CSCI 631
Foundations of Computer Vision

Ifeoma Nwogu
ion@cs.rit.edu

Logistic Regression

Schedule

- Last 2 classes
  - Classifiers
    - Nearest neighbor
    - SVM with different kernels
- Today
  - Regression vs. classification
  - Logistic Regression classifier
- Readings for today:
  - Intro to Statistical Learning in R (pages 128-138) – link provided on myCourses

Supervised vs. unsupervised learning

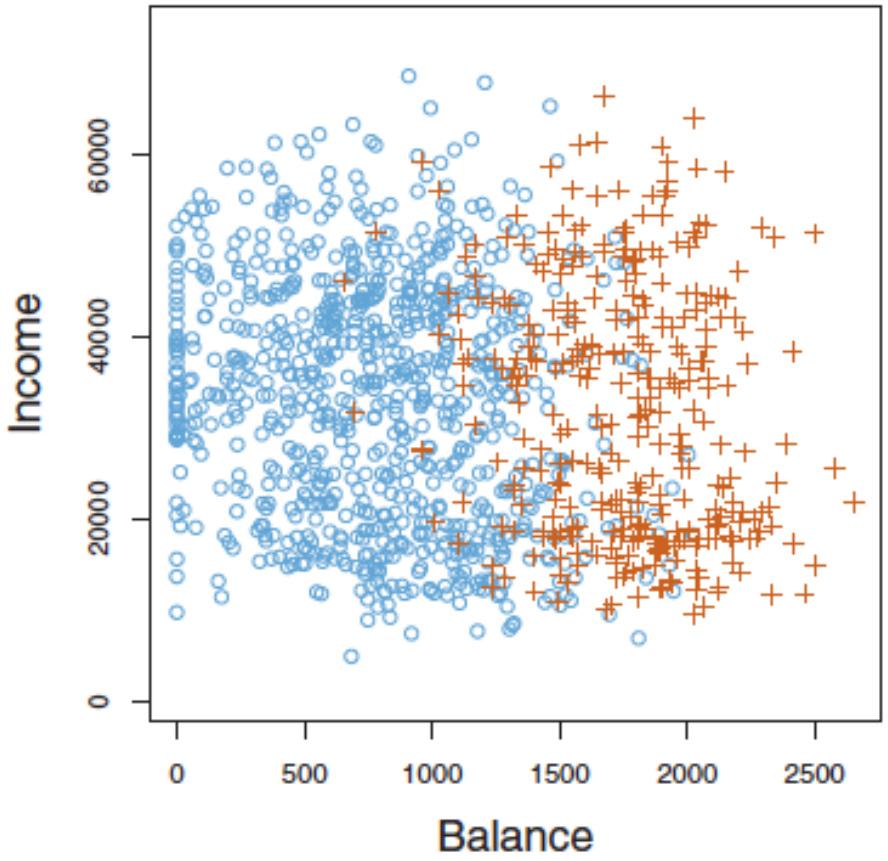
- Unsupervised learning - $x_1, x_2 \ldots\ldots\ldots x_N$
  - No labels given
  - Goal is to determine the natural clusters of data
    - Useful for segmentation and other unconstrained learning
- Supervised learning - $(x_1,y_1)\ldots\ldots\ldots(x_N,y_N)$
  - Class labels are provided for the data given
  - Goal is to learn
    - what characterizes each class (different from other classes) and
    - how to assign a new point to the right class

Two forms of supervised learning

- Regression – used to predict continuous values
  - Find a surface that best conforms to the distribution of the data
- Classification – used to predict the class of a data point
  - Find the decision boundaries that best separate different classes of data
- Other forms of learning - reinforcement learning, transfer learning etc.

Regression or Classification Problem? Why?

