Regression, Classification, Logistic regression, Projections

Narayana Santhanam

EE 645
Jan 15, 2025

Weather data

	d	cc	z	o3	pv	r	ciwc	clwc	q	crwc	cswc	t	u	v	w	vo	cc +6
326	-0.000033	0.000000	1355.880746	9.158514e-08	0.000005	74.670949	0.000000	-1.084202e-19	0.001228	0.0	1.673474e-07	288.839977	-0.361970	-0.120074	0.159247	-0.000004	0.312499
327	-0.000044	0.000000	1628.316900	9.155199e-08	0.000008	83.140626	0.000000	-1.084202e-19	0.001201	0.0	1.673474e-07	287.705038	-0.389090	-0.235887	0.169665	-0.000002	0.492195
328	-0.000051	0.000000	1848.214115	9.151515e-08	0.000009	89.451752	0.000000	-1.084202e-19	0.001175	0.0	1.255106e-07	286.363458	-0.286911	-0.426284	0.181739	0.000002	0.437505
329	-0.000058	0.000000	2049.537613	9.149919e-08	0.000015	104.877487	0.000002	-1.084202e-19	0.001150	0.0	2.510211e-07	285.192490	-0.271845	-0.637063	0.183555	0.000009	0.601560
330	-0.000063	0.007813	2095.192587	9.149919e-08	0.000014	111.802835	0.000002	2.871108e-06	0.001124	0.0	2.510211e-07	284.475603	-0.244451	-0.681304	0.204124	0.000011	0.703127
331	-0.000062	0.117193	2157.018384	9.154708e-08	0.000016	116.302008	0.000002	5.006932e-05	0.001051	0.0	2.510211e-07	284.045364	-0.248560	-0.781135	0.225922	0.000010	0.765630
332	-0.000068	0.312499	2243.977147	9.156181e-08	0.000012	110.556843	0.000009	1.177855e-04	0.000904	0.0	6.693896e-07	283.157866	-0.332659	-0.734115	0.230142	0.000012	0.804694
333	-0.000073	0.492195	2308.075951	9.157532e-08	0.000011	108.295192	0.000025	6.626890e-05	0.000810	0.0	1.464290e-06	282.398591	-0.342247	-0.845758	0.252047	0.000016	0.249996
334	-0.000074	0.437505	2351.133203	9.169811e-08	0.000011	107.716068	0.000011	1.335182e-05	0.000746	0.0	5.438791e-07	282.447337	-0.476749	-0.823291	0.276569	0.000020	0.234370

Design/Training matrix X

Target: cc $+6$ hours

Weather data

	d	cc	z	o3	pv	r	ciwc	clwc	q	crwc	cswc	t	u	v	w	vo	cc +6
326	-0.000033	0.000000	1355.880746	9.158514e-08	0.000005	74.670949	0.000000	-1.084202e-19	0.001228	0.0	1.673474e-07	288.839977	-0.361970	-0.120074	0.159247	-0.000004	0.312499
327	-0.000044	0.000000	1628.316900	9.155199e-08	0.000008	83.140626	0.000000	-1.084202e-19	0.001201	0.0	1.673474e-07	287.705038	-0.389090	-0.235887	0.169665	-0.000002	0.492195
328	-0.000051	0.000000	1848.214115	9.151515e-08	0.000009	89.451752	0.000000	-1.084202e-19	0.001175	0.0	1.255106e-07	286.363458	-0.286911	-0.426284	0.181739	0.000002	0.437505
329	-0.000058	0.000000	2049.537613	9.149919e-08	0.000015	104.877487	0.000002	-1.084202e-19	0.001150	0.0	2.510211e-07	285.192490	-0.271845	-0.637063	0.183555	0.000009	0.601560
330	-0.000063	0.007813	2095.192587	9.149919e-08	0.000014	111.802835	0.000002	2.871108e-06	0.001124	0.0	2.510211e-07	284.475603	-0.244451	-0.681304	0.204124	0.000011	0.703127
331	-0.000062	0.117193	2157.018384	9.154708e-08	0.000016	116.302008	0.000002	5.006932e-05	0.001051	0.0	2.510211e-07	284.045364	-0.248560	-0.781135	0.225922	0.000010	0.765630
332	-0.000068	0.312499	2243.977147	9.156181e-08	0.000012	110.556843	0.000009	1.177855e-04	0.000904	0.0	6.693896e-07	283.157866	-0.332659	-0.734115	0.230142	0.000012	0.804694
333	-0.000073	0.492195	2308.075951	9.157532e-08	0.000011	108.295192	0.000025	6.626890e-05	0.000810	0.0	1.464290e-06	282.398591	-0.342247	-0.845758	0.252047	0.000016	0.249996
334	-0.000074	0.437505	2351.133203	9.169811e-08	0.000011	107.716068	0.000011	1.335182e-05	0.000746	0.0	5.438791e-07	282.447337	-0.476749	-0.823291	0.276569	0.000020	0.234370

Design/Training matrix X

Target: cc $+6$ hours

rows are examples/instances, cols are features/attributes

Recap: Visualizing Linear Regression

Design/Training matrix X, Target y

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

High school approach: if there is only one feature,

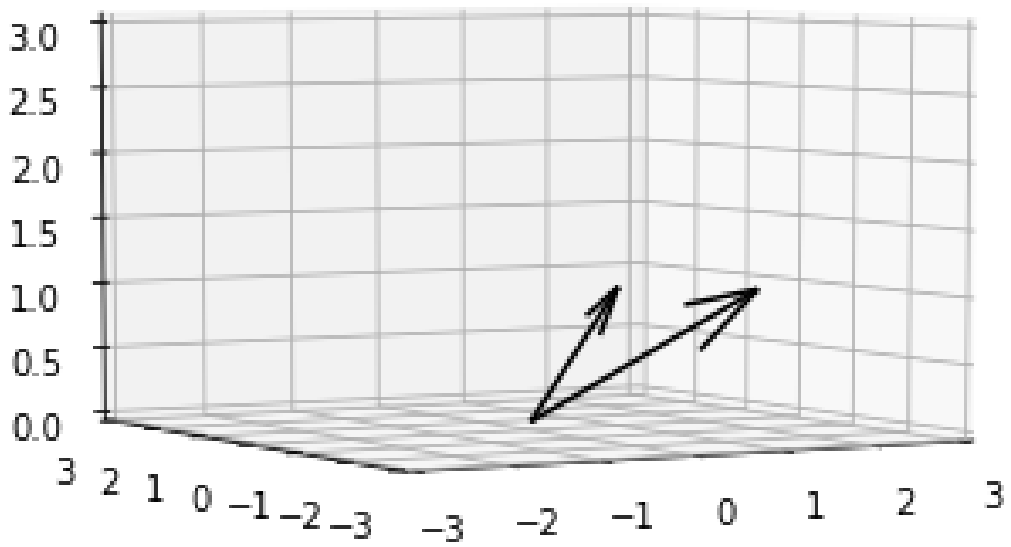
plot points (feature, target) (so rows of X and y)

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

actual computations (differentiate) laborious and not really insightful

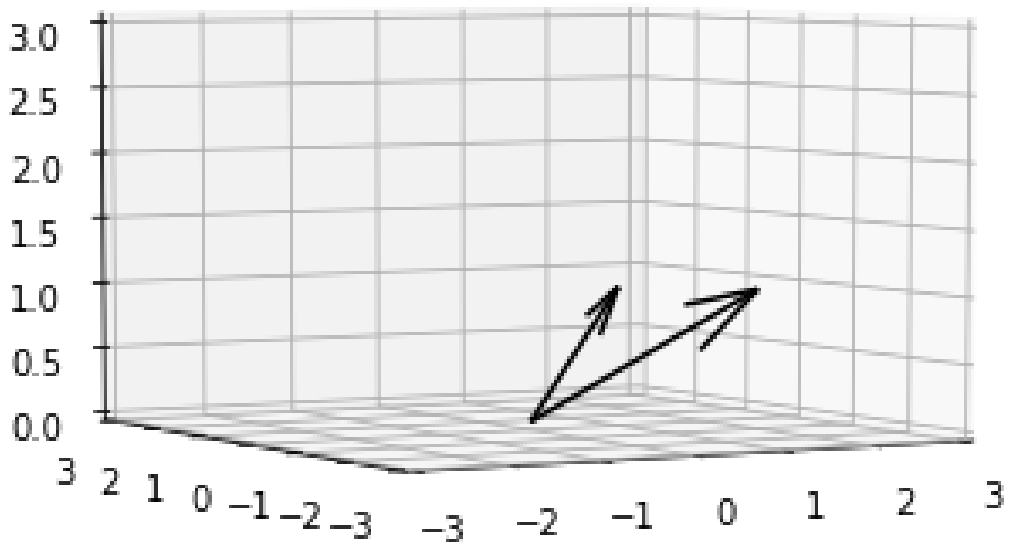
Instead: visualize the columns of X, project y into column space of X

