AMMI Deep Learning DIY:

Day 1

Alexandre Sablayrolles, Pierre Stock

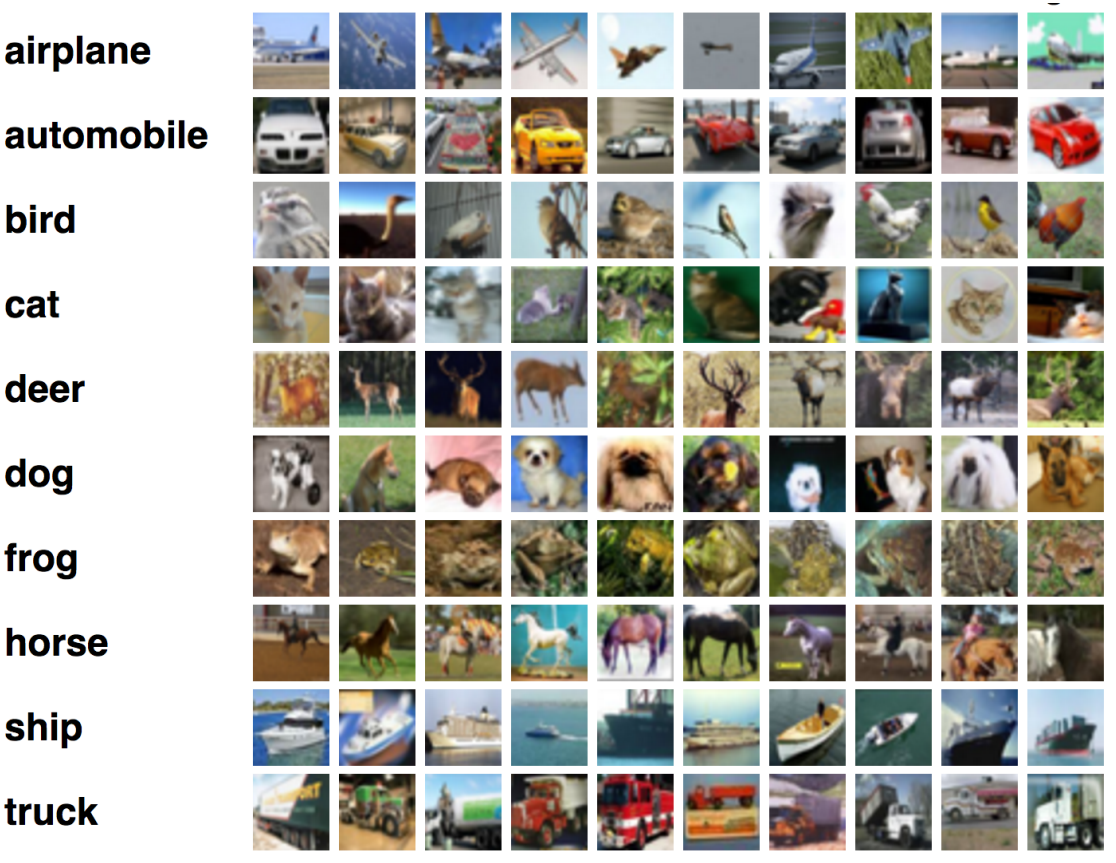
Deep Learning DIY: Objectives

Given an idea or a project, be able to implement it in a robust way

Deep Learning DIY: Objectives

Given an idea or a project, be able to implement it in a robust way

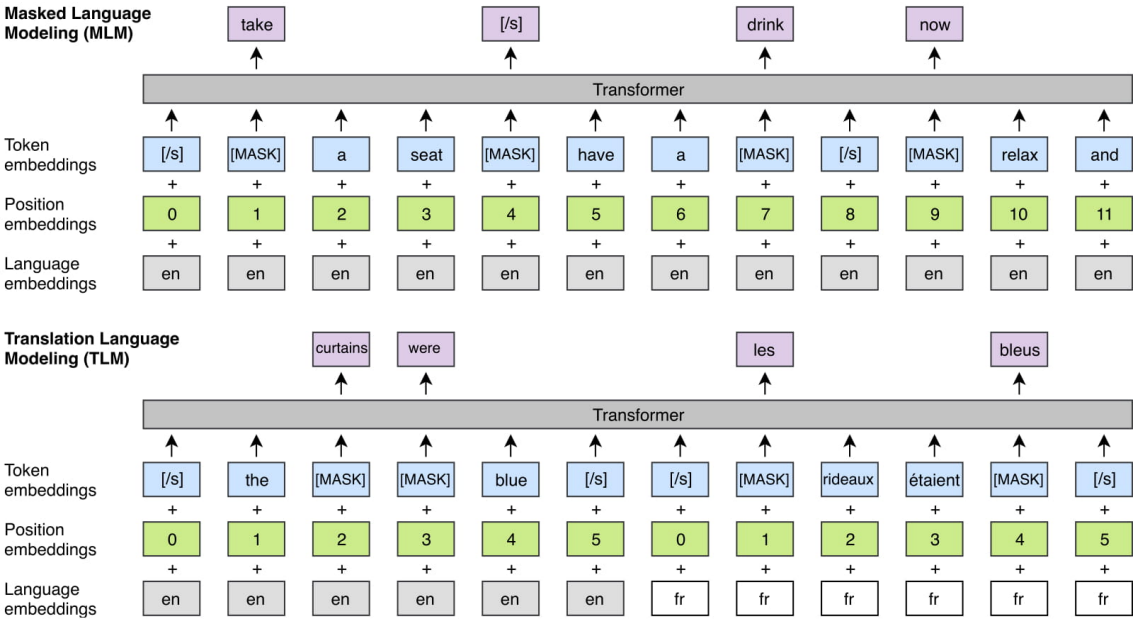
- Ex: "Given a set of images, can I find the best classifier ?"



Deep Learning DIY: Objectives

Given an idea or a project, be able to implement it in a robust way

- Ex: "With a conversation dataset, how can I create a chatbot ? How can I build a translation model ?"



Deep Learning DIY: Objectives

Given an idea or a project, be able to implement it in a robust way

- Ex: "I don't know anything about speech recognition. Can I read a couple of papers and get started on a model ?"

Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, Jie Chen, Jingdong Chen, Zhijie Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Ke Ding, Niandong Du, Erich Elsen, Jesse Engel, Weiwei Fang, Linxi Fan, Christopher Fougner, Liang Gao, Caixia Gong, Awni Hannun, Tony Han, Lappi Vaino Johannes, Bing Jiang, Cai Ju, Billy Jun, Patrick LeGresley, Libby Lin, Junjie Liu, Yang Liu, Weigao Li, Xiangang Li, Dongpeng Ma, Sharan Narang, Andrew Ng, Sherjil Ozair, Yiping Peng, Ryan Prenger, Sheng Qian, Zongfeng Quan, Jonathan Raiman, Vinay Rao, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Kavya Srinet, Anuroop Sriram, Haiyuan Tang, Liliang Tang, Chong Wang, Jidong Wang, Kaifu Wang, Yi Wang, Zhijian Wang, Zhiqian Wang, Shuang Wu, Likai Wei, Bo Xiao, Wen Xie, Yan Xie, Dani Yogatama, Bin Yuan, Jun Zhan, Zhenyao Zhu

Baidu Silicon Valley AI Lab[1], 1195 Bordeaux Avenue, Sunnyvale CA 94086 USA

Baidu Speech Technology Group, No. 10 Xibeiwang East Street, Ke Ji Yuan, Haidian District, Beijing 100193 CHINA

Abstract

We show that an end-to-end deep learning approach can be used to recognize either English or Mandarin Chinese speech–two vastly different languages. Because it replaces entire pipelines of hand-engineered components with neural networks, end-to-end learning allows us to handle a diverse variety of speech including noisy environments, accents and different languages. Key to our approach is our application of HPC techniques, enabling experiments that previously took weeks to now run in days. This allows us to iterate more quickly to identify superior architectures and algorithms. As a result, in several cases, our system is competitive with the transcription of human workers when benchmarked on standard datasets. Finally, using a technique called Batch Dispatch with GPUs in the data center, we show that our system can be inexpensively deployed in an online setting, delivering low latency when serving users at scale.

1. Introduction

Decades worth of hand-engineered domain knowledge has gone into current state-of-the-art automatic speech recognition (ASR) pipelines. A simple but powerful alternative solution is to train such ASR models end-to-end, using deep learning to replace most modules with a single model as in (Hannun et al., 2014a) and (Graves & Jaitly, 2014b). This "end to end" vision of training simplyfies the training process by removing the engineering required for the bootstrapping/alignment/clustering/HMM machinery often used to build state-of-the-art ASR models. On such a system, built on end-to-end deep learning, we can employ a range of deep learning techniques: capturing large training sets, training larger models with high performance computing, and methodically exploring the space of neural network architectures.

This paper details our contribution to the model architecture, large labeled training datasets, and computational scale for speech recognition. This includes an extensive investigation of model architectures, and our data capturing pipeline that has enabled us to create larger datasets than what is typically used to train speech recognition systems.

We benchmark our system on several publicly available test sets with a goal of eventually attaining human-level performance. To that end, we have also measured the performance of crowd workers on each benchmark for comparison. We find that our best Mandarin Chinese speech system transcribes short voice-query like utterances better than a typical Mandarin Chinese speaker.

The remainder of the paper is as follows. We begin with a review of related work in deep learning, end-to-end speech recognition, and scalability in Section 2. Section 3 describes the architectural and algorithmic improvements to the model and Section 4 explains how to efficiently compute them. We discuss the training data and steps taken to further augment the training set in Section 5. An analysis of results for our system in English and Mandarin is presented in Section 6. We end with a description of the steps needed to deploy our system to real users in Section 7.

