Lecture 17: Classification Models

Goals for today:

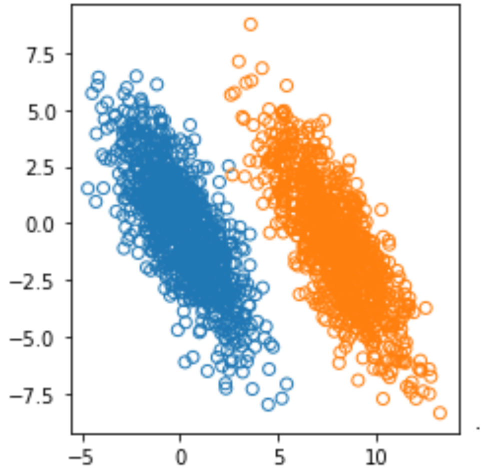
- Distinct types of classification problems.
- New error metrics for classification models
- Several new models for classification.

Examples

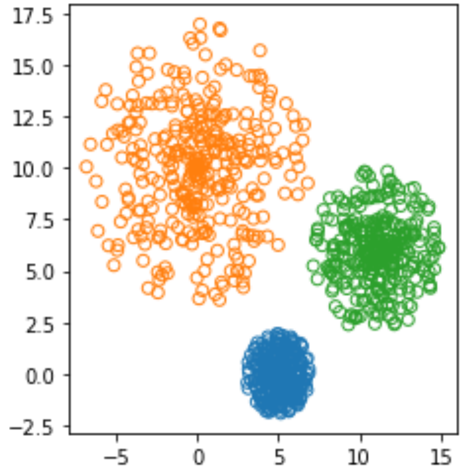
Given features $\mathbf{x}$, predict $y_\mathrm{i} = f\left(\mathbf{x}_\mathrm{i}\right)$, where $y_\mathrm{i} \in [1, 2, ..., C]$

- **Binary** (C=2) versus **Multiclass** (C>2)

Binary

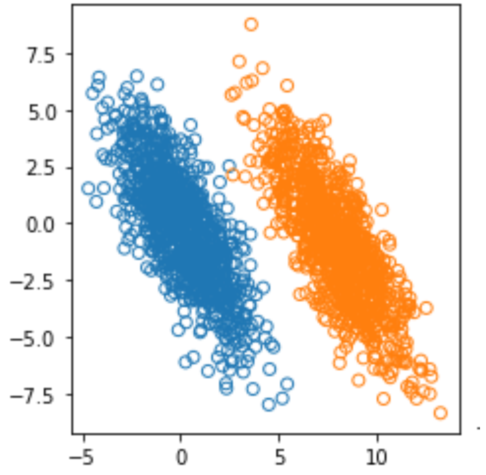


Multiclass

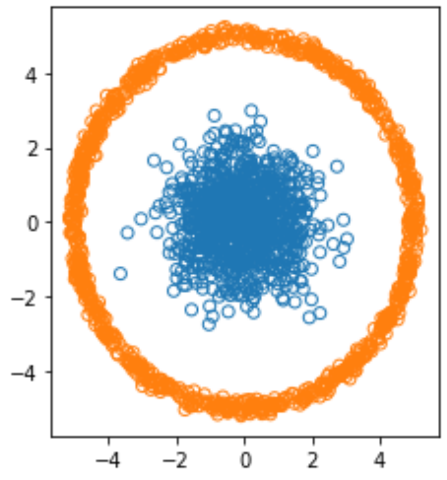


- **Linear**ly vs **Nonlinear**ly Separable

Linear



Nonlinear



Error Metrics - Definitions

There are several distinct error metrics for classification, based on the four types of outcomes for each class of label:

$$\mathrm{TP} = \mathrm{True\ Positive} \qquad \mathrm{FP} = \mathrm{False\ Positive}$$
$$\mathrm{TN} = \mathrm{True\ Negative} \qquad \mathrm{FN} = \mathrm{False\ Negative}$$

- **Recall** - fraction labeled correctly over all that should have been labeled

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Measures the fraction of a given label "recalled" by the model.

- **Precision** – fraction labeled correctly out of all that were labeled

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Measures how "precise" the model is when it assigns a label.

Note: recall and precision are generally inversely related, and should be discussed in relation to one another.

What would the precision for label "1" be for a model that labeled every sample "1"?

Error Metrics - Definitions

- Recall and Precision leave out the TN cases and can't be easily interpreted in isolation. Two average metrics try to give a holistic view:

- **Accuracy** – fraction of correction predictions over all conditions:

$$\mathrm{A} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}} = \frac{1}{N}\sum_{\mathrm{i}}^{N}\mathbf{1}\left(y_{\mathrm{i}} = y_{\mathrm{pred,i}}\right)$$

- **F_β** – "F-score" is the product of recall and precision or their mean (**β=1**):

$$\mathrm{F}_\beta = \left(1+\beta^2\right)\frac{\mathrm{P}\cdot\mathrm{R}}{\left(\beta^2\cdot\mathrm{P}\right)+\mathrm{R}} = \frac{\left(1+\beta^2\right)\mathrm{TP}}{\left(1+\beta^2\right)\mathrm{TP}+\beta^2\mathrm{FN}+\mathrm{FP}}$$

β=1 weights both **P** and **R** equally, **β>1** weights **P** more strongly, **β<1** weights **R** more strongly. Typically averaged over classes.