Recap: Columns of the design matrix



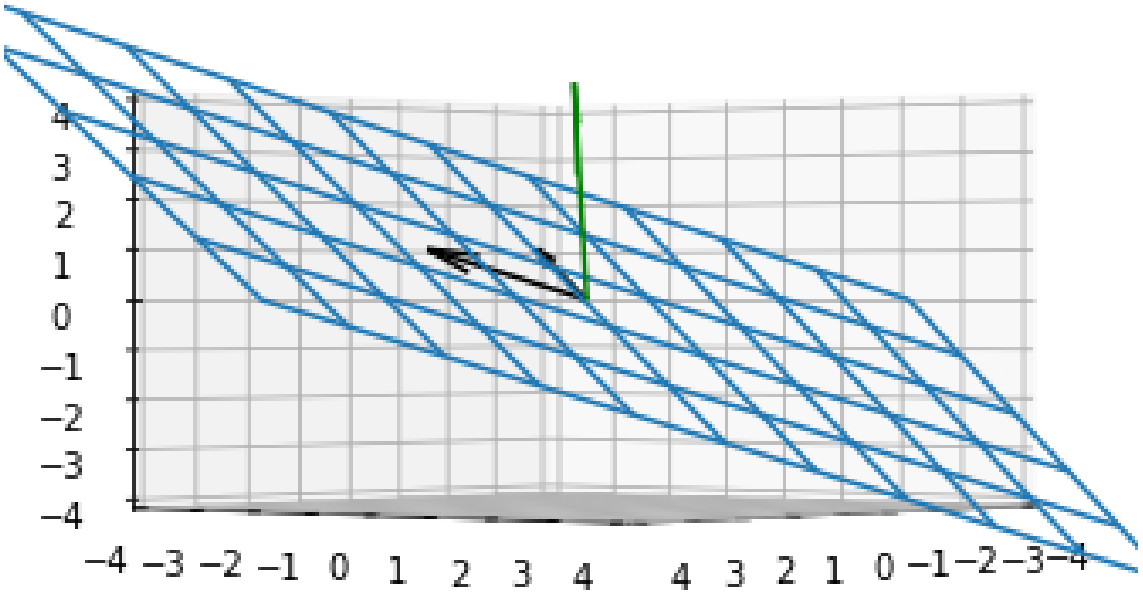
(above fig: X has 3 rows/examples and 2 columns/features)

Recap: Columns of the design matrix



(above fig: X has 3 rows/examples and 2 columns/features)
caricature pictures like the above really help

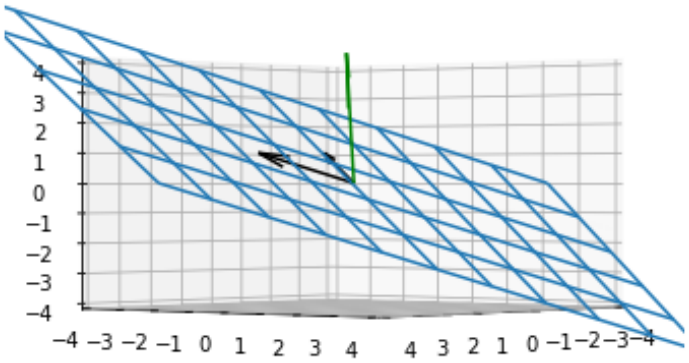
Recap: Linear Regression



Columns, Column space, Target

(above fig: X has 3 rows/examples and 2 columns/features
y is a numerical target for each example,
hence 3-coordinate vector
Xw traces all points in blue plane when w varies)

Recap: Projection



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

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

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

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

Question

If you want to do linear regression with an intercept, ie model the target as $X\mathrm{w} + b1$, how should you do it?

Question

If you want to do linear regression with an intercept, ie model the target as $X\mathrm{w} + b1$, 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 $X\mathrm{w} + b1$, how should you do it?

Center X and run regression without an intercept

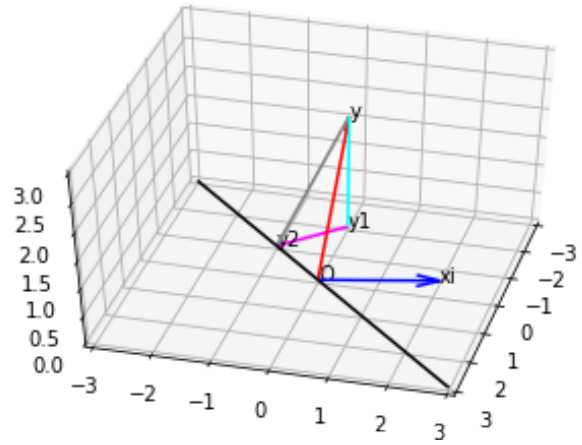
compute b as difference between means of target and prediction

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 notebook for class, we add 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 $||\mathrm{y} - X\mathrm{w}||^2$, restrict w to be in a special set of vectors.

LASSO (ℓ_1-regularization): Constrain $||\mathrm{w}||_1 < B$

B is chosen by validation

LASSO successfully disregarded fake features in our notebook example

Ridge (ℓ_2-regularization): Constrain $||\mathrm{w}||_2 < B$

again, B is chosen by validation

Helps with stable solutions

Weather data

	d	cc	z	o3	pv	r	ciwc	clwc	q	crwc	cswc	t	u	v	w	vo	cc +6
326	-0.000033	0.000000	1355.880746	9.158514e-08	0.000005	74.670949	0.000000	-1.084202e-19	0.001228	0.0	1.673474e-07	288.839977	-0.361970	-0.120074	0.159247	-0.000004	0.312499
327	-0.000044	0.000000	1628.316900	9.155199e-08	0.000008	83.140626	0.000000	-1.084202e-19	0.001201	0.0	1.673474e-07	287.705038	-0.389090	-0.235887	0.169665	-0.000002	0.492195
328	-0.000051	0.000000	1848.214115	9.151515e-08	0.000009	89.451752	0.000000	-1.084202e-19	0.001175	0.0	1.255106e-07	286.363458	-0.286911	-0.426284	0.181739	0.000002	0.437505
329	-0.000058	0.000000	2049.537613	9.149919e-08	0.000015	104.877487	0.000002	-1.084202e-19	0.001150	0.0	2.510211e-07	285.192490	-0.271845	-0.637063	0.183555	0.000009	0.601560
330	-0.000063	0.007813	2095.192587	9.149919e-08	0.000014	111.802835	0.000002	2.871108e-06	0.001124	0.0	2.510211e-07	284.475603	-0.244451	-0.681304	0.204124	0.000011	0.703127
331	-0.000062	0.117193	2157.018384	9.154708e-08	0.000016	116.302008	0.000002	5.006932e-05	0.001051	0.0	2.510211e-07	284.045364	-0.248560	-0.781135	0.225922	0.000010	0.765630
332	-0.000068	0.312499	2243.977147	9.156181e-08	0.000012	110.556843	0.000009	1.177855e-04	0.000904	0.0	6.693896e-07	283.157866	-0.332659	-0.734115	0.230142	0.000012	0.804694
333	-0.000073	0.492195	2308.075951	9.157532e-08	0.000011	108.295192	0.000025	6.626890e-05	0.000810	0.0	1.464290e-06	282.398591	-0.342247	-0.845758	0.252047	0.000016	0.249996
334	-0.000074	0.437505	2351.133203	9.169811e-08	0.000011	107.716068	0.000011	1.335182e-05	0.000746	0.0	5.438791e-07	282.447337	-0.476749	-0.823291	0.276569	0.000020	0.234370

