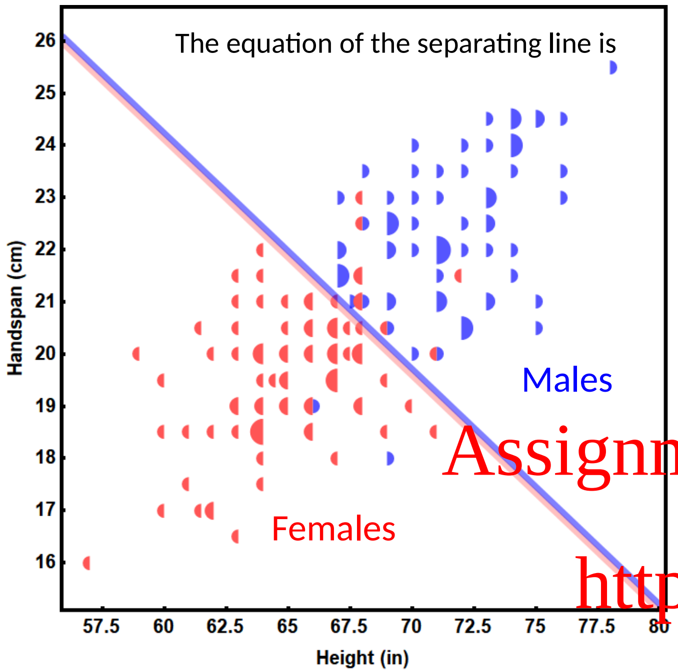
Topic 4: Linear Classifiers

Linear Classifier – Basic Idea



Based on the equation of the separating line, a classifier to predict gender given height and handspan can be programmed as

$$55.55 - 0.49\,Height - 1.08\,Handspan \;\underset{Female}{\overset{Male}{\gtrless}}\; 0$$

These parameters need to be determined from the given training set

General form of a linear classifier

With math swag

Class 1

$$w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d \quad ¿¿ \quad 0 \qquad \text{OR} \qquad \sum_{j=0}^{d} w_j x_j \quad ¿¿ \quad 0 \;\left(x_0 \overset{\text{def}}{=} 1\right)$$

Class 2

Linear Classifier Design Given a training set , find an optimal set of weights .

Class 1

$$\mathbf{x}\,\mathbf{w} \quad ¿¿ \quad 0$$

Class 2

Dot product

Linear Classifier – Perceptron

Initialize classifier

Pick a random sample and its associated class label

Class label (+1 or -1)

Augmented version of

Is the sample classified correctly?

