Seminar in Data Science

Lecture 7: Classification

Laurent El Ghaoui

Seminar in Data Science and Information Technology, Summer 2020
TBSI – UC Berkeley

7/27/2020

Outline

Outline

Basics of binary classification

Data

We are given a *training* data set with m measurements:

- *Feature vectors:* data points $x_i \in \mathbf{R}^n$, $i = 1, \ldots, n$.
- *Labels:* $y_i \in \{-1, 1\}$, $i = 1, \ldots, m$.

Examples:

Feature vectors	Labels
Companies' corporate info	default/no default
Stock price data	price up/down
News data	price up/down
News data	sentiment (positive/negative)
Emails	presence of a keyword
Genetic measures	presence of disease

Using the training data set $\{x_i, y_i\}_{i=1}^m$, our goal is to find a classification rule $\hat{y} = f(x)$ allowing to predict the label $\hat{y}$ of a new data point x.

Popular classification algorithms

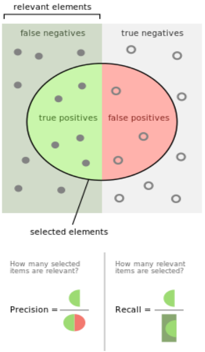
- Support vector machines;
- Logistic regression;
- Decision trees and random forests;
- Neural networks;
- Etc.

In this lecture, we focus on SVM and logistic regression.

Metrics

In regression, we can use average prediction error (on the test set) to evaluate a particular prediction algorithm.

In classification, we need to capture false positives and false negatives, and we can use similar metrics (evaluated on the *test* set):



- *Precision* p: the number of correctly predicted positive results divided by the number of all predicted positive results,

$$p = \frac{TP}{TP + FP}.$$

- *Recall* r: the number of correctly predicted positive results divided by the number of actual positives in the data,

$$r = \frac{TP}{TP + FN}.$$

Confusion matrix and F1 score

Denote by

- P, N the positive and negative classes;
- TP, TN: true positive and negative results;
- FP, FN: false positive and negative results.

The confusion matrix shows all the types of error rates:

	Condition positive (15)	Condition negative (15)	
Predicted Positive (21)	True Positive TP = 15	False Positive FP = 6	Positive Predictive Value PPV = TP/(TP+FNP) = 0.71
Predicted Negative (9)	False Negative FN = 0	True Negative TN = 9	Negative Predictive Value NPV = TN/(FN+TN) = 1
	Sensitivity TPR = TP/(TP+FN) = 1	Specificity TNR = TN/(FP+TN) = 0.6	

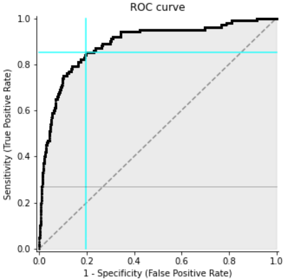
Confusion matrix

F1 Score and ROC curve

To combine precision and recall one often uses the F1 score , which is the harmonic mean of the two:

$$F1 = \frac{2rp}{r + p}.$$

When the classifier depends on some hyper-parameter λ, it is useful to develop the ROC curve plotting the true positive rate against false positive rate:



ROC curve

A high area under the curve denotes a better classifier.

Linear classification

Linear classification rule: assumes f is a combination of the sign function and a linear (in fact, affine) function:

$$\hat{y} = \mathbf{sign}(w^T x + b),$$

where $w \in \mathbf{R}^n$, $b \in \mathbf{R}$ are given.

The goal of a linear classification algorithm is to find w, b, using the training data.

Support Vector Machines

Separable data

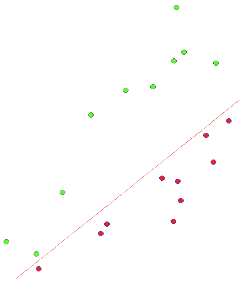
The data is linearly separable if there exist a linear classification rule that makes no error on the training set.

This is a set of linear inequalities constraints on (w, b):

$$y_i(w^T x_i + b) \geq 0, \quad i = 1, \ldots, m.$$

Strict separability corresponds the the same conditions, but with strict inequalities.

Geometry

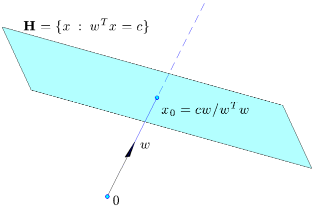


Geometrically: the hyperplane

$$\{x \;:\; w^T x + b = 0\}$$

perfectly separates the positive and negative data points.