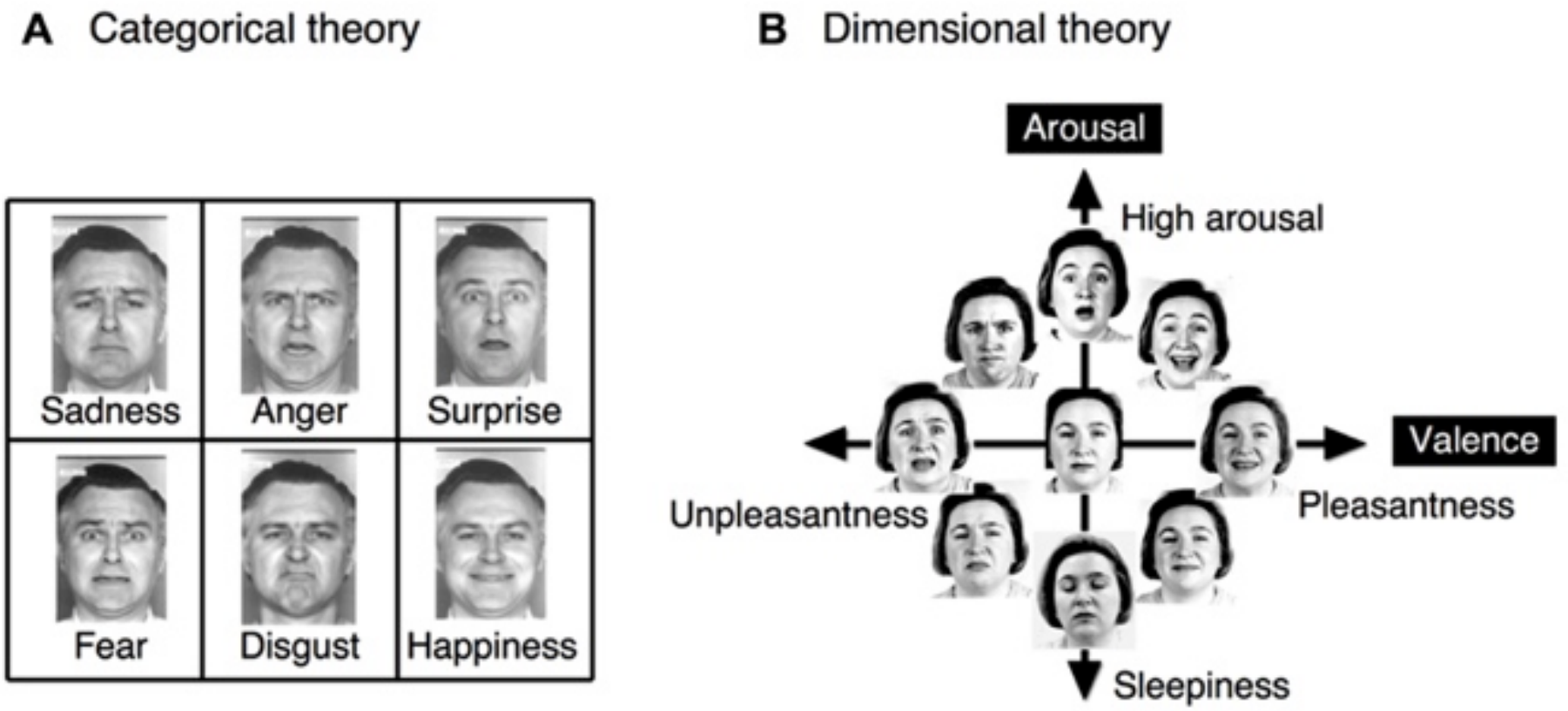
- Goal: Given the income and balance values of a client, find out if he/she would default on credit card payment



Example taken from Intro to Stat Learning in R

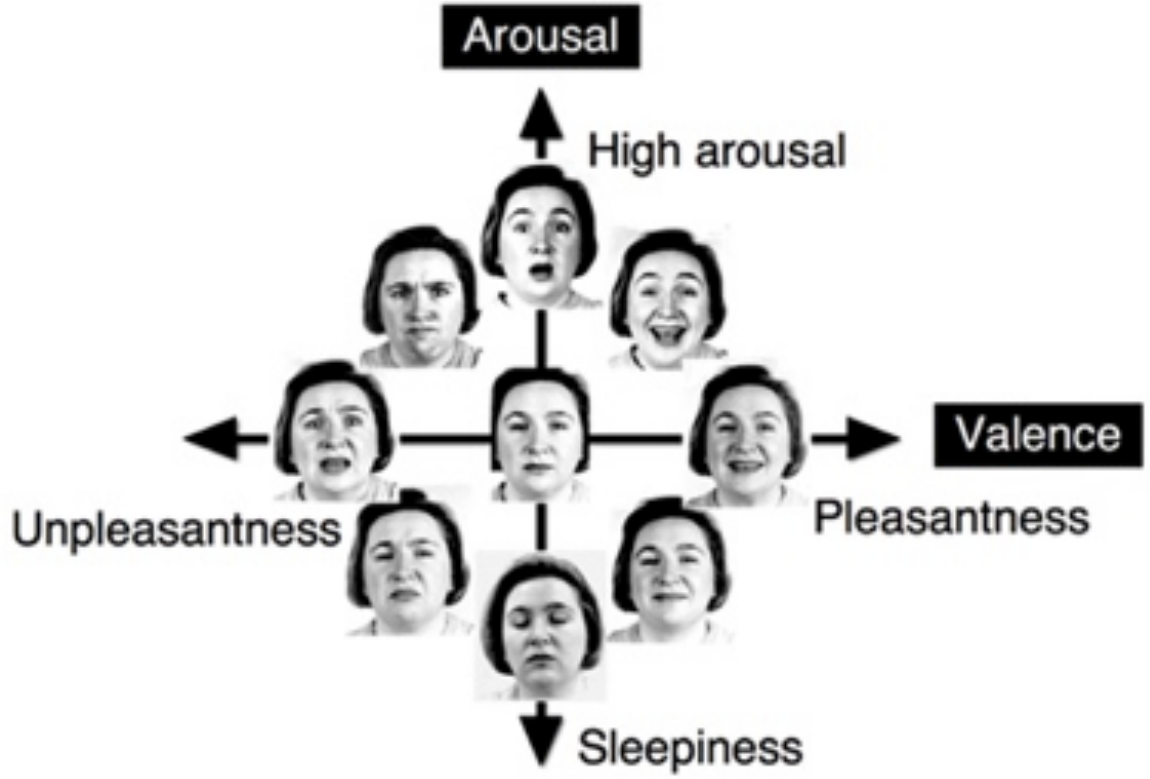
Regression vs. classification in computer vision

- Most problems in computer vision are more related to classification than regression