For : Is

For : Is

Yes

No

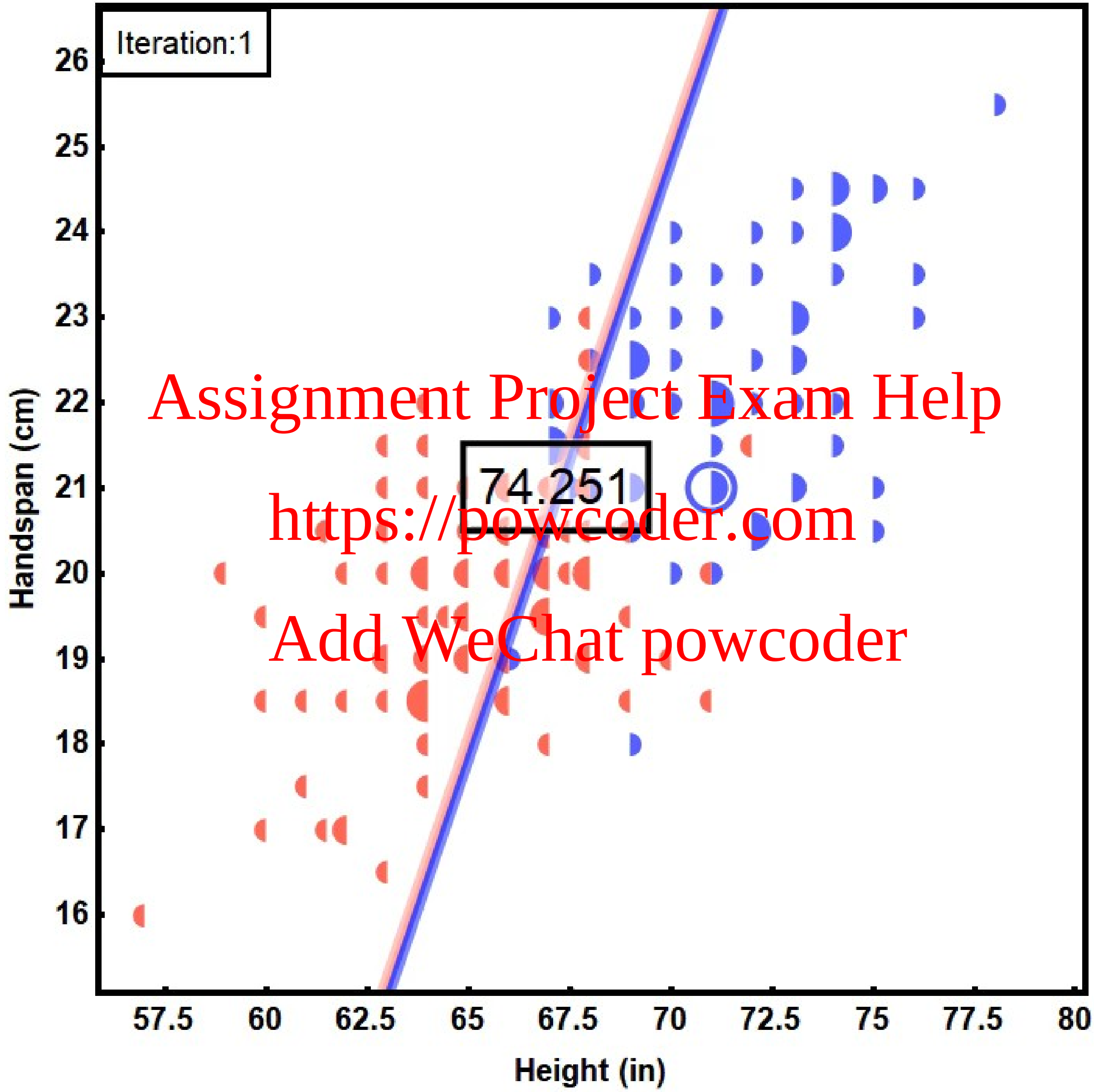
Modify classifier:

For :

For :

- Converges if samples are linearly separable.
- Keep track of current best classifier at each iteration. If tired of iterating, stop and produce the best classifier found.
- Simple "beginners" classifier.
- It might be necessary to "update gently" as in
  or

Iterations of the perceptron classifier for height-handspan data



Why does the update rule work?

Assume that the algorithm encounters a sample with class label +1 that is not correctly classified by the current classifier.

That is, should have been , but is in fact (for example).

When the weight vector is updated, a new weight vector is computed as . If the algorithm is any good, we naturally expect that this change should favor the correct classification of the same sample. Let us check if this is indeed true by feeding the same sample through the updated classifier. In order to do this we need to examine the dot product and check its sign. Let us do this.

Linear Classifier – Mean-square error minimizer

Given a training set, find set of weights such that
for positive points the value of is forced to 1
for negative points the value of is forced to -1

$$
\begin{bmatrix}
x_{10}¿1 & x_{11} & x_{12} & x_{13} & \cdots \\
x_{20}¿1 & x_{21} & x_{22} & x_{23} & \cdots \\
x_{30}¿1 & x_{31} & x_{32} & & \cdots \\
 & . & . & . & . \\
x_{N0}¿1 & x_{N1} & x_{N2} & x_{N3} & \cdots
\end{bmatrix}
\begin{bmatrix}
w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_d
\end{bmatrix}
=
\begin{bmatrix}
t_1 \\ t_2 \\ \vdots \\ t_N
\end{bmatrix}
$$

