Mathematical Background and the Perceptron

Héctor Corrada Bravo

University of Maryland, College Park, USA

CMSC643: 2017-09-12

ML Preliminaries

A common situation in data analysis is that one has an outcome Y and one or more independent covariates or predictors $X_1, \ldots, X_p$.

One usually observes these variables for multiple "instances".

ML Preliminaries

A common situation in data analysis is that one has an outcome Y and one or more independent covariates or predictors $X_1, \ldots, X_p$.

One usually observes these variables for multiple "instances".

(Note) We use upper case to denote a random variable. To denote actual numbers we use lower case. One way to think about it: Y has not happened yet, and when it does, we see $Y = y$.

ML Preliminaries

One may be interested in various things:

- What effects do the covariates have on the outcome?
- How well can we describe these effects?
- Can we predict the outcome using the covariates?, etc...

Motivating Example: Credit Analysis

default	student	balance	income
No	No	729.5265	44361.625
No	Yes	817.1804	12106.135
No	No	1073.5492	31767.139
No	No	529.2506	35704.494
No	No	785.6559	38463.496
No	Yes	919.5885	7491.559

Task: predict account default

What is the outcome Y?

What are the predictors X_j?

Terminology and notation

We will be mixing the terminology of statistics and computer science.

For example, we will sometimes call Y and X the outcome/predictors, sometimes observed/covariates, and even input/output. We may call each instance an observation or example.

Terminology and notation

We will be mixing the terminology of statistics and computer science.

For example, we will sometimes call Y and X the outcome/predictors, sometimes observed/covariates, and even input/output. We may call each instance an observation or example.

We will denote predictors with X and outcomes with Y (quantitative) and G (qualitative). Notice G are not numbers, so we cannot add or multiply them.

Terminology and notation

Height and weight are quantitative measurements. These are sometimes called continuous measurements.

Terminology and notation

Height and weight are quantitative measurements. These are sometimes called continuous measurements.

Gender is a qualitative measurement. They are also called categorical or discrete. This is a particularly simple example because there are only two values. With two values we sometimes call it binary.

Terminology and notation

We will use G to denote the set of possible values. For gender it would be $G = \{Male, Female\}$.

A special case of qualitative variables are ordered qualitative where one can impose an order. With men/women this can't be done, but with, say, $G = \{low, medium, high\}$ it can.

From data to feature vectors

The vast majority of ML algorithms we see in class treat instances as "feature vectors".

We can represent each instance as a vector in Euclidean space $\langle x_1, \ldots, x_p, y \rangle$

.

From data to feature vectors

The vast majority of ML algorithms we see in class treat instances as "feature vectors".

We can represent each instance as a vector in Euclidean space $\langle x_1, \ldots, x_p, y \rangle$

.

This means:

- every measurement is represented as a continuous value
- in particular, categorical variables become numeric (e.g., one-hot encoding)

From data to feature vectors

Here is the same credit data represented as a matrix of feature vectors

default	student	balance	income
1	0	1991.6491	42133.37
1	1	2461.5070	11878.56
-1	1	1357.4235	25733.31
-1	0	135.9982	59369.65
1	0	1753.0844	48965.35
-1	0	1143.4314	35773.40

Technical notation

- Observed values will be denoted in lower case. So x_i means the ith observation of the random variable X.

- Matrices are represented with bold face upper case. For example $\mathbf{X}$ will represent all observed predictors.

Technical notation

- N will usually mean the number of observations, or length of Y. i will be used to denote which observation and j to denote which covariate or predictor.
- Vectors will not be bold, for example x_i may mean all predictors for subject i, unless it is the vector of a particular predictor $\mathbf{x}_j$.
- All vectors are assumed to be column vectors, so the i-th row of $\mathbf{x}$ will be x_i', i.e., the transpose of x_i.

Geometry and Distances

Now that we think of instances as vectors we can do some interesting operations.

