Applied Deep Learning

Chapter 3: Advancing Our Toolbox

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

University of Toronto

Fall 2025

Data Preparation

Frankly speaking, preparing data for training and testing is

most time-consuming part of a practical project

+ *How hard it could be? Really harder than finding right hyperparameters, regularizing or adjusting the optimizer?*
– Sure! We get used to such *design tasks* and start to have feeling about NNs, as they repeat so much. *The main thing that is new is data*

How to prepare data that *works well for our purpose* is an individual topic discussed in *courses on data science*: we only briefly touch it in this section

Data Preparation

Procedure of processing raw data into a form suitable for underlying model

Data Preprocessing Procedures

There is a long list of techniques for data preparation

- *Data augmentation which we do when training dataset is too small*
- *Data cleaning that we do to either remove or modify unwanted samples in training dataset*
  - ↳ Samples like outliers, duplicates and nulls
- *Data transform which aims to transform data into a form that lead to more robust training*
- *Dimensionality reduction which reduces the redundancy by extracting lower-dimensional features with minimal confusion from samples*
- ...

We discuss the first two items in this section: *let's start with the first one that we already have some idea about*

Expanding Dataset via *Augmentation*

Recall that one conclusion from overfitting was that dataset is too small

In practice, we may expanding our dataset by *augmenting* it

+ *Say we have a set of cat and dog images! How can we expand it?*
– Well! Let's take a look at this simple example!

Say we have a dataset of cat and dog N-pixel images. We write it as

$$\mathbb{D} = \{(\boldsymbol{x}_b, v_b) : b = 1, \dots, B\} \qquad v_b = 0 : \text{cat} \qquad v_b = 1 : \text{dog}$$

Say $\boldsymbol{x}_1$ is pixel-vector of a cat image. We are given function $\mathcal{A} : \mathbb{R}^N \mapsto \mathbb{R}^N$: it gets $\boldsymbol{x}_1$ and returns $\hat{\boldsymbol{x}} = \mathcal{A}(\boldsymbol{x}_1)$ which satisfies two conditions

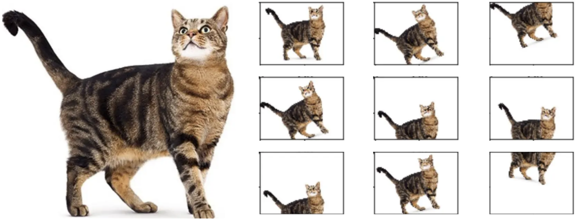
1. *After plotting $\hat{\boldsymbol{x}}$ we still see a cat*
2. *This new cat image does not belong to the dataset, i.e., $(\hat{\boldsymbol{x}}, 0) \notin \mathbb{D}$*

We can then expand our dataset as $\mathbb{D} \leftarrow \{(\hat{\boldsymbol{x}}, 0)\} \cup \mathbb{D}$!

Data Augmentation: *Example*

+ *But, how could we know such a function $\mathcal{A}(\cdot)$?*
– For images we know some!

We can simply *rotate*, *shift*, *zoom-in*, *zoom-out*, *change intensity and so on!*



How can we rotate an image? Multiply it by a rotation matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$

These are examples of data augmentation *by engineering*

we can find $\mathcal{A}(\cdot)$ analytically using properties of our data

Data Augmentation by *Engineering*

Data augmentation by *engineering depends* on *data* and *learning task*

- If we are classifying a set of images
  ↳ *We can apply geometric transformations, e.g., random rotating, flipping, stretching, zooming, and cropping*
  ↳ *We may use kernel filters* to make *random* filtering, *e.g., changing sharpness or blurring*
  ↳ *We can apply random color-space transformations, e.g., changing intensity or brightness, and exchanging RGB channels*
  ↳ *We can randomly remove pixels, i.e., set their value to some reference value*
  ↳ *We can randomly mix images, e.g., get multiple cat images and make a new one out of them*

Note that *we may train a separate NN to do any of those transforms for us: just think about the last one!*