$\{N \times (d+1)\}$ $\{(d+1) \times 1\}$ $\{N \times 1\}$

Matrix

Low dim vector

is a column vector

High dim vector

is a column vector

is solved using the pseudoinverse as

Designing a linear classifier = Solving a matrix equation

$$X a \, w = T$$

Tall and thin matrix (Given)

Low dimensional column vector (Unknown)

High dimensional column vector (Given)

In order to understand this equation in a familiar framework, we define and giving the alternate form

$$w A = b$$

Low dimensional row vector (Unknown)

Short and fat matrix (Given)

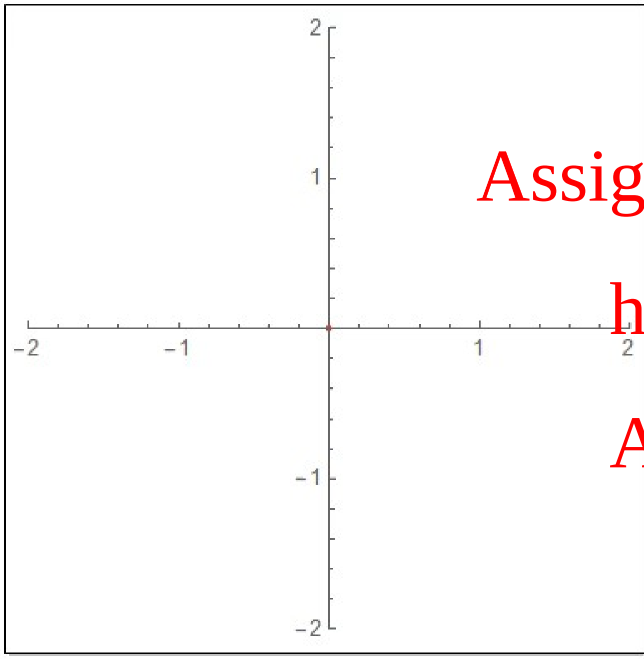
High dimensional row vector (Given)

