Distinctive-touch

gesture-based authentication for touch-screen displays

max van kleek : mas.622j : dec.7.2004
"smart" digital touch-screen displays in places people congregate, accidentally meet one another most often located in high-traffic spaces: building entrances, elevator lobbies, hallways, lounges, kitchenettes, cafeterias,
personalization requires user identification

biometrics violate privacy - users can’t change their biologies

RFID requires user to carry around a badge

usernames/passwords are cumbersome, slow

these techniques are overkill, and not scalable
distinctive touch:
recognizes you by your “passdoodle”
a multi-stroke gesture-based signature

users train 5-10 training examples: < 5 seconds each
distinctive touch interaction times

```
>> min(times)
1.0290  0.4060  1.5410  0.5180  0.3540  1.1390  1.3990
1.0310  2.2330  1.0930

>> mean(times)
1.3309  0.9380  2.4711  0.9367  0.5284  1.4328  1.5674
1.1913  2.4735  1.3355

>> max(times)
2.0140  1.5030  2.9120  1.3070  0.6900  1.8130  1.9180
1.3330  2.8690  1.7490

>> std(times)
1.2539  0.3313  1.1927  4.3841  2.1382  0.3453  2.2351
0.3209  0.7660  1.6609

>> mean(mean(times))
1.4206

< 5 seconds

>> vstd(times)
0.6467
```
training doodles → feature extraction → build a model/train classifier → trained classifier → “max!”

building a dt classifier
training doodles

feature extraction

build a model/train classifier

“max!”

(1) 

(2) 

(3) 

(4) 

talk outline

trained classifier
training doodles

feature extraction

build a model/train classifier

trained classifier

“max!”

training set
training set: hiragana character set
training set: hiragana character set

train set:
45 classes
10 ex each
1-5 strokes

test set:
45 classes
5 ex each
training set: hiragana character set

train set:
45 classes
10 ex each
1-5 strokes

test set:
45 classes
5 ex each
training set: hiragana character set set

histogram of number of data points ("length") of stroke color-coded by class

- train set:
  - 45 classes
  - 10 ex each
  - 1-5 strokes

- test set:
  - 45 classes
  - 5 ex each
dt versus handwriting recognition

- stroke order
- rejection
- velocity and timing
- # ex / # of classes
Training doodles → Feature extraction → Build a model/train classifier → Trained classifier → “max!” → Feature extraction
# feature extraction

**Dean Rubine: Specifying gestures by example**

13 unistroke features: 11 geometric, 2 time

<table>
<thead>
<tr>
<th>dt extractor</th>
<th>rubine extr</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>distfv</td>
<td>f8</td>
<td>total euclidean distance traversed by a stroke</td>
</tr>
<tr>
<td>bboxfv2</td>
<td>f3</td>
<td>dimension of bounding box of a stroke</td>
</tr>
<tr>
<td>f4fv</td>
<td>f4</td>
<td>stroke bounding box aspect ratio</td>
</tr>
<tr>
<td>startendfv</td>
<td>f5</td>
<td>euclidean distance between start and end pts of a stroke</td>
</tr>
<tr>
<td>f1fv</td>
<td>f1,f2,f6,f7</td>
<td>sine and cosine of start and end angles of a stroke</td>
</tr>
<tr>
<td>f9fv</td>
<td>f9</td>
<td>sum of angle traversed by the stroke</td>
</tr>
<tr>
<td>f10fv</td>
<td>f10</td>
<td>sum of absolute angle traversed by the stroke (“curviness”)</td>
</tr>
<tr>
<td>f11fv</td>
<td>f11</td>
<td>sum of squared angles traversed by stroke (“jagginess”)</td>
</tr>
<tr>
<td>f12fv</td>
<td>f12</td>
<td>max instantaneous velocity within a stroke</td>
</tr>
<tr>
<td>f13fv</td>
<td>f13</td>
<td>total time duration of a stroke</td>
</tr>
</tbody>
</table>
generalizing unistroke feature vectors to variable #s of strokes for fld, svm, nnets, glds...

allocate space for all stroke features in each “feature frame”; preserve frame alignment

perils:
sparse vectors

“0” sentinel unintentionally misinformative?

try & see..!
generalizing unistroke feature vectors to variable #s of strokes

solution #2:

represent each stroke as a sequence; use appropriate techniques
training doodles → feature extraction

build a model/train classifier

trained classifier → “max!”

comparing classifiers

(1) (2) (3)
k-nearest neighbors - all features

knn algorithm performance for various values of k using various feature selectors

- leave-one-out all fvs
- test all fvs

accuracy (%) vs. k
**k-nearest neighbors - combinations**

Performance using feature extractors f1fv bboxfv2 f13fv f9fv distfv startendfv

- **Leave-one-out validation**
- **Test set**

Accuracy (%) vs. `k`
strokewise ML - simplest multistroke generative model

represent each class as separate sequences of states, each representing a stroke.