Linear algebra flashback: hyperplanes



Geometrically, a hyperplane

$$\mathbf{H} = \left\{ w \,:\, w^T x = c \right\}$$

is a translation of the set of vectors orthogonal to w. The direction of the translation is determined by w, and the amount by $c/\|w\|_2$. Indeed, the projection of 0 onto $\mathbf{H}$ is

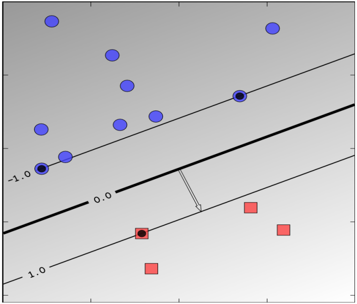
$$x_0 = cw/(w^T w).$$

Geometry (cont'd)

Assuming strict separability, we can always rescale (w, b) and work with

$$y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, m.$$

Amounts to make sure that negative (resp. positive) class contained in half-space $w^T x + b \leq -1$ (resp. $w^T x + b \geq 1$).



The distance between the two "± 1" boundaries turns out the be equal to $2/\|w\|_2$.

Thus the "margin" $\|w\|_2$ is a measure of how well the hyperplane separates the data apart.

Non-separable data

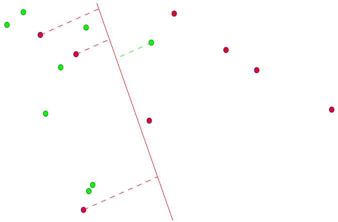
Separability constraints are homogeneous, so WLOG we can work with

$$y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, m.$$

If the above is infeasible, we try to minimize the "slacks"

$$\min_{w,b,s} \sum_{i=1}^{m} s_i \; : \; s \geq 0, \quad y_i(w^T x_i + b) \geq 1 - s_i, \quad i = 1, \ldots, m.$$

The above can be solved as a "linear programming" (LP) problem (in variables w, b, s).



Geometry of LP formulation.

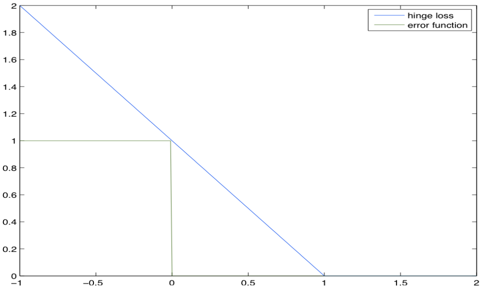
Geometry of LP formulation: we minimize the sum of the distances (seen as dotted segments) from mis-classified points to the boundary.

Hinge loss function

The previous LP can be interpreted as minimizing the hinge loss function

$$L(w, b) := \sum_{i=1}^{m} \max(1 - y_i(w^T x_i + b), 0).$$

This serves as an approximation to the number of errors made on the training set:



Regularization

The solution might not be unique, so we add a regularization term $\|w\|_2^2$:

$$\min_{w,b} \frac{1}{2}\|w\|_2^2 + C \cdot \sum_{i=1}^{m} \max(1 - y_i(w^T x_i + b), 0)$$

where $C > 0$ allows to trade-off the accuracy on the training set and the prediction error (more on why later). This makes the solution unique.

The above model is called the *Support Vector Machine* . It is a quadratic program (QP). It can be reliably solved using special fast algorithms that exploit its structure.

If C is large, and data is separable, reduces to the maximal-margin problem

$$\min_{w,b} \frac{1}{2}\|w\|_2^2 \ : \ y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, m.$$

Logistic model for binary classification

We model the probability of a label Y to be equal $y \in \{-1, 1\}$, given a data point $x \in \mathbf{R}^n$, as:

$$P(Y = y \mid x) = \frac{1}{1 + \exp(-y(w^T x + b))}.$$

This amounts to modeling the *log-odds ratio* as a linear function of X:

$$\log \frac{P(Y = 1 \mid x)}{P(Y = -1 \mid x)} = w^T x + b.$$

- The decision boundary $P(Y = 1 \mid x) = P(Y = -1 \mid x)$ is the hyperplane with equation $w^T x + b = 0$.
- The region $P(Y = 1 \mid x) \geq P(Y = -1 \mid x)$ (*i.e.*, $w^T x + b \geq 0$) corresponds to points with predicted label $\hat{y} = +1$.

