

Practical Deep Neural Networks

GPU computing perspective

Introduction

Yuhuang Hu Chu Kiong Loo

Advanced Robotic Lab
Department of Artificial Intelligence
Faculty of Computer Science & IT
University of Malaya

Outline

1 Introduction

2 Linear Algebra

3 Dataset Preparation

- Data Standardization
- Principle Components Analysis
- ZCA Whitening

Outline

1 Introduction

2 Linear Algebra

3 Dataset Preparation

- Data Standardization
- Principle Components Analysis
- ZCA Whitening

Objectives

- Light introduction of numerical computation.
- Fundamentals of Machine Learning.
- Support Vector Machine, Softmax Regression.
- Feed-forward Neural Network.
- Convolutional Networks.
- Recurrent Neural Networks.

Prerequisites

- ★ Basic training in Calculus
- ★ Basic training in Linear Algebra
 - Matrix operations
 - Matrix properties: transform, rank, norm, determinant, etc
 - Eigendecomposition, Singular Value Decomposition.
- ★ Basic programming skills
 - If-else conditioning
 - Loops
 - Function, class, library
 - Source code control: Git (optional)

References

- ◆ Deep Learning: An MIT Press book in preparation
Main reference in this workshop, still in development, awesome structure, awesome contents.
- ◆ Machine Learning: A probabilistic perspective
One of the best Machine Learning books on the market.
- ◆ CS231n Convolutional Neural Networks for Visual Recognition Notes
Nice structured, well written, loads of pictures.
- ◆ CS229 Machine Learning Course Materials
For basic knowledge, well written, easy-to-read.

Software and Tools

- ★ Ubuntu 14.04
- ★ CUDA Toolkit 7
- ★ Python 2.7.9 (Why not 3.*?)
- ★ Theano
- ★ numpy, scipy, etc
- ★ Eclipse+PyDev

Reading List

A reading list is prepared for this workshop, all reading materials can be found at:

<http://rt.dgyblog.com/ref/ref-learning-deep-learning.html>

The list keeps expanding!!

Outline

1 Introduction

2 Linear Algebra

3 Dataset Preparation

- Data Standardization
- Principle Components Analysis
- ZCA Whitening

Scalar, Vector, Matrix and Tensor

Scalars A scalar is a single number.

Vectors A vector is an array of numbers.

Matrices A matrix is a 2-D array of numbers.

Tensors A tensor is an array of numbers arranged on a regular grid with a variable number of axes.

Matrix Operations and Properties

Identity $\mathbf{I}_n \in \mathbb{R}^{n \times n}$, $\mathbf{I}\mathbf{A} = \mathbf{A}\mathbf{I} = \mathbf{A}$.

Transpose $(\mathbf{A}^\top)_{ij} = A_{ji}$.

Addition $\mathbf{C} = \mathbf{A} + \mathbf{B}$ where $C_{ij} = A_{ij} + B_{ij}$.

Scalar $\mathbf{D} = a \cdot \mathbf{B} + c$ where $D_{ij} = a \cdot B_{ij} + c$.

Multiply $\mathbf{C} = \mathbf{A}\mathbf{B}$ where $C_{ij} = \sum_k A_{ik}B_{kj}$.

Element-wise Product $\mathbf{C} = \mathbf{A} \odot \mathbf{B}$ where $C_{ij} = A_{ij}B_{ij}$.

Distributive $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{A}\mathbf{B} + \mathbf{A}\mathbf{C}$.

Associative $\mathbf{A}(\mathbf{B}\mathbf{C}) = (\mathbf{A}\mathbf{B})\mathbf{C}$.

Transpose of Matrix Product $(\mathbf{A}\mathbf{B})^\top = \mathbf{B}^\top \mathbf{A}^\top$.

Inverse Matrices

The *matrix inverse* of \mathbf{A} is denoted as \mathbf{A}^{-1} , and it is defined as the matrix such that

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}_n$$

if \mathbf{A} is not singular matrix.

If \mathbf{A} is not square or is square but singular, it's still possible to find it's generalized inverse or pseudo-inverse.

Norms

We usually measure the size of vectors using an L^p norm:

$$\|\mathbf{x}\|_p = \left(\sum_i |x_i|^p \right)^{\frac{1}{p}}$$

There are few commonly used norms

- L_1 norm

$$\|\mathbf{x}\|_1 = \sum_i |x_i|.$$

- l_∞ (Max norm)

$$\|\mathbf{x}\|_\infty = \max_i |x_i|.$$

- Frobenius norm: Measure the size of matrix, analogy to the L^2 norm.