Data Augmentation by *Engineering*

Data augmentation by engineering depends on data and learning task

- If we are working with audio signals
  - ↳ *We can apply noise injection, e.g., add background noise*
  - ↳ *We may change sampling rate to change the speed of audio data*
  - ↳ *We can apply random shifts, e.g., sample signals at a bit deviated points*
- If we are dealing with text data
  - ↳ *We may apply random replacements, e.g., replace a word with its synonyms*
  - ↳ *We may manipulate syntax-tree, e.g., rephrase a sentence*
  - ↳ *We can do random shuffling is some applications*
  - ↳ *We may apply removal and insertion of redundant words, e.g., so*

Synthetic Data Generation

An alternative approach to expand a small dataset is to

generate synthetic data

We did this in Assignment 1 for the dummy projectile example

> *Recall we had a projectile with velocity v and height h: we knew by Newton's laws that the hitting distance d is given by*
>
> $$d = 0.45 v \sqrt{h}$$
>
> *To make dataset, we generated lots of velocities v_i and heights h_i at random and for each pair we determined d_i by above equation*

Synthetic Data vs Augmented Data

When we generate synthetic data

- *we need to know the process of data being generated from a seed*
- *we make new data by simulating the process with a random seed*

When we augment data

- *we need to know transforms that keep data-points inside dataset*
- *we apply those transforms on the existing data*

\+ *It sounds that synthetic data is only feasible in scientific problems, where we know physics! Right?!*

– Until few years ago the answer was Yes! But, currently No! Nowadays, *we can generate images of what we want from noise using generative adversarial networks (GANs) or diffusion models!*

Data Augmentation: *Formulation*

$\boxed{\mathcal{A}(\cdot) \text{ gets data-point } \boldsymbol{x} \text{ and returns new data-point } \hat{\boldsymbol{x}} = \mathcal{A}(\boldsymbol{x})}$

In general, we do not need $\mathcal{A}(\cdot)$ operate on single data-point

$\mathcal{A}(\cdot)$ can get multiple samples and generate a new one

For instance, *it combines multiple cat images and makes a new one*

Also, $\mathcal{A}(\cdot)$ is not enforced to return data-points with same labels

label of what $\mathcal{A}(\cdot)$ returns is only required to be a valid label

For instance, *it gets multiple cat images and makes a new dog image; however, if our dataset includes only cats and dogs it should not return a horse image*

We can now formulate data augmentation more precisely

Augmentation and Synthetic Generation: *Formulation*

Data Augmentation

A data augmentation technique $\mathcal{A}(\cdot)$ takes the training dataset as the input and returns new samples from data distribution

We can think of it as the following block

$$\text{training dataset } \mathbb{D} \rightsquigarrow \boxed{\mathcal{A}(\cdot)} \rightsquigarrow (\boldsymbol{x}_{\mathrm{new},1}, v_{\mathrm{new},1}), (\boldsymbol{x}_{\mathrm{new},2}, v_{\mathrm{new},2}), \ldots$$

Synthetic Data Generation

A synthetic data generator $\mathcal{S}(\cdot)$ takes a random seed as the input and returns samples from data distribution

We can think of it as the following block

$$\text{random seed } \mathrm{s} \rightsquigarrow \boxed{\mathcal{S}(\cdot)} \rightsquigarrow (\boldsymbol{x}_1, v_1), (\boldsymbol{x}_2, v_2), \ldots$$

Augmentation and Synthetic Generation: *Formulation*

+ *There is one point left bothering in definitions!*

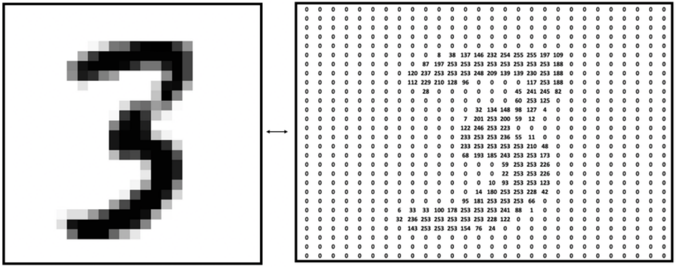
  What does data distribution mean?

– In simple words it means an *abstract machine that generates only data-points that we need*

+ *But, why we call it distribution*

– Well! Let's get it clear!