Each state thus has an associated parametric distribution over the values of that stroke’s feature vector

strictly encodes our previous assumption that strokes of the same class always arrive in the same order... otherwise, we’d need an HMM.
strokewise ML - 1 gaussian per stroke

estimation - $\mu_s \Sigma_s$ obtained from ML estimate of mean, cov of feature vectors for each stroke in that class

classification - maximum log posterior prob over models (during comparison, models with different # of strokes than a gesture are immediately rejected)

easily generalizable to gaussian mixtures - just apply EM
**strokewise ML - 1 gaussian per stroke**

Performance with individual features

<table>
<thead>
<tr>
<th>features</th>
<th>l-o-o performance</th>
<th>test performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>distfv</td>
<td>0.5422</td>
<td>0.3467</td>
</tr>
<tr>
<td>bboxfv2</td>
<td>0.8067</td>
<td>0.6756</td>
</tr>
<tr>
<td>f4fv</td>
<td>0.5156</td>
<td>0.4267</td>
</tr>
<tr>
<td>startendfv</td>
<td>0.6067</td>
<td>0.4400</td>
</tr>
<tr>
<td>f1fv</td>
<td>0.8889</td>
<td>0.7511</td>
</tr>
<tr>
<td>f9fv</td>
<td>0.6511</td>
<td>0.5822</td>
</tr>
<tr>
<td>f10fv</td>
<td>0.5467</td>
<td>0.4222</td>
</tr>
<tr>
<td>f11fv</td>
<td>0.4311</td>
<td>0.3333</td>
</tr>
<tr>
<td>f12fv</td>
<td>0.2756</td>
<td>0.1289</td>
</tr>
<tr>
<td>f13fv</td>
<td>0.6378</td>
<td>0.4178</td>
</tr>
</tbody>
</table>
**strokewise ML - 1 gaussian per stroke**

performance with multiple features

<table>
<thead>
<tr>
<th>features</th>
<th>l-o-o performance</th>
<th>test performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>f9fv, bboxfv2, startendfv</td>
<td>0.9733</td>
<td>0.9156</td>
</tr>
<tr>
<td>f1fv, startendfv, f9fv, distfv</td>
<td>0.9644</td>
<td>0.8889</td>
</tr>
<tr>
<td>f1fv, f9fv, distfv, startendfv</td>
<td>0.9778</td>
<td>0.9644</td>
</tr>
<tr>
<td>all combined: distfv, bboxfv2, f4fv, startendfv, f1fv, f9fv, f10fv, f12fv, f13fv</td>
<td>0.9711</td>
<td>0.8800</td>
</tr>
</tbody>
</table>
fisher linear discriminant - (1-dimensional)

OVA (one-versus-all):
train C FLD binary classifiers on the fvs
evaluate each one on the test point
+1 if it gets the label right, 0 otherwise / (C*N)

<table>
<thead>
<tr>
<th>features</th>
<th>l-o-o performance</th>
<th>test performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>distfv</td>
<td>0.9422</td>
<td>0.8311</td>
</tr>
<tr>
<td>bboxfv2</td>
<td>0.9356</td>
<td>0.8711</td>
</tr>
<tr>
<td>f4fv</td>
<td>0.8444</td>
<td>0.7556</td>
</tr>
<tr>
<td>startendfv</td>
<td>0.9600</td>
<td>0.880</td>
</tr>
<tr>
<td>f1fv</td>
<td>0.9244</td>
<td>0.9022</td>
</tr>
<tr>
<td>f9fv</td>
<td>0.8711</td>
<td>0.7644</td>
</tr>
<tr>
<td>f10fv</td>
<td>0.8467</td>
<td>0.5778</td>
</tr>
<tr>
<td>f11fv</td>
<td>0.7867</td>
<td>0.5556</td>
</tr>
<tr>
<td>f12fv</td>
<td>0.8467</td>
<td>0.7067</td>
</tr>
<tr>
<td>f13fv</td>
<td>0.9333</td>
<td>0.8400</td>
</tr>
</tbody>
</table>
**fisher linear discriminant** - combined features

