Master Thesis.

A Library for Fast Kernel Expansions with Applications to Computer Vision and Deep Learning.

J. de Curtó i Díaz.

8th December 2014.

curto@cmu.edu

http://www.andrew.cmu.edu/user/curto/

Carnegie Mellon

Outline

Introduction

Description

- Time period: 26th May 2014 - 5th December 2014.
- Carnegie Mellon.
- Location: Pittsburgh (Pennsylvania).
- Office 8018. GATES HILLMAN Center.
- School of Computer Science.
  ML Department.

- Supervisors: **A. Smola** and **C. W. Ngo**.

Introduction

How It All Began?

- Starts as a summer internship at Carnegie Mellon.
  - Working at the HS Laboratory in Computer Vision and Machine Learning.
- Grows into a dissertation in the ML Department.
  - Main focus: Fast Kernel Expansions and their efficient implementation.
- What's next? Deep Learning.

Introduction

A Primer on Machine Learning and Computer Vision

The goal: solve the problem of estimation of ethnicity.

Setup:

- Dataset from Scratch: Images and Labels.
- Landmarks and Affine Transform.
- Extraction of Handcrafted Features.
- Classification.
- Crossvalidation.
- ML Tricks.

C&Z Dataset from Scratch

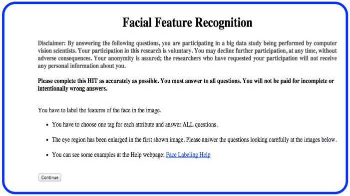
API Weaknesses Exploitation in Flickr

- PYTHON code to retrieve URLs given a list of attributes. Filtering by time, image quality and avoiding negative tags.
- MATLAB code to crawl massively images from the internet.

Cleaning the Data

- Extract faces.
- Label images using MTurk.
- MTurk labels to JSON format.

Labeling the Images



Facial Feature Recognition

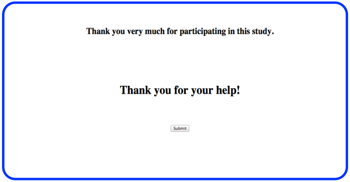
Disclaimer: By answering the following questions, you are participating in a big data study being performed by computer vision scientists. Your participation in this research is voluntary. You may decline further participation, at any time, without adverse consequences. Your anonymity is assured; the researchers who have requested your participation will not receive any personal information about you.

Please complete this HIT as accurately as possible. You must answer to all questions. You will not be paid for incomplete or intentionally wrong answers.

You have to label the features of the face in the image.

- You have to choose one tag for each attribute and answer ALL questions.
- The eye region has been enlarged in the first shown image. Please answer the questions looking carefully at the images below.
- You can see some examples at the Help webpage: Face Labeling Help

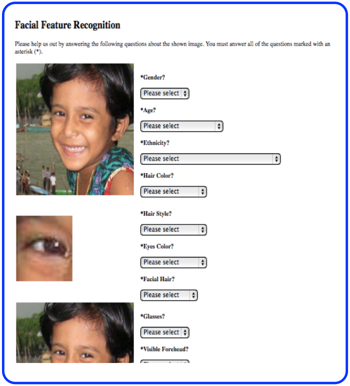
Continue



Thank you very much for participating in this study.

Thank you for your help!

Submit



Facial Feature Recognition

Please help us out by answering the following questions about the shown image. You must answer all of the questions marked with an asterisk (*).

***Gender?** Please select

***Age?** Please select

***Ethnicity?** Please select

***Hair Color?** Please select

***Hair Style?** Please select

***Eyes Color?** Please select

***Facial Hair?** Please select

***Glasses?** Please select

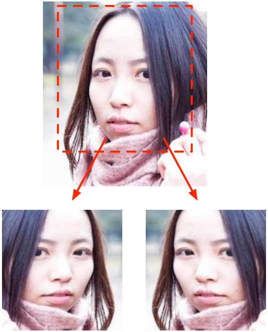
***Visible Forehead?**

Affine Transform and Extraction of Landmarks

- Affine Transform and Normalization.
- Extract Landmark Points: 49 facial points.