Design/Training matrix X

Target: cc $+6$ hours

rows are examples/instances, cols are features/attributes

Weather data

	d	cc	z	o3	pv	r	ciwc	clwc	q	crwc	cswc	t	u	v	w	vo	cc +6
326	-0.000033	0.000000	1355.880746	9.158514e-08	0.000005	74.670949	0.000000	-1.084202e-19	0.001228	0.0	1.673474e-07	288.839977	-0.361970	-0.120074	0.159247	-0.000004	0.0
327	-0.000044	0.000000	1628.316900	9.155199e-08	0.000008	83.140626	0.000000	-1.084202e-19	0.001201	0.0	1.673474e-07	287.705038	-0.389090	-0.235887	0.169665	-0.000002	1.0
328	-0.000051	0.000000	1848.214115	9.151515e-08	0.000009	89.451752	0.000000	-1.084202e-19	0.001175	0.0	1.255106e-07	286.363458	-0.286911	-0.426284	0.181739	0.000002	1.0
329	-0.000058	0.000000	2049.537613	9.149919e-08	0.000015	104.877487	0.000002	-1.084202e-19	0.001150	0.0	2.510211e-07	285.192490	-0.271845	-0.637063	0.183555	0.000009	1.0
330	-0.000063	0.007813	2095.192587	9.149919e-08	0.000014	111.802835	0.000002	2.871108e-06	0.001124	0.0	2.510211e-07	284.475603	-0.244451	-0.681304	0.204124	0.000011	1.0
331	-0.000062	0.117193	2157.018384	9.154708e-08	0.000016	116.302008	0.000002	5.006932e-05	0.001051	0.0	2.510211e-07	284.045364	-0.248560	-0.781135	0.225922	0.000010	1.0
332	-0.000068	0.312499	2243.977147	9.156181e-08	0.000012	110.556843	0.000009	1.177855e-04	0.000904	0.0	6.693896e-07	283.157866	-0.332659	-0.734115	0.230142	0.000012	1.0
333	-0.000073	0.492195	2308.075951	9.157532e-08	0.000011	108.295192	0.000025	6.626890e-05	0.000810	0.0	1.464290e-06	282.398591	-0.342247	-0.845758	0.252047	0.000016	0.0
334	-0.000074	0.437505	2351.133203	9.169811e-08	0.000011	107.716068	0.000011	1.335182e-05	0.000746	0.0	5.438791e-07	282.447337	-0.476749	-0.823291	0.276569	0.000020	0.0

Design/Training matrix X

Target: cc $+6$ hours $> .4$

rows are examples/instances, cols are features/attributes

Binary classification

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

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

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

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

Is this "hack" any good?

Binary classification

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

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

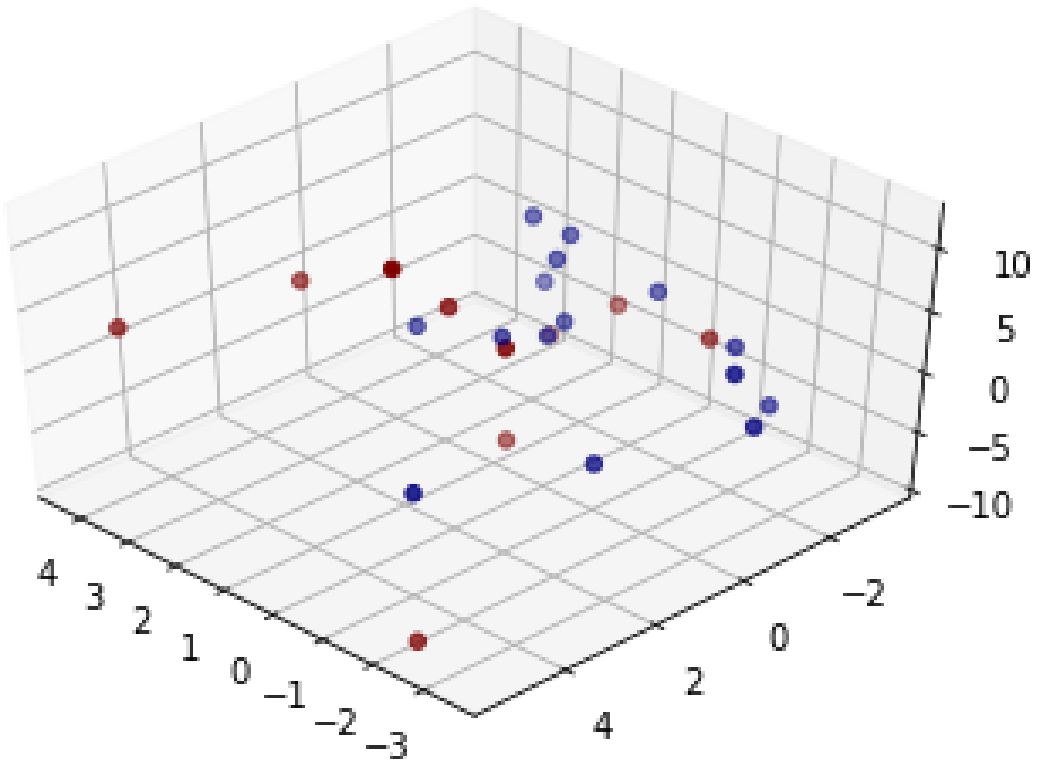
To predict on a test example z, obtain dot product $z^T\hat{w}$,

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

Is this "hack" any good?

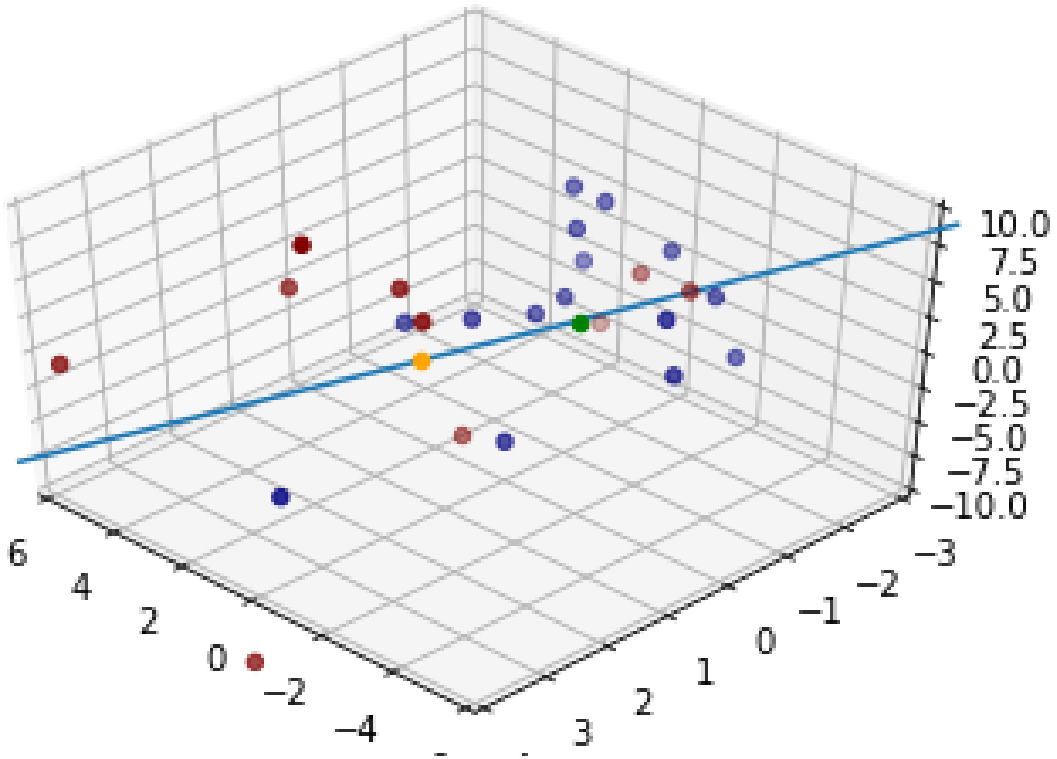
Fisher Discriminant!