Let's try a first one: define a distance between two instances using Euclidean distance

$$d(x_1, x_2) = \sqrt{\sum_{j=1}^{p} (x_{1j} - x_{2j})^2}$$

K-nearest neighbor classification

Now that we have a distance between instances we can create a classifier. Suppose we want to predict the class for an instance x.

K-nearest neighbors uses the closest points in predictor space predict Y.

$$\hat{Y} = \frac{1}{k} \sum_{x_k \in N_k(x)} y_k.$$

$N_k(x)$ represents the k-nearest points to x. How would you use $\hat{Y}$ to make a prediction?

K-nearest neighbors algorithm

function KNN-CLASSIFY(x, X, y, K)
    $S \leftarrow []$ ▷ Compute distance to all points in X
    for all $i = 1, \ldots, N$ **do**
        $S \oplus \langle d(x, x_i), i \rangle$
    end for
    $S \leftarrow sort(S)$ ▷ Find K nearest points
    $\hat{y} \leftarrow 0$
    for all $k = 1, \ldots, K$ **do**
        $\langle d(x, x_i) \rangle \leftarrow S_k$
        $\hat{y} \leftarrow \hat{y} + y_i$ ▷ Update prediction
    end for
    return $\text{sign}(\hat{y})$ ▷ Return $+1$ if $\hat{y} > 0$, -1 otherwise
end function

K-nearest neighbors algorithm

An important notion in ML and prediction is inductive bias.

What assumptions we make about our data that allow us to make predictions.

K-nearest neighbors algorithm

An important notion in ML and prediction is inductive bias.

What assumptions we make about our data that allow us to make predictions.

In KNN, our inductive bias is that points that are **nearby** will be of the same class.

K-nearest neighbors algorithm

Parameter K is a hyper-parameter, it's value may affect prediction accuracy significantly.

K-nearest neighbors algorithm

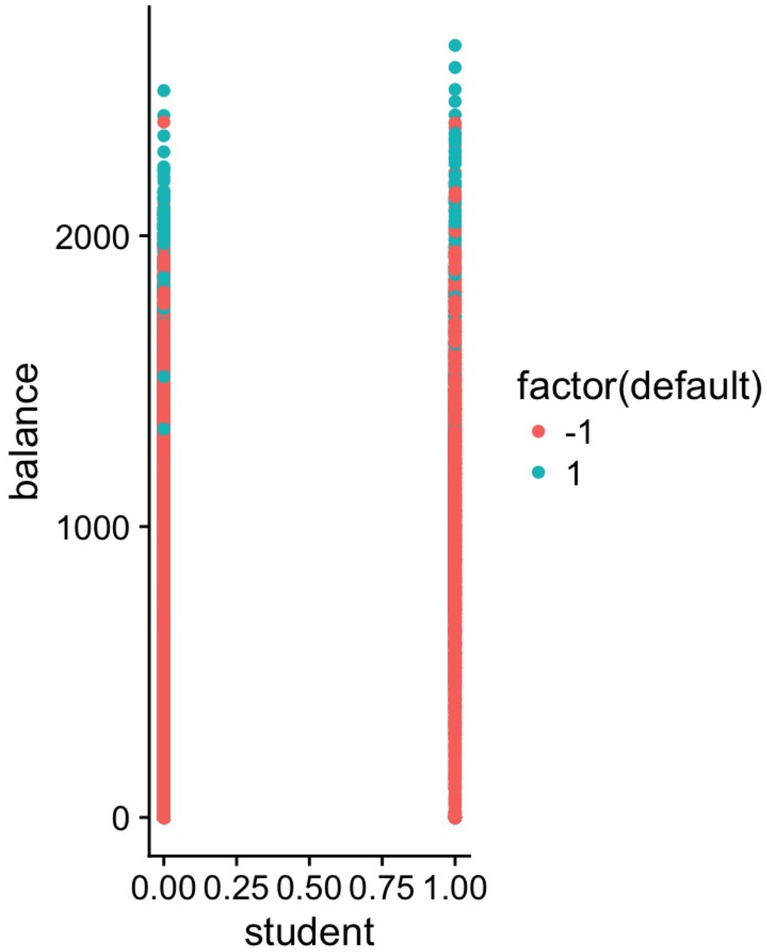
Parameter K is a hyper-parameter, it's value may affect prediction accuracy significantly.