[1] Contact author: sanjeevsatheesh@baidu.com

Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

Deep Learning DIY: Objectives

Given an idea or a project, be able to implement it in a robust way

- Write code from scratch

Deep Learning DIY: Objectives

Given an idea or a project, be able to implement it in a robust way

- Write code from scratch
  - Write a training loop, a dataloader, etc.

Deep Learning DIY: Objectives

Given an idea or a project, be able to implement it in a robust way

- Write code from scratch
  - Write a training loop, a dataloader, etc.
  - Already know what to do (generic machine learning code) when the project starts

Deep Learning DIY: Objectives

Given an idea or a project, be able to implement it in a robust way

- Write code from scratch
- Debug the project, from code to machine learning assumptions

Deep Learning DIY: Objectives

Given an idea or a project, be able to implement it in a robust way

- Write code from scratch
- Debug the project, from code to machine learning assumptions
  - did I use the right parameters (learning rate, optimizer, etc.) ?

Deep Learning DIY: Objectives

Given an idea or a project, be able to implement it in a robust way

- Write code from scratch
- Debug the project, from code to machine learning assumptions
  - did I use the right parameters (learning rate, optimizer, etc.) ?
  - am I sure that my evaluation is correct ?

Deep Learning DIY: Objectives

Given an idea or a project, be able to implement it in a robust way

- Write code from scratch
- Debug the project, from code to machine learning assumptions
  - did I use the right parameters (learning rate, optimizer, etc.) ?
  - am I sure that my evaluation is correct ?
  - are the training and test set different ?

Deep Learning DIY: Objectives

Given an idea or a project, be able to implement it in a robust way

- Write code from scratch
- Debug the project, from code to machine learning assumptions
  - did I use the right parameters (learning rate, optimizer, etc.) ?
  - am I sure that my evaluation is correct ?
  - are the training and test set different ?
  - my model does not work on this data, does it work on other data ?

Deep Learning DIY: Objectives

Given an idea or a project, be able to implement it in a robust way

- Write code from scratch
- Debug the project, from code to machine learning assumptions
- Top-down approach to understanding

Deep Learning DIY: Objectives

Given an idea or a project, be able to implement it in a robust way

- Write code from scratch
- Debug the project, from code to machine learning assumptions
- Top-down approach to understanding
  - there are a lot of moving parts, but we nail them down progressively

Deep Learning DIY: Objectives

Given an idea or a project, be able to implement it in a robust way

- Write code from scratch
- Debug the project, from code to machine learning assumptions
- Top-down approach to understanding

Teachers

Pierre Stock

2nd year PhD Student at Facebook AI Research.

Working on model compression, computer vision, deep learning.

Teachers

Timothée Lacroix

Finishing PhD Student at Facebook AI Research.

Working on learning embeddings for large-scale datasets, tensor factorization, etc.

Teachers

Alexandre Sablayrolles

3rd year PhD Student at Facebook AI Research.

Working on fast nearest neighbor search and differential privacy.

Schedule (Tentative)

First week

Schedule (Tentative)

First week

- Monday: Logistic regression & the basics of debugging

Schedule (Tentative)

First week

- Monday: Logistic regression & the basics of debugging
- Tuesday: Homework (debugging, model & parameter search)

Schedule (Tentative)

First week

- Monday: Logistic regression & the basics of debugging
- Tuesday: Homework (debugging, model & parameter search)
- Wednesday: Data visualization (PCA, t-SNE)

Schedule (Tentative)

First week

- Monday: Logistic regression & the basics of debugging
- Tuesday: Homework (debugging, model & parameter search)
- Wednesday: Data visualization (PCA, t-SNE)
- Thursday: Homework (Data visualization)

Schedule (Tentative)

First week

- Monday: Logistic regression & the basics of debugging
- Tuesday: Homework (debugging, model & parameter search)
- Wednesday: Data visualization (PCA, t-SNE)
- Thursday: Homework (Data visualization)
- Friday: Convnets, vision

Schedule (Tentative)

First week

- Monday: Logistic regression & the basics of debugging
- Tuesday: Homework (debugging, model & parameter search)
- Wednesday: Data visualization (PCA, t-SNE)
- Thursday: Homework (Data visualization)
- Friday: Convnets, vision

Second week

Schedule (Tentative)

First week

- Monday: Logistic regression & the basics of debugging
- Tuesday: Homework (debugging, model & parameter search)
- Wednesday: Data visualization (PCA, t-SNE)
- Thursday: Homework (Data visualization)
- Friday: Convnets, vision

Second week

- Monday: Natural Language Processing

Schedule (Tentative)

First week

- Monday: Logistic regression & the basics of debugging
- Tuesday: Homework (debugging, model & parameter search)
- Wednesday: Data visualization (PCA, t-SNE)
- Thursday: Homework (Data visualization)
- Friday: Convnets, vision

Second week

- Monday: Natural Language Processing
- Tuesday: Homework (NLP)

Schedule (Tentative)

First week

- Monday: Logistic regression & the basics of debugging
- Tuesday: Homework (debugging, model & parameter search)
- Wednesday: Data visualization (PCA, t-SNE)
- Thursday: Homework (Data visualization)
- Friday: Convnets, vision

Second week

- Monday: Natural Language Processing
- Tuesday: Homework (NLP)
- Wednesday: Reinforcement Learning

Schedule (Tentative)

First week

- Monday: Logistic regression & the basics of debugging
- Tuesday: Homework (debugging, model & parameter search)
- Wednesday: Data visualization (PCA, t-SNE)
- Thursday: Homework (Data visualization)
- Friday: Convnets, vision

Second week

- Monday: Natural Language Processing
- Tuesday: Homework (NLP)
- Wednesday: Reinforcement Learning
- Thursday: Homework (project)

Schedule (Tentative)

First week

- Monday: Logistic regression & the basics of debugging
- Tuesday: Homework (debugging, model & parameter search)
- Wednesday: Data visualization (PCA, t-SNE)
- Thursday: Homework (Data visualization)
- Friday: Convnets, vision

Second week

- Monday: Natural Language Processing
- Tuesday: Homework (NLP)
- Wednesday: Reinforcement Learning
- Thursday: Homework (project)
- Friday: Homework & grading

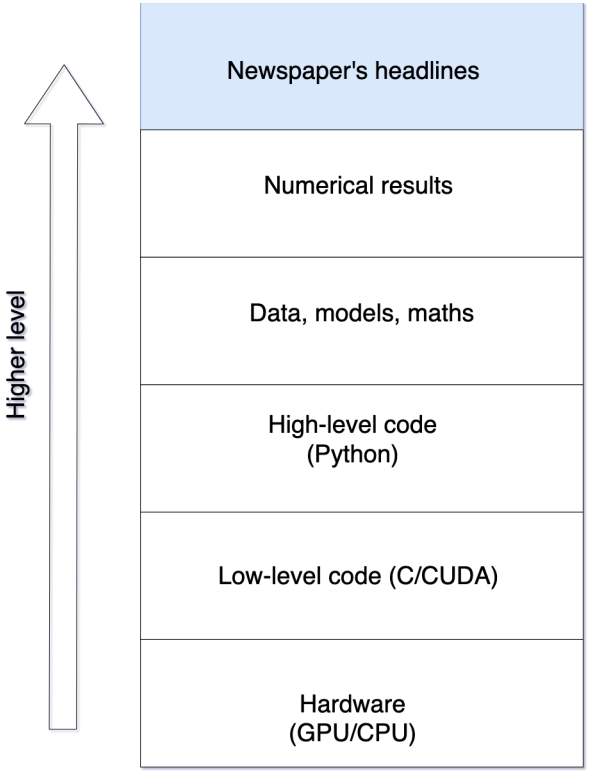
Outline

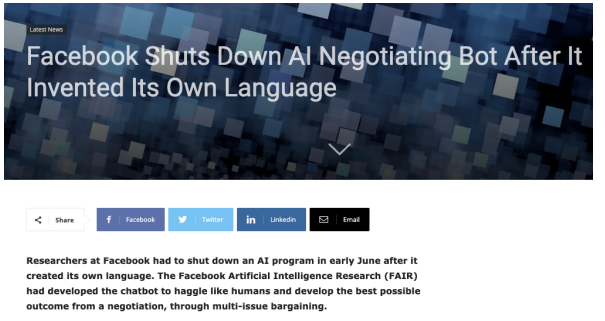
1. The machine learning stack
2. Brief recap: gaussian random variables
3. Logistic regression and linear classification
4. How to debug a machine learning code?

Outline

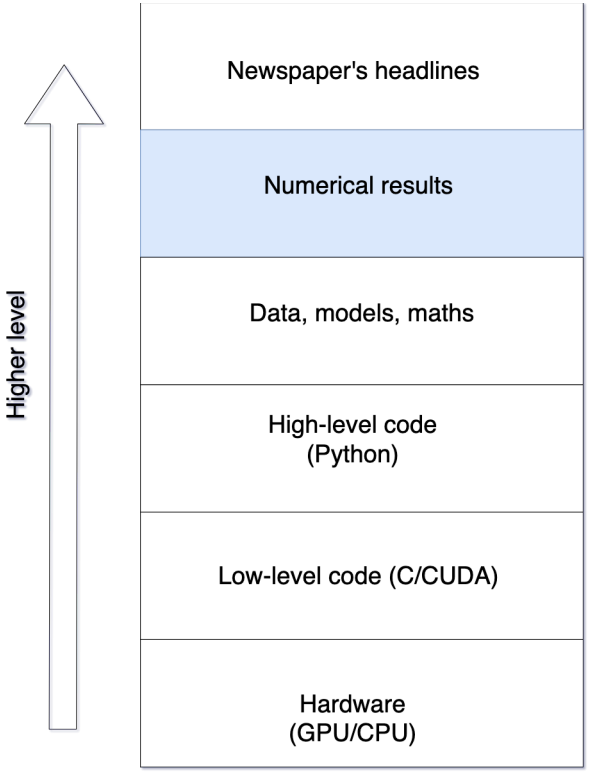
1. The machine learning stack
2. Brief recap: gaussian random variables
3. Logistic regression and linear classification
4. How to debug a machine learning code?