Let's say that we had the following result:
y = [0,0,0,0,0,0,0,0,0,1] ; y_{pred} = [0,0,0,0,0,0,0,0,0,0]
What would the accuracy be? What information would it leave out?

Error Metrics - Example

- Consider the following binary class case (0 = in spec; 1 = out of spec):

Labels: $y = [0,0,0,0,1,1,1,1]$ $y_{\mathrm{p}} = [0,0,0,0,1,1,1,0]$

Positives Negatives:

$$\mathrm{TP}_0 = 4 \quad \mathrm{FP}_0 = 1 \quad \mathrm{TN}_0 = 3 \quad \mathrm{FN}_0 = 0$$
$$\mathrm{TP}_1 = 3 \quad \mathrm{FP}_1 = 0 \quad \mathrm{TN}_1 = 4 \quad \mathrm{FN}_1 = 1$$

Precision:

$$\mathrm{P}_0 = \frac{4}{4+1} = 0.8 \qquad \mathrm{P}_1 = \frac{3}{3+0} = 1.0$$

Only 80% of the time that the model labeled something in specifications was it actually in specifications.

100% of the time it labeled something out of specifications, it was out of specifications.

Error Metrics - Example

- Consider the following binary class case (0 = in spec; 1 = out of spec):

Labels: $y = [0,0,0,0,1,1,1,1]$ $\quad y_\mathrm{p} = [0,0,0,0,1,1,1,0]$

Positives **Negatives** **:**

$$\mathrm{TP}_0 = 4 \quad \mathrm{FP}_0 = 1 \quad \mathrm{TN}_0 = 3 \quad \mathrm{FN}_0 = 0$$

$$\mathrm{TP}_1 = 3 \quad \mathrm{FP}_1 = 0 \quad \mathrm{TN}_1 = 4 \quad \mathrm{FN}_1 = 1$$

Precision:

$$\mathrm{P}_0 = \frac{4}{4+1} = 0.8 \qquad \mathrm{P}_1 = \frac{3}{3+0} = 1.0$$

Recall:

$$\mathrm{R}_0 = \frac{4}{4+0} = 1.0 \qquad \mathrm{R}_1 = \frac{3}{3+1} = 0.75$$

100% of the in-spec samples were correctly labeled

Only 75% of the out-of-spec samples were correctly labeled.

Note that there is generally an inverse relationship between precision and recall. You can be more careful to only label samples correctly (increase precision), but you will tend to miss some (reduce recall).

Error Metrics - Example

- Consider the following binary class case (0 = in spec; 1 = out of spec):

Labels: $y = [0,0,0,0,1,1,1,1]$ $\quad y_\mathrm{p} = [0,0,0,0,1,1,1,0]$

Positives Negatives:

$$\mathrm{TP}_0 = 4 \quad \mathrm{FP}_0 = 1 \quad \mathrm{TN}_0 = 3 \quad \mathrm{FN}_0 = 0$$

$$\mathrm{TP}_1 = 3 \quad \mathrm{FP}_1 = 0 \quad \mathrm{TN}_1 = 4 \quad \mathrm{FN}_1 = 1$$

Precision:

$$\mathrm{P}_0 = \frac{4}{4+1} = 0.8 \qquad \mathrm{P}_1 = \frac{3}{3+0} = 1.0$$

Recall:

$$\mathrm{R}_0 = \frac{4}{4+0} = 1.0 \qquad \mathrm{R}_1 = \frac{3}{3+1} = 0.75$$

Accuracy:

$$\mathrm{A} = \frac{1+1+1+1+1+1+1+0}{8} = \frac{7}{8}$$

F1:

$$\mathrm{F}_1(0) = \frac{2\cdot 4}{2\cdot 4+0+1} = \frac{2\cdot 0.8}{0.8+1.0} = \frac{8}{9} \qquad \mathrm{F}_1(1) = \frac{1.5}{1.0+0.75} = \frac{6}{7}$$

Note: In the first case, we are showing the equivalence of the two formulae.