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i,j} A_{ij}^2}$$

Unit vector, Orthogonal, Orthonormal, Orthogonal Matrix

A *unit vector* is a vector with *unit norm*:

$$\|\mathbf{x}\|_2 = 1$$

A vector \mathbf{x} and \mathbf{y} are *orthogonal* to each other if $\mathbf{x}^\top \mathbf{y} = 0$. In \mathbb{R}^n , at most n vectors may be mutually orthogonal with nonzero norm. If vectors are not only orthogonal but also have unit norm, we call them *orthonormal*.

An *orthogonal matrix* is a square matrix whose rows are mutually orthonormal and whose columns are mutually orthonormal:

$$\mathbf{A}^\top \mathbf{A} = \mathbf{A} \mathbf{A}^\top = \mathbf{I}$$

This implies that

$$\mathbf{A}^{-1} = \mathbf{A}^\top$$

Eigendecomposition

An *eigenvector* of a square matrix A is a non-zero vector v such that

$$Av = \lambda v$$

where scalar λ is known as the *eigenvalue* corresponding to the eigenvector. If v is an eigenvector of A , so is any rescaled vector sv for $s \in \mathbb{R}, s \neq 0$. Therefore, we usually only look for unit eigenvectors.

We can represent the matrix A using an *eigendecomposition*.

$$A = V \text{diag}(\lambda) V^{-1}$$

where $V = [v^{(1)}, v^{(2)}, \dots, v^{(n)}]$ (one column per eigenvector) and $\lambda = [\lambda_1, \dots, \lambda_n]$.

Every real symmetric matrix can be decomposed into

$$A = Q \Lambda Q^T$$

where Q is an orthogonal matrix composed of eigenvectors of A , and Λ is a diagonal matrix, with λ_{ii} being the eigenvalues.

Singular Value Decomposition (SVD)

SVD is a general purpose matrix factorization method that decompose an $n \times m$ matrix \mathbf{X} into:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{W}^\top$$

where \mathbf{U} is an $n \times n$ matrix whose columns are mutually orthonormal, $\mathbf{\Sigma}$ is a rectangular diagonal matrix and \mathbf{W} is an $m \times m$ matrix whose columns are mutually orthonormal. Elements along the main diagonal of $\mathbf{\Sigma}$ are referred to as the singular values. Columns of \mathbf{U} and \mathbf{W} are referred to as the *left-singular vectors* and *right-singular vectors* of \mathbf{X} respectively.

- Left-singular vectors of \mathbf{X} are the eigenvectors of $\mathbf{X}\mathbf{X}^\top$.
- Right-singular vectors of \mathbf{X} are the eigenvectors of $\mathbf{X}^\top\mathbf{X}$.
- Non-zero singular values of \mathbf{X} are the square roots of the non-zero eigenvalues for both $\mathbf{X}\mathbf{X}^\top$ and $\mathbf{X}^\top\mathbf{X}$.

Trace, Determinant

The trace operator is defined as

$$\text{Tr}(\mathbf{A}) = \sum_i A_{ii}$$

Frobenius norm in trace operator:

$$\|\mathbf{A}\|_F = \sqrt{\text{Tr}(\mathbf{A}^\top \mathbf{A})}$$

The determinant of a square matrix, denoted $\det(\mathbf{A})$ is a function mapping matrices to real scalars. The determinant is equal to the product of all matrix's eigenvalues.

Outline

1 Introduction

2 Linear Algebra

3 Dataset Preparation

- Data Standardization
- Principle Components Analysis
- ZCA Whitening

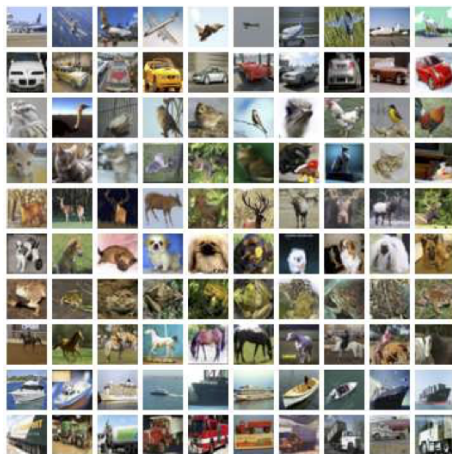
MNIST

MNIST dataset contains 60,000 training samples and 10,000 testing samples (Lecun, Bottou, Bengio, & Haffner, 1998). Each sample is a hand-written digit from 0-9.



CIFAR-10

CIFAR-10 dataset consists 60,000 images in 10 classes (Krizhevsky, 2009). There are 50,000 training images and 10,000 testing images.



Dataset: How to split the data?