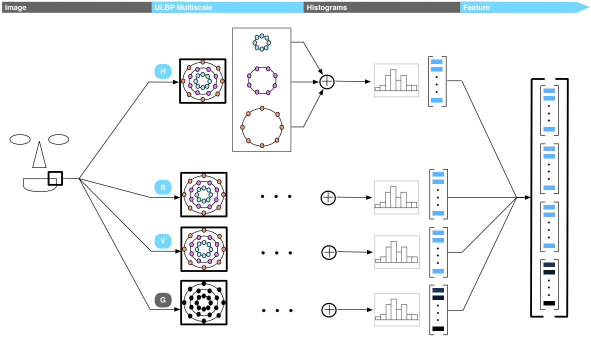
Extraction of Landmarks

Normalization, Affine Transform and Mirroring



Feature Extraction

- ULBP Multiscale.
- HSV Color Space.
- 4th informative channel. Preprocessing of (Tan and Triggs 2007).



Classification and Crossvalidation

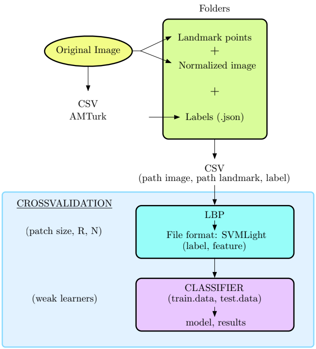
Classifier

- SVM Linear: LIBSVM (Chang and Lin 2007).

Crossvalidation

- Choose 3 best radius.
- Choose patch size and number of neighbors.
- Choose appropriate SVM C parameter.
- Choose number of weak learners for AdaBoost.

Classifier



Accuracy

- Around 86% in our dataset.

Fast Kernel Expansions

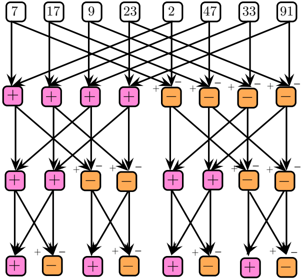
McKernel

- Motivation: use kernel methods in large-scale data.
- Based on Random Kitchen Sinks by (Rahimi and Recht 2007).
- Main idea: approximate a random matrix Gaussian by a multiplication of random matrices diagonal.
- Why it speeds up the computation? Uses Hadamard, which can be computed in $O(n \log n)$ time.

Fast Kernel Expansions

Walsh Hadamard

- Can be computed using the Fast Walsh Hadamard (FWH).
- Algorithm Divide and Conquer that recursively halves the input vector.



McKernel

Writing Fast Numerical Code

- SIMD Intrinsics: SSE2, SSSE3 and AVX.
- Cache-friendly code.
- Loop unrolling.

```
__mm128 a = _mm_loadu_ps(&data[0]);
__mm128 b = _mm_loadu_ps(&data[4]);
__mm128 s = _mm_add_ps(a, b);
__mm128 d = _mm_sub_ps(a, b);
_mm_storeu_ps(s, &data[0]);
_mm_storeu_ps(d, &data[4]);
```

McKernel

McKernel

In Random Kitchen Sinks instead of computing RBF GAUSSIAN Kernel

$$k(x, x') = \exp(-||x - x'||^2/(2\sigma^2)) \quad (1)$$

the method computes

$$k(x, x') = \exp(i[Zx]_c) \quad (2)$$

where z_c is drawn from a random distribution Normal.
Z is now parametrized by V as

$$V := \frac{1}{\sigma\sqrt{d}} SHG\Pi HB. \quad (3)$$

McKernel

Fastfood

$$V := \frac{1}{\sigma\sqrt{d}} SHG\Pi HB \tag{4}$$

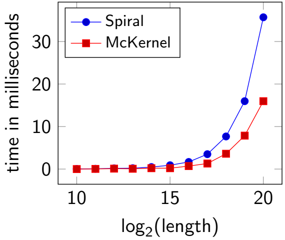
where

- B is a random matrix diagonal with i.i.d. entries $+1$ and -1.
- H is the matrix Walsh Hadamard computed in-place with FWH in McKernel.
- Π is the matrix of permutation generated using FISHER YATES algorithm in McKernel.
- G is a random matrix diagonal with random numbers i.i.d. Normal.
- S is a random matrix diagonal with random numbers i.i.d. Chi.