The various levels of machine learning...



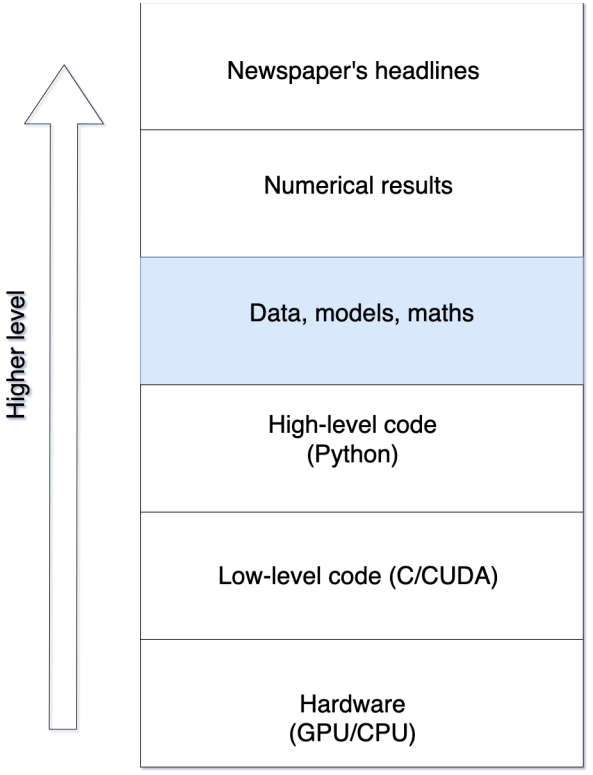


The various levels of machine learning...



Model	vs. LIKELIHOOD				vs. Human			
	Score (all)	Score (agreed)	% Agreed	% Pareto Optimal	Score (all)	Score (agreed)	% Agreed	% Pareto Optimal
LIKELIHOOD	5.4 vs. 5.5	6.2 vs. 6.2	87.9	49.6	4.7 vs. 5.8	6.2 vs. 7.6	**76.5**	66.2
RL	7.1 vs. 4.2	7.9 vs. 4.7	89.9	58.6	4.3 vs. 5.0	6.4 vs. 7.5	67.3	69.1
ROLLOUTS	7.3 vs. 5.1	7.9 vs. 5.5	92.9	63.7	**5.2 vs. 5.4**	7.1 vs. 7.4	72.1	78.3
RL+ROLLOUTS	**8.3 vs. 4.2**	**8.8 vs. 4.5**	**94.4**	**74.8**	4.6 vs. 4.2	**8.0 vs. 7.1**	57.2	**82.4**

The various levels of machine learning...



$$R(x_{n..n+k}) = \mathbb{E}_{x_{(n+k+1..T;o)} \sim p_\theta}[r(o)p_\theta(o|x_{0..T})] \tag{16}$$

We then return the utterance maximizing R.

$$u^* = \underset{u \in U}{\operatorname{argmax}}\, R(u) \tag{17}$$

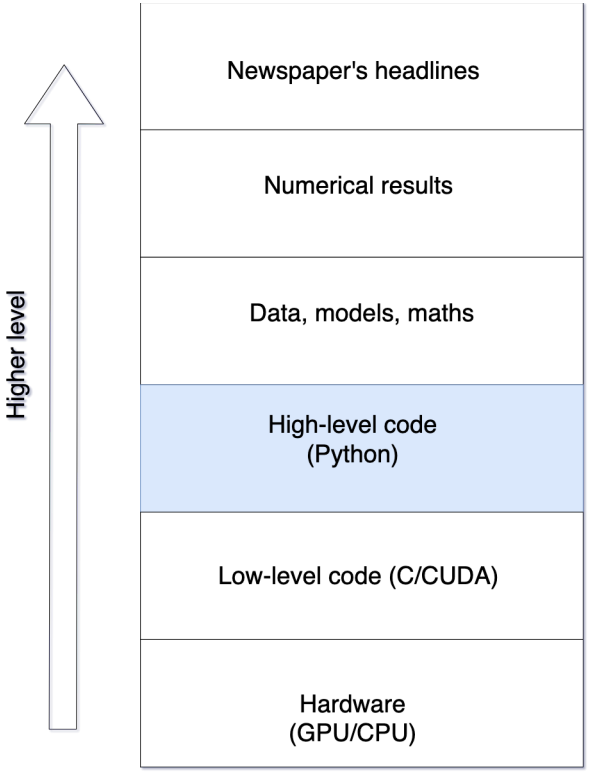
We use 5 rollouts for each of 10 candidate turns.

6 Experiments

6.1 Training Details

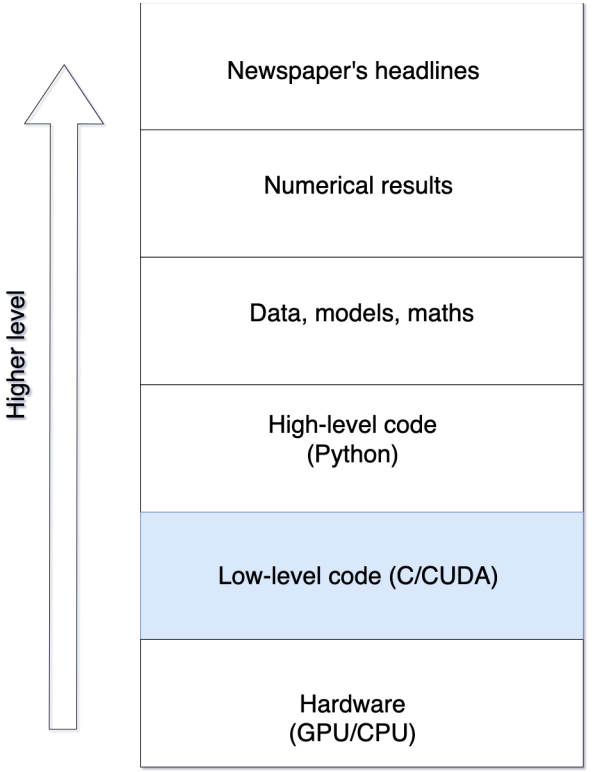
We implement our models using PyTorch. All hyper-parameters were chosen on a development dataset. The input tokens are embedded into a 64-dimensional space, while the dialogue tokens are embedded with 256-dimensional embeddings (with no pre-training). The input GRU_g has a hidden layer of size 64 and the dialogue GRU_w is of size 128. The output $\text{GRU}_{\overrightarrow{o}}$ and $\text{GRU}_{\overleftarrow{o}}$ both have a hidden state of size 256, the size of h^s is 256 as well. During supervised training, we optimise using stochastic gradient descent with a minibatch size of 16, an initial learning rate of 1.0, Nesterov momentum with μ=0.1 (Nesterov, 1983), and clipping gradients whose L^2 norm exceeds 0.5. We train the model for 30 epochs and pick the snapshot of the model with the best validation perplexity. We then annealed the learning rate by a factor of 5 each epoch. We weight the terms in the loss function (Equation 10) using α=0.5. We do not train against output decisions where humans selected different agreements. Tokens occurring fewer than 20 times are replaced with an 'unknown' token.

The various levels of machine learning...



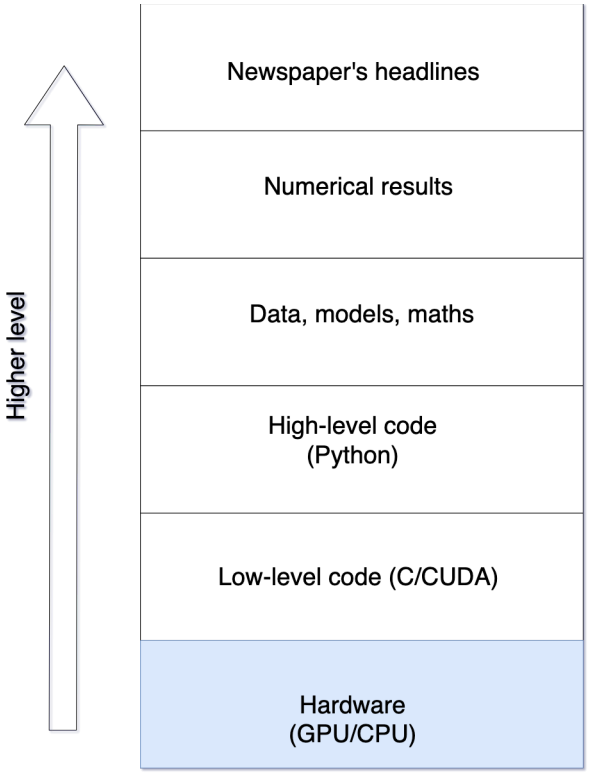

```python
def train(train_loader, model, criterion, optimizer, epoch, args):
    batch_time = AverageMeter('Time', ':6.3f')
    data_time = AverageMeter('Data', ':6.3f')
    losses = AverageMeter('Loss', ':.4e')
    top1 = AverageMeter('Acc@1', ':6.2f')
    top5 = AverageMeter('Acc@5', ':6.2f')
    progress = ProgressMeter(
        len(train_loader),
        [batch_time, data_time, losses, top1, top5],
        prefix="Epoch: [{}]".format(epoch))

    # switch to train mode
    model.train()

    end = time.time()
    for i, (images, target) in enumerate(train_loader):
        # measure data loading time
        data_time.update(time.time() - end)

        if args.gpu is not None:
            images = images.cuda(args.gpu, non_blocking=True)
        target = target.cuda(args.gpu, non_blocking=True)

        # compute output
        output = model(images)
        loss = criterion(output, target)

        # measure accuracy and record loss
        acc1, acc5 = accuracy(output, target, topk=(1, 5))
        losses.update(loss.item(), images.size(0))
        top1.update(acc1[0], images.size(0))
        top5.update(acc5[0], images.size(0))

        # compute gradient and do SGD step
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # measure elapsed time
        batch_time.update(time.time() - end)
        end = time.time()

        if i % args.print_freq == 0:
            progress.display(i)

```

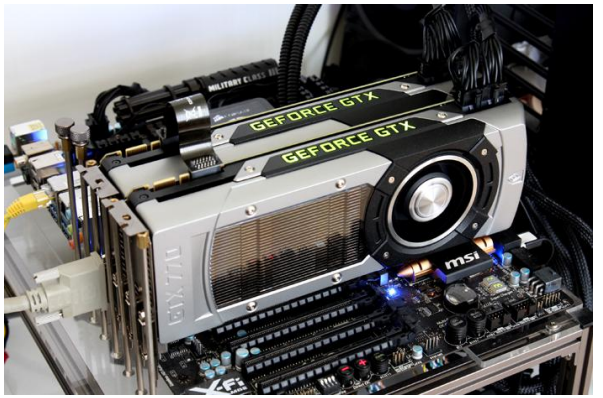
The various levels of machine learning...