Example of image-driven regression

- Facial expression analysis using the dimensional theory of emotion



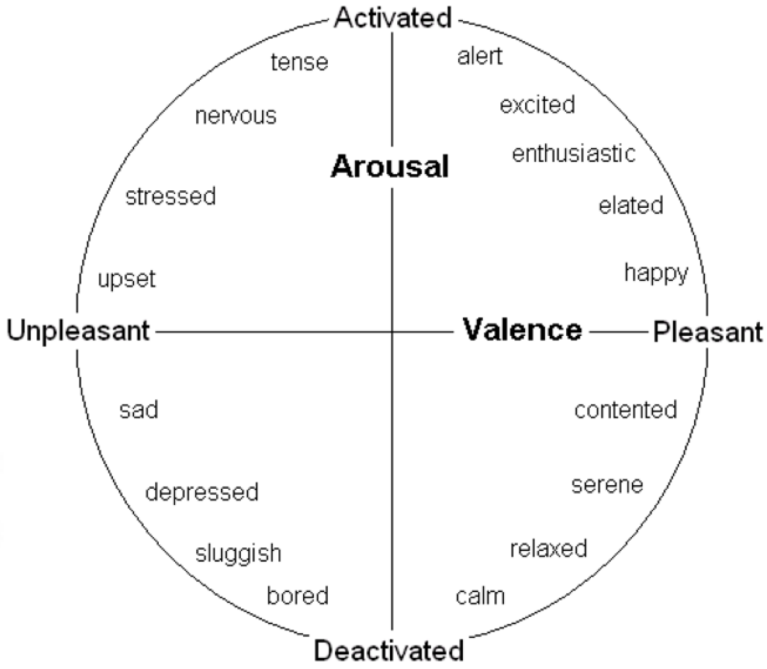


Image-driven classification

- One of the longest standing problems in vision
  - Land mass analysis, botany, robotics, scene recognition, document analysis, satellite imagery, astronomy, etc
- See Kaggle.com for several image classification competitions

Classification example

Image from Land Info website

Classification strategies

- Non-parametric
  - nearest neighbor
- Probabilistic
  - histogram
  - logistic regression
- Decision boundary
  - SVM

Linear regression review

- *Given a set of points* $(x_1,y_1)……(x_N,y_N)$,
- Find the linear model that best fits the points
- Note that y_i is a continuos value
  - (the dependent variable)

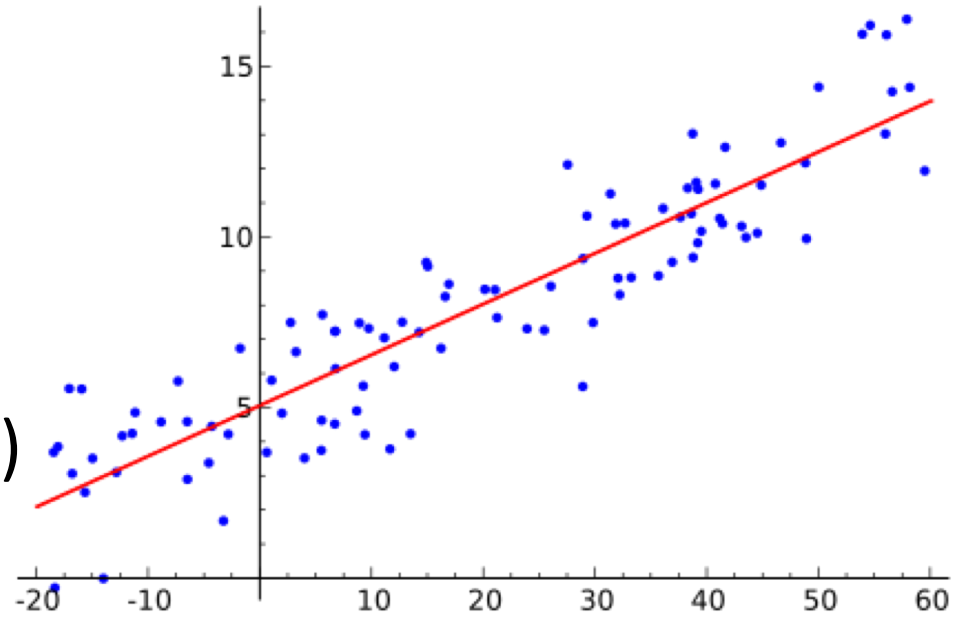


Image by Sewaqu - https://commons.wikimedia.org/w/index.php?curid=11967659

Components of the learning process

1. Extract features of interest from images
2. Determine your general prediction function
   - Parameters will need to be learned
3. Adopt a cost/error/loss function to learn the optimal parameter values
4. Evaluate the classifier performance

Linear regression continued

Given a data set with *n* points:

$$\{y_i, x_{i1}, x_{i2}, \ldots, x_{ip}\}^n$$

We assume a linear regression model, so that

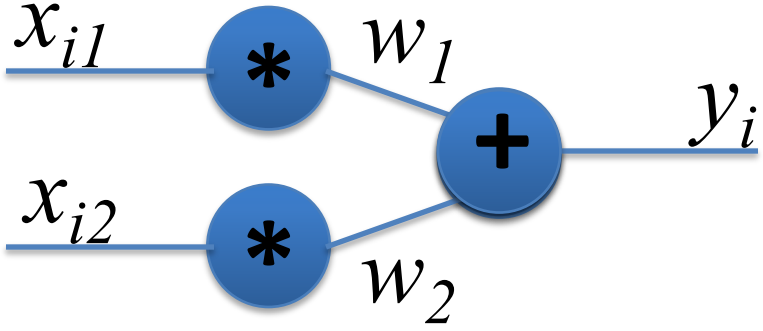
$$y_i = w_1 x_{i1} + w_2 x_{i2} + \ldots + w_p x_{ip} + \epsilon$$