Maximum-likelihood

The likelihood function is

$$l(w,b) = \prod_{i=1}^{m} \frac{1}{1 + e^{-y_i(w^T x_i + b)}}.$$

Now minimize the **negative** log-likelihood:

$$\min_{w,b} \; -L(w,b) := -\sum_{i=1}^{m} \log(1 + e^{-y_i(w^T x_i + b)})$$

- In practice, we may consider adding a regularization term

$$\max_{w,b} \; -L(w,b) + \lambda \|w\|_2^2.$$

- Many packages exist for logistic regression, *e.g.* [4].

Logistic regression

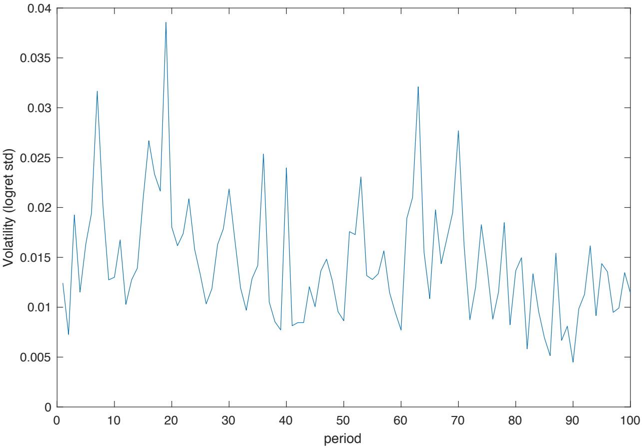
Example: predicting volatility direction

- We use a logistic model to predict if a stock's volatility is going up or down in the next period
- We select a period duration of 15 days
- From each period, we extract from the data some *features*, such as:
  (1) mean return over the current period,
  (2) mean return over the previous period,
  (3) volatility over the current period,
  (4) volatility over the previous period,
  (5) average price level over the current period,
  (6) average price level over the previous period
- We compute the volatility over the next period and set label $y = +1$ if it increased and label $y = 0$ otherwise
- We learn a Logistic model using data, and then test it on validation data
- We use AAPL stock data, from Dec. 5 2011 to Dec. 1 2017.

Logistic regression

Example: predicting volatility direction

- We use $N_{id} = 68$ periods for training the model, and $N_v = 30$ periods for verification.

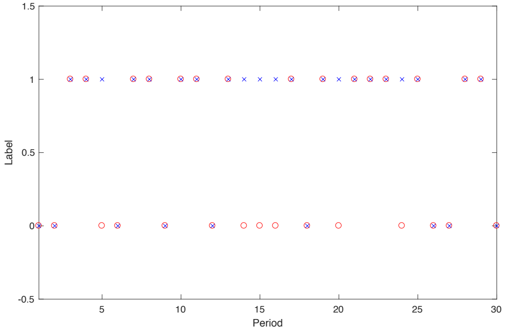


Logistic regression

Example: predicting volatility direction

- Logistic model:

$$w = [90.92,\ -107.58,\ 184.70,\ 68.18,\ 0.11,\ -0.1], \quad b = -5.11.$$



RED= actual, BLUE= predicted.

- Error rate of 20% on the verification set.

See `logistic_volatility_ex.m`

Prediction of volatility direction

In the current example:

	Condition positive (15)	Condition negative (15)	
Predicted Positive (21)	True Positive TP = 15	False Positive FP = 6	Positive Predictive Value PPV = TP/(TP+FNP) = 0.71
Predicted Negative (9)	False Negative FN = 0	True Negative TN = 9	Negative Predictive Value NPV = TN/(FN+TN) = 1
	Sensitivity TPR = TP/(TP+FN) = 1	Specificity TNR = TN/(FP+TN) = 0.6	

Outline

Multi-class problems

In some problems, the "'labels" y_i, $i = 1, \ldots, m$ are not binary, but correspond to more than two categories (*e.g.*, star ratings, analysts recommendations, etc).

- A common practice is to transform the problem into a sequence of binary classification problems, doing multiple "one-vs-all" approaches.
- Some of the approaches seen before can handle directly multi-class problems nicely; for detail the example of logistic regression next.
- If the categories are ordered (such as "buy", "hold", "sell"), we can use methods seen in the context of generalied low-rank models.

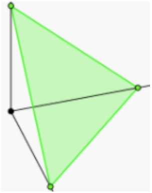
Multi-class logistic regression

Assume there are $p > 1$ classes. The outputs can be encoded as "one-hot vectors": specifically, the output is $y_i \in \mathbf{R}^p$, with $y_i(j) = 1$ if data point x_i belongs to class j, 0 otherwise.