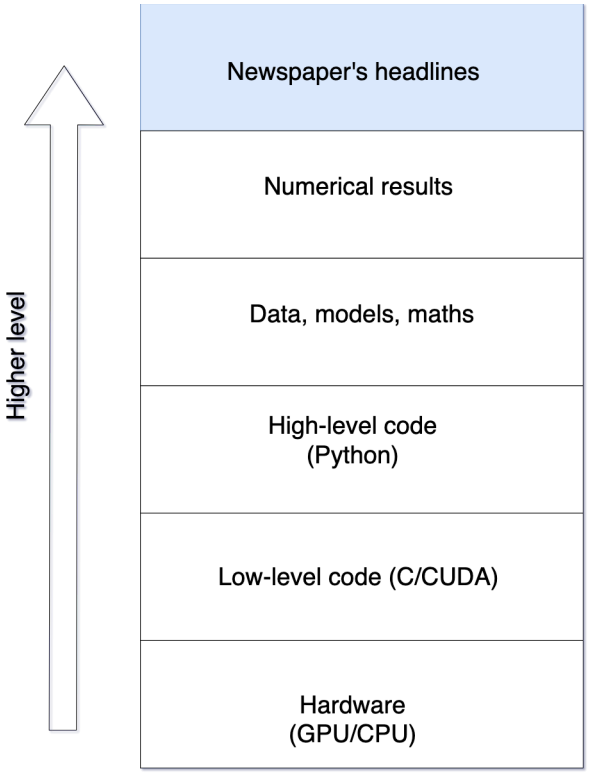

```
117  template <typename Comparator, typename K, typename V, int Power2SortSize>
118  __device__ inline void warpBitonicSort(K* keys,
119                                         V* values,
120                                         const Comparator& comp) {
121    // Smaller sorts should use a warp shuffle sort
122    static_assert(Power2SortSize > kWarpSize,
123                  "sort not large enough");
124    static_assert(integerIsPowerOf2(Power2SortSize),
125                  "sort size must be power of 2");
126    static_assert(Power2SortSize <= kMaxBitonicSortSize,
127                  "sort size <= 4096 only supported");
128
129    // If what we are sorting is too large, then lanes must loop more
130    // than once
131    constexpr int loopPerThread = (Power2SortSize / 2) / kWarpSize;
132    int laneId = getLaneId();
133
134  #pragma unroll
135    for (int size = 2; size < Power2SortSize; size *= 2) {
136
137  #pragma unroll
138      for (int stride = size / 2; stride > 0; stride /= 2) {
139
140  #pragma unroll
141        for (int loop = 0; loop < loopPerThread; ++loop) {
142          int threadId = loop * kWarpSize + laneId;
143          bool flag = ((threadId & (size / 2)) != 0);
144
145          int pos = 2 * threadId - (threadId & (stride - 1));
146
147          bitonicSwap<Comparator, K, V>(
148            keys[pos], values[pos],
149            keys[pos + stride], values[pos + stride],
150            flag, comp);
151
152          __threadfence_block();
153        }
154      }
155    }
```

The various levels of machine learning...



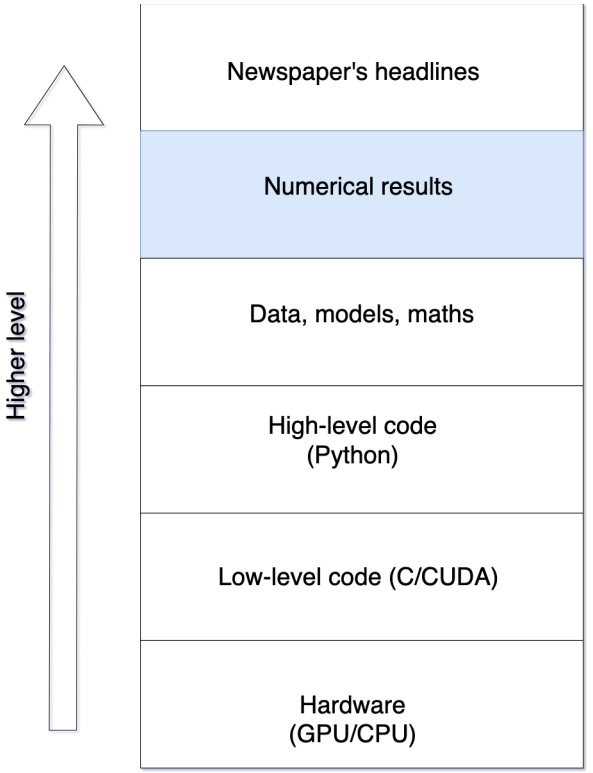


... and their corresponding bugs



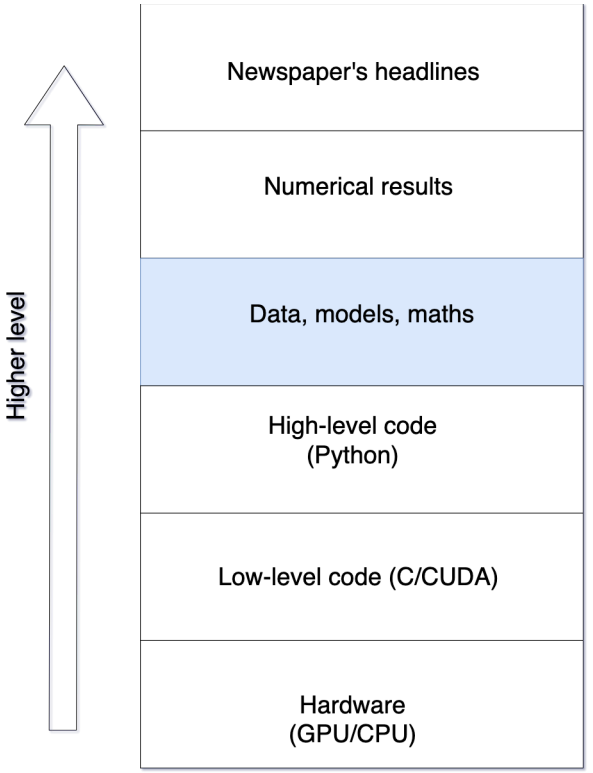
"We created a Terminator" VS "Small neural nets learned to communicate"

... and their corresponding bugs



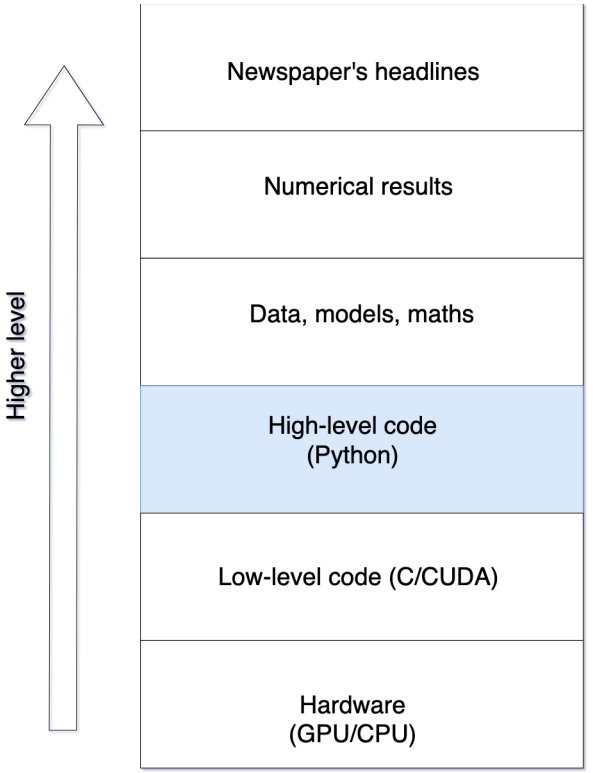
"We solved vision" VS "We got 1% improvement on Imagnet"

