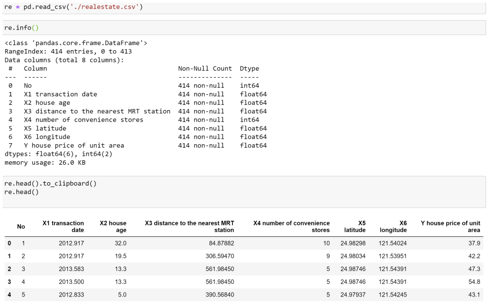
# Classification and Projections

Narayana Santhanam

EE 645
Jan 17, 2023

## Linear Regression



```
re = pd.read_csv('./realestate.csv')
```

```
re.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 8 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   No                                      414 non-null    int64
 1   X1 transaction date                     414 non-null    float64
 2   X2 house age                            414 non-null    float64
 3   X3 distance to the nearest MRT station  414 non-null    float64
 4   X4 number of convenience stores         414 non-null    int64
 5   X5 latitude                             414 non-null    float64
 6   X6 longitude                            414 non-null    float64
 7   Y house price of unit area              414 non-null    float64
dtypes: float64(6), int64(2)
memory usage: 26.0 KB
```

```
re.head().to_clipboard()
re.head()
```

| | No | X1 transaction date | X2 house age | X3 distance to the nearest MRT station | X4 number of convenience stores | X5 latitude | X6 longitude | Y house price of unit area |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2012.917 | 32.0 | 84.87882 | 10 | 24.98298 | 121.54024 | 37.9 |
| **1** | 2 | 2012.917 | 19.5 | 306.59470 | 9 | 24.98034 | 121.53951 | 42.2 |
| **2** | 3 | 2013.583 | 13.3 | 561.98450 | 5 | 24.98746 | 121.54391 | 47.3 |
| **3** | 4 | 2013.500 | 13.3 | 561.98450 | 5 | 24.98746 | 121.54391 | 54.8 |
| **4** | 5 | 2012.833 | 5.0 | 390.56840 | 5 | 24.97937 | 121.54245 | 43.1 |

Design/Training matrix $X$, Target $\mathbf{y}$

: rows are examples/instances, cols are features/attributes

## Recap: Visualizing Linear Regression

Design/Training matrix $X$, Target $\mathbf{y}$

: rows are examples/instances, cols are features/attributes

High school approach: if there is only one feature,

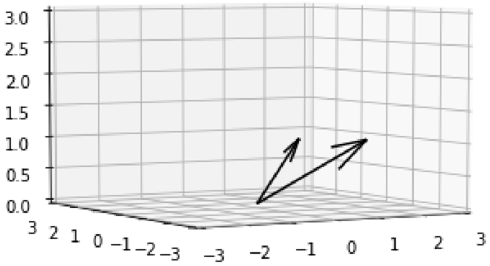
plot points (feature, target) (so rows of $X$ and $\mathbf{y}$)

draw line that minimizes the sum of squares of ordinate (along $y-$axis) distances

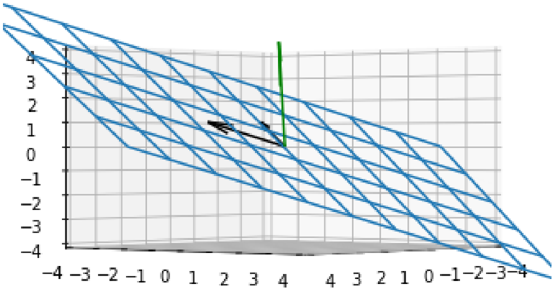
actual computations (differentiate) laborious and not really insightful

Instead: visualize the columns of $X$, project $\mathbf{y}$ into column space of $X$

## Recap: Columns of the design matrix



## Recap: Linear Regression



Columns, Column space, Target

## Recap: Projection

Ordinary Least squares: Find $\mathbf{w}$ that minimizes the training/mean-square-error $||\mathbf{y} - X\mathbf{w}||^2$

minimizer: $\hat{\mathbf{w}}$

Projection of $\mathbf{y}$ into column space of $X$: $X\hat{\mathbf{w}}$,

where $\hat{\mathbf{w}} = (X^T X)^{-1} X^T \mathbf{y}$

## Question

If you want to do linear regression with an intercept, ie model the target as $X\mathbf{w} + b\mathbf{1}$, how should you do it?