*(warning: figures are a bit misleading; we’ll describe why in the next section)*

<table>
<thead>
<tr>
<th>features</th>
<th>l-o-o performance</th>
<th>test performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>bbox2, f12fv, startendfv</td>
<td>0.9822</td>
<td>0.8489</td>
</tr>
<tr>
<td>f1fv, f9fv, startendfv</td>
<td>0.9533</td>
<td>0.9733</td>
</tr>
<tr>
<td>f1fv, f12fv, distfv, startendfv, bboxfv2</td>
<td>0.9867</td>
<td>0.9156</td>
</tr>
<tr>
<td>f1fv, f9fv, f11fv, distfv, startendfv</td>
<td>0.9778</td>
<td>0.9644</td>
</tr>
</tbody>
</table>
support vector machines -

OSU SVM Toolkit for Matlab [http://www.ece.osu.edu/~maj/osu_svm/]

training took too long - no l-o-o

<table>
<thead>
<tr>
<th>features</th>
<th>linear k test perf.</th>
<th>quad k test perf</th>
</tr>
</thead>
<tbody>
<tr>
<td>distfv</td>
<td>0.9790</td>
<td>0.9790</td>
</tr>
<tr>
<td>bboxfv2</td>
<td>0.9789</td>
<td>0.9789</td>
</tr>
<tr>
<td>f4fv</td>
<td>0.9784</td>
<td>0.9784</td>
</tr>
<tr>
<td>startendfv</td>
<td>0.9778</td>
<td>0.9778</td>
</tr>
<tr>
<td>f1fv</td>
<td>0.9831</td>
<td>0.9831</td>
</tr>
<tr>
<td>f9fv</td>
<td>0.9778</td>
<td>0.9778</td>
</tr>
<tr>
<td>f10fv</td>
<td>0.9778</td>
<td>0.9778</td>
</tr>
<tr>
<td>f11fv</td>
<td>0.9778</td>
<td>0.9778</td>
</tr>
<tr>
<td>f12fv</td>
<td>0.9774</td>
<td>0.9775</td>
</tr>
<tr>
<td>f13fv</td>
<td>0.9778</td>
<td>0.9778</td>
</tr>
<tr>
<td>all combined</td>
<td>0.9923</td>
<td>0.9923</td>
</tr>
</tbody>
</table>
training doodles

feature extraction

build a model/train classifier

comparing results:
classifier behavior

trained classifier

“max!”
comparison

**k-nearest-neighbors** - simple, most sensitive to choice of feature extractors

**sequential ML** - simple to estimate, strictly requires stroke order

**fisher linear discriminant (1d)** - performed well

**support vector machines (lin, quad kernel)** - outperformed other methods, took significant training time
rejection

\[ \text{knn - greedily chooses } k \text{ nearest neighbors} \]
\[ \text{strokewise ML - chooses largest log likelihood} \]

\[ \implies \text{choose thresholds empirically using l-o-o validation (in theory, tricky in practice - soft thresholds difficult to manage)} \]

\[ \text{FLD and SVMs - gauge ‘specificity’ of discriminants by measuring performance as follows:} \]
\[ +1 \text{ iff all } C \text{ FLDs/SVMs are correct} \]
\[ 0 \text{ otherwise} \]
\[ \implies \text{“strict criterion”} \]
support vector machines - strict criterion

test performance with polynomial kernel:

<table>
<thead>
<tr>
<th>features</th>
<th>linear k</th>
<th>quad k</th>
<th>cubic k</th>
<th>quartic k</th>
</tr>
</thead>
<tbody>
<tr>
<td>all features</td>
<td>0.6978</td>
<td>0.6222</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f1fv, bboxfv2, distfv, f9fv, startendfv</td>
<td>0.7244</td>
<td>0.8444</td>
<td>0.8311</td>
<td>0.7867</td>
</tr>
<tr>
<td>f1fv, f9fv, f11fv, bboxfv2, distfv</td>
<td>0.6489</td>
<td>0.8400</td>
<td>0.8000</td>
<td>0.7511</td>
</tr>
</tbody>
</table>
comments?

http://people.csail.mit.edu/~emax/dt
write me: max@mit.edu

(by the way, matlab runs much faster + crashes less without the GUI!
matlab -nodesktop )

..good night!