K-nearest neighbors algorithm

Feature scaling is an important issue in distance-based methods.

Which of these two features will affect distance the most?

We will see in later lectures how to address this.



Quick vector algebra review

- A (real-valued) vector is just an array of real values, for instance $x = \langle 1, 2.5, -6 \rangle$ is a three-dimensional vector.

- Vector sums are computed pointwise, and are only defined when dimensions match, so

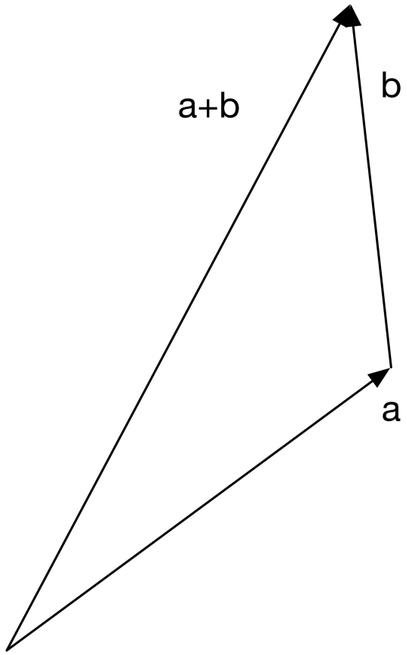
$$\langle 1, 2.5, -6 \rangle + \langle 2, -2.5, 3 \rangle = \langle 3, 0, -3 \rangle$$

.

Quick vector algebra review

In general, if $c = a + b$ then $cd = ad + bd$ for all vectors d.

Vector addition can be viewed geometrically as taking a vector a, then tacking on b to the end of it; the new end point is exactly c.



Quick vector algebra review

Scalar Multiplication: vectors can be scaled by real values;

$$2\langle 1, 2.5, -6\rangle = \langle 2, 5, -12\rangle$$

In general, $ax = \langle ax_1, ax_2, \ldots, ax_p\rangle$

Quick vector algebra review

The norm of a vector x, written $\|x\|$ is its length.

Unless otherwise specified, this is its Euclidean length, namely:

$$\|x\| = \sqrt{\sum_{j=1}^{p} x_j^2}$$

Quick vector algebra review

Quiz

Write Euclidean distance of vectors u and v as a vector norm

Quick vector algebra review

The dot product, or inner product of two vectors u and v is defined as

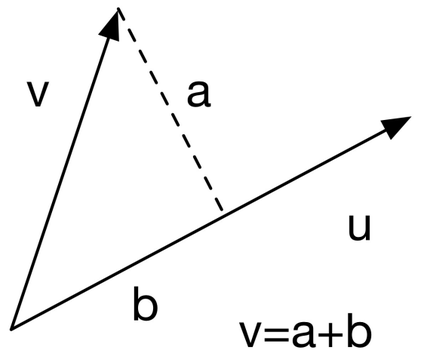
$$u'v = \sum_{j=1}^{p} u_i v_i$$

Quick vector algebra review

The dot product, or inner product of two vectors u and v is defined as

$$u'v = \sum_{j=1}^{p} u_i v_i$$

A useful geometric interpretation of the inner product $v'u$ is that it gives the projection of v onto u (when $\|u\| = 1$).



Back to KNN classification

The algorithm we saw scans the complete training set X to make a prediction for observation x.

This is not a good idea when the training set is massive.

Back to KNN classification

Suppose I give you function `hash_vector` that does the following:

- Generate random unit vector u, $\|u\| = 1$
- Given vector x, compute $a = x'u$
- Round a to the nearest integer

Back to KNN classification

Suppose I give you function `hash_vector` that does the following:

- Generate random unit vector u, $\|u\| = 1$
- Given vector x, compute $a = x'u$
- Round a to the nearest integer

Quiz

Sketch a system that uses locality sensitive hashing (LSH) with `hash_vector` to find candidate near neighbors of x to avoid computing distance to all vectors in dataset X.