... and their corresponding bugs



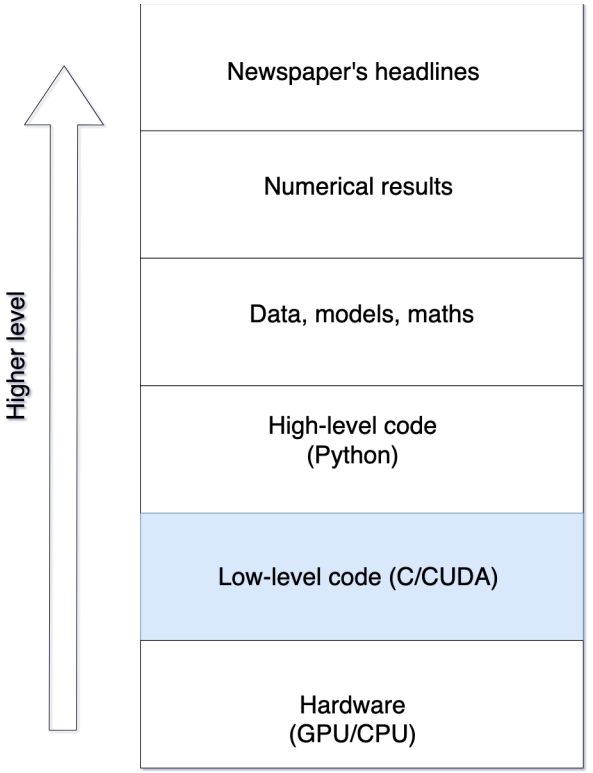
"We can recognize tanks/cars VS we can recognize day/night"

... and their corresponding bugs



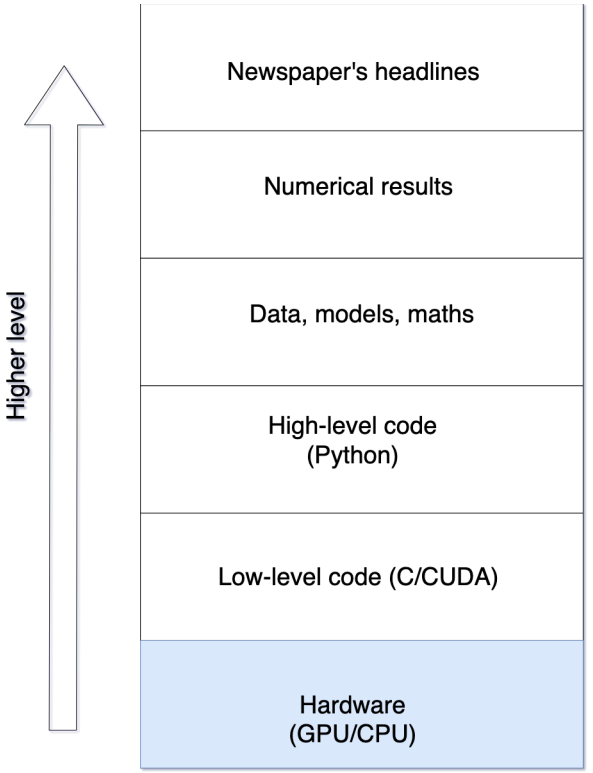
"This model does not work" VS "I forgot to learn the parameters"

... and their corresponding bugs



"My model is slow" VS "I did not use optimized functions"

... and their corresponding bugs

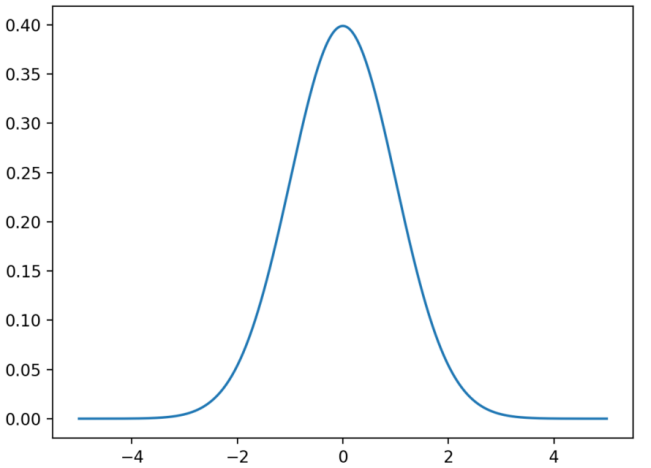


"My model cannot fit large data" VS "I did not use the right GPU setting "

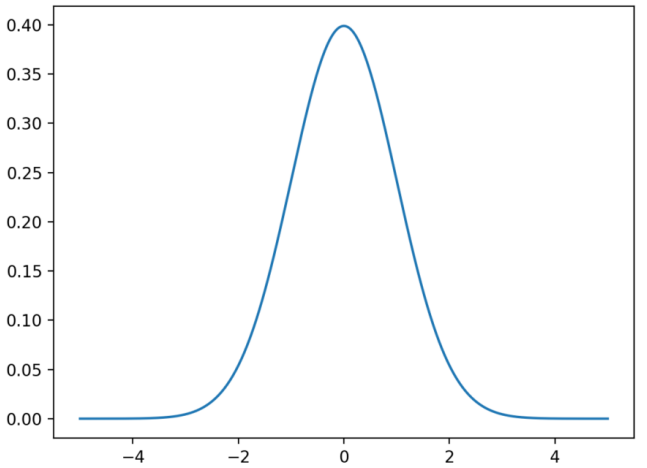
Outline

1. The machine learning stack
2. Brief recap: gaussian random variables
3. Logistic regression and linear classification
4. How to debug a machine learning code?

Brief recap on distributions



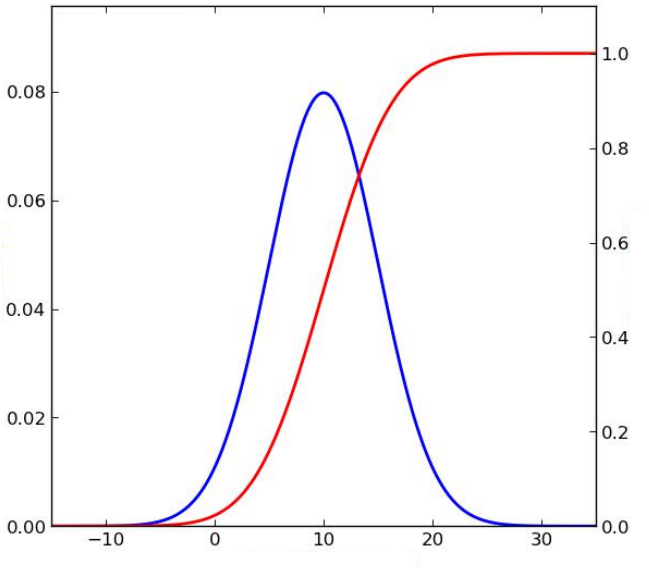
Brief recap on distributions



- Gaussian or Normal Distribution
- Ubiquitous in statistics and Machine Learning

PDF vs. CDF

- Probability Density Function (PDF) vs. Cumulative Distribution Function (CDF)



One-dimensional Gaussian

$$P(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \text{ for } x \in \mathbf{R}$$

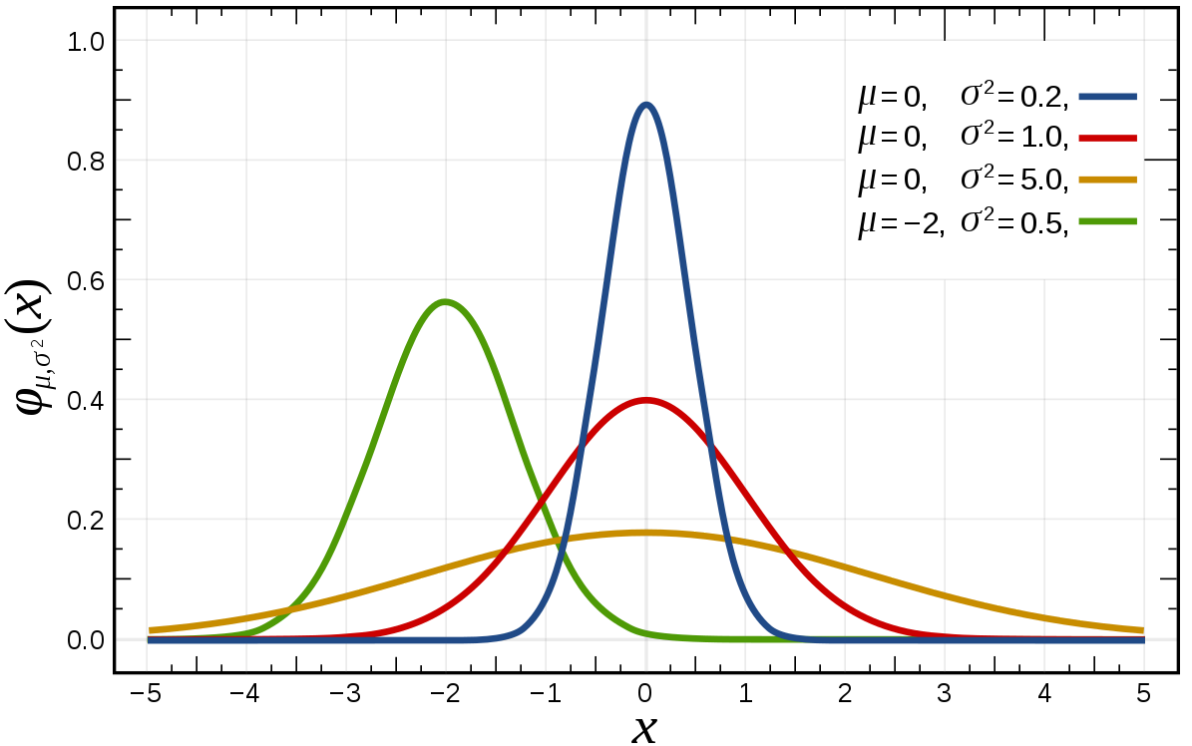
- $\mu, \sigma \in \mathbf{R}$
- μ is the *mean*, σ^2 is the *variance* (σ is the *standard deviation*)

One-dimensional Gaussian

$$P(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \text{ for } x \in \mathbf{R}$$

- $\mu, \sigma \in \mathbf{R}$
- μ is the *mean*, σ^2 is the *variance* (σ is the *standard deviation*)
- Question 1: discuss the influence of μ and σ
- Question 2: how to compute a probability given the PDF graph ?

