Machine Learning Crash Course: 3. Classification

Carlo Zanella

September 12, 2018

This module

So far we have looked at regression problems - problems where the response variable is quantitative. In this module we look at classification problems where the response variable we want to predict is qualitative.

We will look at how to evaluate classification problems, learn the workhorse method of logistic regression, and end with a state-of-the-art non-linear classifier called Support Vector Machines.

Classification

- Again we have some data X and we want to predict a variable Y.
- But now Y takes values in some discrete set $\{A, B, C, \ldots\}$.
- Example: we have data on emails and want to predict whether an email is spam or not.
- Economic example: suppose we look at patents at the EPO and we want to know whether a patent is about automation or not.
- Or a bank wants to predict the probability of default given a vector of creditor characteristics
- There may also be multiple classes
- Classification problems are almost more prevalent than regression problems in Machine Learning.

The Classification Task

- There are some issues involved with classification that are not present with regression
- We will look at them as we encounter them
- Note that the model

$$Y = f(X) + \epsilon,$$

  is now less relevant because Y is categorical.
- Better to think of there being some joint distribution $\Pr(Y, X)$ and we want to estimate that Y is in some class k given X: $\Pr(Y = k|X)$
- One objective would be to minimize the expected number of misclassifications

$$E\left[\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}(\tilde{y}_i \neq y_i)\right]$$

Bayes classifier

- Suppose we know $\Pr(Y|X)$.
- There are k classes for Y.
- Then given x_0 the classifier

$$\tilde{y}_0 = \arg\max_y \Pr(Y = y | X = x_0)$$

  is called the Bayes classifier and minimizes the expected number of misclassifications

$$E\left[\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(\tilde{y}_i \neq y_i)\right].$$

- Many (but not all) methods try to estimate $P(Y|X)$.

Logistic Regression

- A tool familiar to economists that is often used is logistic regression.
- Suppose we have a binary classification problem with $Y \in \{0, 1\}$.
- Recall that we assume that the probability of $Y = 1$ is given by

$$p_\beta(x) = \Pr(Y = 1|X) = \frac{e^{\beta_0 + \beta_1 X_1 + \dots}}{1 + e^{\beta_0 + \beta_1 X_1 + \dots}}$$

- The cost function (i.e. negative of the log likelihood function) is then

$$J(\beta) = -\left[\frac{1}{n}\sum_{i=1}^{n} y_i \log p_\beta(x_i) + (1 - y_i)\log(1 - p_\beta(x_i))\right].$$

- Can be easily extended to more than two classes.

Logistic Regression: An example

- Suppose we have data on credit.
- We want to predict the probability of a default: $\Pr(\textit{default} = \text{"Yes"})$ where $\textit{default} \in \{\text{"Yes"}, \text{"No"}\}$.
- We estimate a logistic model and for each observation we get a prediction for each observation.
- We can summarize the classification in a confusion matrix:

	True label	
Predicted	No	Yes
No	9627	228
Yes	40	105

- Two kinds of errors: False positives and false negatives.

	True label	
Predicted	No	Yes
No	True Negative	False Negative
Yes	False Positive	True Positive

Decision Boundary

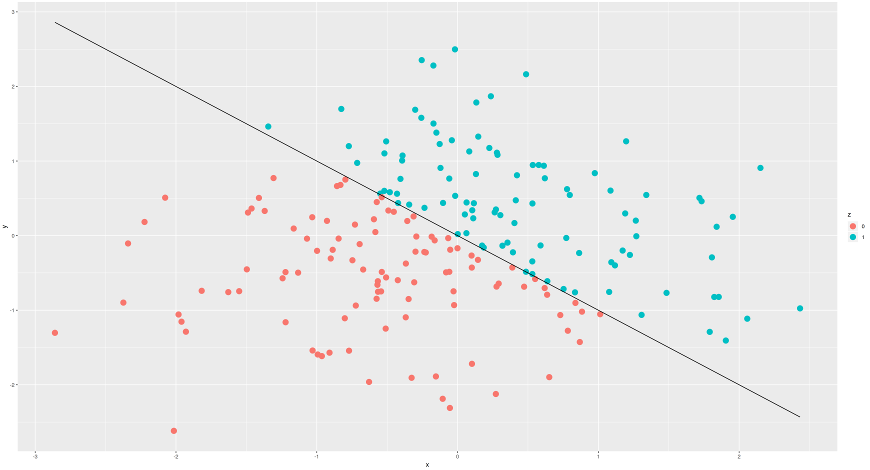

Figure: Decision Boundary

Decision Boundary in the Logistic Model

- Recall that $\Pr(Y = 1|X) = g(\beta_0 + \beta_1 X_1 + \ldots)$, where

$$g(z) = \frac{\exp(z)}{1 + \exp(z)}$$

is the logistic function.