The curse of dimensionality

Distance-based methods like KNN can be problematic in high-dimensional problems

The curse of dimensionality

Consider the case where we have many covariates. We want to use k-nearest neighbor methods.

The curse of dimensionality

Consider the case where we have many covariates. We want to use k-nearest neighbor methods.

Basically, we need to define distance and look for small multi-dimensional "balls" around the target points. With many covariates this becomes difficult.

The curse of dimensionality

Imagine we have equally spaced data and that each covariate is in $[0, 1]$. We want to something like kNN with a local focus that uses 10% of the data in the local fitting.

The curse of dimensionality

Imagine we have equally spaced data and that each covariate is in $[0, 1]$. We want to something like kNN with a local focus that uses 10% of the data in the local fitting.

If we have p covariates and we are forming p-dimensional cubes, then each side of the cube must have size l determined by $l \times l \times \cdots \times l = l^p = .10$.

The curse of dimensionality

If the number of covariates is p=10, then $l = .1^{1/10} = .8$. So it really isn't local!

If we reduce the percent of data we consider to 1%, $l = 0.63$. Still not very local.

The curse of dimensionality

If the number of covariates is p=10, then $l = .1^{1/10} = .8$. So it really isn't local! If we reduce the percent of data we consider to 1%, $l = 0.63$. Still not very local.

If we keep reducing the size of the neighborhoods we will end up with very small number of data points in each average and thus predictions with very large variance.

The curse of dimensionality

This is known as the curse of dimensionality.

Because of this so-called curse, it is not always possible to use KNN. But other methods, like Decision Trees, thrive on multidimensional data.

Geometric Representation of Classification Problems

Recall that our setting is that we observe for subject i predictors (covariates) x_i, and qualitative outcomes (or classes) g_i, which can takes values from a discrete set G.

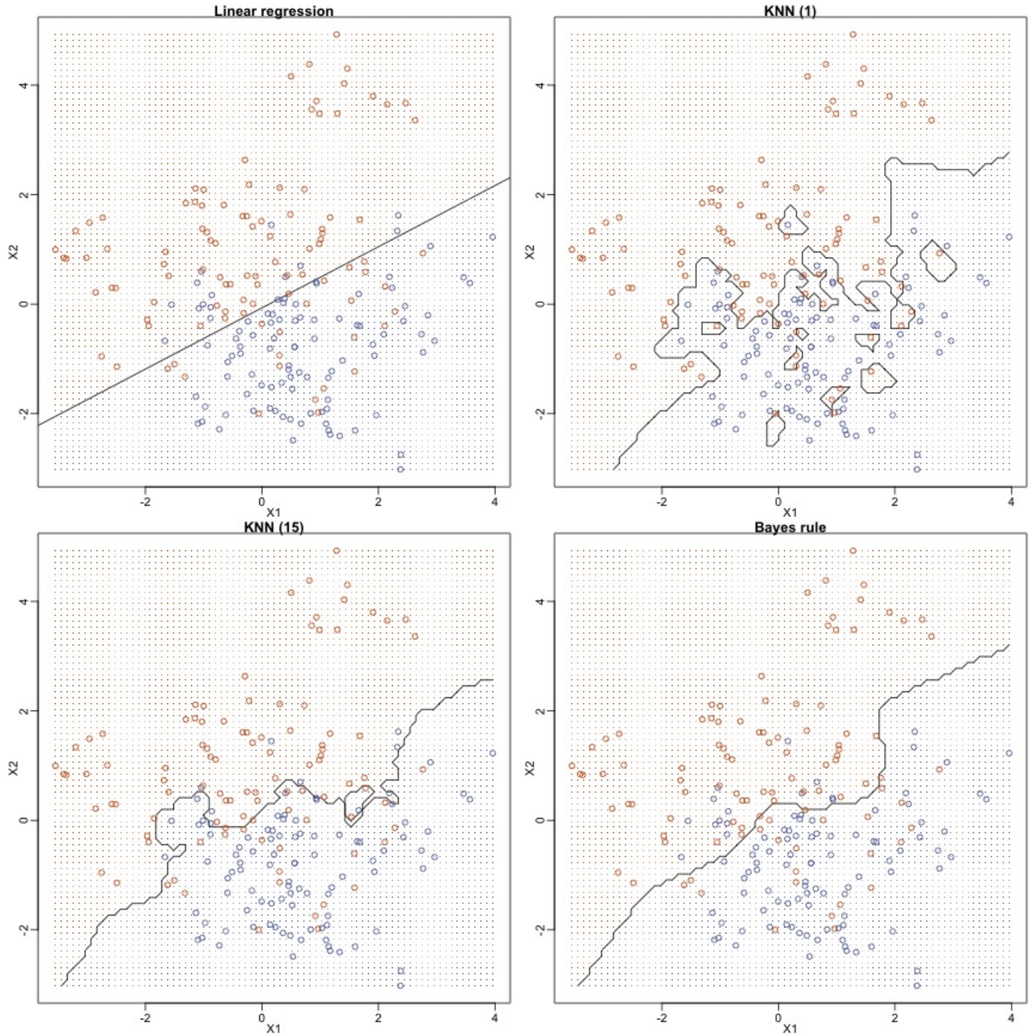
Geometric Representation of Classification Problems

