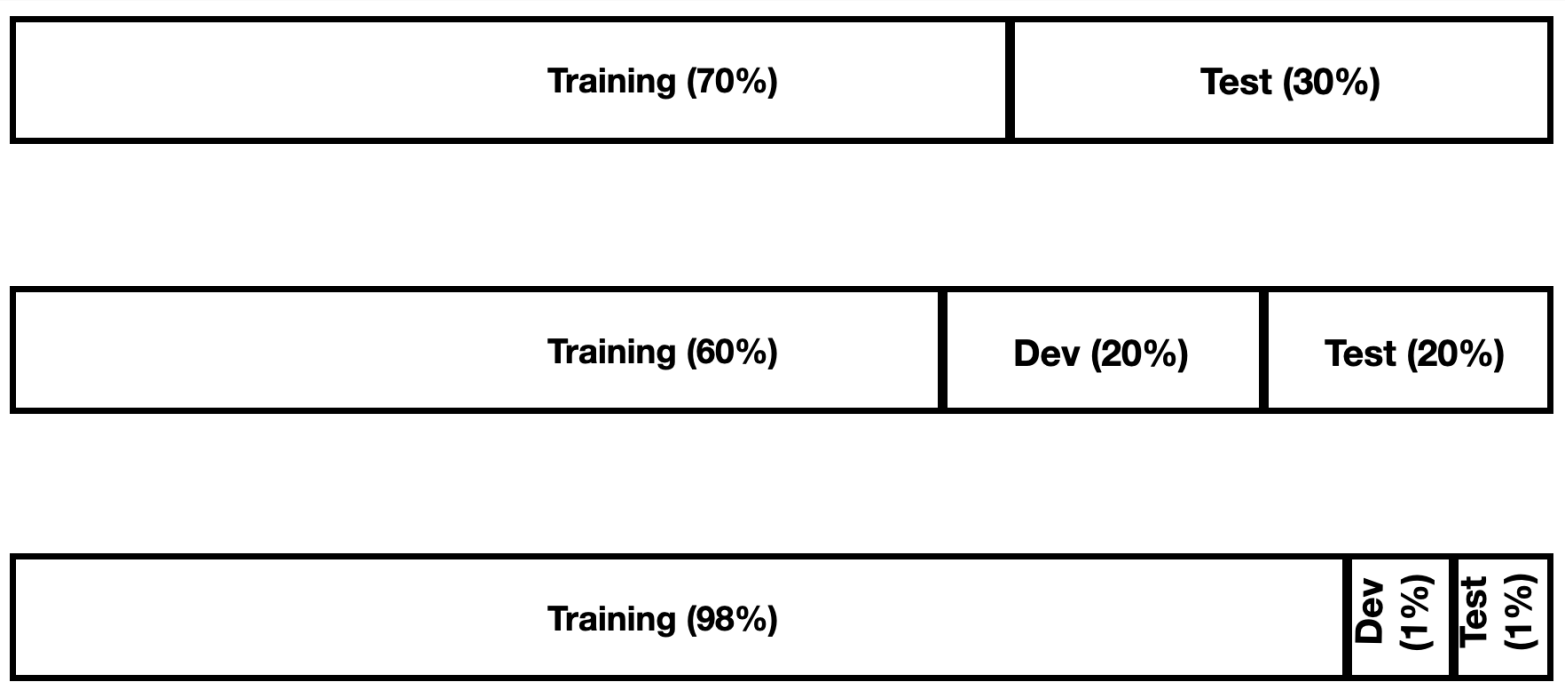
- Congo
- France
- Russia, Others

Recommandations

- Randomly shuffle into dev/test sets
- Both dev/test should have data that come from the same distributions

Setting up your goal: training/dev/test data sets

Old/new way of splitting data



Size of the test set

- Big enough to give high confidence in the overall performance of the model
- For some applications we could have no test set

Setting up your goal: training/dev/test data sets

When to change dev/test sets and metrics?

Let us say you have the following scenario:

- Metric: classification error $J(Y, \hat{Y}) = \frac{1}{m} \sum_{i=1}^{i=m} \mathcal{L}\{Y^{(i)} \neq \hat{Y}^{(i)}\}$
- Algorithm A: 3% error $\rightarrow$ but wrongly classifies pornographic pics
- Algorithm B: 5% error $\rightarrow$ but no porn picture

Model analysis

1. Metric + Dev: Algorithm A is the best
2. You/ users or production: Algorithm B is the best

In this case, what should we change? The dev/test sets or the metric?

