Learning with Perceptrons; User Controls

February 11, 2009

Problem Set 3

- What: Classifying handwritten digits as 3s or 5s.
- When: Problem set will be posted tomorrow, due February 24.
- Who: You and a partner.
- How: Perceptron learning.

1. Learning with Perceptrons

2. User Controls

The classification problem.

Setup

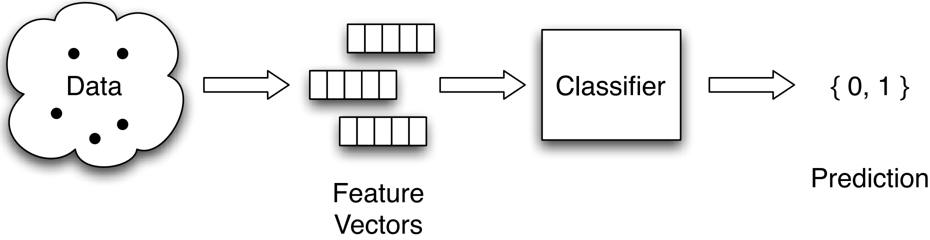
Data from some domain which can be given meaningful labels.

Data	**Labels**
emails	{Spam, NotSpam}
stock charts	{Buy, Sell, Hold}
handwritten letters	{A, B, C . . . }

Goal

Train a program to assign labels to domain elements.

Binary classifiers take vectors and return bits.



Arbitrary data elements are mapped into *feature vectors*.
Splits classification into two problems:

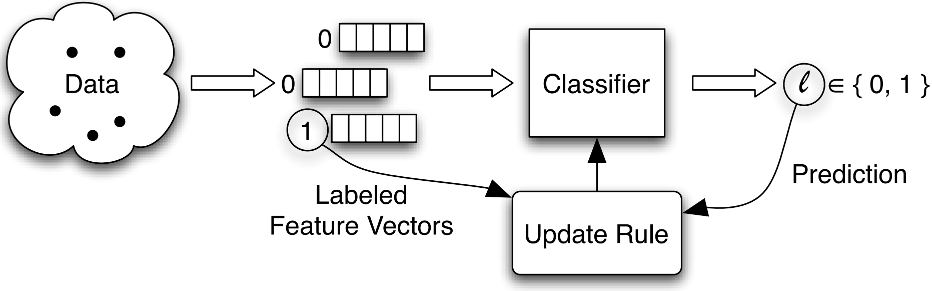
- Generating feature vectors—domain specific
- Classifying feature vectors—general purpose

Not in this course: Picking Features

Feature Selection

- Requires lots of trial and error
- Deep knowledge of application domain helpful
- Essential to getting good classification results

A Machine Learning Approach: Train the classifier.



- Pick a *training example*, **x**, with known label ℓ.
- Call the classifier input **x**.
- Let the classifier make a prediction: label *p*.
- If $\ell \neq p$, update the classifier using ℓ, *p*, and **x**.
- Repeat many, many times.

Perceptrons are simple classifiers.

- Perceptrons feature vectors with components in $[-1, +1]$,
- and label examples with a 1 or a 0.
- A perceptron maintains a weight, w_i, for each feature.
  - $w_i > 0 \Rightarrow$ feature i correlates with 1 label.
  - $w_i < 0 \Rightarrow$ feature i correlates with 0 label.
  - Large $|w_i| \Rightarrow$ feature i is important.

Making a perceptron prediction (I/II)

To make a prediction about $\mathbf{x} = \langle x_1, x_2 \ldots, x_n \rangle$

- For each feature, i, calculate a vote, $v_i = w_i x_i$.
- Sum the votes

$$v_{total} = \sum_{i=1}^{n} w_i x_i = \mathbf{w} \cdot \mathbf{x}.$$

- If the tally is positive ($v_{total} > 0$) return 1, else return 0.

Making a perceptron prediction (II/II)

Another way: $p = H(\mathbf{w} \cdot \mathbf{x})$ where

- p: the prediction
- $\mathbf{w}$: the weight vector
- $\mathbf{x}$: the feature vector
- H: the threshold function,

$$H(z) = \begin{cases} 1 & z > 0 \\ 0 & \text{otherwise} \end{cases}$$

Training a perceptron

Let’s say we know a particular feature vector $\mathbf{x}$ should be labeled ℓ_x. And our perceptron returns $p = H(\mathbf{w} \cdot \mathbf{x})$.

Two main cases:

Training a perceptron

Let's say we know a particular feature vector $\mathbf{x}$ should be labeled ℓ_x. And our perceptron returns $p = H(\mathbf{w} \cdot \mathbf{x})$.

Two main cases:

- $p = \ell_x$. The perceptron did the right thing. We're done.
- $p \neq \ell_x$. Two sub-cases:

Training a perceptron

Let's say we know a particular feature vector $\mathbf{x}$ should be labeled ℓ_x. And our perceptron returns $p = H(\mathbf{w} \cdot \mathbf{x})$.