Fisher Discriminant



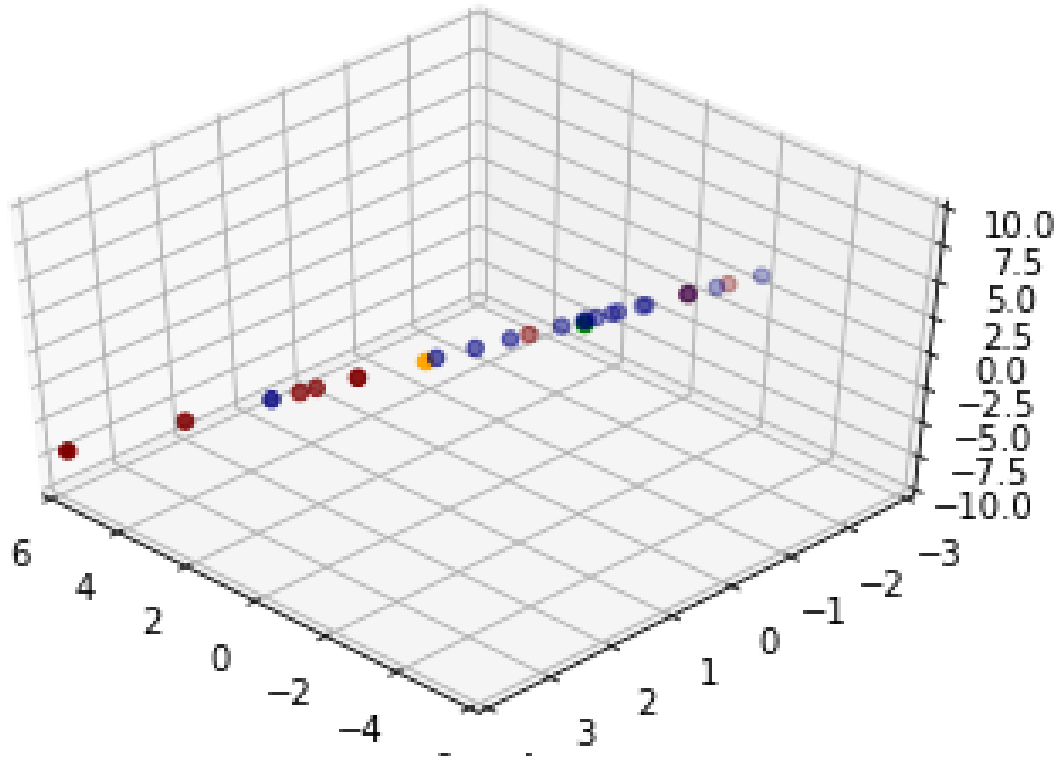
n_1 +1s (red) and n_2 -1s (blue)