In matrix form, we now have:

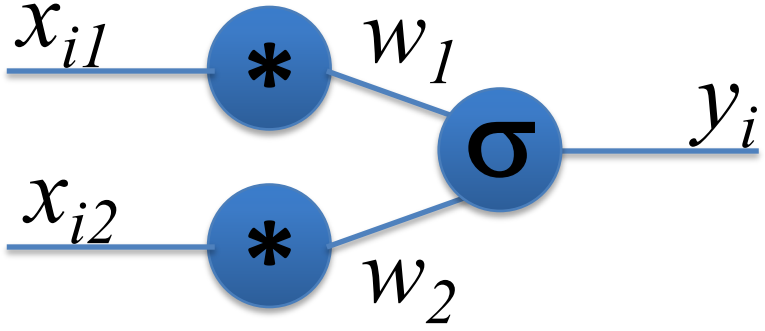
$$\mathbf{y} = \mathbf{x}^\top \mathbf{W} + \epsilon$$

Logistic regression as an extension of linear regression

Linear Regression



Logistic regression



- σ(z) =
- σ(z) = tanh(z) or $\dfrac{1}{1+e^{-z}}$

Components of the logistic regression process

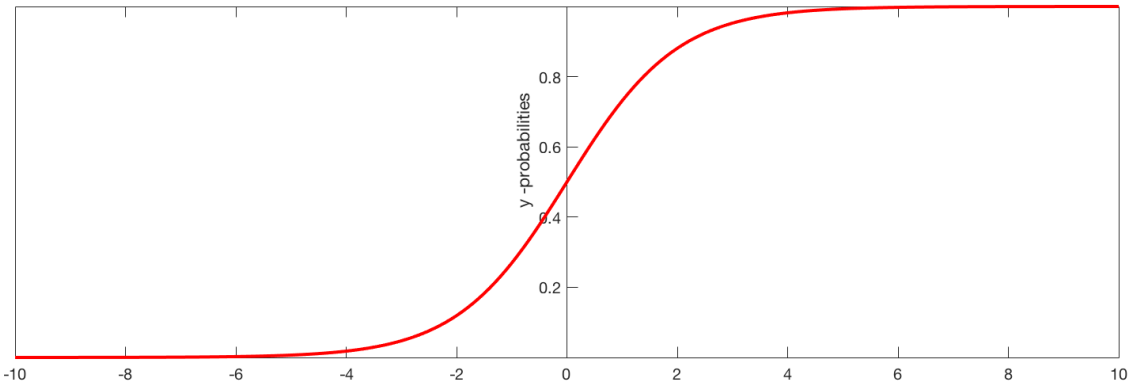
1. Extract features of interest from images
2. **Determine your general prediction function**
   - **Parameters will need to be learned**
3. Adopt a cost/error/loss function to learn the optimal parameter values
4. Evaluate the classifier performance

Logistic regression – prediction function

The sigmoid function $\sigma(z)$ = $\dfrac{1}{1+e^{-z}}$

```
>> z =  -10:0.1:10;
>> y = 1./(1 + exp(-z));
>> figure; plot(z,y,'r','LineWidth',2)
>> |
```

y_



Predictor function

$y = \sigma(W^T x)$

Logistic regression – prediction function

- The prediction function $\sigma(.) = \dfrac{1}{1+e^{-(W^T X)}}$
  - we are given $\boldsymbol{X}$, how can we get $\mathbf{W}$?
    - try random guesses?
    - accuracy = 25%, 34%, constantly changing, highly unstable!!!
    - Find the optimal solution for W using a cost function

Components of the logistic regression process

1. Extract features of interest from images
2. Determine your general prediction function
   - Parameters will need to be learned
3. **Adopt a cost/error/loss function to learn the optimal parameter values**
4. Evaluate the classifier performance

Logistic regression-loss function

Cross-entropy loss function

- $J = -\{t \log(y) + (1-t) \log(1-y)\}$, where

  t = target label

  y = predicted label (from the logistic regression)

- For all the training examples,

$$J = -\sum_{n=1}^{N} t_n \log(y_n) + (1 - t_n) \log(1 - y_n)$$

- Compare with mean squared loss:

$$J = \sum_{n=1}^{N} (t_n - y_n)^2$$

Working with loss functions

- Having experience with different loss functions based on the nature of your data and parameters
- Maximizing the likelihood function (related to above)

Gradient descent

$$\frac{\partial J}{\partial w} = \sum_{n=1}^{N} (y_n - t_n) x_n$$

In vector form, $\frac{\partial J}{\partial w} = X^T (Y - T)$

For the bias term,

$$\frac{\partial J}{\partial w_0} = \sum_{n=1}^{N} (y_n - t_n)$$

Gradient descent

We use gradient descent to find the optimal weights for the predictions

- J(w) is a convex function (-ve sign), no local optima
- But no closed form solution to maximize J
- Convex functions easy to optimize with gradient descent
  - Write the expression for the the gradient of J(w)
  - Use an update rule to traverse the space until a min is reached