McKernel

FWH

- Iterative algorithm. It computes half of the vector going down in depth, and then it goes from bottom to top solving iteratively the remaining computations.
- Recursions are avoided to decrease stack overhead and cache hits are maximized by using this structure.
- SIMD Intel Intrinsics: SSE2 and SSSE3 using 128 bit registers and AVX using 256 bit registers.
- FWH at 1Gflop. **Faster than current state-of-the art methods (Spiral).**



McKernel

Permutation by Fisher Yates

- Π can be generated with order $O(n \log n)$:
  - Augmenting the integers $1, \ldots, n$ with random keys, forming value key pairs $(r_1, 1) \cdots (r_n, n)$.
  - r_z is a random number Uniform $[0, 1]$.
  - Sort these elements by key using an $O(n \log n)$ algorithm, for instance Quicksort.
  - Discard the keys to get the permutation.
- Shuffle Fisher Yates: optimum $(O(n))$ algorithm to permute an array of n elements.
  - Start from the first element of an array $\{1 \ldots n\}$.
  - Pick another element uniformly from the remaining set.
  - Swap this new selected element with the current item.
  - Repeat this procedure till you get to the $n - 1$ position to obtain the desired permutation.

McKernel

Factory McKernel: Object Oriented Design

McKernel provides an API based on a factory, which is an object oriented paradigm where:

- Interface creates object.
- Subclass decides class to instantiate.

```
McKernel* mckernel =
FactoryMcKernel::createMcKernel(FactoryMcKernel::RBF,
 data, nv, dn, sigma);
```

where we can use RBF or RBF MATÉRN:

- `FactoryMcKernel::RBF`
- `FactoryMcKernel::MRBF`

Each kernel contains methods:

- `McFeatures()` Computes V.
- `McEvaluate()` Computes the mapping of complex features.

McKernel

Distributed Version

- Pseudo-random Numbers are generated using functions of hashing $h(c, z)$ with range $[0 \dots N]$ just by setting $U_c = h(c, z)/N$.
- From these random numbers Uniform, we generate random numbers Normal using BOX MULLER Transform (Box and Muller 1958)

$$P_{cz} = (-2 \log h_1(c, z)/N)^{1/2} \cos(2\pi h_2(c, z)/N). \tag{5}$$

- and deviates Chi using the approximation of (Wilson and Hilferty 1931)

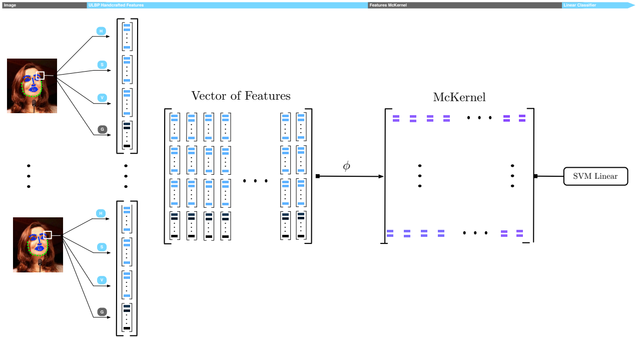
$$\chi_d^2 = d\left(\sqrt{\frac{2}{9d}}w + \left(1 - \frac{2}{9d}\right)\right)^3 \tag{6}$$

where w is a standard distribution Normal $N(0, 1)$.

By using hashing we get Pseudo-random Numbers than can be generated on the fly!

An Application of Computer Vision

The idea is to use McKernel before the linear classifier in our system of estimation of ethnicity.