Fisher Discriminant



Class means: m_1 and m_2

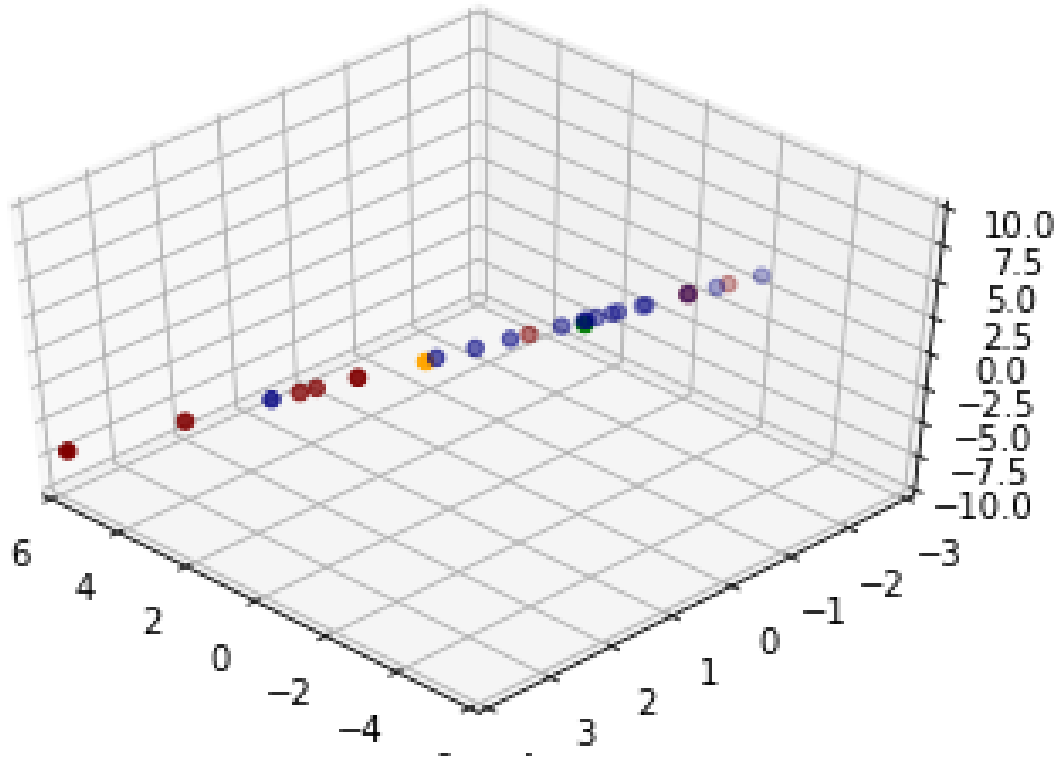
Fisher Discriminant



All points projected onto one line

Set threshold to be some point on the line for classification

Fisher Discriminant

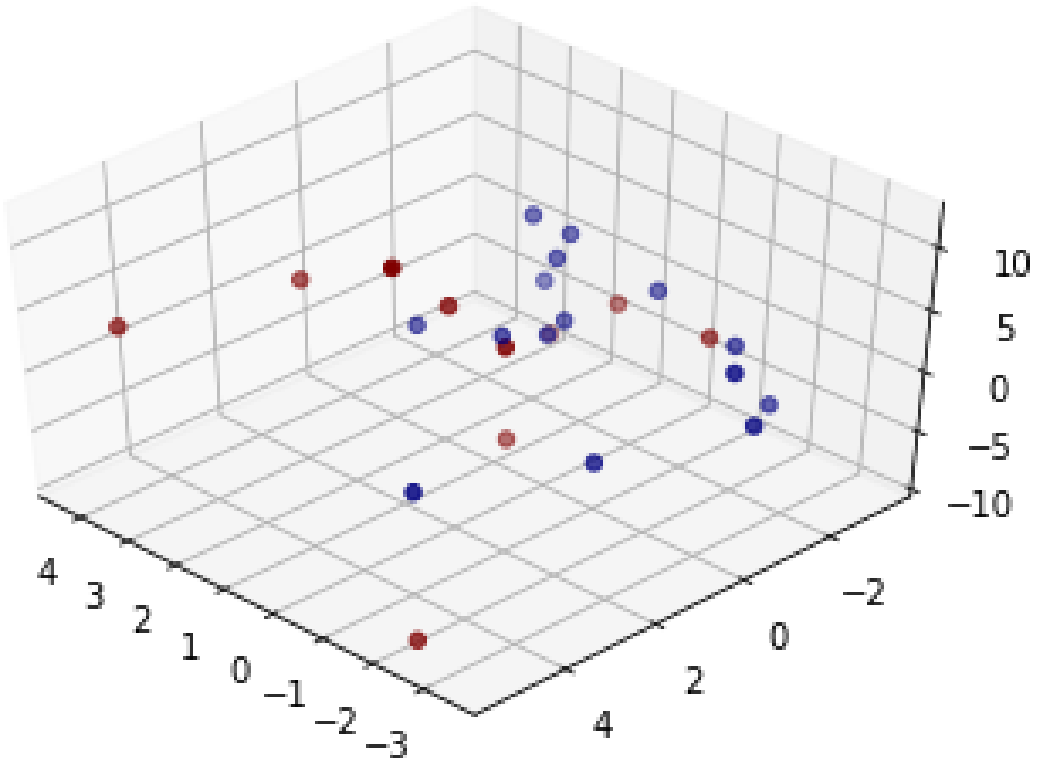


All points projected onto one line

Set threshold to be some point on the line for classification

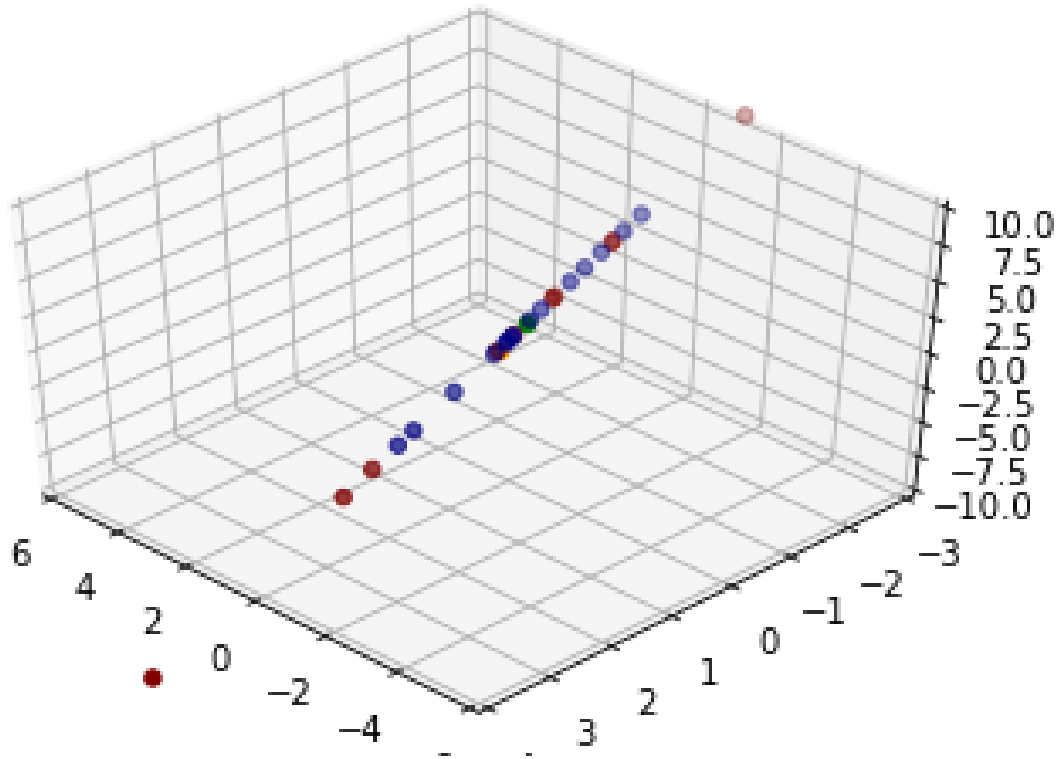
Don't like this line? No matter

Fisher Discriminant



Take the points again and ...

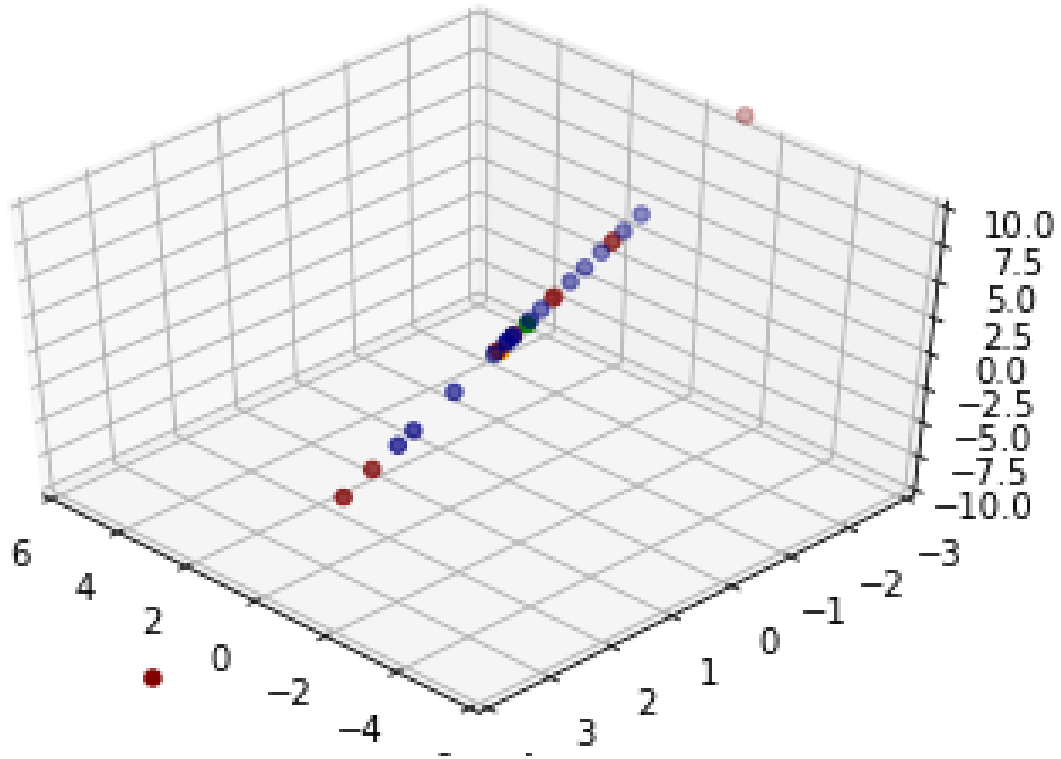
Fisher Discriminant



Project onto another line

Set threshold to be some point on the line

Fisher Discriminant



Project onto another line

Set threshold to be some point on the line

Which line is the best?

Fisher Discriminant

Which direction? Let u be any vector (length 1, direction arbitrary)

Spread between class means: $\left(u^T(m_1 - m_2)\right)^2$

We love matrices: the above is simply $u^T S_b u$,
where $S_b = (m_1 - m_2)(m_1 - m_2)^T$

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 u be any vector (length 1, direction arbitrary)

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

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

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 u be any vector (length 1, direction arbitrary)

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

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

Fisher Discriminant

Turns out the solution to

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

is exactly to choose the vector along the linear regression solution

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

with y being the vector of class labels ± 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?