Step size, $\eta > 0$

Update rule: $\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w})$

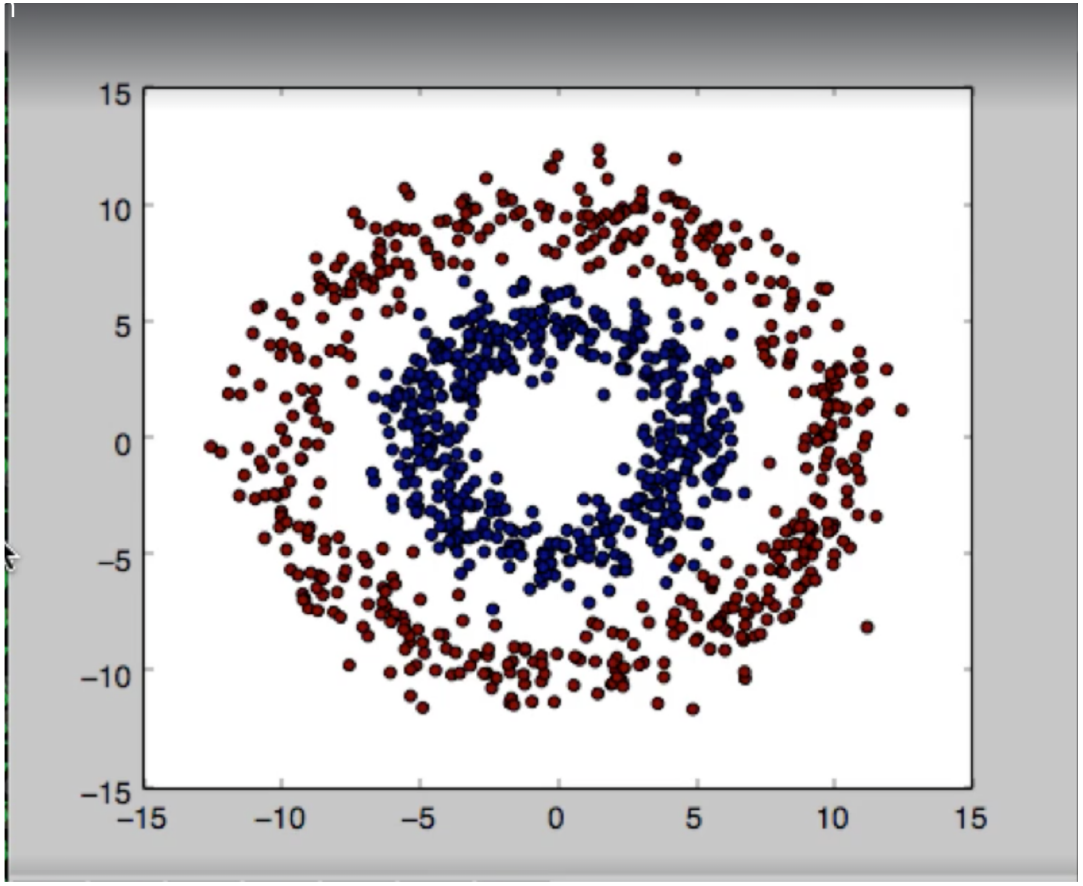
$$w_i^{(t+1)} \leftarrow w_i^{(t)} - \eta \frac{\partial l(\mathbf{w})}{\partial w_i}$$

Logistic regression as a classifier

- Does not work as well when classes are linearly well separated!
- To address this problem, we use regularizers to induce penalties on the cost function
- A multiclass extension of LR is called the softmax classifier which we will use later
- Linear discriminant analysis (LDA) and naïve Bayes classifiers are special cases of LR

The donut problem

- How can we use LR to classify this dataset?



Components of the logistic regression process

1. Extract features of interest from images
2. Determine your general prediction function
   - Parameters will need to be learned
3. Adopt a cost/error/loss function to learn the optimal parameter values
4. **Evaluate the classifier performance**

Classifiers: Crucial Points

- Classifiers accept features, return decision
  - and are often trained using data
- Very mature training processes now
  - Always try standard methods first
    - SVM
    - kNN
    - Logistic regression
- Evaluate with separate test data
  - look at total error rate
  - ROC
  - class confusion matrix
  - (occasionally) total risk
    - typically when one has a loss model

Summary: Classifiers

- Nearest-neighbor and k-nearest-neighbor classifiers
- Support vector machines
  - Linear classifiers
  - Margin maximization
  - Non-separable case
  - The kernel trick
  - Multi-class SVMs
- Logistic regressor (pre-cursor to neural networks)
- There are so many other classifiers out there
  - Nneural networks, boosting, decision trees/forests, …

Evaluating binary classifiers

- Always
  - train on training set, check on validation set, evaluate on test set
    - test set performance might/should be worse than training set
- Options
  - Error rates
    - Total error must be less than 50% for two classes
  - Receiver operating curve
    - When we use a hand-tuned parameter (e.g.threshold)
      - we might use different thresholds
  - Class confusion matrix
    - for multiclass

Evaluating classifiers

- Always
  - train on training set, evaluate on validation dataset, final test on test data
    - test set performance might/should be worse than training set
- Options
  - Total error rate
    - always less than 50% for two class
  - Receiver operating curve
    - because we might use different thresholds
  - Class confusion matrix
    - for multiclass
  - Average precision