Recall that our setting is that we observe for subject i predictors (covariates) x_i, and qualitative outcomes (or classes) g_i, which can takes values from a discrete set G.

Since our prediction $\hat{G}(x)$ will always take values in the discrete set G, we can always divide the input space into a collection of regions taking the same predicted values.

Geometric Representation of Classification Problems

Boundaries can be smooth or rough depending on the prediction function.



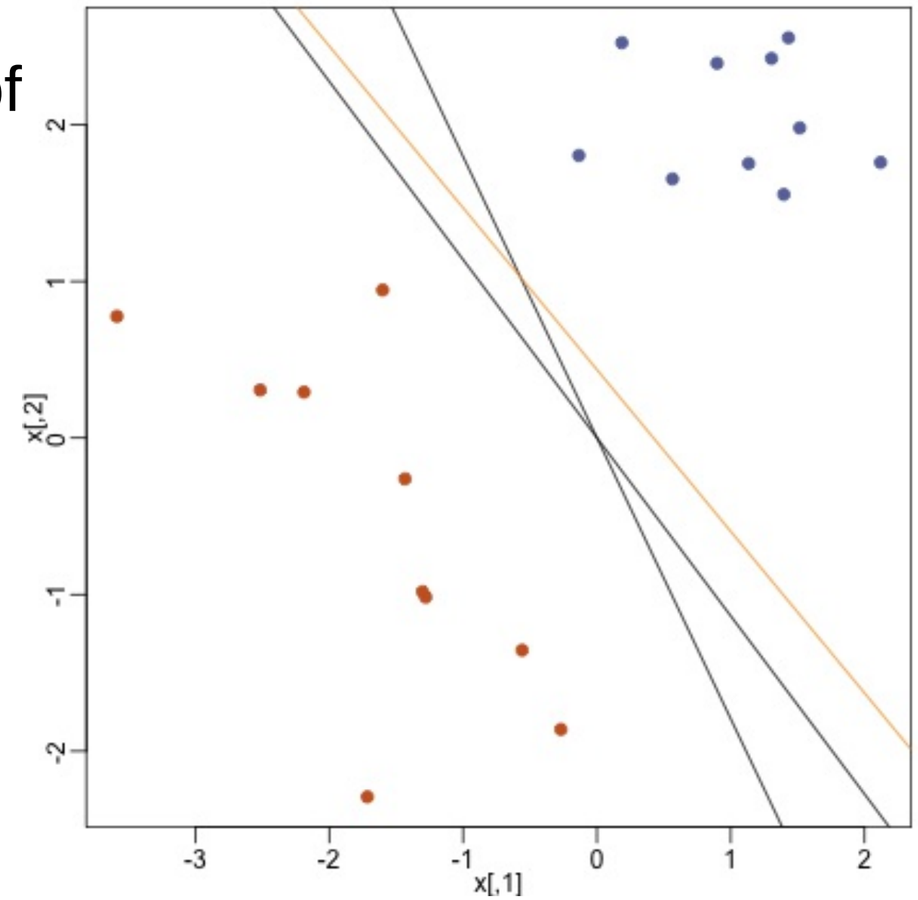
Geometric Representation of Classification Problems

For an important class of procedures, these decision boundaries are linear.