Weather data

	d	cc	z	o3	pv	r	ciwc	clwc	q	crwc	cswc	t	u	v	w	vo	cc +6
326	-0.000033	0.000000	1355.880746	9.158514e-08	0.000005	74.670949	0.000000	-1.084202e-19	0.001228	0.0	1.673474e-07	288.839977	-0.361970	-0.120074	0.159247	-0.000004	0.0
327	-0.000044	0.000000	1628.316900	9.155199e-08	0.000008	83.140626	0.000000	-1.084202e-19	0.001201	0.0	1.673474e-07	287.705038	-0.389090	-0.235887	0.169665	-0.000002	1.0
328	-0.000051	0.000000	1848.214115	9.151515e-08	0.000009	89.451752	0.000000	-1.084202e-19	0.001175	0.0	1.255106e-07	286.363458	-0.286911	-0.426284	0.181739	0.000002	1.0
329	-0.000058	0.000000	2049.537613	9.149919e-08	0.000015	104.877487	0.000002	-1.084202e-19	0.001150	0.0	2.510211e-07	285.192490	-0.271845	-0.637063	0.183555	0.000009	1.0
330	-0.000063	0.007813	2095.192587	9.149919e-08	0.000014	111.802835	0.000002	2.871108e-06	0.001124	0.0	2.510211e-07	284.475603	-0.244451	-0.681304	0.204124	0.000011	1.0
331	-0.000062	0.117193	2157.018384	9.154708e-08	0.000016	116.302008	0.000002	5.006932e-05	0.001051	0.0	2.510211e-07	284.045364	-0.248560	-0.781135	0.225922	0.000010	1.0
332	-0.000068	0.312499	2243.977147	9.156181e-08	0.000012	110.556843	0.000009	1.177855e-04	0.000904	0.0	6.693896e-07	283.157866	-0.332659	-0.734115	0.230142	0.000012	1.0
333	-0.000073	0.492195	2308.075951	9.157532e-08	0.000011	108.295192	0.000025	6.626890e-05	0.000810	0.0	1.464290e-06	282.398591	-0.342247	-0.845758	0.252047	0.000016	0.0
334	-0.000074	0.437505	2351.133203	9.169811e-08	0.000011	107.716068	0.000011	1.335182e-05	0.000746	0.0	5.438791e-07	282.447337	-0.476749	-0.823291	0.276569	0.000020	0.0

Design/Training matrix X

Target: cc $+6$ hours $> .4$

rows are examples/instances, cols are features/attributes

Logistic Regression

Generative approach

Very productive: probabilistic worldview
probability model for X and Y

Output is also randomized
Given some example x,
output is a probability distribution on labels
rather than one label
soft prediction

Modeling X

Recall: each example (row) of X is a 16-coordinate vector
Visualize the space of all possible examples

Modeling X

Recall: each example (row) of X is a 16-coordinate vector

Visualize the space of all possible examples

of course unless you are an alien, it isn't truly possible

Modeling X

Recall: each example (row) of X is a 16-coordinate vector

Visualize the space of all possible examples

of course unless you are an alien, it isn't truly possible

but there is such a region in the 16-dimensional real vector space

Modeling X

Recall: each example (row) of X is a 16-coordinate vector

Visualize the space of all possible examples

of course unless you are an alien, it isn't truly possible

but there is such a region in the 16-dimensional real vector space

Require a probability model

Modeling X

Recall: each example (row) of X is a 16-coordinate vector

Visualize the space of all possible examples

of course unless you are an alien, it isn't truly possible

but there is such a region in the 16-dimensional real vector space

Require a probability model

we need a probability model jointly for X and Y

Modeling X

Recall: each example (row) of X is a 16-coordinate vector

Visualize the space of all possible examples

of course unless you are an alien, it isn't truly possible

but there is such a region in the 16-dimensional real vector space

Require a probability model

we need a probability model jointly for X and Y

but one step at a time

Modeling X

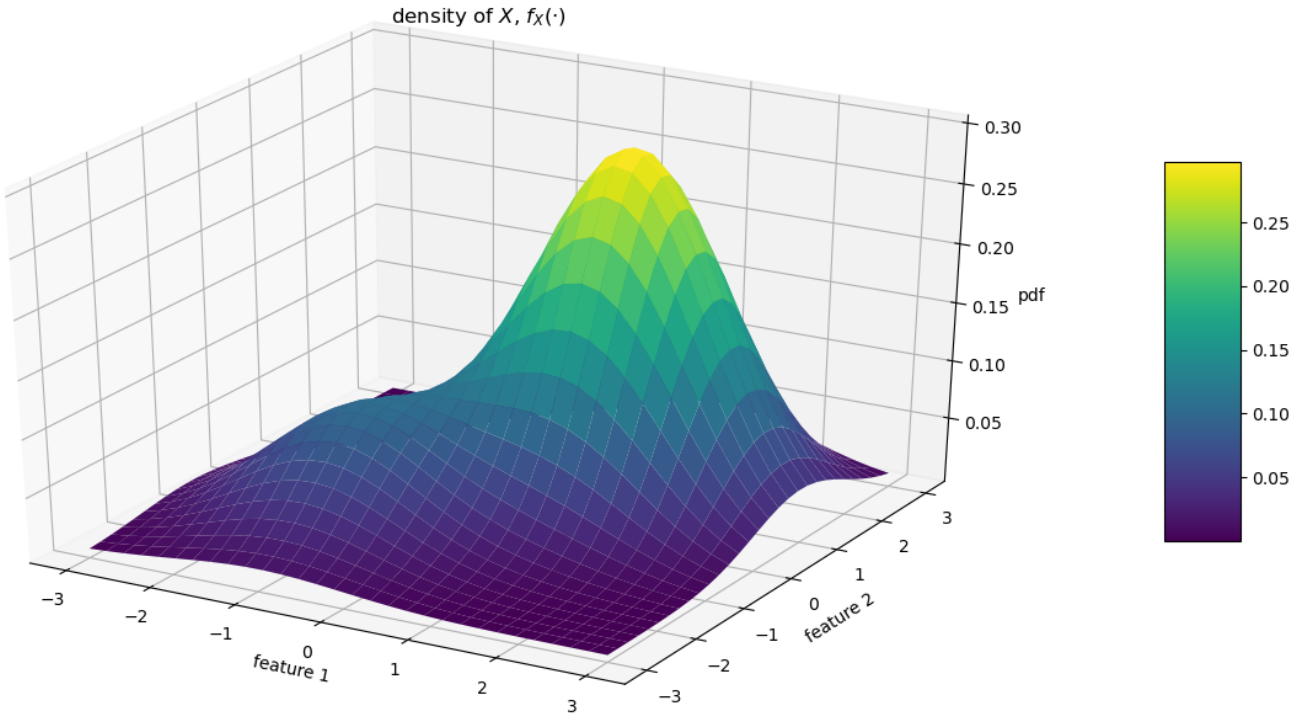
Probability model on the space of all possible examples

- depends on X categorical (discrete)/real-valued (continuous)
- In our example, X is continuous
- density of X alone is marginal

Modeling X

Probability model on the space of all possible examples

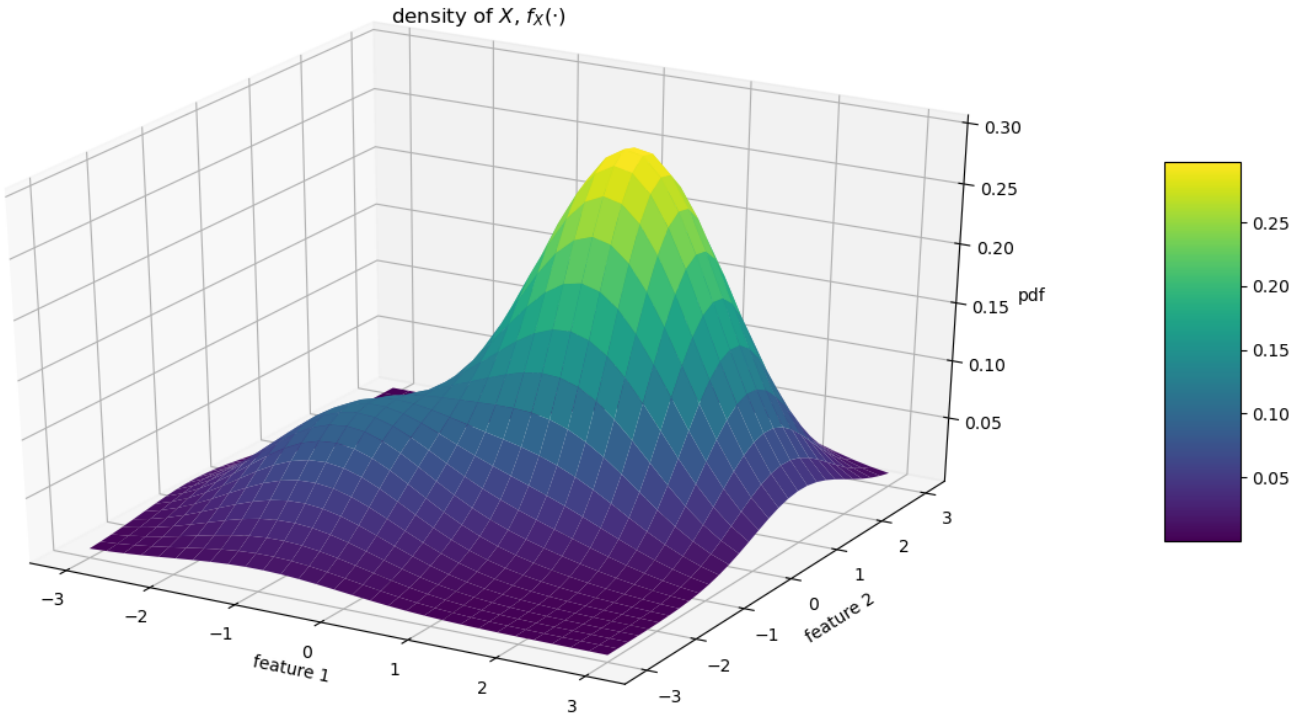
- depends on X categorical (discrete)/real-valued (continuous)
- In our example, X is continuous
- density of X alone is marginal