Topic 4 TOC

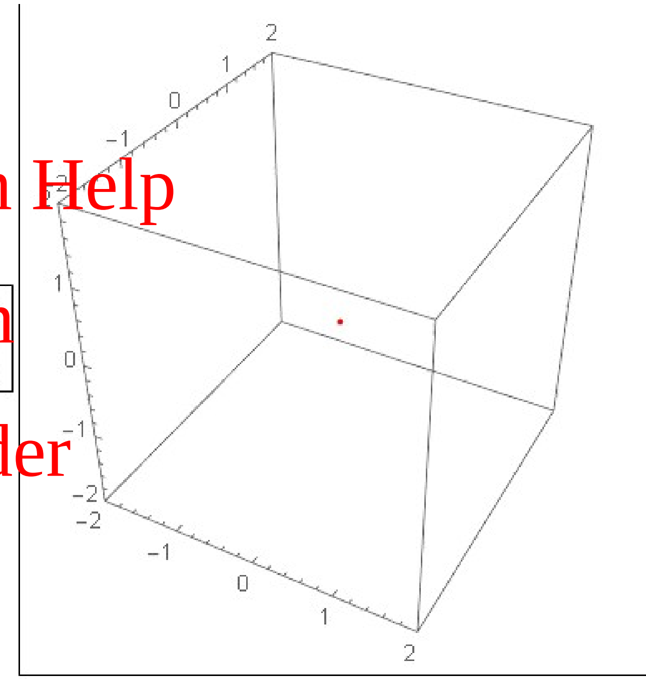
Press ESC to go to Index

Matrix A bridging 2D and 3D space

Bookmarks
1. Begin drawing
2. Plane curve in space
3. Show plane



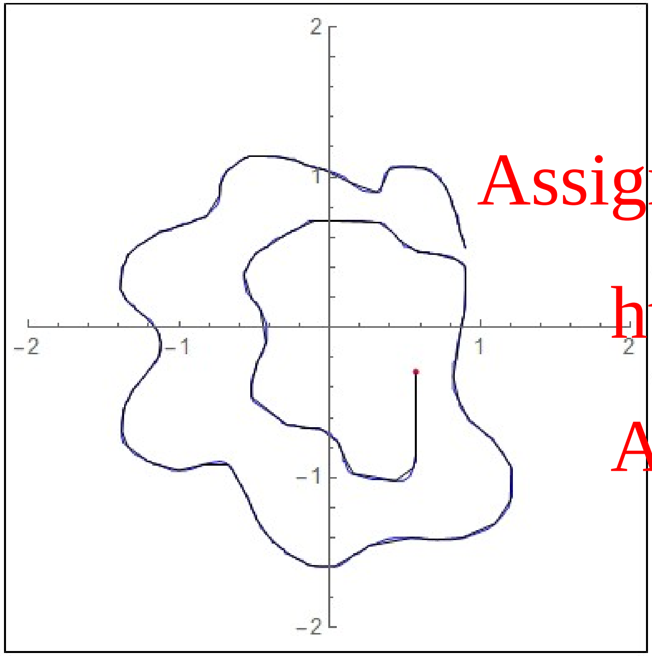
$$A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$



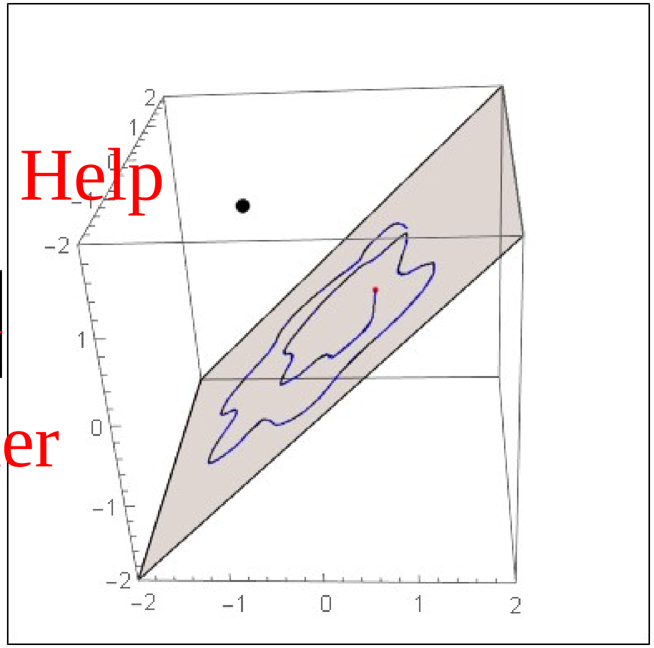
Note that the matrix defines the plane

Inverse image may not exist

Bookmarks
1. Start moving T
2. T on plane
3. Edge view
4. Start moving T
5. Perpendicular

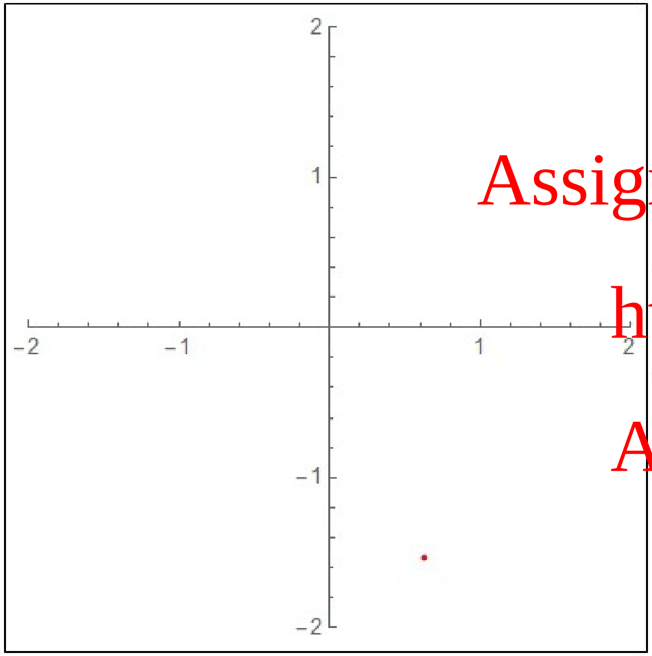


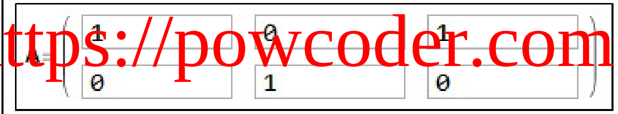
$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

