Pattern Information Processing: Final Project

24 July, 2012

Stroke based classification of handwritten digits

Christo du Plessis

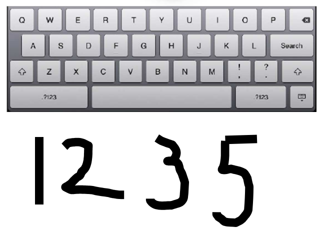
Akihiro Yamashita

Sugiyama Laboratory

Tokyo Institute of Technology

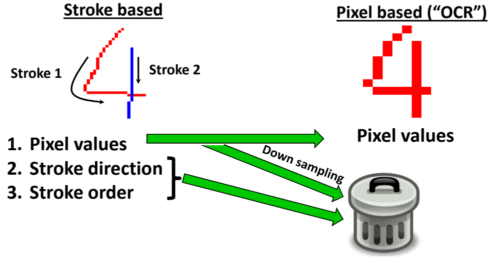
Introduction

- Many modern devices have touch displays instead of keyboards
- Text input is tedious
- Drawing text is more natural than software keyboards



Stroke-based digit classification

- When digits are treated as images, information is discarded
  - "OCR" problem more difficult than stroke-based approach



Feature selection

- Stroke-based features should be:
  - Size invariant
  - Low-dimensional description
  - Accurate describe the shape and stroke direction
  - Smoothen over discrete pixel values
- Bezier-curve derivatives satisfy all of these properties

Bezier curves

(Watt, A.H., 2005)

- Bezier curve defined as:

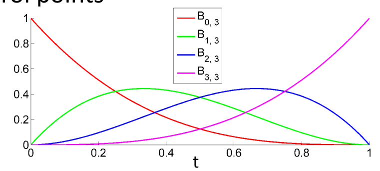
$$B(t) = \sum_{i=0}^{n} B_{i,n}(t)\boldsymbol{c}_i,$$

(unknown) Control points

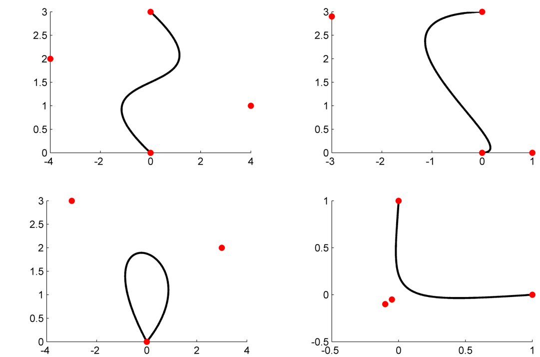
Bernstein basis functions:

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

- Bernstein basis functions smoothens between control points



Bezier curves: Examples



Square loss fitting of a stroke

- The point list is defined as

$\{(x_j, y_j)\}_{j=1}^{N}$ $t = \frac{j}{n}$ we fit $\{(x_j, t_j)\}_{j=1}^{N}$ (Each dimension separately)

- Dimension-wise square-loss fit:

$$E = (\boldsymbol{x} - \bar{\boldsymbol{c}}_x^\top \boldsymbol{M}\boldsymbol{T})(\boldsymbol{x} - \boldsymbol{c}_x^\top \boldsymbol{M}\boldsymbol{T})^\top, \\ = \boldsymbol{x}\boldsymbol{x}^\top - 2\bar{\boldsymbol{c}}_x^\top \boldsymbol{M}\boldsymbol{T}\boldsymbol{x}^\top + \bar{\boldsymbol{c}}_x^\top \boldsymbol{M}\boldsymbol{T}\boldsymbol{T}^\top \boldsymbol{M}^\top \bar{\boldsymbol{c}}_x$$

$$\Rightarrow \bar{\boldsymbol{c}}_x = \left(\boldsymbol{M}\boldsymbol{T}\boldsymbol{T}^\top \boldsymbol{M}\right)^{-1} \boldsymbol{M}\boldsymbol{T}\boldsymbol{x}^\top \qquad \boldsymbol{c} = \begin{bmatrix} \bar{\boldsymbol{c}}_x^\top \\ \bar{\boldsymbol{c}}_y^\top \end{bmatrix}$$

$$\boldsymbol{T} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ t_1 & t_2 & \dots & t_m \\ t_1^2 & t_2^2 & \dots & t_m^2 \\ \vdots & \vdots & \dots & \vdots \\ t_1^n & t_2^n & \dots & t_m^n \end{bmatrix}$$

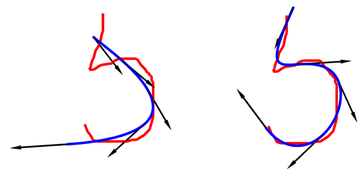
M From Bernstein polynomial

Derivatives

- Derivative calculated as

$$\frac{\partial B(t)}{\partial t} = \sum_{i=0}^{n} \frac{\partial B_{i,n}(t)}{\partial t}\boldsymbol{c}_i, \qquad \frac{\partial B_{i,n}(t)}{\partial t} = nB_{i-1,n-1}(t) - nB_{i,n-1}(t)$$