## Question

If you want to do linear regression with an intercept, ie model the target as $X\mathbf{w} + b\mathbf{1}$, how should you do it?

Center $X$ and run regression without an intercept

compute $b$ as difference between means of target and prediction

## Question

If you want to do linear regression with an intercept, ie model the target as $X\mathbf{w} + b\mathbf{1}$, how should you do it?

Center $X$ and run regression without an intercept

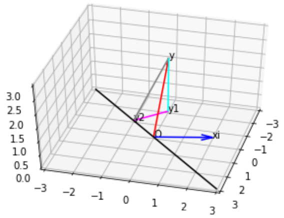
compute $b$ as difference between means of target and prediction

Why?

## Geometry isn't enough

Any (independent) feature reduces training error

including a randomly generated column we can add to $X$



But clearly randomly generated columns cannot help in prediction on test examples

Test/Generalization error isn't captured by training error alone!

In our notebook, we added enough additional features to bring down training error to 0. But as expected, such an approach does terribly.

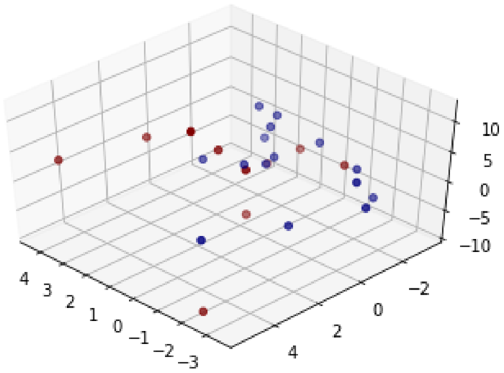
## Regularization

Instead of finding minimum over all possible **w** of $||\mathbf{y} - X\mathbf{w}||^2$, restrict **w** to be in a special set of vectors.

LASSO ($\ell_1$-regularization): Constrain $||\mathbf{w}||_1 < B$
    $B$ is chosen by validation
    LASSO successfully disregarded fake features in our notebook example

Ridge ($\ell_2$-regularization): Constrain $||\mathbf{w}||_2 < B$
    again, $B$ is chosen by validation
    Helps with stable solutions

## Classification



Training matrix $X$, class labels $\mathbf{y}$ (categorical)

## Binary classification

How about we try linear regression (on the centered $X$), but interpret the categorical $\mathbf{y}$ as a numerical vector (one class gets label 1 and another gets -1)

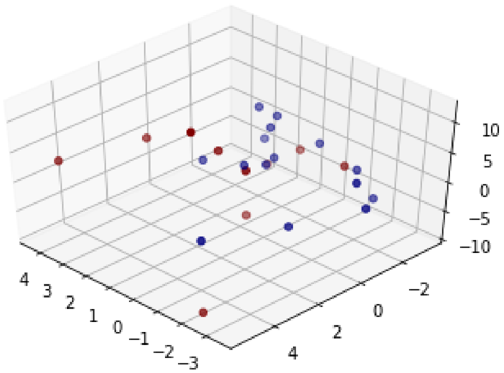
LinearRegression fits $\mathbf{y} \approx X\hat{\mathbf{w}}$

To predict on a test example $\mathbf{z}$, obtain dot product $\mathbf{z}^T\mathbf{w}$,

predict class label 1 if $\mathbf{z}^T\mathbf{w} > 0$, -1 else.

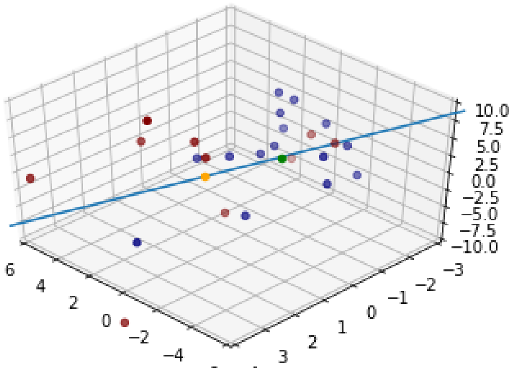
Is this “hack” any good?