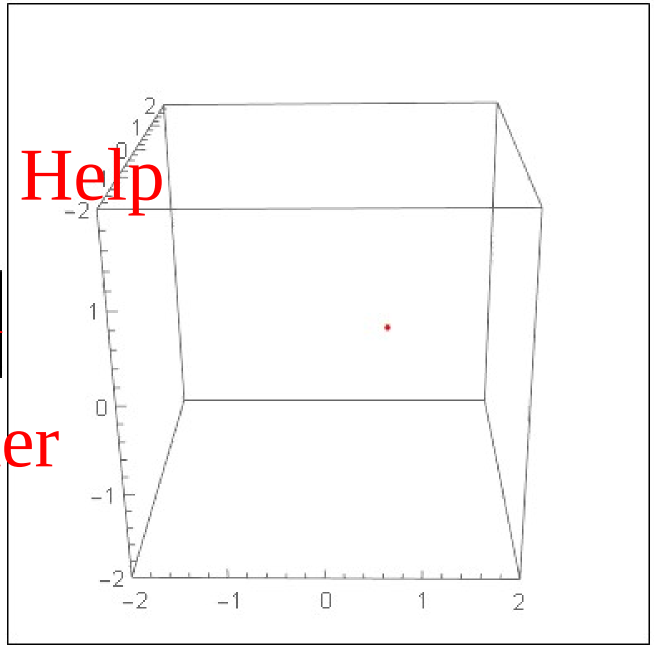


Minimizing the sum-of-squares

Bookmarks
1. Show plane
2. Show b
3. Show perpendicular
4. Show line & dist
5. Move point
6. Min
7. Min







Topic 4 TOC

Press ESC to go to Index

Replacing an insolvable with a solvable

$$w\,A = b$$

Not in the range space of

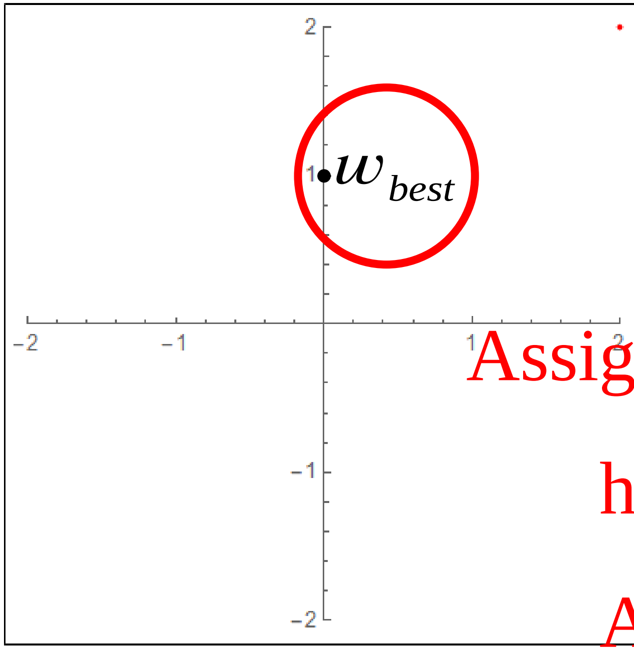
We abandon the goal of solving this equation