Setting up your goal: training/dev/test data sets

Another example

- Metric: classification error $J(Y, \hat{Y}) = \frac{1}{m} \sum_{i=1}^{i=m} \mathcal{L}\{Y^{(i)} \neq \hat{Y}^{(i)}\}$
- Algorithm **A**: 3% error
- Algorithm **B**: 5% error

1. Metric + Dev



2. User images

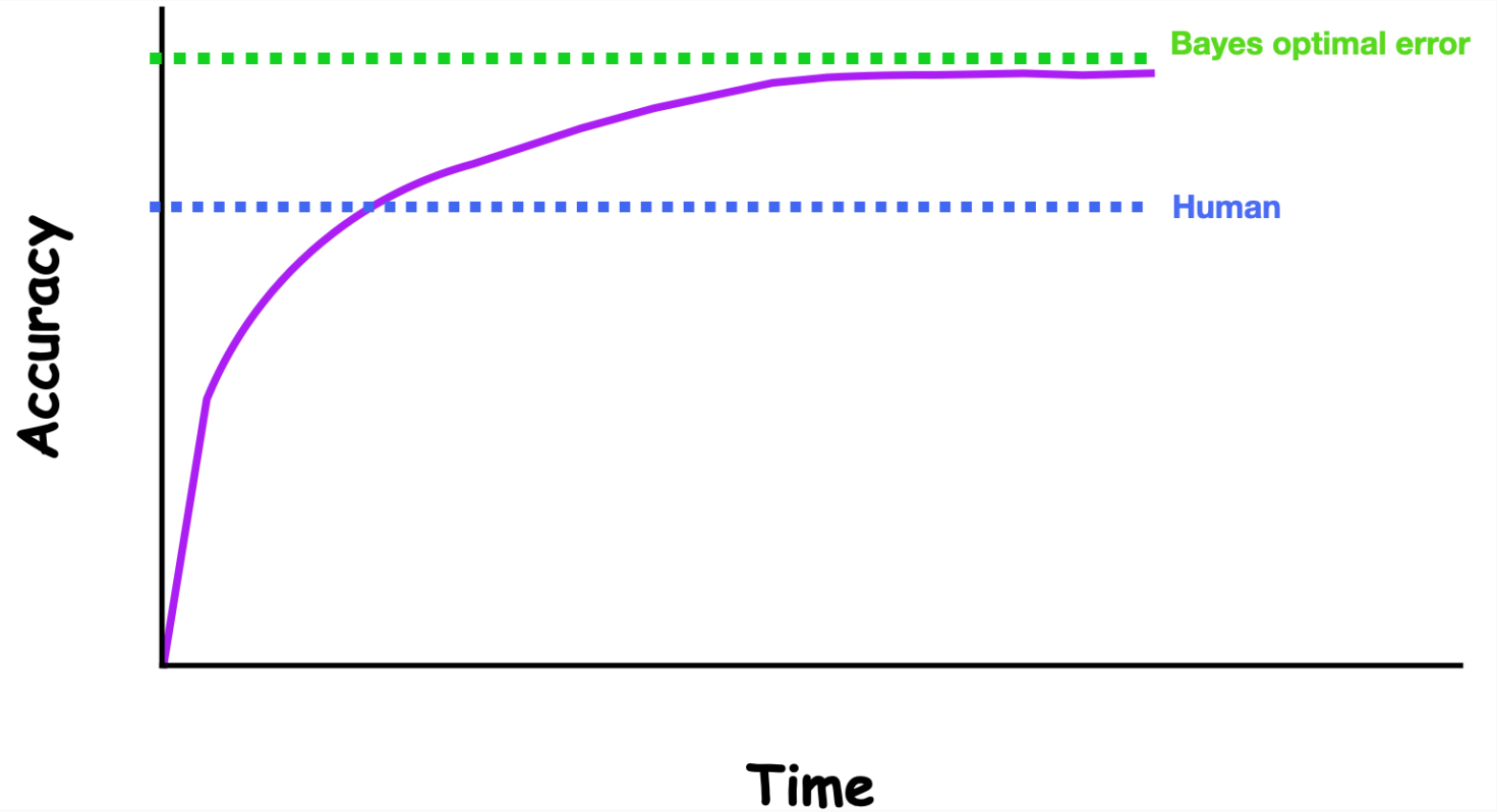


In this case, what should we change? The dev/test sets or the metric?