Two main cases:

- $p = \ell_x$. The perceptron did the right thing. We're done.
- $p \neq \ell_x$. Two sub-cases:
    - $p = 1, \ell_x = 0$. The vote was too high. Next time we get a vector like $\mathbf{x}$ we should vote lower. Update weights by $\mathbf{w} = \mathbf{w} - \alpha\mathbf{x}$.

Training a perceptron

Let's say we know a particular feature vector $\mathbf{x}$ should be labeled ℓ_x. And our perceptron returns $p = H(\mathbf{w} \cdot \mathbf{x})$.

Two main cases:

- $p = \ell_x$. The perceptron did the right thing. We're done.
- $p \neq \ell_x$. Two sub-cases:
  - $p = 1, \ell_x = 0$. The vote was too high. Next time we get a vector like $\mathbf{x}$ we should vote lower. Update weights by $\mathbf{w} = \mathbf{w} - \alpha\mathbf{x}$.
  - $p = 0, \ell_x = 1$. The vote was too low. Next time we get a vector like $\mathbf{x}$ we should vote higher. Update weights by $\mathbf{w} = \mathbf{w} + \alpha\mathbf{x}$.

Training a perceptron

Let's say we know a particular feature vector $\mathbf{x}$ should be labeled ℓ_x. And our perceptron returns $p = H(\mathbf{w} \cdot \mathbf{x})$.

Two main cases:

- $p = \ell_x$. The perceptron did the right thing. We're done.
- $p \neq \ell_x$. Two sub-cases:
  - $p = 1, \ell_x = 0$. The vote was too high. Next time we get a vector like $\mathbf{x}$ we should vote lower. Update weights by $\mathbf{w} = \mathbf{w} - \alpha\mathbf{x}$.
  - $p = 0, \ell_x = 1$. The vote was too low. Next time we get a vector like $\mathbf{x}$ we should vote higher. Update weights by $\mathbf{w} = \mathbf{w} + \alpha\mathbf{x}$.

Learning rate α is a "knob" that controls how sensitive the perceptron is to new information.

The Perceptron Update Rule

$$\mathbf{w}_{new} = \mathbf{w}_{old} + (\ell_x - p)\alpha\mathbf{x}$$

where

- $\mathbf{w}_{new}$: the new weight vector
- $\mathbf{w}_{old}$: the original weight vector
- $\mathbf{x}$: a training example
- ℓ_x: correct label for training example $\mathbf{x}$
- p: the perceptron's prediction for $\mathbf{x}$ (obtained when still using weight vector $\mathbf{w}_{old}$)
- α: a fixed learning rate

Note term $(\ell_x - p)$ encodes last slide's case analysis.

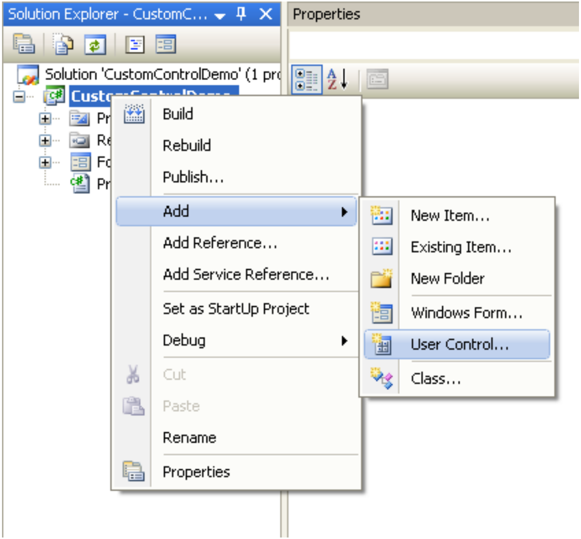
1 Learning with Perceptrons

2 User Controls

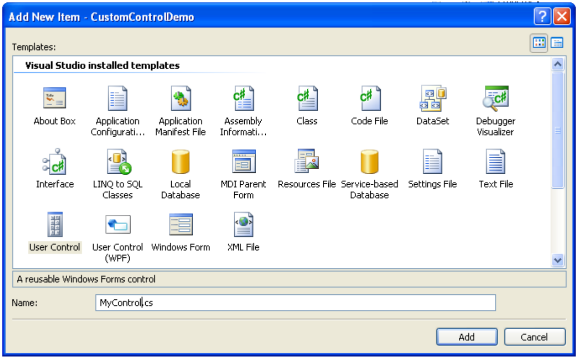
User Controls Summary

- User controls are programmer build controls that can be added to the Visual Studio designer.
- User controls integrate with other elements when designing forms in VS Viewer.
- As with rest of Windows.Forms framework, there's nothing special about custom controls: everything maps to C# code.