$$w\,A = bp$$

In the range space of

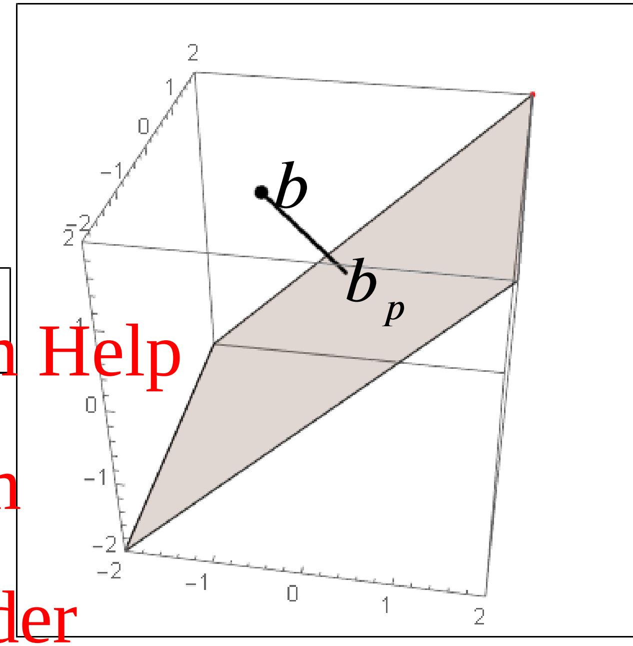
We will choose to solve this closely related equation

What the pseudoinverse does



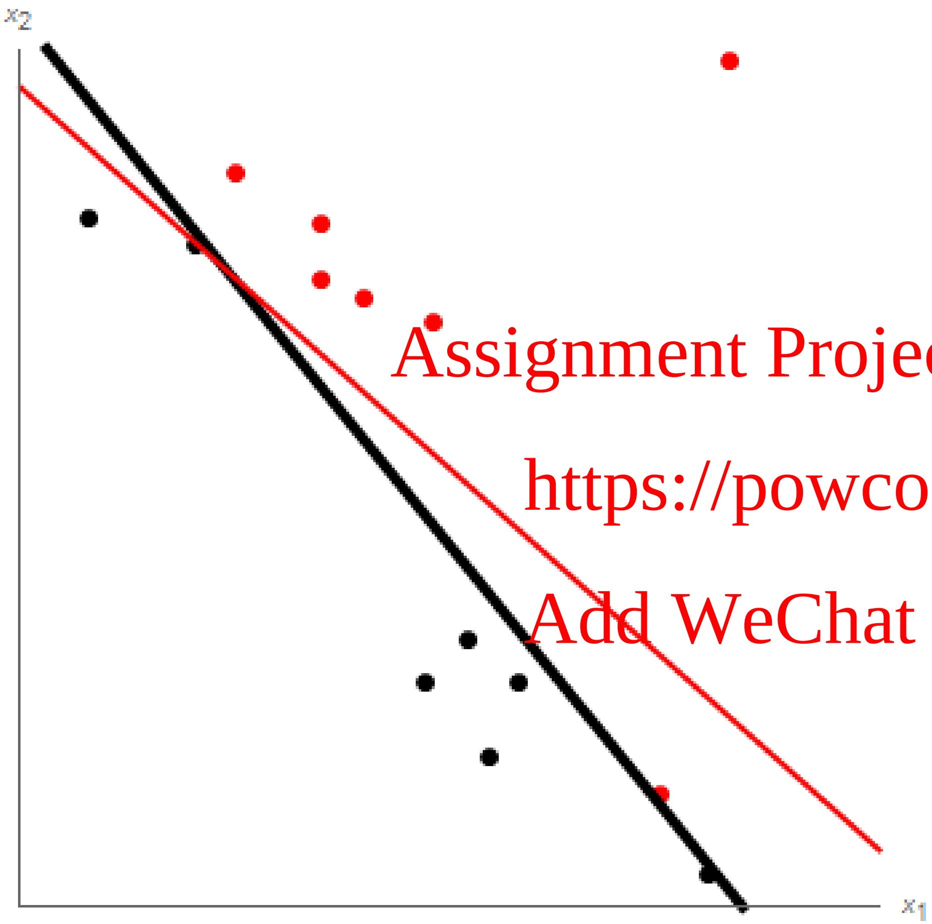
$$A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$A^{+} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1 \\ 1/2 & 0 \end{pmatrix}$$