## Fisher Discriminant



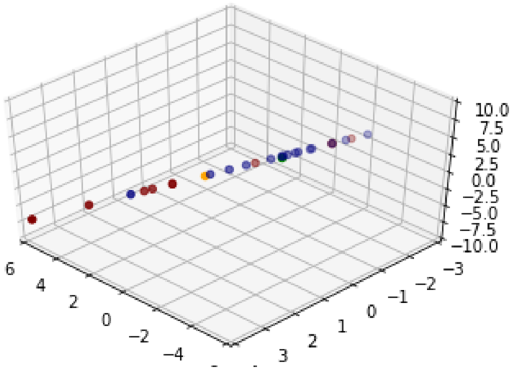
$n_1$ red and $n_2$ blue

## Fisher Discriminant



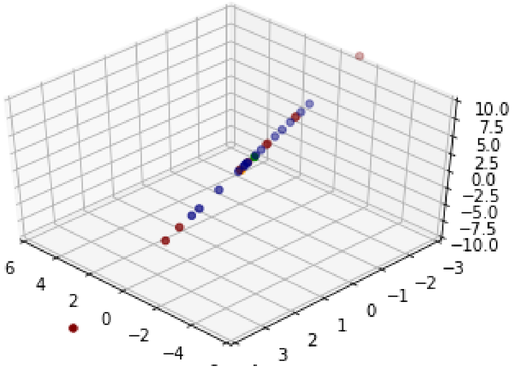
Class means: $\mathbf{m}_1$ and $\mathbf{m}_2$

## Fisher Discriminant



Set threshold to be some point on the line

## Fisher Discriminant



## Fisher Discriminant

Which direction? Let $\mathbf{u}$ be any vector (length 1, direction arbitrary)

Spread between class means: $\left(\mathbf{u}^T(\mathbf{m}_1 - \mathbf{m}_2)\right)^2$

We love matrices: the above is simply $\mathbf{u}^T S_b \mathbf{u}$,
where $S_b = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T$

If you are not that adept with matrices, not to worry. We will practice it. Reading "matrix equations" properly is a very useful skill in ML/AI. Once you acquire it, you can read something and figure out what the author was thinking

Bigger this spread, the better!

## Fisher Discriminant

Which direction? Let $\mathbf{u}$ be any vector (length 1, direction arbitrary)

Spread of projections within class 1: $\sum \left(\mathbf{u}^T(\mathbf{x}_i - \mathbf{m}_1)\right)^2$
 sum over all $n_1$ red examples
Spread of projections within class 1: $\sum \left(\mathbf{u}^T(\mathbf{x}_j - \mathbf{m}_2)\right)^2$
 sum over all $n_2$ blue examples
Total spread: sum over the two classes

We love matrices: the above is simply $\mathbf{u}^T S_w \mathbf{u}$,
 where $S_w = X^T X - n_1 \mathbf{m}_1 \mathbf{m}_1^T - n_2 \mathbf{m}_2 \mathbf{m}_2^T$

If you are not that adept with matrices, not to worry. We will practice it. Reading "matrix equations" properly is a very useful skill in ML/AI. Once you acquire it, you can read something and figure out what the author was thinking

Smaller this spread, the better!

## Fisher Discriminant

Which direction? Let $\mathbf{u}$ be any vector (length 1, direction arbitrary)

Maximize the spread between class means, while controling for spread within classes

Formulation: $\max \mathbf{u}^T S_b \mathbf{u}$ subject to $\mathbf{u}^T S_w \mathbf{u} = 1$.

## Fisher Discriminant

Turns out the solution to

Formulation: $\max \mathbf{u}^T S_b \mathbf{u}$ subject to $\mathbf{u}^T S_w \mathbf{u} = 1$.

is exactly to choose the vector along the linear regression solution

$$(X^T X)^{-1} X^T \mathbf{y},$$

with $\mathbf{y}$ being the vector of class labels $\pm 1$

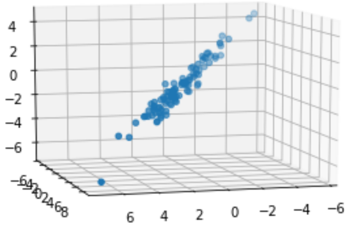
the values of the class label are important, you can't have labels to be 1 and 0 for example.

## Fisher Discriminant

Can you figure out why the Fisher discriminant must coincide with the linear regression solution?

## Principal Components Analysis

A lot of things we eyeball in 2 or 3-d can be obtained by analyzing the information matrix $X^TX$



## Principal Components Analysis

Not unlike our intuition