above: prob density just made up for illustration

Modeling X

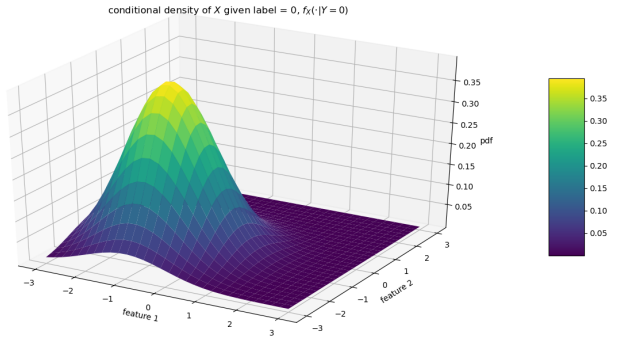
Probability model on the space of all possible examples



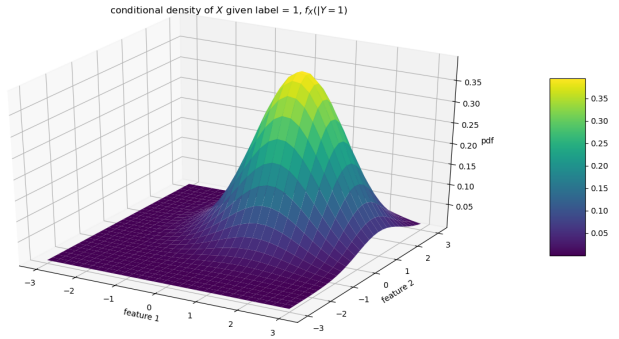
Modeling X and Y together

Probability model on the space of all possible examples for each value of Y

conditional density of X given $Y = 0$ (alternatively $Y = 1$)

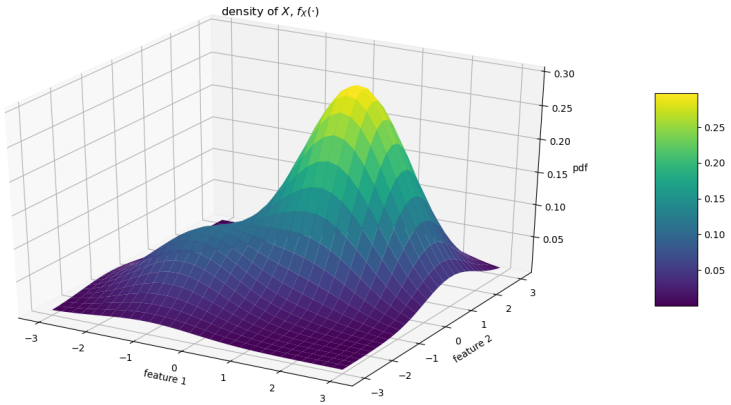


.25 +



.75

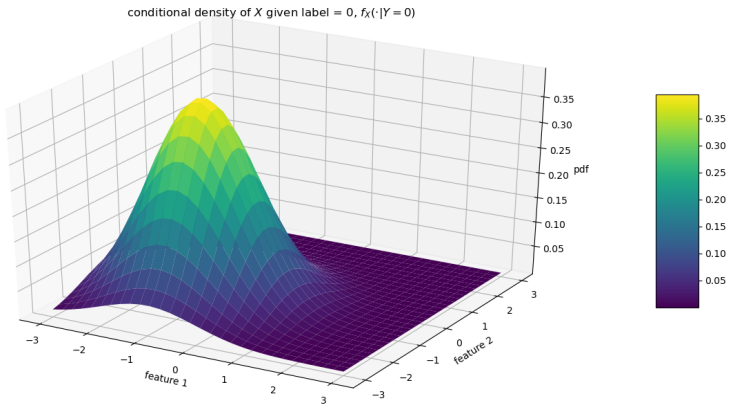
=

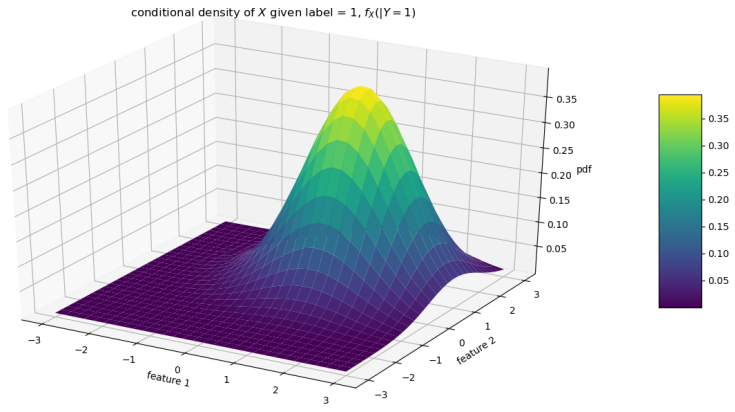


Modeling X and Y together

Probability model on the space of all possible examples for each value of Y

conditional density of X given $Y = 0$ and $Y = 1$ resp.





What probability model for conditional densities?

What probability model for conditional densities?

Rather than presuppose a form for the model, why not one with as little additional assumptions as possible?

What probability model for conditional densities?

Rather than presuppose a form for the model, why not one with as little additional assumptions as possible?

We have data corresponding to each label

average (expectation) of examples corresponding to $Y = 0$ (resp $Y = 1$)

What probability model for conditional densities?

Rather than presuppose a form for the model, why not one with as little additional assumptions as possible?

We have data corresponding to each label

average (expectation) of examples corresponding to $Y = 0$ (resp $Y = 1$)

Infinity of models whose expectation matches observed expectation

pick one that has "maximum entropy" among them

Maximum entropy models

Information is quantified using probabilities
How many bits of information do you get when you observe a random variable X?

Entropy of X, $H(X)$ calculated using X's probability mass/density

when X is continuous, this is a white lie

Intuition behind model with maximum entropy: any model with lesser entropy implicitly makes some assumption, which is why we don't get as much information when we observe a sample from this model

Math behind logistic regression

A detailed explanation is available on the class website.
You can take away a quantitative/qualitative understanding
Deliverables for this module:

Fill in mathematical details below or implement logistic regression
    If you do both, a small prize from me
Submit writeup (handwritten/scanned ok) or code on Laulima

Maximum entropy: Step 1

Training data: (x_i, y_i), $i = 1, \ldots, n$. $\mathrm{x}_i \in \mathbb{R}^d$, $\mathrm{y}_i \in \{0, 1\}$

Subject to $\mathbb{E}[X|Y = 0] = c_0$ for some $c_0 \in \mathbb{R}^d$, max entropy conditional density given $Y = 0$ evaluated at x is

$$f(X = \mathrm{x}|Y = 0) = \exp(\beta_0^{(0)} + \beta_1^{(0)\,T} \mathrm{x})$$

(explain how $\beta_0^{(0)}$ and $\beta_1^{(0)}$ can be determined in principle). We handle $f(X|Y = 1)$ similarly.