- ✌ If dataset has been splitted initially, please keep that way.
- ✌ If dataset comes as a whole, usually use 60% as trianing data, 20% as cross-validation and 20% as testing data.
- ✌ Another alternative is using 70% as training data, 30% as testing data.

Dataset: Cross validation

- ❖ Cross validation is used to keep track the performance of learning algorithm (overfitting? underfitting?).
- ❖ Used to be a small portion of training dataset, randomly selected.
- ❖ Serve as testing data when testing data is not visible.
- ❖ Help to choose and adjust parameters of the learning model.

Mean subtraction, Unit Variance

Let μ as mean image

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

And then subtract mean image on every image:

$$x^{(i)} = x^{(i)} - \mu$$

Let

$$\sigma_j^2 = \frac{1}{m} \sum_i \left(x_j^{(i)} \right)^2$$

and then

$$x_j^{(i)} = \frac{x_j^{(i)}}{\sigma_j}$$

This is also called PCA whitening if it's applied to rotated data, the variance here can also be viewed as eigenvalues.

PCA

Principle Components Analysis (PCA) is a dimension reduction algorithm that can be used to significantly speed up unsupervised feature learning algorithm. PCA will find a lower-dimensional subspace onto which to project our data.

Given a set of data $\mathbf{X} = \{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$, where $x^{(i)} \in \mathbb{R}^n$, covariance is computed by

$$\Sigma = \frac{\mathbf{X}\mathbf{X}^\top}{m}$$

Use eigendecomposition, we can obtain the eigenvectors \mathbf{U} :

$$\mathbf{U} = \begin{bmatrix} | & | & \cdots & | \\ u_1 & u_2 & \cdots & u_n \\ | & | & \cdots & | \end{bmatrix}$$

and corresponding eigenvalues $\mathbf{\Lambda} = \{\lambda_1, \dots, \lambda_n\}$. Note that usually $\mathbf{\Lambda}$ is a sorted in descending order.

PCA

Columns in U are principle bases, so that we can rotate our data to the new bases:

$$\mathbf{X}_{\text{rot}} = \mathbf{U}^T \mathbf{X};$$

Noted that $\mathbf{X} = \mathbf{U} \mathbf{X}_{\text{rot}}$.

To one of the data x , we can reduce the dimension by

$$\tilde{x} = \begin{bmatrix} x_{\text{rot},1} \\ \vdots \\ x_{\text{rot},k} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \approx \begin{bmatrix} x_{\text{rot},1} \\ \vdots \\ x_{\text{rot},k} \\ x_{\text{rot},k+1} \\ \vdots \\ x_{\text{rot},n} \end{bmatrix} = x_{\text{rot}}$$

The resulted $\tilde{\mathbf{X}}$ can approximate the distribution of \mathbf{X}_{rot} since it's dropping only small components. And we reduced $n - k$ dimensions of the original data

PCA

We can recover the approximation of original signal $\hat{\mathbf{X}}$ from $\tilde{\mathbf{X}}$:

$$\hat{x} = U \begin{bmatrix} \tilde{x}_1 \\ \vdots \\ \tilde{x}_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

PCA: Number of components to retain

To determine k , we usually look at the *percentage of variance retained*,

$$p = \frac{\sum_{i=1}^k \lambda_j}{\sum_{j=1}^n \lambda_j} \quad (1)$$

One common heuristic is to choose k so as to retain 99% of the variance ($p \geq 0.99$).

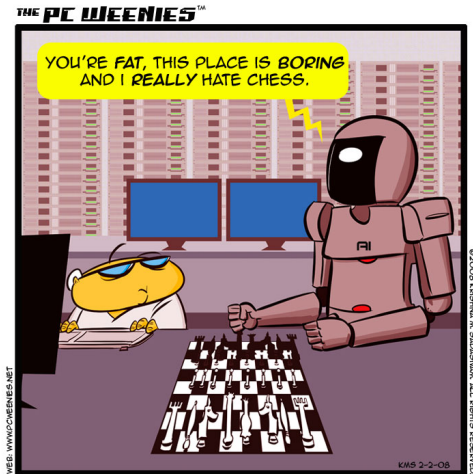
ZCA Whitening

ZCA whitening is just a recover version of PCA whitening:

$$x_{\text{ZCA}} = U x_{\text{PCA}} = U \frac{x_{\text{rot}}}{\mathbf{\Lambda} + \epsilon}$$

where ϵ is a regularization term. When x takes values around $[-1, 1]$, a value of $\epsilon \approx 10^{-5}$ might be typical.

Q&A



HOW YOU'LL KNOW WHEN YOU'VE TRULY
SUCCEEDED IN THE FIELD OF A.I. RESEARCH.

References

- Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images* (Master Thesis).
- Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998, Nov). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.