Models – Logistic Regression

- Logistic Regression is a linear model for a binary classification (C=2)

- Key idea is to turn the discreet classification problem into a continuous problem of predicting the relative probability of the two classes (p_0, p_1):

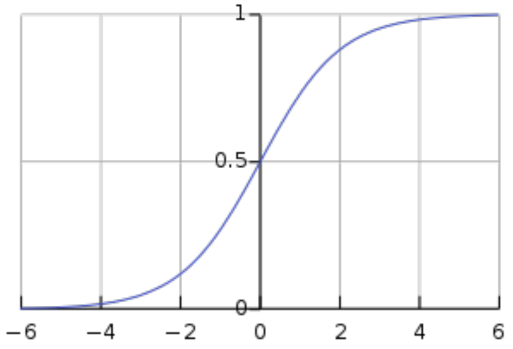
$$\log\text{odds} = \ln\left(\frac{p_1}{p_0}\right) = \ln\left(\frac{p_1}{1-p_1}\right) = \beta\mathbf{x}$$

where, **x** is a vector of one or more features, β holds our linear model parameters

- Taking the exponential and solving for p_1:

$$\frac{p_1}{1-p_1} = \mathrm{e}^{\beta\mathbf{x}} \rightarrow p_1 = \frac{\mathrm{e}^{\beta\mathbf{x}}}{1+\mathrm{e}^{\beta\mathbf{x}}} = \frac{1}{1+\mathrm{e}^{-\beta\mathbf{x}}}$$

Logistic Function



- Given β, prediction comes down to finding whether p_0 or p_1 is larger.

- You can visualize training as finding the optimal positions in **x**-space where the logistic function should switch.

- **Note:** other log-bases are also used (e.g., 10), with marginal impact.

Models – Logistic Regression

• Although the log odds is linear in β, the probability is not. Thus, unlike linear regression, there typically isn't any closed form solution for β.

• Thus, β is solved for by gradient-based optimization, typically with L1 or L2 regularization:

$$L[\mathbf{x},\mathbf{y},\beta] = \sum_{\mathrm{i}=1}^{N} \log\left(\mathrm{e}^{-y_i \mathbf{x}_\mathrm{i}\beta} + 1\right) + \alpha||\beta||_2$$ **(L2 case)**

• Internally, y is usually converted to [-1,1] labels rather than [0,1].

• The logic here is that when y_i and $\mathbf{x}_i\boldsymbol{\beta}$ have the same sign, the loss is minimized.

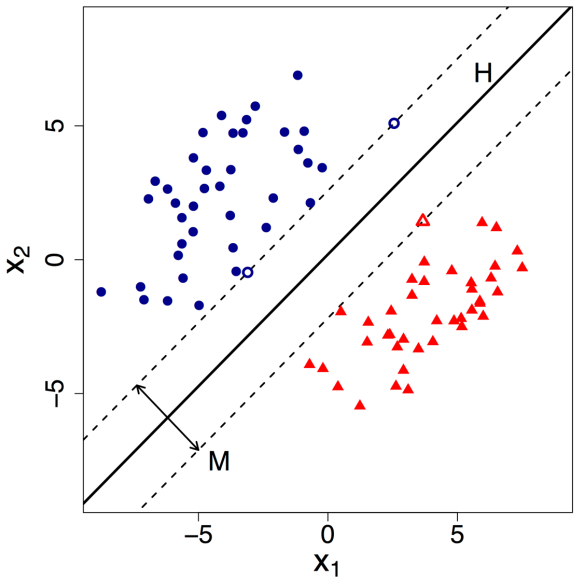
• Logistic regression can be extended to multi-class prediction, essentially by training n-1 pair-wise models.

• Logistic Regression is still very commonly used because of its simplicity and strong connection with probability, but clearly has limited expressibility.

Models – Support Vector Machine

• The object SVMs is to identify the optimal boundary that separates two or more classes of objects in **x**-space.

• SVM draws a *linear* boundary between the two classes. That is, if you plot the data classes with respect to **x**, the SVM solution is represented by the "best" straight line that divides them.



• **What if the classes overlap?** The typical objective function for training SVMs utilizes a "soft" boundary that linearly penalizes observing a label in the wrong region. In this case the "best" line, will still have errors in training.

• **What if the line separating the classes is not straight?** When the best boundary between your classes is anything but a straight line, that means your problem is nonlinear in your chosen **x** features. You may be able to find better descriptors that linearize the problem. More likely you are stuck with **x**. But you may be able to linearize the boundary using the "kernel trick", which transforms your **x** into a higher dimension where the problem is linear.

• **What is a support vector?** It turns out the optimal SVM solution only depends on the samples that are at the edge of the domain. The vectorial positions of these boundary data are the "support vectors" and hence the name of the model.

Models – k-nearest neighbors and radius nearest neighbors

- **KNN/RNN** are examples of non-parametric/”model-free” classifiers
- Instead of training a model, we actually just store the training data (**x**,**y**)
- Predictions are based on the following criteria for **KNN/RNN**:

  1. KNN: For a new point $\mathbf{x_i}$, we query the training data and find the **k nearest-neighbors** to that point. The majority vote (i.e., majority **y** label) is used as the prediction for $\mathbf{x_i}$.

  2. RNN: For a new point $\mathbf{x_i}$, we query the training data and find the **nearest-neighbors** with radius **r** of that point. The majority vote (i.e., majority **y** label) is used as the prediction for $\mathbf{x_i}$.

Note: ties can be handled in various ways (adding neighbors until breaking the tie, weighting neighbors based on distance, etc.)

- Main hyperparameters are either **k** or **r**.

These classifiers have some overlap with common unsupervised learning algorithms that we’ll cover later.