Confusion Matrix

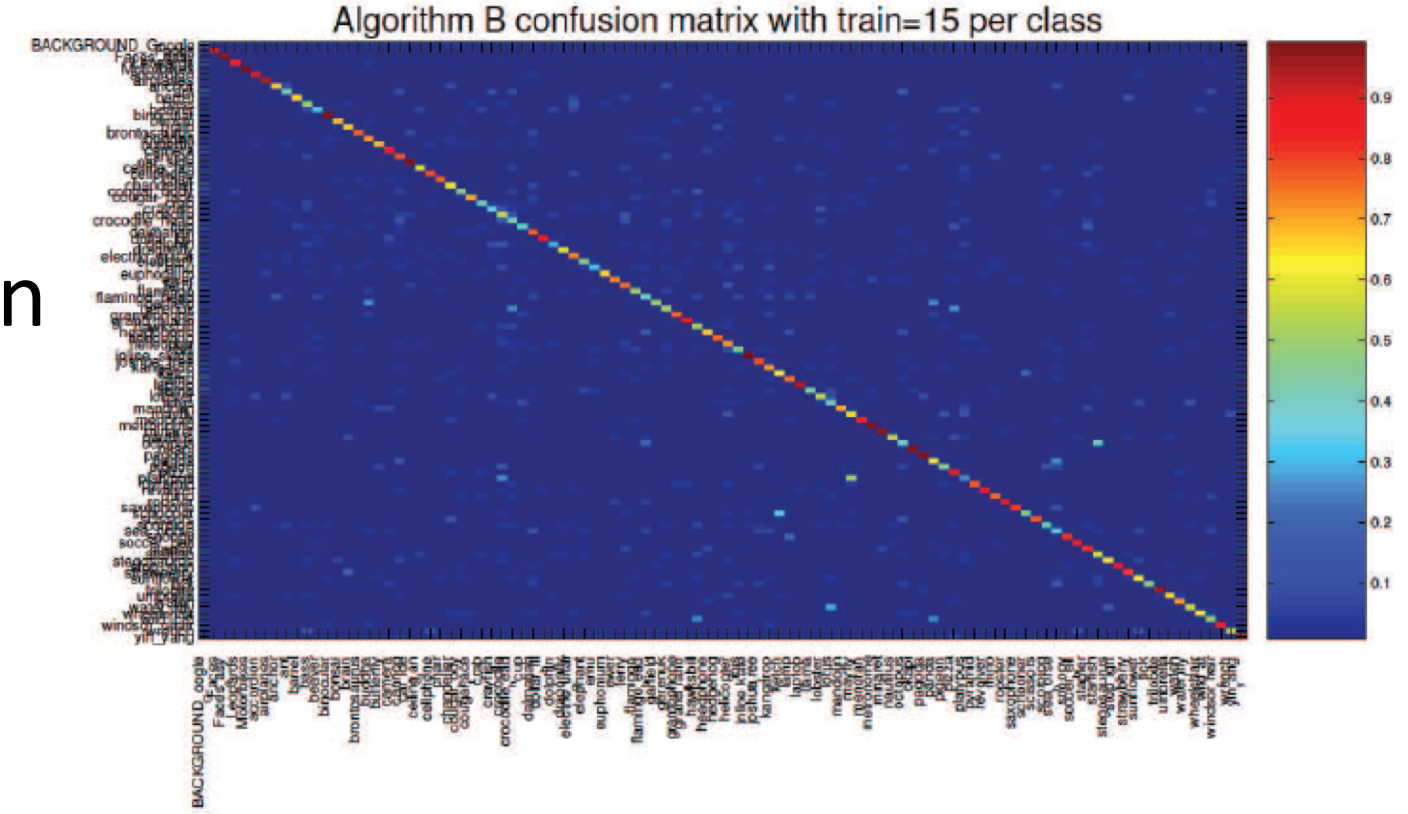


FIGURE 15.3: An example of a class confusion matrix from a recent image classification system, due to Zhang *et al.* (2006*a*). The vertical bar shows the mapping of color to number (warmer colors are larger numbers). Note the redness of the diagonal; this is good, because it means the diagonal values are large. There are spots of large off-diagonal values, and these are informative, too. For example, this system confuses: schooners and ketches (understandable); waterlily and lotus (again, understandable); and platypus and mayfly (which might suggest some feature engineering would be a good idea). *This figure was originally published as Figure 5 of "SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition," by H. Zhang, A. Berg, M. Maire, and J. Malik, Proc. IEEE CVPR, 2006,* © *IEEE, 2006.*

Evaluating a binary classifier

Predicted \ Actuals	+	-
+	True Positive	False Positive
-	False Negative	True Negative