Influence of μ and σ



Multi-dimensional Gaussian

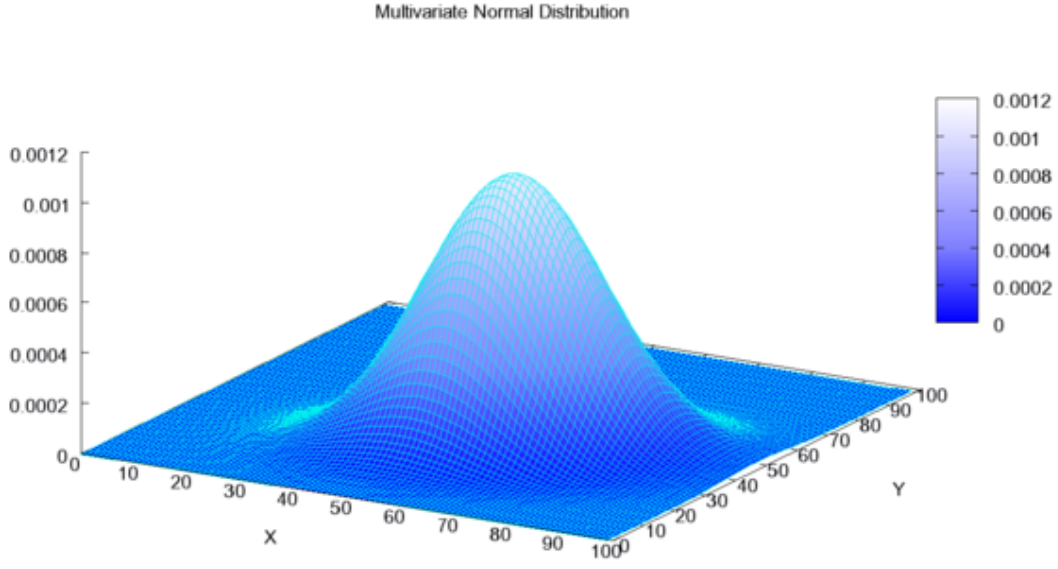
$$P(x|\mu,\Sigma) = \frac{1}{\sqrt{|\Sigma|(2\pi)^n}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \text{ for } x \in \mathbf{R}$$

- $\mu \in \mathbf{R}^n$ is a *vector*, $\Sigma \in \mathcal{M}_n(\mathbf{R})$ is a *matrix*
- μ is the *mean vector*, Σ is the *covariance matrix* (assumed to be positive semi-definite)

Multi-dimensional Gaussian

$$P(x|\mu,\Sigma) = \frac{1}{\sqrt{|\Sigma|(2\pi)^n}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \text{ for } x \in \mathbf{R}$$

- $\mu \in \mathbf{R}^n$ is a *vector*, $\Sigma \in \mathcal{M}_n(\mathbf{R})$ is a *matrix*
- μ is the *mean vector*, Σ is the *covariance matrix* (assumed to be positive semi-definite)



Multi-dimensional Gaussian

$$P(x|\mu,\Sigma) = \frac{1}{\sqrt{|\Sigma|(2\pi)^n}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \text{ for } x \in \mathbf{R}^n$$

- $\mu \in \mathbf{R}^n$ is a *vector*, $\Sigma \in \mathcal{M}_n(\mathbf{R})$ is a *matrix*
- μ is the *mean vector*, Σ is the *covariance matrix* (assumed to be positive semi-definite)
- Question: re-write $P(x|\mu,\Sigma)$ for $\Sigma = I_n$

Multi-dimensional Gaussian

$$P(x|\mu,\Sigma) = \frac{1}{\sqrt{|\Sigma|(2\pi)^n}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \text{ for } x \in \mathbf{R}$$

- $\mu \in \mathbf{R}^n$ is a *vector*, $\Sigma \in \mathcal{M}_n(\mathbf{R})$ is a *matrix*
- μ is the *mean vector*, Σ is the *covariance matrix* (assumed to be positive semi-definite)
- If $\Sigma = I_n$, then $|\Sigma| = 1$ and $\Sigma^{-1} = I_n$, thus

$$(x-\mu)^T\Sigma^{-1}(x-\mu) = (x-\mu)^T(x-\mu) = ||x-\mu||^2$$

and

$$P(x|\mu, I_n) = \frac{1}{\sqrt{(2\pi)^n}} \exp\left(-\frac{1}{2}||x-\mu||^2\right)$$

Central Limit Theorem

- Say you draw at random n balanced dice that have each 100 faces numbered from 1 to 99

Central Limit Theorem

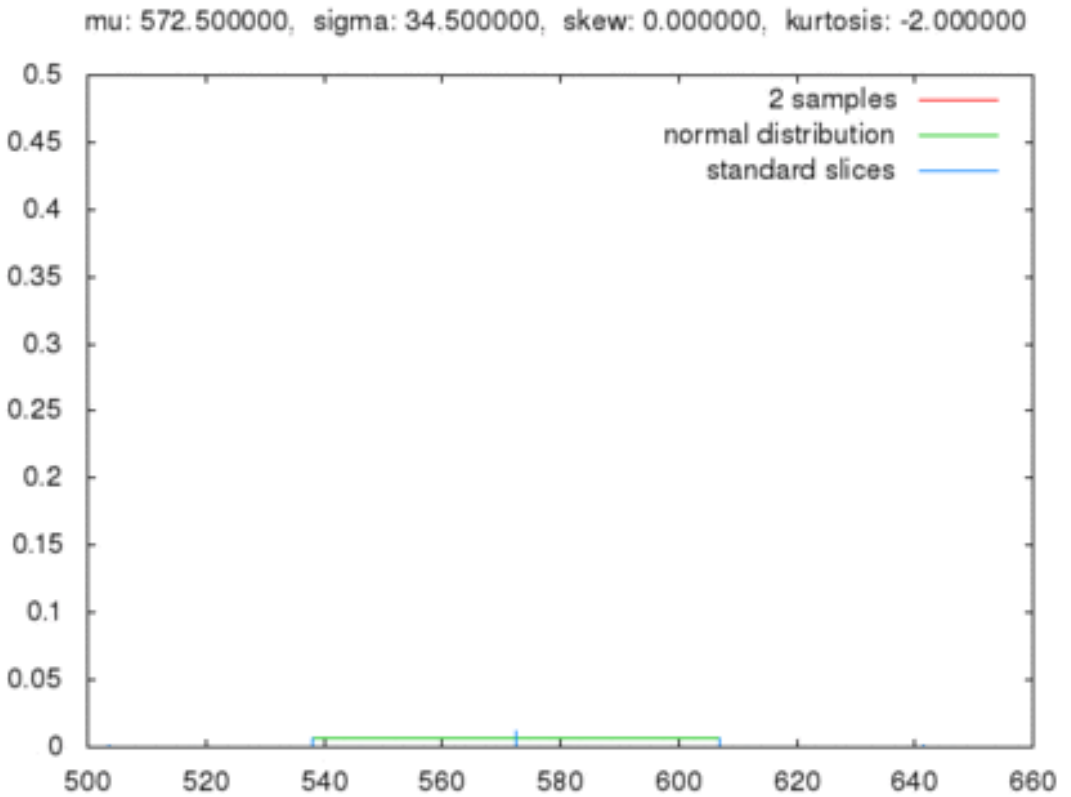
- Say you draw at random n balanced dice that have each 100 faces numbered from 1 to 99
- What can you say about the distribution of the average sum of the n dice $\overline{X_n}$?
- Central Limit theorem: when $n \mapsto +\infty$

$$\overline{X_n} \sim \mathcal{N}(\mu, \sigma/\sqrt{n})$$

- Here $\mu = 50$ and $\sigma \approx 28.7$

Central Limit Theorem

- Say you draw at random n balanced dice that have each 100 faces numbered from 1 to 99



Outline

1. The machine learning stack
2. Brief recap: gaussian random variables
3. Logistic regression and linear classification
4. How to debug a machine learning code?

Logistic regression

- We want to separate two gaussians
- Generating process: for a class $y \in \{0, 1\}$, we generate x:

$$P(x|y=0) = C \exp(-\|x - \mu_0\|^2)$$

$$P(x|y=1) = C \exp(-\|x - \mu_1\|^2)$$

Logistic regression

- In machine learning, we observe x and want to infer y

$$P(y = 1|x) = ?$$

Logistic regression

- In machine learning, we observe x and want to infer y

$$P(y = 1|x) = ?$$

- We can apply Bayes'rule:

$$P(y = 1|x) = \frac{P(y = 1, x)}{P(x)}$$

$$P(y = 1|x) = \frac{P(x|y = 1)P(y = 1)}{P(x|y = 1)P(y = 1) + P(x|y = 0)P(y = 0)}$$

Logistic regression

- Logistic function $\sigma(x) = 1/(1 + \exp(-x))$

Logistic regression

- Logistic function $\sigma(x) = 1/(1 + \exp(-x))$
- We have:

$$\frac{a}{a+b} = \sigma\left(-\log\left(\frac{b}{a}\right)\right)$$

Logistic regression

- Logistic function $\sigma(x) = 1/(1 + \exp(-x))$
- We have:

$$\frac{a}{a+b} = \sigma\left(-\log\left(\frac{b}{a}\right)\right)$$

- Hence:

$$P(y=1|x) = \sigma\left(-\log\left(\frac{P(x|y=0)P(y=0)}{P(x|y=1)P(y=1)}\right)\right)$$