Comparing to human-level performance

Why human-level performance?

1. Advances in DL and ML algorithms with application in many areas
2. The workflow of designing and building an ML system is much more efficient for tasks that humans can also do



Comparing to human-level performance

Why comparing your model performance to human-level?

Humans are quite good at a lot of tasks. So long as ML is worse than humans, we can:

1. Get labelled data from humans
2. Gain insights from manual error analysis: Why did a person get this right?
3. Better analysis of bias/variance

Cat classification example

Human error ($\approx$ Bayes error)	1%	7.5%
Training error	8%	8%
Dev error	10%	10%

Focusing on bias or variance reduction techniques?

Understanding the human-level performance

Human-level as a proxy for Bayes error

Medical image classification example:

Let us suppose you have this:

1. Typical human ..... 3% error
2. Typical doctor ..... 1% error
3. Experienced doctor ..... 0.7% error
4. Team of experienced doctors ..... 0.5% error

What is the "human-level" error here?

	case 1	case 2	case 3
Human error ($\approx$ Bayes error)	1/0.7/0.5%	1/0.7/0.5%	1/0.7/0.5%
Training error	5%	1%	0.7%
Dev error	6%	5%	0.8%

Carrying out error analysis

When working on an ML project and your model does not achieve the human-level performance, you should:

1. Look at dev examples to evaluate ideas
2. Evaluate several ideas in parallel

Should you try to make your cat classifier do better on dogs?

- Get ≈ 100 mislabeled dev set example
- Count up how many dogs are

Error analysis

Evaluating multiple ideas in parallel

Ideas for our cat detection:

1. Fix pictures of dogs being recognized as cats
2. Fix great cats (lions, panthers, etc...) being misrecognized
3. Improve performance on blurry images

Images	Dog	Great cats	Blurry	Instagram	Comments
1	*Yes*	...	...	...	*Pitball*
2	...	...	*Yes*	*Yes*	...
3	*Yes*	...	*Yes*	...	Rainy day
⋮	⋮	⋮	⋮	⋮	⋮
Total %	8%	43%	6%	12%	...

Error analysis

Conclusion

1. Incorrectly labeled data *vs.* mislabeled data
2. Consider adding a column in your error analysis for incorrectly labelled
3. DL models are good at handling random incorrectly labelled in the training set

Correcting ”incorrect” dev/test set examples

- Apply the same process to your dev and test sets to make sure they continue to come from the same distribution
- Consider examining examples your algorithm got right as well as ones it got wrong
- Train and dev/test data may now come from slightly different distributions

Mismatched training and dev/test sets

Training and testing on different distributions

Let's consider our cat app example:

1. Data from the internet, e.g. from web pages: 200K images of high quality
2. Data from the mobile app: 10K images of average or, let us say, low quality

What are the options for the Training and dev/test data sets?

Two suggestions:

- Option 1: Shuffle the two datasets into one distribution

Training: 205K	**Dev**: 2.5K	**Test**: 2.5K

- Option 2: Take the dev/test from the mobile app data

Training: 205K	**Dev**: 2.5K	**Test**: 2.5K

Mismatched training and dev/test sets

Bias and variance with mismatched data distributions

Let's consider our cat app example and assume humans get $\approx 0\%$ error

1. Training error ..... 1% error
2. Dev error ..... 10% error

When applying the error analysis, what problem do we have here?

What to do?

- Define a training-dev set: same distribution as the training set, but not used for training

Training: 200K	**Training-dev**: 5K	**Dev**: 2.5K	**Test**: 2.5K

Mismatched training and dev/test sets

Addressing the data mismatch problem

1. Carry out manual error analysis to try to understand the difference between training and dev/test sets
2. Make training data more similar, or collect more data similar to dev/test sets

Remarks

- Artificial data synthesis: you could generate more data. e.g. Speech recognition task
- Problem with overfitting to a single noise: could be better with more noise types
- The synthesized data could be less representative than all audio with random noise