- We now seek to predict a vector of probabilities in $\mathbf{R}^p$: our predicted output $\hat{y}(x)$ will be non-negative, and sum to one.
- For a test point x we choose the largest entry in $\hat{y}(x)$ to predict the most likely class, based on the logistic model.

Soft-max function

The soft-max function maps an arbitrary vector $z \in \mathbf{R}^p$ into one that belongs to the probability simplex (*i.e.* non-negative components that sum to one).



The probability simplex in $\mathbf{R}^3$.

Precisely, $S : \mathbf{R}^p \to \mathbf{R}^p_+$ takes values as follows: for $z \in \mathbf{R}^p$, $S(z)$ is the vector with elements

$$[S(z)]_i = \frac{e^{z_i}}{\sum_{j=1}^p e^{z_j}}, \quad i = 1, \ldots, p.$$

Prediction model and training problem

Putting this together, we set

$$\hat{y}_\theta(x) = S(Wx + b\mathbf{1})$$

with $\theta := [W, b] \in \mathbf{R}^{p\times(n+1)}$ contain the parameters of the logistic model, and S is the soft-max function.

The training problem consists in minimizing a loss function that measures the discrepancy between the predicted probability distributions and their observed (one-hot) counterpart, for each data point:

$$\min_\theta \ \mathcal{L}(\theta) := \sum_{i=1}^m l(\hat{y}_\theta(x_i), y_i)$$

It is common to choose the cross-entropy measure to assess how close two distributions are:

$$l(\hat{y}, y) := \sum_{j=1}^p y_j \log \hat{y}_j.$$

The logistic regression problem thus formulated is convex.

Outline

Generalized classification

We consider the problem

$$\min_{w} \mathcal{L}(X^T w + b\mathbf{1}, y) + \lambda p(w),$$

where

- $\mathcal{L}$ is a convex loss function that encodes the error between the observed value and the predicted value;
- (w, b) are the model parameters;
- p is a penalty on the regression parameters;
- $\lambda > 0$ is a penalty parameter.

When $\mathcal{L}(z, y) = \mathbf{1}^T(1 - yz)_+$, $p(w) = \|w\|_2^2$, we recover regularized SVM.

Playing with loss functions and penalties

Changing loss functions allos to cover these types of regression methods:

- SVMs
- Logistic regression
- Naïve Bayes classification

Typical penalties allow to

- l_1-norm: to enforce sparsity;
- l_2-norm (often, squared): to control statistical noise and improve prediction error;
- sum-block norms enable to enforce whole blocks of w to be zero.

Motivations

In some applications, we have access to a measure of uncertainty associated with each data point, and model this as $X \in \mathcal{X}$, with $\mathcal{X}$ a matrix set that describe the uncertainty around a given data set $\hat{X} \in \mathcal{X}$.

Robust model:

$$\min_{w,b} \max_{X \in \mathcal{X}} \mathcal{L}(X^T w + b\mathbf{1}, y).$$

Example: interval model

Assume that each entry in the data matrix is only known to belong to a given interval:

$$X_{ij} \in [\hat{X}_{ij} - R_{ij}, \hat{X}_{ij} + R_{ij}],$$

with $\hat{X}_{ij}$, $R_{ij} > 0$ given, $1 \leq i \leq n$, $1 \leq j \leq m$.

This corresponds to the robust model

$$\min_{w,b} \max_{X \in \mathcal{X}} \mathcal{L}(X^T w + b\mathbf{1}, y),$$

with $\mathcal{X} = [\hat{X} - R, \hat{X} + R]$ an interval matrix (here $R = (R_{ij})$).

Explicit form

Key fact: for given $\hat{x} \in \mathbf{R}^n$, $\rho \in \mathbf{R}^n_+$:

$$\max_{x \,:\, |x-\hat{x}| \leq r} w^T x = w^T \hat{x} + r^T |w|,$$

where $|z|$ denotes the vector of magnitudes of elements in vector z.

Example
Robust SVM

For the SVM (hinge loss) case, we obtain

$$\min_{w,b} \sum_{i=1}^{m} \max(1 - y_i(w^T \hat{x}_i + b) + R_i^T |w|, 0),$$

where R_i stands for the i-th colum of R. This provides some form of l_1-regularization.

The above can be further approximated with the upper bound

$$\min_{w,b} \sum_{i=1}^{m} \max(1 - y_i(w^T \hat{x}_i + b), 0) + \sigma^T |w|,$$

with $\sigma := \sum_i R_i$.