- Suppose we classify $\tilde{y}_i = 1$ if $g(\beta_0 + \beta_1 x_1 + \ldots) > 0.5$.
- This holds whenever $\beta_0 + \beta_1 x_1 + \ldots > 0$.
- This is a linear decision boundary.
- Upshot: logistic model estimates decision boundaries linear in the explanatory variables
- Do you think the logistic model can estimate the boundary on the previous page?

Decision Boundary

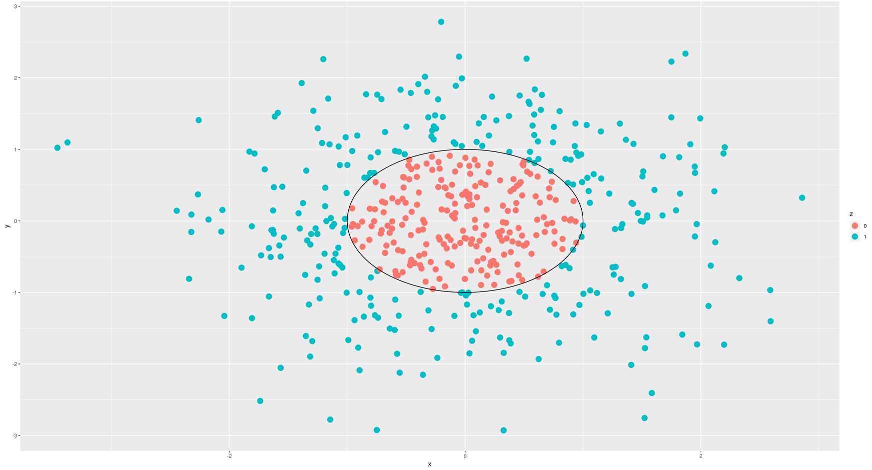

Figure: Nonlinear Decision Boundary: Can the logistic model capture this?

(Notebook “04 Classification”)

Receiver Operator Characteristics

- The 0.5 threshold corresponds to the Bayes classifier which minimizes the mis-classification rate.
- Depending on whether type 1 or type 2 errors are more costly this may not be what we want.
- Varying the threshold can give different trade-offs between type 1 and type 2 error.
- This is usually visualized in the so called Receiver Operator Characteristics curve, or ROC curve.
- It plots the True Positive Rate vs the False Positive Rate.
- (see the notebook "04 Classification" for an example)

AUC

- The Area Under the ROC Curve (AUC or AUROC) is simply the integral of the ROC curve.
- The best model would get a AUC of 1.0.
- The model which predicts exactly the opposite of the true label gets an AUC of 0.0.
- A model assigning random predictions have an 0.5.
- In practice AUCs are therefore between 0.5 and 1.0.
- Interpretation: draw a random positive observation x_p and a random negative observation x_n.
  Let $p(x_p)$ and $p(x_n)$ be the probabilities assigned by our model.
- The AUC is the probability that $p(x_p) > p(x_n)$.
- That is the probability that our model correctly ranks negative and positive examples.

AUC

- This is from a medical course "Interpreting Diagnostic Tests" at the University of Nebraska Medical Center:

.90-1 = excellent
.80-.90 = good
.70-.80 = fair
.60-.70 = poor
.50-.60 = fail
Source: `http://gim.unmc.edu/dxtests/roc3.htm`

- More seriously, what is a good fit depends on the variable and the context of the classification.

Logistic regression: Regularization

- Overfitting is also a major concern for logistic regression.
- Especially with many variables the decision boundaries are prone to overfitting.
- Regularization is a general method and can easily be applied to Logistic Regression.
- Actually both Linear Regression and Logistic Regression are instances of "Generalized Linear Models" which can all be regularized in the same fashion.
- The Elastic Net penalized cost function is

$$J(\beta) = -\left[\frac{1}{n}\sum_{i=1}^{n} y_i \log p_\beta(x_i) + (1-y_i)\log(1-p_\beta(x_i))\right] + \lambda\left(\alpha\sum_{j=1}^{k}|\beta_k| + (1-\alpha)\sum_{j=1}^{k}\beta_k^2\right).$$

Logistic regression: Regularization (2)

- In R the `glmnet` package can be used to run regularized logistic regressions.
- Rather than writing as we did for Linear Regression

  ```
  fit <- glmnet(x,y)
  ```

  simply write

  ```
  fit <- glmnet(x,y,family="binomial"),
  ```

  which instructs R to do a Logistic Regression.