Classification rate =

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Also known as **accuracy**

- We want to **maximize TP and TN**
- We want to **minimize FP and FN**

Evaluating a binary classifier

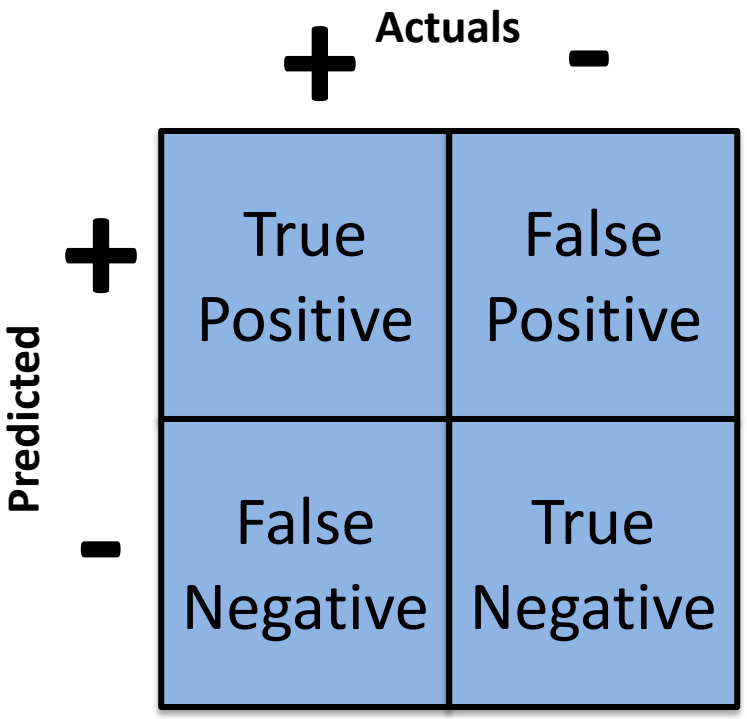
Predicted \ Actuals	+	-
+	True Positive	False Positive
-	False Negative	True Negative

Misclassification rate =

$$\frac{FP + FN}{TP + TN + FP + FN}$$

Also known as **error rate**

Evaluating a binary classifier cont'd



$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

Precision: When it predicts yes, how often is it correct?

Recall: When it's actually yes, how often does it predict yes?

- Recall is also known as **TPR** or **sensitivity**

Evaluating a binary classifier cont'd

Predicted \ Actuals	+	-
+	True Positive	False Positive
-	False Negative	True Negative

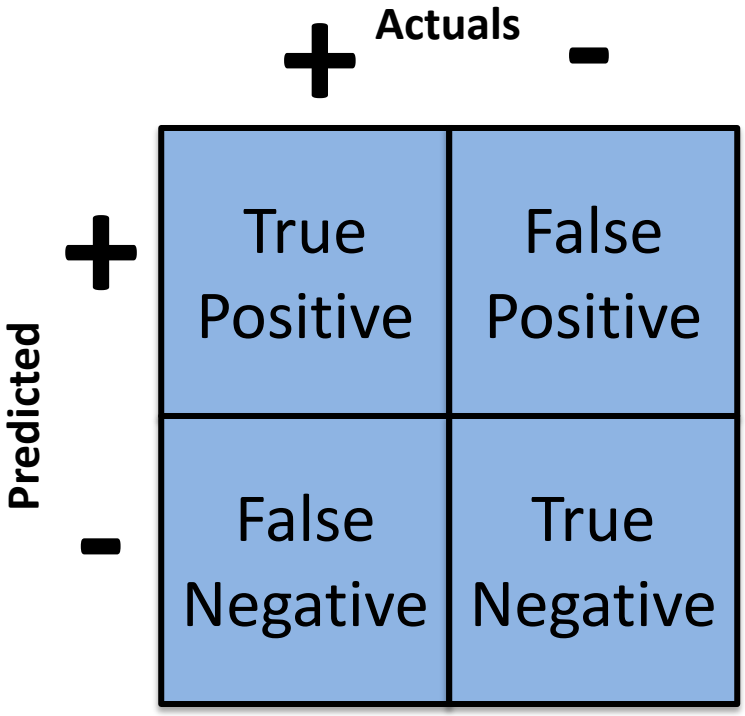
$$\text{Specificity} = \frac{TN}{TN + FP}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