- Example, '5'



3rd order curve 5rd order curve

- Derivatives are features

Dataset

- 2250 multi-stroke ('4', '5', '=', '+', 'x') samples
- 4498 single-stroke ('1', '2', etc…) samples
- Dataset collected from only 6 users
  - May not be representative of all users
  - This perhaps explains the different performance in training and demonstration

Selection of Features

- Calculate the derivative at d points
- Compared the features using 5-nearest neighbour
- Cross-validation score on whole dataset

d	5	8	10	15
3rd order Bezier	2.77%	2.65%	2.68%	2.71%
4rd order Bezier	2.68%	1.85%	2.11%	**1.85%**
5rd order Bezier	5.04%	3.46%	2.83%	2.73%

Misclassification rate – single stroke characters

Method	Misclassification rate
LSPC	1.98%
1-KNN	1.67%
3-KNN	1.64%
5-KNN	1.85%
9-KNN	1.99%
SVM, linear	1.47%
SVM, Polynomial p=2	1.08%
SVM, Polynomial p=3	0.98%
SVM, Polynomial p=4	**0.84%**
SVM, Gaussian	1.00%

Discussion and conclusion

- Bezier-curves fitted via least-squares and derivatives used as features
- Stroke-based approach yields excellent results
- SVM with 4rd order polynomial gave highest accuracy:
  - 0.85% Misclassification rate
- Stroke-based approach simplifies tasks such as segmenting (not discussed)
- Lower practical performance perhaps due to **small dataset** (few samples, small amount of subjects)
  - Improve by enlarging dataset, and
  - Consider invariances (e.g. slight rotation)

Questions?

Acknowledgements:

- Program written in Matlab, uses:
  - libSVM: http://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - LSPC: http://sugiyama-www.cs.titech.ac.jp/~sugi/software/LSPC/
- Thanks to the following people for entering data:
  - Duong Nguyen
  - Tomoya Sakai
  - Keisuke Nakata
  - Hyunha Nam

References

- Watt, Alan H., 3D Computer Graphics, Addison-Wesley, 2005

Square loss fitting of a stroke (1)

- Let $B(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$ $\quad \boldsymbol{c}_i = \begin{bmatrix} c_i^x \\ c_i^y \end{bmatrix}$
- The curve can be expressed as

$$x(t) = \bar{\boldsymbol{c}}^{x\top} \boldsymbol{b}(t) \qquad \bar{\boldsymbol{c}}^x = [c_0^x \quad c_1^x \quad \dots \quad c_n^x]$$

$$\boldsymbol{b}(t) = [B_{0,n}(t) \quad B_{1,n}(t) \quad \dots \quad B_{n,n}(t)]^\top$$

- The point list is defined as

$\{(x_i, y_i)\}_{j=1}^N \quad t = \frac{j}{n}$ we fit $\{(x_i, t_i)\}_{j=1}^N$ (Each dimension separately)

- Output at different time-steps are $\bar{c}^x \boldsymbol{M}\boldsymbol{T}$

$$\boldsymbol{T} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ t_1 & t_2 & \dots & t_m \\ t_1^2 & t_2^2 & \dots & t_m^2 \\ \vdots & \vdots & \dots & \vdots \\ t_1^n & t_2^n & \dots & t_m^n \end{bmatrix}$$

M **From Bernstein polynomial, e.g.**

$$\boldsymbol{M}_3 = \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}^\top$$

Square loss fitting of a stroke (2)

- Square-loss fit:

$$\begin{aligned} E &= \left(\boldsymbol{x} - \bar{\boldsymbol{c}}_x^\top \boldsymbol{M}\boldsymbol{T}\right)\left(\boldsymbol{x} - \boldsymbol{c}_x^\top \boldsymbol{M}\boldsymbol{T}\right)^\top, \\ &= \boldsymbol{x}\boldsymbol{x}^\top - 2\bar{\boldsymbol{c}}_x^\top \boldsymbol{M}\boldsymbol{T}\boldsymbol{x}^\top + \bar{\boldsymbol{c}}_x^\top \boldsymbol{M}\boldsymbol{T}\boldsymbol{T}^\top \boldsymbol{M}^\top \bar{\boldsymbol{c}}_x \end{aligned}$$

$$\Rightarrow \bar{\boldsymbol{c}}_x = \left(\boldsymbol{M}\boldsymbol{T}\boldsymbol{T}^\top \boldsymbol{M}\right)^{-1} \boldsymbol{M}\boldsymbol{T}\boldsymbol{x}^\top$$

- Same can be done for $\boldsymbol{y}(t)$
- Regularize $\boldsymbol{M}\boldsymbol{T}\boldsymbol{T}^\top \boldsymbol{M}$ with $\epsilon \boldsymbol{I}$ for the case where the pixels are less than the control points