(Let's do Exercise 3)

Support Vector Machines (SVM)

- A very powerful class of classifiers are the Support Vector Machines (SVM).
- In many areas they do as good a job as neural networks.
- But much more robust and faster to train
- Based on some rather groovy mathematics

SVM: Large margin classifier

- Take a binary classification problem with $Y \in \{-1, 1\}$ and features $X = (X_1, X_2, \ldots)$.
- The classes are linearly separable if there exists a hyperplace β such that

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots > 0 \text{ if } y_i = 1 \text{ and}$$
$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots < 0 \text{ if } y_i = -1.$$

- Such a hyperplace is called a separating hyperplace.

SVM: Large margin classifier

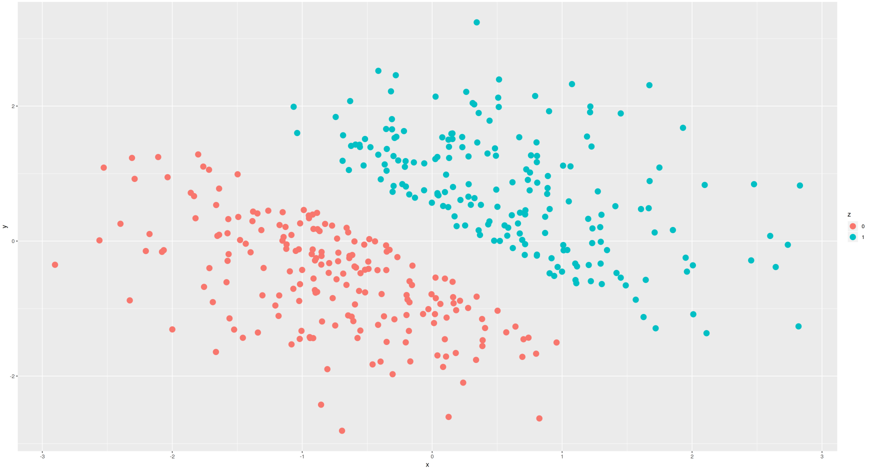

Figure: Decision Boundary with a Margin

SVM: Large margin classifier

Figure: Many separating hyperplanes

SVM: Large margin classifier

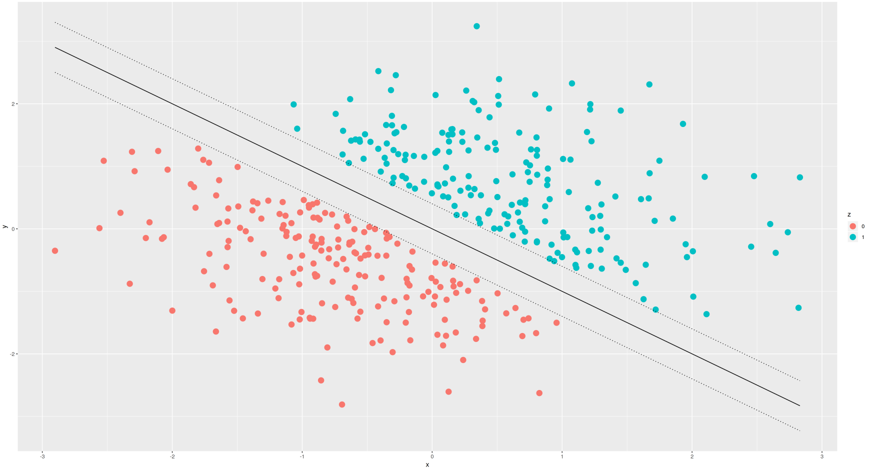

Figure: Maximal margin hyperplane

SVM: Large margin classifier

- The hyperplane on the last page solves the problem

$$\begin{aligned} \min_{M,\beta_0,\ldots,\beta_m} \quad & M \\ \text{s.t} \quad & \\ \beta_0 + \beta_1 x_{i1} + \ldots \beta_m x_{im} > M & \text{ if } y_i = 1 \quad (1) \\ \beta_0 + \beta_1 x_{i1} + \ldots \beta_m x_{im} < -M & \text{ if } y_i = -1 \text{ and} \quad (2) \\ \sum_{i=0}^{m} \beta_i^2 = 1. & \end{aligned}$$

- The classifier that comes out of this is called the maximal margin classifier.
- Note that we can write constraints (1) and (2) succinctly as