Creating a user control with the gui



Creating a user control with the gui



Add New Item - CustomControlDemo

Templates:

Visual Studio installed templates

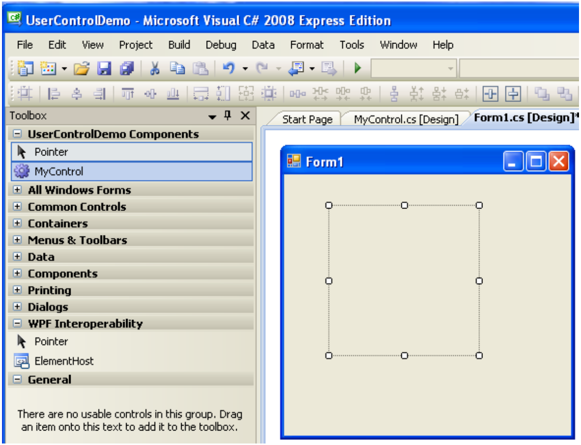
A reusable Windows Forms control

Name: MyControl.cs

Creating a user control with the gui

Build the project…

Creating a user control with the gui

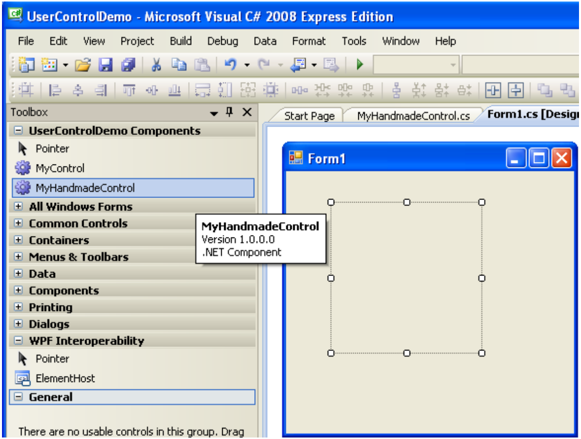


There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

Creating a user control by hand

```
using System.Windows.Forms;

class MyHandmadeControl : UserControl { }
```



MyHandmadeControl
Version 1.0.0.0
.NET Component

There are no usable controls in this group. Drag

Drawing in a user control

- Whenever Windows displays your control, it raises a Paint event.
- Calling the control's .Invalidate() method will also raise a Paint event. (Useful for forcing redraws.)
- A user control's OnPaint(PaintEventArgs e) virtual method usually handles paint requests. Override this to do custom drawing.
- It's also possible to handle the Paint event directly using a delegate.

Handling paint requests: Code example

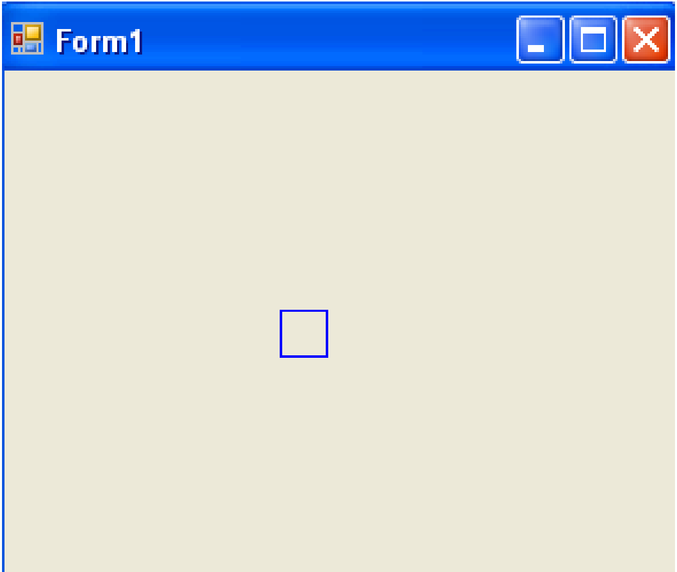
```
using System.Windows.Forms;

// Drawing namespace includes Graphics class
// and primitives used by Graphics class
using System.Drawing;

class MyHandmadeControl : UserControl
{
  protected override void OnPaint(PaintEventArgs e)
  {
    // Graphics object contains methods to
    // actually draw
    var g = e.Graphics;
    g.DrawRectangle(Pens.Blue, 0, 0, 20, 20);
  }
}
```

Handling paint requests: Screenshot

Adding a MyHandmadeControl to a form (using the designer) yields the following program:



Some Drawing and Graphics concepts.

- Drawing.Pen—Objects describing color, etc. of lines and curves.
- Drawing.Brush—Objects describing color, etc. of filled in regions.
- Drawing.Font—Objects describing fonts.

⋮

- Graphics.DrawCurve()–Draws a curve using a pen.
- Graphics.FillRectangle()—Fills a rectangular region using a brush.

⋮