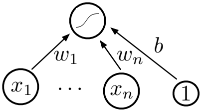
An Application of Deep Learning

A Simple Neural Network

$$a(\mathbf{x}) = \sum_c w_c x_c + b = \mathbf{w}^T \mathbf{x} + b \quad (7)$$

$$h(\mathbf{x}) = g(a(\mathbf{x})) = g\left(\sum_c w_c x_c + b\right) \quad (8)$$

where **w** are the connecting weights, b the neuron bias and $g()$ the activation function.

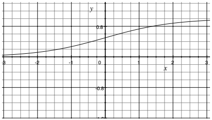


Introduction to Deep Learning

Activation Function

$$g(a) = \text{sigmoid}(a) = \frac{1}{1 + e^{-a}}. \quad (9)$$

- Understand an artificial neuron as an estimator of $p(y = 1|\mathbf{x})$, logistic regression classifier.
- It works like this: if the output is greater than 0.5 the logistic regression outputs class 1, otherwise it outputs class 0.



Introduction to Deep Learning

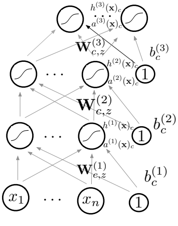
Multi-layer Neural Network

$$\mathbf{a}^{(k)}(\mathbf{x}) = \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}(\mathbf{x}) + \mathbf{b}^{(k)} \tag{10}$$

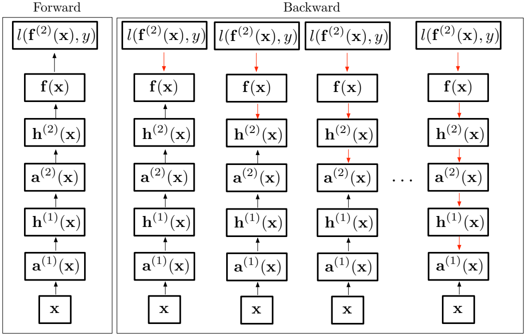
$$\mathbf{h}^{(k)}(\mathbf{x}) = g(\mathbf{a}^{k}(\mathbf{x})). \tag{11}$$

Output layer:

$$\mathbf{h}^{(L+1)}(\mathbf{x}) = \mathbf{o}(\mathbf{a}^{(L+1)}(\mathbf{x})) = \mathbf{f}(\mathbf{x}). \tag{12}$$



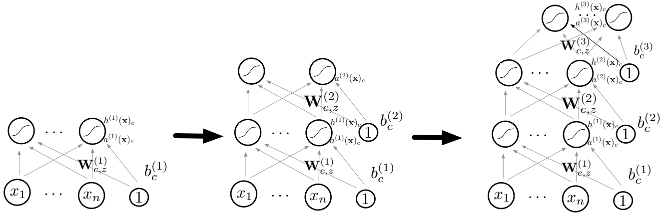
Backpropagation



- Each node has a function of forward propagation which depends on children.
- Each node has a function of backward propagation which depends on parent.

Deep Neural Network

- Pretraining: initialize the parameters.
- Once all layers are pretrained, train the whole network using supervised learning: fine-tuning.
  - Forward propagation.
  - Backward propagation.
  - Update.



Deep Neural Network McKernel

- Implementation of the Code: two-layer stacked autoencoder, with sparse autoencoders as hidden layers.
- Autoencoder: makes the input equal to the output and extracts features in the hidden layer $f(\mathbf{x}) \equiv \hat{\mathbf{x}}$.
- We have embedded McKernel in a neural network as non-linear mapping to the activation function.
- Improved performance of 3% just by wiring this kernel expansion.

Conclusions

Contributions

- SIMD FWH implementation faster than current state-of-the-art methods.
- Library McKernel for Fast Kernel Expansions in Log-linear Time.
- C&Z Dataset.
- Implementation from scratch of a system for estimation of ethnicity, including the tools to crawl massively images from the net.
- Implementation of a simple architecture of Deep Learning to test our library of approximating kernel expansions, it is the ground for future research.

Thank You

DE CURTÓ I DÍAZ Joaquim.

Thank you.

Special thanks to all the people at the ML Department and Robotics that made this possible.