Possible Samples of Data



Let's start with MNIST dataset: *in MNIST we have 28 $\times$ 28 pixel images*

- *They are 8-bit images: each pixel value is an integer between 0 and 255*
- *These images are all hand-written numbers*

But, we note that they are

- *neither all* 28 $\times$ 28 pixel *8-bit images*
- *nor all possible* images of *hand-written numbers*

Possible Samples of Data

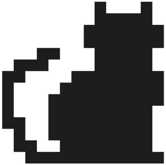
MNIST images are not all 28×28 pixel 8-bit images

A 28×28 pixel image has in total 784 pixels with each pixel being one of 256 different possible values values: in total we have

$$\textit{total number of images} = 256^{784} = 2^{6272} > 10^{1881}$$

MNIST has only $70,000 < 10^5$ images!

We also note that *not all of those 2^{6272} can get into MNIST!* For instance,



Possible Samples of Data

MNIST images are not all possible images of hand-written numbers

We can imagine that *among those* 2^{6272} *images there are much more than only* $70,000$ *images* of *hand-written numbers: just take an image of my handwriting and convert it into a 28* $\times$ *28 pixel image!*

Space of Possible Data *(Data Space)*

Space of possible data is *the set of all labeled data-points* whose *labels* are *valid*

In our example, *the space of possible data* is

$$\mathbb{X} = \left\{ \boldsymbol{x} \in \{0, \dots, 255\}^{784} : \text{image of } \boldsymbol{x} \text{ is classified as hand-written number} \right\}$$

Of course space of possible data is *only a definition: most of the time, it is impossible to specify it explicitly* like *above example*

Data-Point as *Sample of Random Object*

We obviously see that a dataset is a subset of data space: *MNIST is a subset of $\mathbb{X}$ defined in the last slide and it is much much smaller*

In machine learning, we have a specific way to look at datasets

dataset is collection of samples drawn randomly from the data space

This can be easily understood as the example below

Recall the data space $\mathbb{X}$ defined in last slide

- *We assume that there exist a machine with a button*
  - ↳ *each time we push this button the machine randomly generates data-point x from $\mathbb{X}$*

MNIST is then generated by pushing this button for $70,000$ times

Data Distribution

Data Distribution

Data distribution is the probability distribution by which the dataset has been generated from the data space

+ *Do we know this distribution?!*
– No! We can *neither fully specify the space of possible data nor the data distribution!* They are mainly abstract definitions!

But, we can have a partial understanding: *assume I say such a sentence*

> *"MNIST contains 70,000 samples drawn from data distribution* $p(\boldsymbol{x})$*"*

We cannot find out what $p(\boldsymbol{x})$ *is, but we know for sure*

 $\equiv \boldsymbol{x}_1 \rightsquigarrow p(\boldsymbol{x}_1) \neq 0$

 $\equiv \boldsymbol{x}_2 \rightsquigarrow p(\boldsymbol{x}_2) = 0$

Data Distribution

From now on, if we get into such a sentence in a paper

> the learner has access to a *small reference dataset* $S_T := \{(x_{T,1}, y_{T,1}), \ldots, (x_{T,m_T}, y_{T,m_T})\}$ of m_T samples drawn i.i.d. from a target distribution $\mathcal{D}_T$

we know what it means!

Last note: although we do not have access to data distribution

we can in practice approximate it from samples that we have collected

For instance, if data-points are heights of different people: we can plot the histogram to approximate data distribution

Data Distribution: *Practical Aspects*

In practice, this looking at data-points as samples of a random process lets us *use statistical methods to preprocess data*

We can use these methods to

- *realize if our dataset is a good representative of data space*
  ↳ *Maybe we have too many non-typical data-points*
- *transform our dataset into a better representative of data space*
  ↳ *Maybe we should add, remove or change some data-points*

This is what we call *data cleaning*: *this can be a separate course! So, we make it very short by discussing only few practical techniques*

Data Cleaning: *Duplicates*

Duplication impacts model training: *assume we have training dataset*

$$\mathbb{D} = \{(\boldsymbol{x}_b, v_b) : b = 1, \ldots, B\}$$

Without any duplication, our training loop solves

$$\min_{\mathbf{w}} \frac{1}{B} \sum_{b=1}^{B} \mathcal{L}\left(y_b, v_b\right) : y_b \text{ output of NN with weights } \mathbf{w} \text{ and input } \boldsymbol{x}_b$$

Now assume that we copy $(\boldsymbol{x}_1, v_1)$ *by mistake* M *times: the training on this duplicated dataset is*

$$\min_{\mathbf{w}} \frac{1}{B+M} \sum_{b=1}^{B} \mathcal{L}\left(y_b, v_b\right) + \frac{M}{B+M} \mathcal{L}\left(y_1, v_1\right)$$

which is not the same thing!