Logistic regression

- We assume balanced classes: $P(y = 0) = P(y = 1)$

Logistic regression

- We assume balanced classes: $P(y = 0) = P(y = 1)$
- Let's work out the term inside σ:

$$-\log\left(\frac{P(x|y=0)P(y=0)}{P(x|y=1)P(y=1)}\right) = -\log\left(\frac{C\exp(-\frac{1}{2}\|x-\mu_0\|^2)}{C\exp(-\frac{1}{2}\|x-\mu_1\|^2)}\right)$$

Logistic regression

- We assume balanced classes: $P(y=0) = P(y=1)$
- Let's work out the term inside σ:

$$-\log\left(\frac{P(x|y=0)P(y=0)}{P(x|y=1)P(y=1)}\right) = -\log\left(\frac{C\exp(-\frac{1}{2}\|x-\mu_0\|^2)}{C\exp(-\frac{1}{2}\|x-\mu_1\|^2)}\right)$$

$$= \frac{1}{2}\|x-\mu_0\|^2 - \frac{1}{2}\|x-\mu_1\|^2$$

Logistic regression

- We assume balanced classes: $P(y=0) = P(y=1)$
- Let's work out the term inside σ:

$$-\log\left(\frac{P(x|y=0)P(y=0)}{P(x|y=1)P(y=1)}\right) = -\log\left(\frac{C\exp(-\frac{1}{2}\|x-\mu_0\|^2)}{C\exp(-\frac{1}{2}\|x-\mu_1\|^2)}\right)$$

$$= \frac{1}{2}\|x-\mu_0\|^2 - \frac{1}{2}\|x-\mu_1\|^2$$

$$= \langle x, \mu_1 - \mu_0\rangle + \frac{\|\mu_1\|^2 - \|\mu_0\|^2}{2}$$

Logistic regression

- We assume balanced classes: $P(y=0) = P(y=1)$
- Let's work out the term inside σ:

$$-\log\left(\frac{P(x|y=0)P(y=0)}{P(x|y=1)P(y=1)}\right) = -\log\left(\frac{C\exp(-\frac{1}{2}\|x-\mu_0\|^2)}{C\exp(-\frac{1}{2}\|x-\mu_1\|^2)}\right)$$

$$= \frac{1}{2}\|x-\mu_0\|^2 - \frac{1}{2}\|x-\mu_1\|^2$$

$$= \langle x, \mu_1 - \mu_0\rangle + \frac{\|\mu_1\|^2 - \|\mu_0\|^2}{2}$$

- Our loss function is

$$-\log\left(P(y=1|x)\right) = -\log\left(\sigma\left(\langle x, \mu_1 - \mu_0\rangle + \frac{\|\mu_1\|^2 - \|\mu_0\|^2}{2}\right)\right)$$

Logistic regression

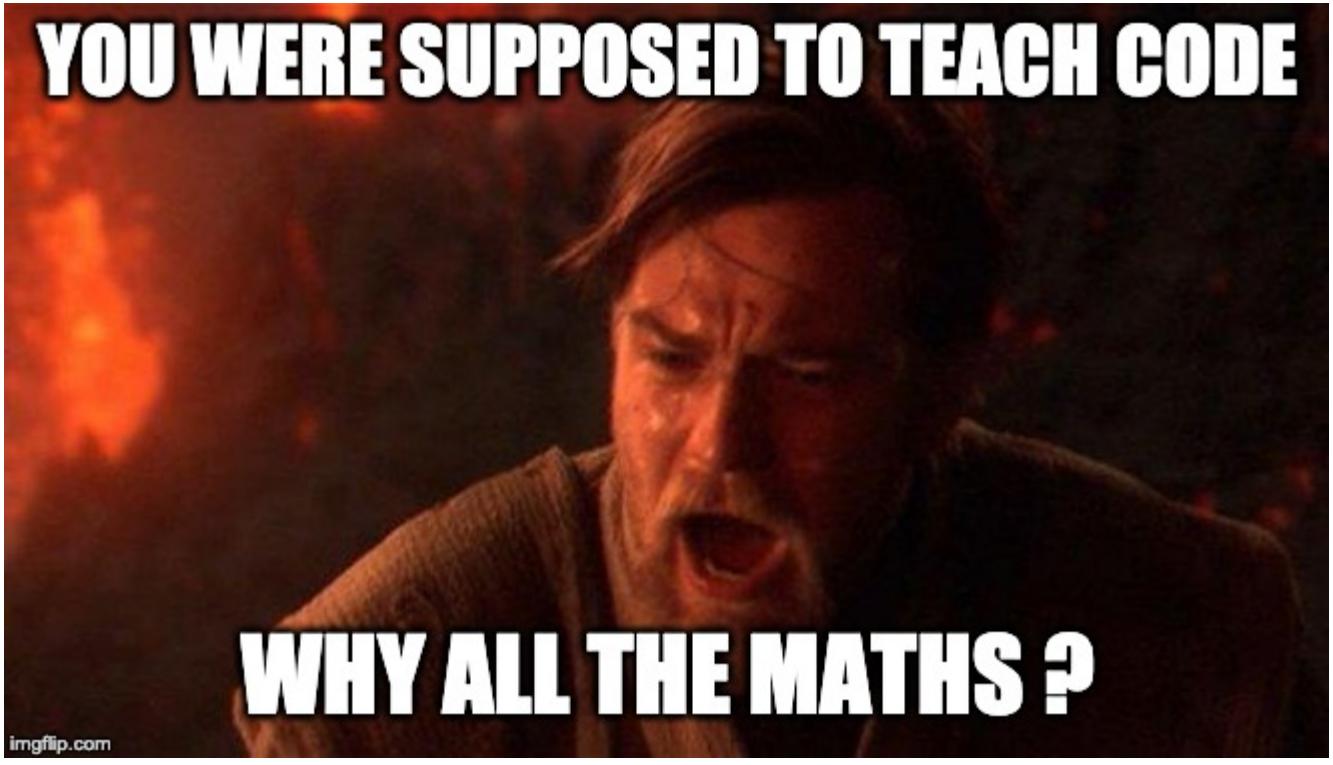
- Our loss function writes as:

$$-\log\left(P(y=1|x)\right) = f\left(\langle x, w\rangle - b\right)$$

where $w = \mu_1 - \mu_0, \quad b = -\frac{\|\mu_1\|^2 - \|\mu_0\|^2}{2}$, and:

$$\begin{aligned} f(u) &= -\log(\sigma(u)) \\ &= -\log\left(\frac{1}{1+\exp(-u)}\right) \\ &= \log(1+\exp(-u)) \end{aligned}$$

Logistic regression



Why the maths

- Why we do the maths

Why the maths

- Why we do the maths
  - **Understand**: why do we do logistic regression ?

Why the maths

- Why we do the maths
  - **Understand**: why do we do logistic regression ?
  - **Debug**: Gaussians are a simple case to test our code

Why the maths

- Why we do the maths
  - **Understand**: why do we do logistic regression ?
  - **Debug**: Gaussians are a simple case to test our code
  - **Visualize**: put some sense back into equations

Logistic regression

- For (x, y=1), we want to maximize $P(y = 1|x)$. We have:

$$-\log\left(P(y = 1|x)\right) = f\left(\langle x, w\rangle - b\right)$$

- For (x,y=0), we want to maximize $P(y = 0|x)$. We can show:

$$-\log\left(P(y = 0|x)\right) = f\left(-\left(\langle x, w\rangle - b\right)\right)$$

- If we call $t_i = 2 * y_i - 1$, we have:

$$-\log\left(P(y = y_i|x_i)\right) = f\left(t_i\left(\langle x_i, w\rangle - b\right)\right)$$

- Over the whole dataset, we minimize:

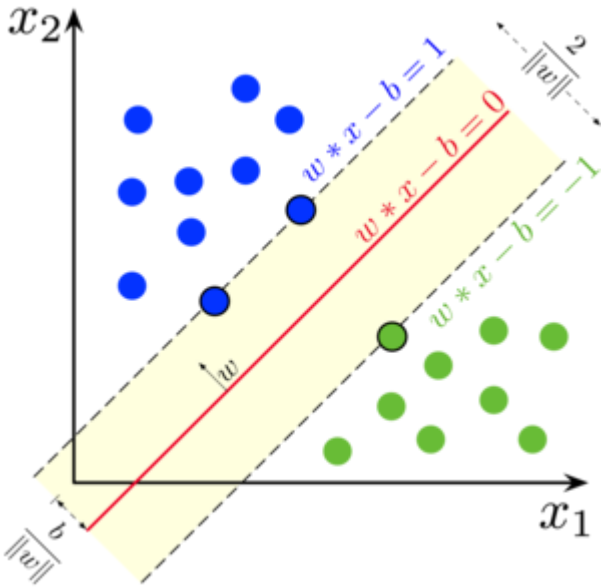
$$\mathcal{L}(w) = \frac{1}{n}\sum_{i=1}^{n} f(t_i\left(\langle x_i, w\rangle - b\right))$$