This is what we will refer to as linear methods for classification.

Geometric Representation of Classification Problems

The perceptron algorithm is a way of finding discriminant functions that are linear with respect to the covariates $X_1, \ldots, X_p$.



Geometric Representation of Classification Problems

In p-dimensional space $\mathbf{R}^p$ these are described by vectors w. The decision boundary is thus

$$L = \{x : w'x = 0\}.$$

Geometric Representation of Classification Problems

Notice that this boundary partitions the input space into two sets on each side of the line.

If we restrict estimates to those for which
$|w|| = 1$

Then the signed distance of any point x to the decision boundary L is $w'x$.

Geometric Representation of Classification Problems

With this we can easily describe the two partitions as

$$L^+ = \{x : w'x > 0\},$$

$$L^- = \{x : w'x < 0\}$$

Geometric Representation of Classification Problems

Intuitively, the w we want as an estimate is one that separates the training data as perfectly as possible.

If we code our classes as $y = -1$ if $g = 1$ and $y = +1$ if $g = 2$, we can describe our intuitive requirement for estimate w as:

$$y_i(w'x_i) > 0, i = 1, \ldots, N$$

Perceptron Algorithm

The Perceptron algorithm is one way of finding a vector w that satisfies the separation requirement as much as possible.

Perceptron Algorithm

Penalize w by how far into the wrong side misclassified points are:

$$D(w) = -\sum_{i \in \mathcal{M}} y_i (w' x_i)$$

$\mathcal{M}$: set of points misclassified by w (on the wrong side of the hyper-plane).

Perceptron Algorithm

The perceptron algorithm estimates w by minimizing D.

Perceptron Algorithm

The perceptron algorithm estimates w by minimizing D. We'll see details how next class, but this introduces an important point:

We will write down learning algorithms as optimization problems where we minimize some cost function determined by prediction error.

Perceptron Algorithm

function PERCEPTRON(X,y,MaxIter)
    $w \leftarrow \langle 0, \ldots, 0 \rangle$
    for all $iter = 1, \ldots, MaxIter$ **do**
        for all $i = 1, \ldots, N$ **do** ▷ For every observation
            if $y_i(w'x_i) < 0$ **then** ▷ Check if x_i is incorrectly classified
                $w \leftarrow w + y_i x_i$
            end if
        end for
    end for
    return w
end function

Perceptron Algorithm

Why does this work?

Quiz

Show that if $y_i(w'x_i) < 0$ then after updating w, $w'x_i$ is changed in the proper direction. That is, it is made larger if $y_i = +1$ and made smaller if $y_i = -1$.

Perceptron Algorithm

Some Notes

- `MaxIter` is a hyper-parameter.
- You can interpret the size of w_j as the importance of feature j for classification

Perceptron Algorithm

There are a few problems with this algorithm:

If there exists w that separates the training points perfectly,

Perceptron Algorithm

There are a few problems with this algorithm:

If there exists w that separates the training points perfectly,

then there are an infinite number of ws that also separate the data perfectly

Perceptron Algorithm

Algorithm will converge in a finite number of steps if the training data is separable

Perceptron Algorithm

Algorithm will converge in a finite number of steps if the training data is separable

However, the number of finite steps can be very large (see CIML for a bound on the number of steps)

Perceptron Algorithm

When the training data is not separable, the algorithm will not converge.

Perceptron Algorithm

Summary

- We will represent many ML algorithms geometrically as vectors
- Vector math review
- K-nearest neighbors and perceptron algorithms