Ellipsoidal uncertainty

Another model involves a spherical (or more generally ellipsoidal) uncertainty, where each data point x_i is only known to belong to a sphere of center $\hat{x}_i$ and radius r_i. More generally:

$$x_i = \hat{x}_i + r_i D u_i$$

with $D = \mathbf{diag}(\sigma_1, \ldots, \sigma_n)$ is a positive-definite diagonal scaling matrix, $\|u_i\|_2 \le 1$, and $r_i > 0$. (Intuition: up to a point-dependent scaling factor r_i, variances are the same across the data points.)

For the SVM (hinge loss) case, we obtain

$$\min_{w,b} \sum_{i=1}^{m} \max(1 - y_i(w^T \hat{x}_i + b) + r_i \|Dw\|_2, 0),$$

This provides some form of l_2-regularization. Model can be further approximated by some form of standard l_2-norm regularized SVM:

$$\min_{w,b} \sum_{i=1}^{m} \max(1 - y_i(w^T \hat{x}_i + b), 0) + \lambda \|Dw\|_2, \quad \lambda := \sum_i r_i.$$

This provides guidance on which scaled penalty to use, and also explains why normalizing data by variance may be beneficial. (Why?)

Robustness interpretation of SVM

Return to separable data in the SVM setup. The set of constraints

$$y_i(w^T x_i + b) \geq 0, \quad i = 1, \ldots, m,$$

has many possible solutions (w, b).

We will select a solution based on the idea of robustness (to changes in data points).

Maximally robust separating hyperplane

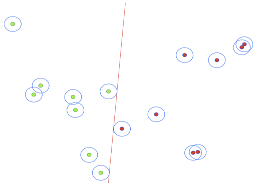
Spherical uncertainty model: assume that the data points are actually unknown, but bounded: for $i = 1, \dots, m$,

$$x_i \in \mathcal{S}_i := \{\hat{x}_i + u_i \;:\; \|u_i\|_2 \le \rho\},$$

where $\hat{x}_i$'s are known, $\rho > 0$ is a given measure of uncertainty, and u_i is unknown.

Robust counterpart: we now ask that the separating hyperplane separates the spheres (and not just the points):

$$\forall x_i \in \mathcal{S}_i \;:\; y_i(w^T x_i + b) \ge 0, \quad i = 1, \dots, m.$$



For separable data we can try to separate spheres around the given points. We'll grow the spheres' radius until sphere separation becomes impossible.

Robust classification

We obtain the equivalent condition

$$y_i(w^T \hat{x}_i + b) \geq \rho \|w\|_2, \quad i = 1, \dots, m.$$

Now we seek (w, b) which maximize ρ subject to the above.

By homogeneity we can always set $\rho \|w\|_2 = 1$, so that problem reduces to

$$\min_{w} \; \|w\|_2 \; : \; y_i(w^T \hat{x}_i + b) \geq 1, \quad i = 1, \dots, m.$$

This is exactly the same problem as the SVM in separable case, a.k.a. the "maximum-margin classifier".

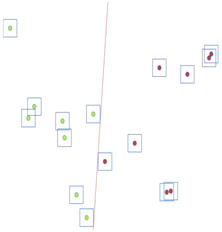
Separating boxes instead of spheres

We can use a box uncertainty model: for $i = 1, \ldots, m$,

$$x_i \in \mathcal{B}_i := \{\hat{x}_i + u_i \ : \ \|u_i\|_\infty \le \rho\}\,.$$

This leads to

$$\min_w \ \|w\|_1 \ : \ y_i(w^T\hat{x}_i + b) \ge 1, \quad i = 1, \ldots, m.$$



Classifiers found that way tend to be sparse. In 2D, the boundary line tends to be vertical or horizontal.

Outline

References

Sahely Bhadra, J Nath, Aharon Ben-Tal, and Chiranjib Bhattacharyya.
Interval data classification under partial information: A chance-constraint approach.
Advances in Knowledge Discovery and Data Mining, pages 208–219, 2009.

Chih-Chung Chang and Chih-Jen Lin.
LIBSVM: A library for SVM classification.

T. Hastie, R. Tibshirani, and J.H. Friedman.
The elements of statistical learning.
Springer, 2009.

David Madigan and David Lewis.
The BBR machine learning package, 2011.

Carolin Strobl, James Malley, and Gerhard Tutz.
An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests.
Psychological methods, 14(4):323, 2009.