Data Cleaning: *Duplicates*

Having the same data-points in dataset does not necessarily mean duplication: *consider the following two simple examples*

1. *We are training an NN that takes age, education and place of birth as input and returns number of children as output*
   - ↳ Dataset has too many $x_b =$ [22, Bachelor, Toronto] and $v_b =$ 0
   - ↳ These are not duplicates since they come from independent samples
2. *We are training an NN that takes age and height as input and returns weight as output*
   - ↳ It is not likely to have multiple $x_b = [48, 176.42]$ and $v_b = 73.31$
   - ↳ They most probably come from the same person and are duplicates

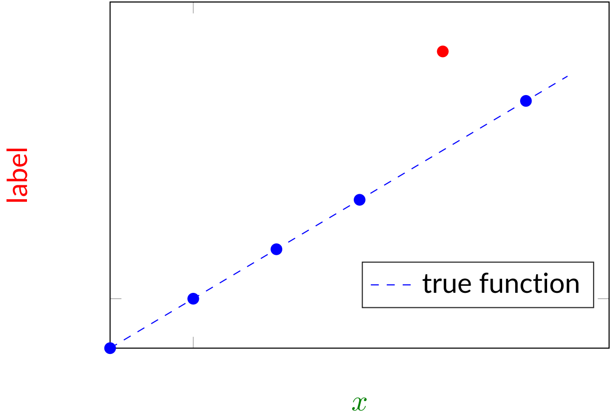
---

Bingo! *Terms like "come from independent samples" and "not likely to have" are used since we look at data-points as samples of a random process*

Data Cleaning: *Outliers*

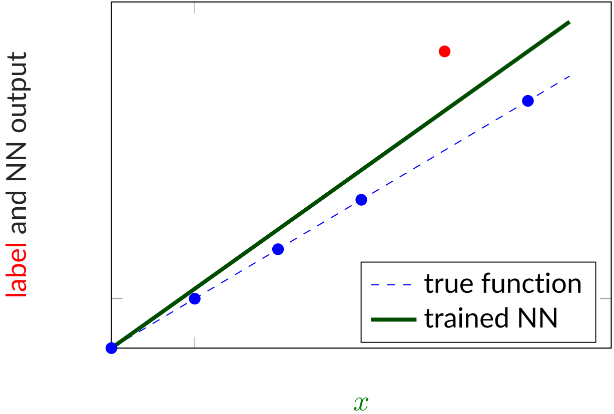
Outliers

Outliers are data-points that lie in an abnormal distance from other samples



Data Cleaning: *Outliers*

Outliers can hinder training even with good tuning and regularization



Data Cleaning: *Outliers*

Finding outliers is required for training of a model that generalizes well

---

There are two types of outliers is a dataset

1. *Univariate outliers which are detected from their marginal distributions*
   - ↳ *These data-points are understood to be outliers from an individual variable in them without comparing to other variables*

   We collect heights and weights: Sultan Kösen[a] *is among our samples with height 2.51 m! Without checking weights, we can say this is an outlier*

   [a] Tallest alive person in the world

Data Cleaning: *Outliers*

Finding outliers is required for training of a model that generalizes well

---

There are two types of outliers is a dataset

2. *Multivatiate outliers which are detected from their joint distributions*
   - ↳ *These are understood to be outliers by comparing different variables*

We collect heights and weights: a sample with height 1.82 m and another with weight 24 kg are individually normal; however a sample with height 1.82 m and weight 24 kg is an outlier

Data Cleaning: *Outliers*

+ *How can we handle outliers?*
– Well! It depends

Conventional approaches to handle outliers are to

1. *remove them from training dataset*
   ↳ *It might be a good idea for small NNs with low model capacity*
   ↳ *It can hinder generalization of our model if we have detected outliers based on poor statistics, e.g., not enough samples to understand data distribution*
2. *use loss functions that are robust against outliers*
   ↳ *They give higher weights to typical data-points*
   ↳ *They give less weights to outliers*
   ↳ *An example is to use Minkowski error instead of squared error for regression*

Take a look at Python library `Pandas` if you need to do any data cleaning