Clumsyboard—A Program for Entering Patterns over the Keyboard

Robert West

December 11, 2006

1 The idea

Thanks 'n' props 'n' respect to Colosso for the idea!

The idea is to be able to enter simple patterns over the keyboard without having to pay attention which particular keys are actually hit. What counts shall rather be the direction and shape of the movement. Possible such patterns are:

- 'east' (finger slides right, e.g., keys `fghj`)
- 'west' (analogous)
- 'south' (finger slides down, e.g., keys `7zhn`)
- 'north' (analogous)
- 'circle' (finger draws a circle counter-clockwise, e.g., keys `7tfvbju7`)
- 'wave' (finger draws a sine wave, e.g., keys `xdrtzhbnk`)

We will achieve the pattern recognition in three steps:

1. An off-line training process: The training program prompts the user to draw a certain pattern. This kind of trial is repeated several times for each pattern (40 times in our case), and so the program collects labeled training data.
2. From these training samples we estimate the parameters of the probability model (see next section).
3. When the program is run and input is entered by the user we compute the pattern that makes this specific input most likely; i.e., we follow a maximum-likelihood approach (see section 3).

2 The model

2.1 The Bayes net

We assumed the following model: the key pressed at time t, $K(t)$, depends only on the pattern P being drawn and on the key $K(t-1)$ that was pressed

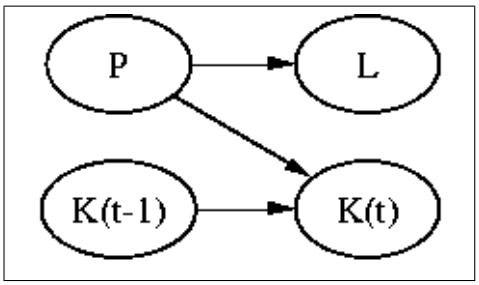

Figure 1: The Bayes model we used. P: the pattern drawn, L: the length of the input key sequence, $K(t)$: the key pressed at time t.

immediately before, at time $t-1$; i.e., we are using a simple Markov model. Also, the length L of the input key sequence depends on which pattern is being drawn. Figure 1 shows a graphical description of the Bayes net.

P is the hidden variable that is to be guessed given the measured data $K(1), \ldots, K(L)$ and L.

The model could be even simpler, by eliminating the variable L: This works most of the time, but difficulties arise: e.g., the key sequence `tfv` could itself mean 'south', but it also occurs in instances of the pattern 'circle': `ztfvbnjuzt`. Our estimation algorithm could then not know which is true since, due to the very simple Markov assumption, it considers only what has happened immediately before. In fact, the pattern 'south' makes the key sequence `tfv` less likely than the pattern 'circle' does; this is because in the training phase the 'south' down-stroke was made at a greater number of different horizontal coordinates on the keyboard than the circle (since the circle is broader than the 'south' stroke it can be made in fewer locations), which means it was done less often at `tfv` than the circle.

So the length provides the kind of global information we need: In the training the program learnt that it is very unlikely that a circle consists of only three keys but that a 'south' stroke is very likely to do so. In our maximum-likelihood estimation this will override the fact that the *sub*-sequence `tfv` is more likely for circles than for 'south' strokes.

2.2 Learning the net parameters

The following probability distributions describe the whole model:

- The probability of a certain key being pressed, given the pattern being drawn and the key pressed before: $\Pr(K(t)|K(t-1), P)$.
- The probability of a key sequence having a certain length, given the pattern being drawn: $\Pr(L|P)$.

We estimate the parameters simply by counting in the example set collected during training:

$$\Pr(K(t) = x | K(t-1) = y, P = p) := \frac{\#(K(t) = x, K(t-1) = y, P = p)}{\#(K(t-1) = y, P = p)}$$

$$\Pr(L = \ell | P = p) := \frac{\#(L = \ell, P = p)}{\#(P = p)}$$

3 Applying the model for on-line estimation

Once the model has been computed off-line from the training samples we can run the actual 'Clumsyboard' program: the user enters a pattern by sliding his fingers over the keyboard. From the data entered we must now guess which one out of the known patterns the user intended to input: We must guess which pattern p^* is most likely given the key strokes recorded:

$$\begin{aligned}
p^* &= \arg\max_p \Pr(P = p | K(1) = k_1, \ldots, K(\ell) = k_\ell, L = \ell) \\
&= \arg\max_p \frac{\Pr(K(1) = k_1, \ldots, K(\ell) = k_\ell, L = \ell | P = p) \Pr(P = p)}{\Pr(K(1) = k_1, \ldots, K(\ell) = k_\ell, L = \ell)} \\
&= \arg\max_p \Pr(K(1) = k_1, \ldots, K(\ell) = k_\ell, L = \ell | P = p) \\
&= \arg\max_p \Pr(K(1) = k_1, \ldots, K(\ell) = k_\ell | P = p) \Pr(L = \ell | P = p) \\
&= \arg\max_p [\Pr(K(\ell) = k_\ell | K(1) = k_1, \ldots, K(\ell - 1) = k_{\ell-1}, P = p) \cdot \\
&\quad \Pr(K(1) = k_1, \ldots, K(\ell - 1) = k_{\ell-1} | P = p) \Pr(L = \ell | P = p)] \\
&= \arg\max_p [\Pr(K(\ell) = k_\ell | K(\ell - 1) = k_{\ell-1}, P = p) \cdot \\
&\quad \Pr(K(\ell - 1) = k_{\ell-1} | K(1) = k_1, \ldots, K(\ell - 2) = k_{\ell-2}, P = p) \cdot \\
&\quad \Pr(K(1) = k_1, \ldots, K(\ell - 2) = k_{\ell-2} | P = p) \Pr(L = \ell | P = p)] \\
&= \ldots \\
&= \arg\max_p [\Pr(L = \ell | P = p) \Pr(K(1) = k_1 | P = p) \cdot \\
&\quad \prod_{t=2}^{\ell} \Pr(K(t) = k_t | K(t - 1) = k_{t-1}, P = p)]
\end{aligned}$$

The last formula can be computed from the model that has been previously learned ($\Pr(K(1) = k_1 | P = p)$ can be computed by summing over the second key in the model). In the transformations we first used Bayes' rule; then the assumption that all patterns are a-priori equally likely (this makes our approach a maximum-likelihood one, as opposed to a maximum-a-posteriori one) and the fact that the a-priori probability of the data measured doesn't depend on the pattern; then the fact that in our Bayes model L and the $K(t)$'s are independent given P; we then pulled some terms into the conditional part; we made use of the Markov assumption; finally we iterated the last two steps.

There was another detail that had to be considered: Single entries of the *sampled* probability table $\Pr(K(t)|K(t-1), P)$ could be zero (because we never came across that specific situation in the few training examples) although the *actual* probability could be a small positive number. Then the product in the last formula would be 'flattended' to zero just because one single factor is (falsely) zero. We obviously don't want this because it destroys all the information we have. So when building the probability table we pretended that we had seen one single transition from each key to every other key (although we actually

hadn't), i.e., we assumed that for every key there is a small probability to reach any other key from there next. We made the same assumption for the other table $\Pr(L|P)$.