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots \beta_m x_{im}) > M.$$

SVM: Support vector classifier

- Usually data are not perfectly linearly-separable.
- We can allow observations to be on the wrong side of the margin
- This is called a soft margin (in constrast to the previous hard margin)
- The slack problem is

$$
\begin{aligned}
&\min_{M,\beta_0,\ldots,\beta_m,\epsilon_1,\ldots,\epsilon_n} \quad M \\
&\text{s.t} \quad y_i(\beta_0 + \beta_1 x_{i1} + \ldots \beta_m x_{im}) > M(1 - \epsilon_i) \text{ if } y_i = 1, \\
&\qquad \sum_{i=0}^{m} \beta_i^2 = 1, \epsilon_i \geq 0 \text{ and } \sum_{i=1}^{n} \epsilon_i \leq C.
\end{aligned}
$$

- ϵ_i: how much an observation violates the constraint
- C: a hyperparameter which acts like a "budget" on how much violations we allow

Kernels

- Let $\mathcal{X}$ be our feature space (usually $\mathbb{R}^m$).
  A kernel is a continuous symmetric function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.
- A kernel is also called a similarity function: if $K(x, y)$ is small then x and y are similar in some sense.
- If the kernel K satisfies a technical condition called Mercer's condition it can be written of the form

$$K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{V}},$$

  where $\mathcal{V}$ is some inner product space and $\phi : \mathcal{X} \to \mathcal{V}$ is a transformation of the features.
- Popular kernels:
  - The Gaussian or radial kernel: $K(x, y) = \exp(-||x - y||^2/\sigma^2)$, where σ^2 controls the dispersion of the function
  - A polynomial kernel: $K(x, y) = (\lambda + \langle x, y \rangle)^d$, where d is the degree of the polynomial

Gaussian Kernel

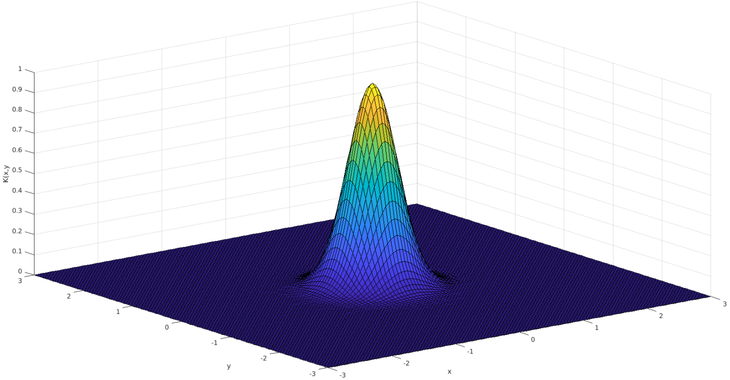

Figure: Gaussian with $\sigma^2 = 0.2$

Gaussian Kernel

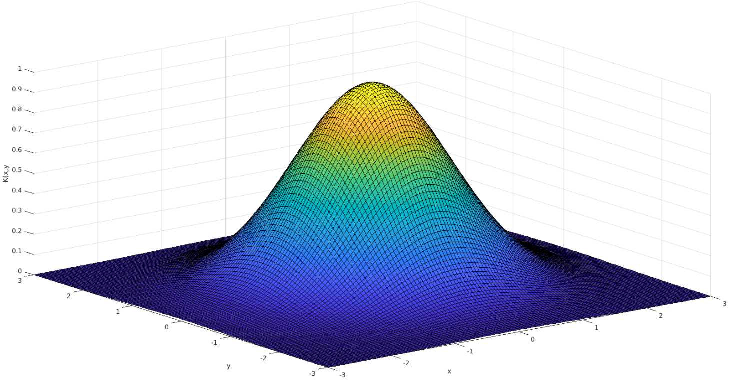

Figure: Gaussian with $\sigma^2 = 2$

Support Vector Machines + Kernels

- Turns out SVM can be easily combined with kernels.
- SVM classifier uses only the inner products $\langle x, x' \rangle$ of all observations x and x'.
- Just switch $\langle x, x' \rangle$ with $K(x, x')$.
- Using kernels allows for highly non-linear decision boundaries.
- Why?
- Using kernels is akin to transforming your observation x with $\phi(x)$.
- SVM+kernels work in a transformed feature space
- The linear decision boundary is happening on the transformed feature space.

Support Vector Machines + Kernels (2)

- Why the fuss with kernels? Why not just transform your features x using ϕ and work with $\phi(x)$?
- Reason 1: Explicitly working with large transformed feature spaces is memory intensive. But SVM doesn't need features, only inner products of features.
- Reason 2: With using kernels the transformed feature space $\mathcal{V}$ is implicit. We don't need to know ϕ.
- Now reason 2 is the real power of SVMs
- In the case of the Gaussian kernel, the feature space $\mathcal{V}$ is even infinite-dimensional
- So we implicily create infinitely many features from our finite features.
- This allows for very highly non-linear boundaries on our original features.
- SVMs are one of the most powerful and popular ML algorithms out there

(Notebook “05 Support Vector Machines”)