Since we have $f(X|Y = j)$,
Bayes rule (using $\mathrm{P}(Y = j)$ for various j) yields $\mathrm{P}(Y = j|X)$

Maximum entropy: Step 2

Training data: (x_i, y_i), $i = 1, \ldots, n$

Show that an unbiased estimate of $\mathbb{E}[X|Y = j]$ is

$$\frac{\sum_{i:y_i=j} x_i}{N_j},$$

where N_j is the number of examples with label j.

Show that $\mathbb{E}[X\mathrm{P}(Y = j|X)] = \mathbb{E}[X|Y = j]\mathrm{P}(Y = j)$

Maximum entropy: Step 3

Monte Carlo estimate of $\mathbb{E}[X\mathrm{P}(Y = j|X)]$ is

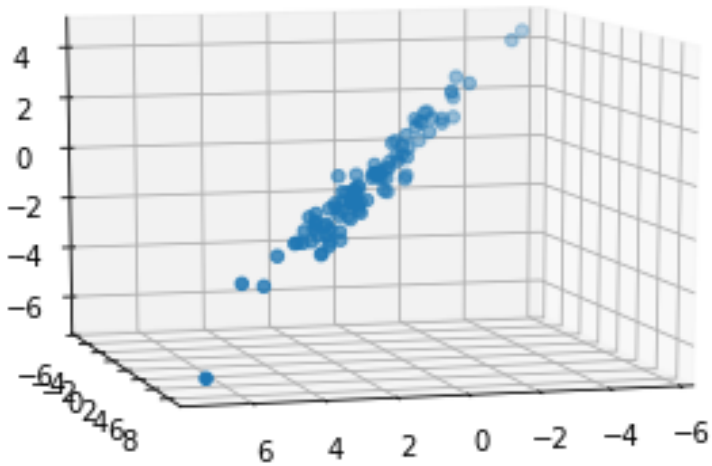
$$\sum_i \mathrm{x}_i \mathrm{P}(Y = j|\mathrm{x}_i)$$

Equate estimate of $\mathbb{E}[X\mathrm{P}(Y = j|X)]$ above to the unbiased estimate of $\mathbb{E}[X|Y = j]$ and $\mathrm{P}(Y = j)$ to obtain the models for $\mathrm{P}(Y = j|X)$

Show that model above is exactly what you would get with logit modeling of probs followed by maximum likelihood.

Principal Components Analysis (PCA)

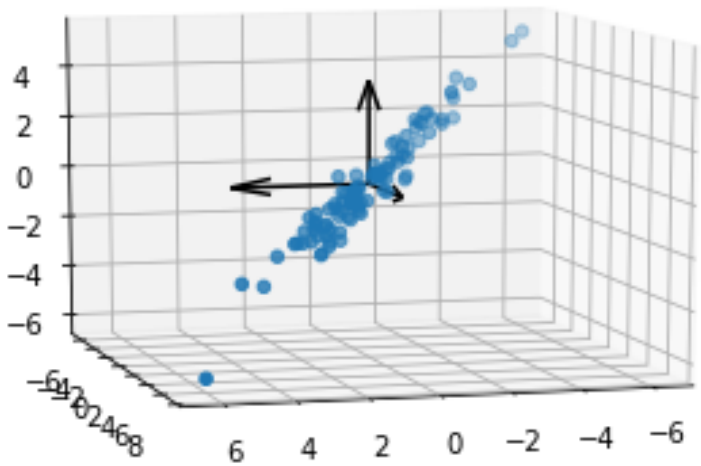
A lot of things we eyeball in 2 or 3-d can be obtained by analyzing the information matrix $X^T X$



Regular coordinates vs ...

The regular coordinate system may not be appropriate to represent row (cols) of a matrix X

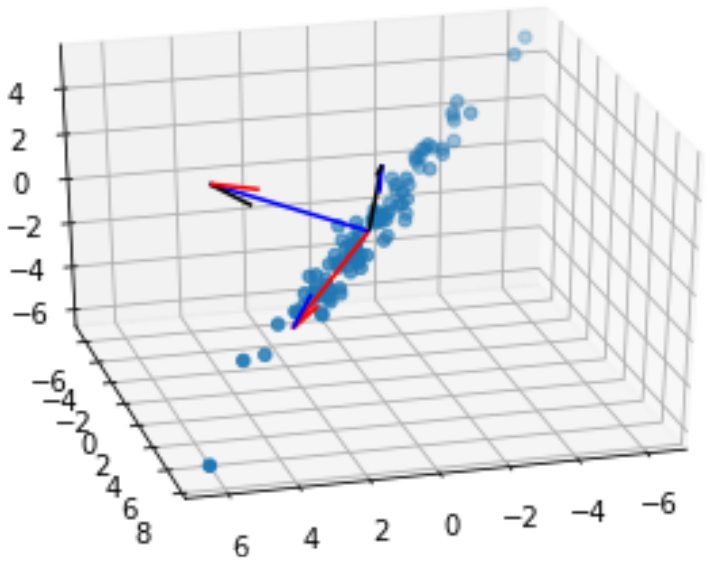
- provides little insight
- usually an arbitrary choice of measurements



... X's own (*eigen*) coordinate system

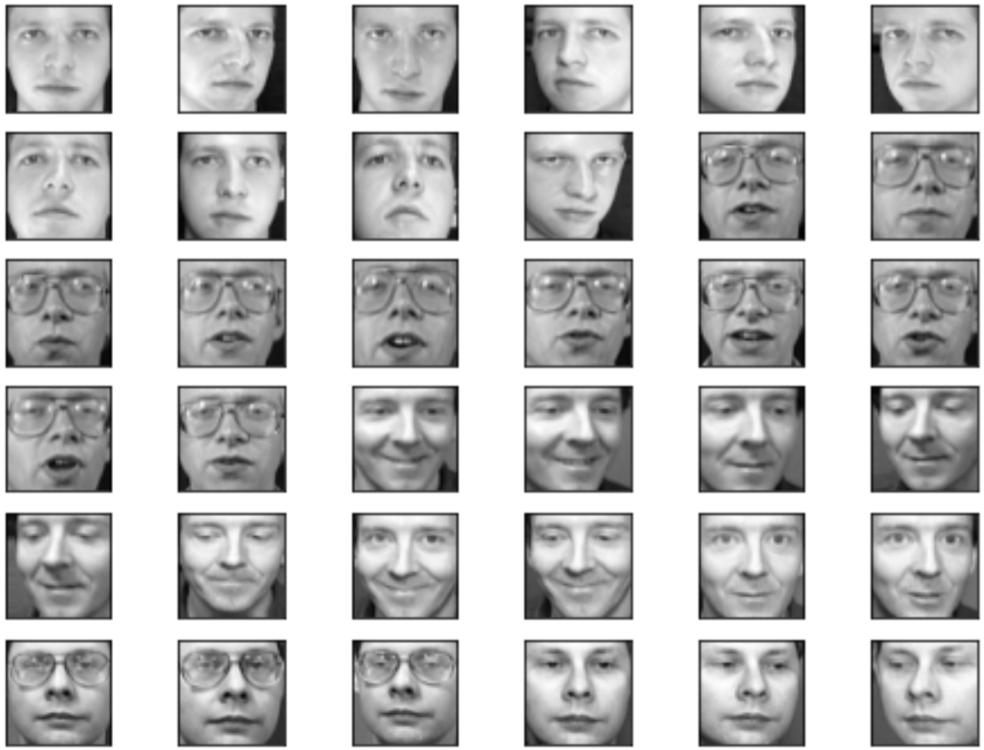
$X^T X$'s eigenvectors capture the variance in rows

- each direction $\Leftrightarrow$ eigenvalue capturing its importance
- all of $X^T X$ eigenvalues ≥ 0
- higher eigenvalues: higher variance among rows



Demo: faces

Small database of faces



Principal Components Analysis

Not unlike our intuition