Sensitivity: Can it mostly find what it's looking for

Specificity: Can it mostly not mistake something else for what it's looking for

Evaluating a binary classifier cont'd



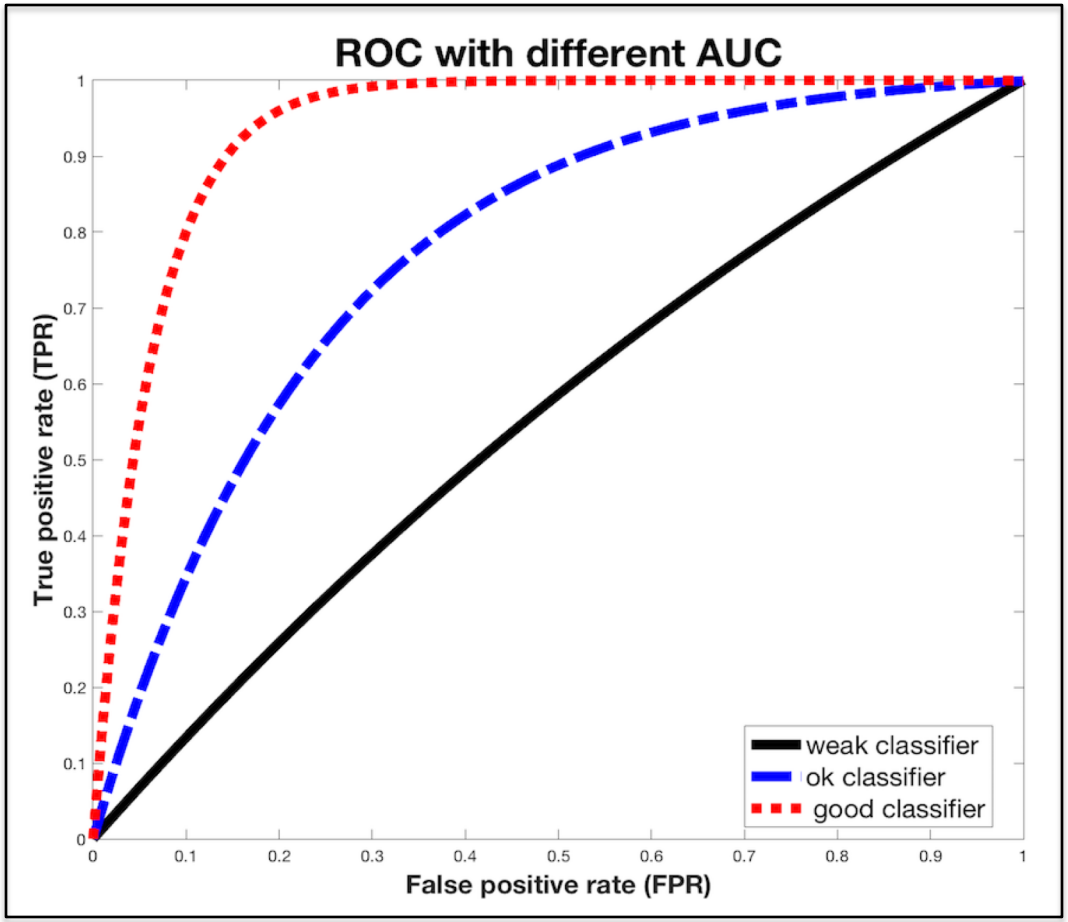
F1-score =

$$\frac{2 * \mathrm{Pr}\,ecison * \mathrm{Re}\,call}{\mathrm{Pr}\,ecision + \mathrm{Re}\,call}$$

Also known as **F-score** or **F-measure**

ROC and AUC

- Threshold does not have to be 0.5;
  - we can use any threshold
- TPR = $\frac{TP}{TP + FN}$
- FPR = $\frac{FP}{TN + FP}$



- **ROC** is generated by Plotting TPR vs. FPR for many threshold values
- **AUC** – area under the curve indicates the goodness of the classifier

Receiver operating curve

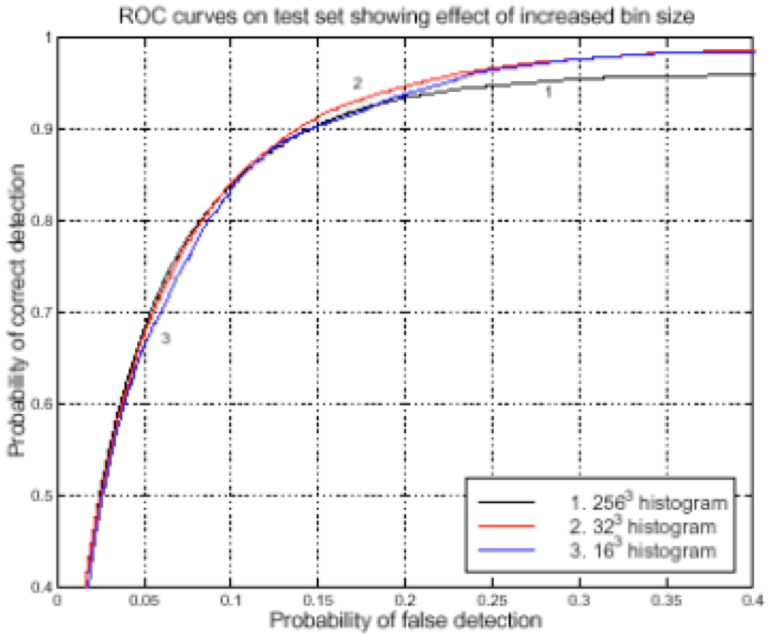


FIGURE 15.4: The receiver operating curve for a classifier, used to build a skin detector by Jones and Rehg. This curve plots the detection rate against the false-negative rate for a variety of values of the parameter θ. A perfect classifier has an ROC that, on these axes, is a horizontal line at 100% detection. There are three different versions of this classifier, depending on the detailed feature construction; each has a slightly different ROC. *This figure was originally published as Figure 7 of "Statistical color models with application to skin detection," by M.J. Jones and J. Rehg, Proc. IEEE CVPR, 1999 © IEEE, 1999.*

Diagnostic testing example

	True Disease State			
		Diseased (+)	Healthy(-)	**Total**
Test results	Diseased (+)	44	10	54
	Healthy(-)	23	60	83
	Total	67	70	137

Total of 137 people

- 67 sick, 70 healthy
  - Acc = 104/137 = 76%
  - Prec = 44/54 = 81%
  - TPR = 44/67 = 65.6%
  - FPR = 10/70 = 14.3%

TPR is the percentage of tested individuals who correctly receive a positive test result.

FPR is the percentage of healthy individuals who incorrectly receive a positive test result.

Sensitivity and Specificity

- Sensitivity (TPR or recall) measures the proportion of actual positives that are correctly identified as such
  - rarely overlooks what it is looking for
- Specificity (TNR) measures the proportion of actual negatives that are correctly identified as such
  - rarely mistakes something else for what it is looking for
- For practical reasons, tests in medical diagnosis with sensitivity and specificity values above 90% have high credibility

Melanoma testing example

	True Disease State			
		Diseased (+)	Healthy(-)	**Total**
Test results	Diseased (+)	0	0	0
	Healthy(-)	2	98	100
	Total	2	98	100

Total of 100 people tested

- 2 sick, 98 healthy
  - Acc = 98/100 = **98%**
  - Sensitivity = 0/2 = 0%
  - Specificity = 98/98 = 100%

Questions