Logistic regression

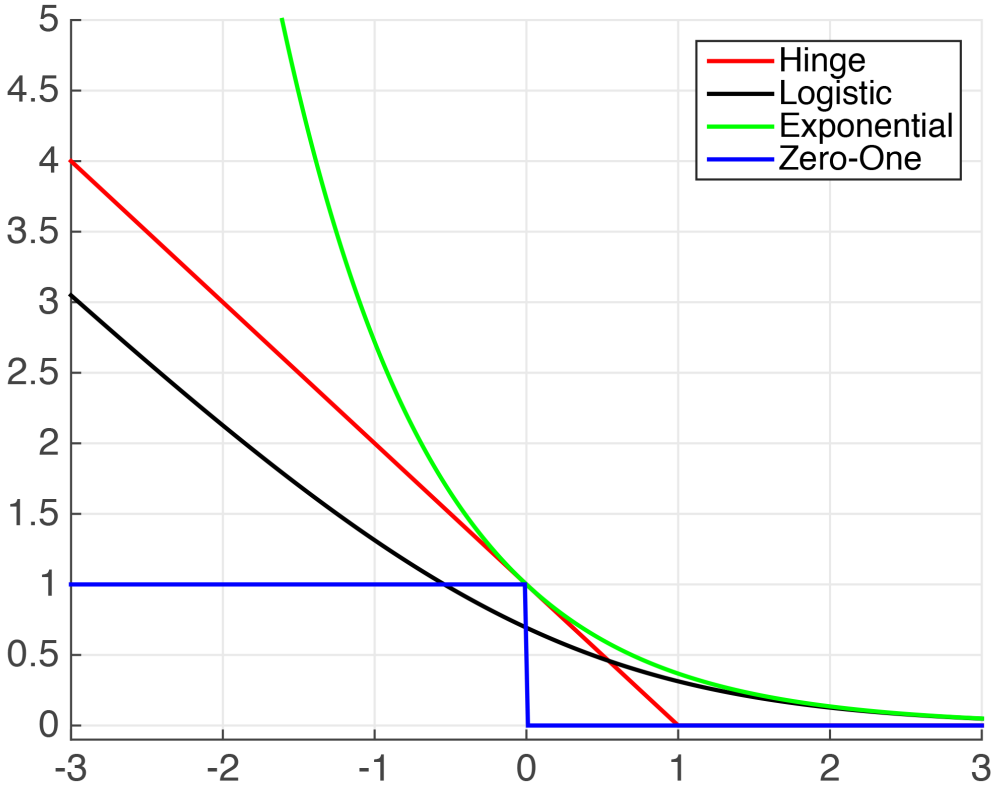
- Trick: $\langle x, w \rangle - b = \langle \tilde{x}, \tilde{w} \rangle$, with
  - $\tilde{x} = [x, -1]$
  - $\tilde{w} = [w, b]$

Logistic regression

- Trick: $\langle x, w \rangle - b = \langle \tilde{x}, \tilde{w} \rangle$, with
  - $\tilde{x} = [x, -1]$
  - $\tilde{w} = [w, b]$
- If $t_i = 1$, we expect $\langle x_i, w \rangle - b \geq 0$
- If $t_i = -1$, we expect $\langle x_i, w \rangle - b \leq 0$
- In general, $t_i(\langle x_i, w \rangle - b) \geq 0$ iff we classified correctly



What does the loss look like ?



Gradients

- How do we learn the model ?

Gradients

- How do we learn the model ?
- Gradient descent!

Gradients

- How do we learn the model ?
- Gradient descent!

Gradients

- How do we learn the model ?
- Gradient descent!

- (Negative) gradient is:

$$-\nabla \mathcal{L}(\tilde{w}) = \frac{1}{n} \sum_{i=1}^{n} \sigma(-t_i \langle \tilde{x}_i, \tilde{w} \rangle) t_i \tilde{x}_i$$

Gradients

- How do we learn the model ?
- Gradient descent!

- (Negative) gradient is:

$$-\nabla \mathcal{L}(\tilde{w}) = \frac{1}{n} \sum_{i=1}^{n} \sigma(-t_i \langle \tilde{x}_i, \tilde{w} \rangle) t_i \tilde{x}_i$$

- Do you believe me ?

Gradients

- How do we learn the model ?
- Gradient descent!

- (Negative) gradient is:

$$-\nabla \mathcal{L}(\tilde{w}) = \frac{1}{n} \sum_{i=1}^{n} \sigma(-t_i \langle \tilde{x}_i, \tilde{w} \rangle) t_i \tilde{x}_i$$

- Do you believe me ?
- Looking at the formula, can you convince yourself ?

Gradients

- How do we learn the model ?
- Gradient descent!

- (Negative) gradient is:

$$-\nabla \mathcal{L}(\tilde{w}) = \frac{1}{n} \sum_{i=1}^{n} \sigma(-t_i \langle \tilde{x}_i, \tilde{w} \rangle) t_i \tilde{x}_i$$

Gradients

- How do we learn the model ?
- Gradient descent!

- (Negative) gradient is:

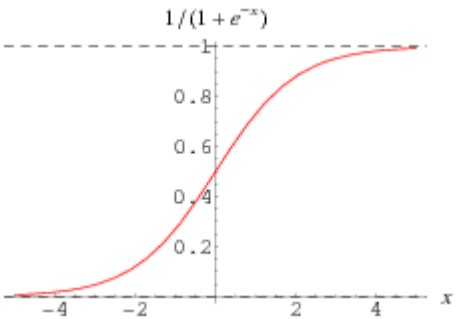
$$-\nabla \mathcal{L}(\tilde{w}) = \frac{1}{n} \sum_{i=1}^{n} \sigma(-t_i \langle \tilde{x}_i, \tilde{w} \rangle) t_i \tilde{x}_i$$

Gradients

- How do we learn the model ?
- Gradient descent!

- (Negative) gradient is:

$$-\nabla \mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \sigma(-t_i \langle \tilde{x}_i, \tilde{w} \rangle) t_i \tilde{x}_i$$



Gradients

- Now you believe the maths, but how do you believe your implementation ?

Gradients

- Now you believe the maths, but how do you believe your implementation ?
- Gradcheck!

Gradients

- Now you believe the maths, but how do you believe your implementation ?
- Gradcheck!
- The gradient is the vector of component-wide derivatives

Gradients

- Now you believe the maths, but how do you believe your implementation ?
- Gradcheck!
- The gradient is the vector of component-wide derivatives
- We can compare the gradient to its finite-difference approximation

$$\nabla \mathcal{L}_j \approx \frac{\mathcal{L}(\tilde{w} + \epsilon\, e_j) - \mathcal{L}(\tilde{w} - \epsilon\, e_j)}{2\,\epsilon}$$

Outline

1. The machine learning stack
2. Brief recap: gaussian random variables
3. Logistic regression and linear classification
4. How to debug a machine learning code?

Code debuging

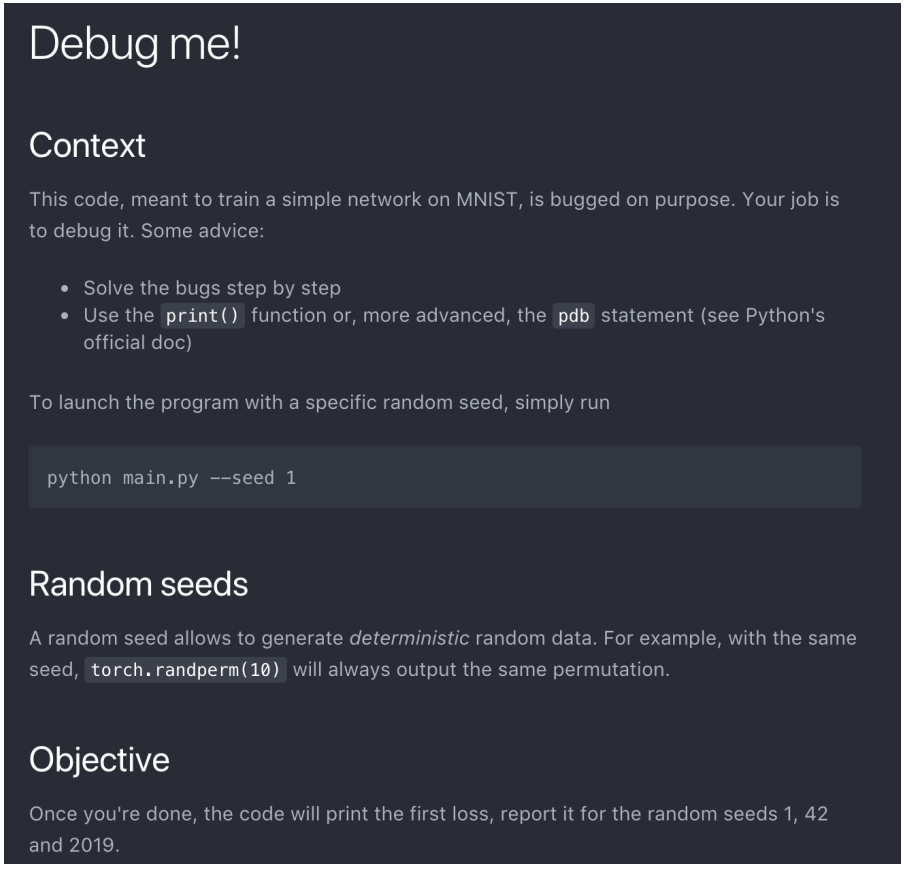
- Debuging 101: use `print()` functions for quantities that matter
- More advanced: use the python debuger `pdb` (see official documentation, more on that on future sessions)
- Beware: a code that *executes* does not necessarily *work* as you expected! (e.g. a network that does not converge properly)

Guidelines

- When *writing* code, check at every step that the code compiles and that it outputs what you expect
  - See the very important `assert` statement
- Once the code is written or when executing someone else's code, perform sanity checks such as:
  - It fails when we expect it to fail
  - It succeeds on very simple and naive cases
  - It outputs the same result another code doing the same thing that you trust (possibly less optimized)

Let's practice!

- One of your homework for tomorrow will be to debug a code that has been *voluntarily* bugged. The folder's name is debug, start with the README!



Debug me!

Context

This code, meant to train a simple network on MNIST, is bugged on purpose. Your job is to debug it. Some advice:

- Solve the bugs step by step
- Use the `print()` function or, more advanced, the `pdb` statement (see Python's official doc)

To launch the program with a specific random seed, simply run

```
python main.py --seed 1
```

Random seeds

A random seed allows to generate *deterministic* random data. For example, with the same seed, `torch.randperm(10)` will always output the same permutation.

Objective

Once you're done, the code will print the first loss, report it for the random seeds 1, 42 and 2019.