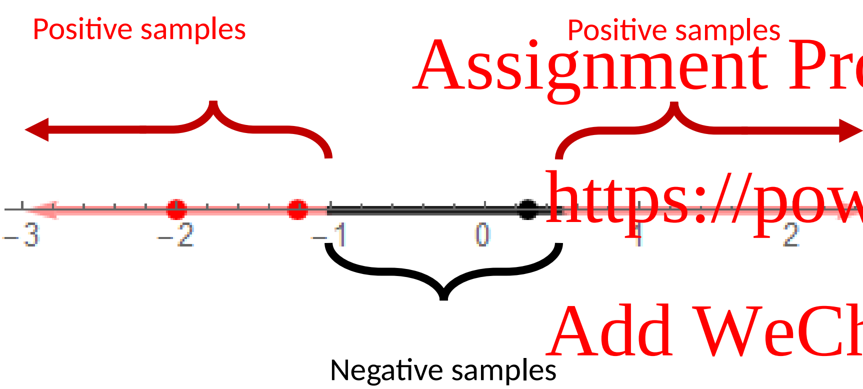
What is the pseudoinverse of a given matrix ?
It is that matrix which maps any given point in the output space to the point in the input space whose image is as close as possible to the given point .

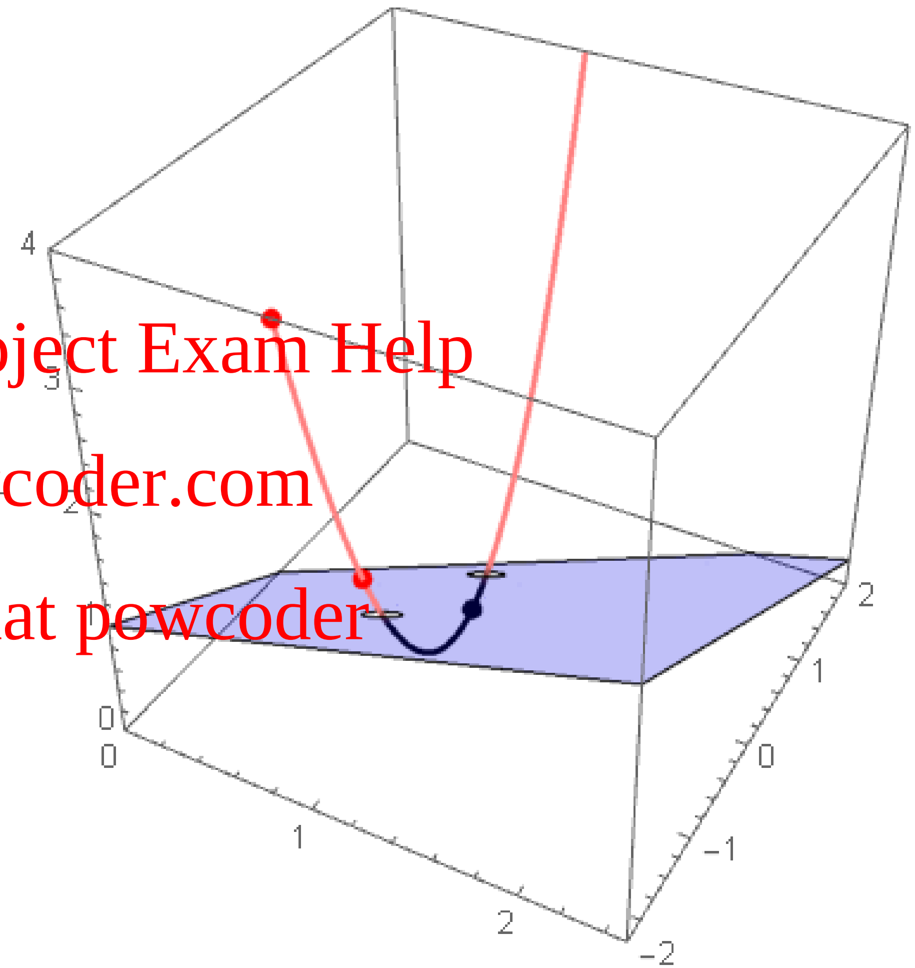
Linear Classifiers – Perceptron versus Pseudoinverse



- The perceptron classifier is unaffected by perturbations of points if they do not cross the decision boundary.
- The pseudoinverse classifier is affected by perturbations of points.

Linear Classifiers – Using nonlinear combinations of features



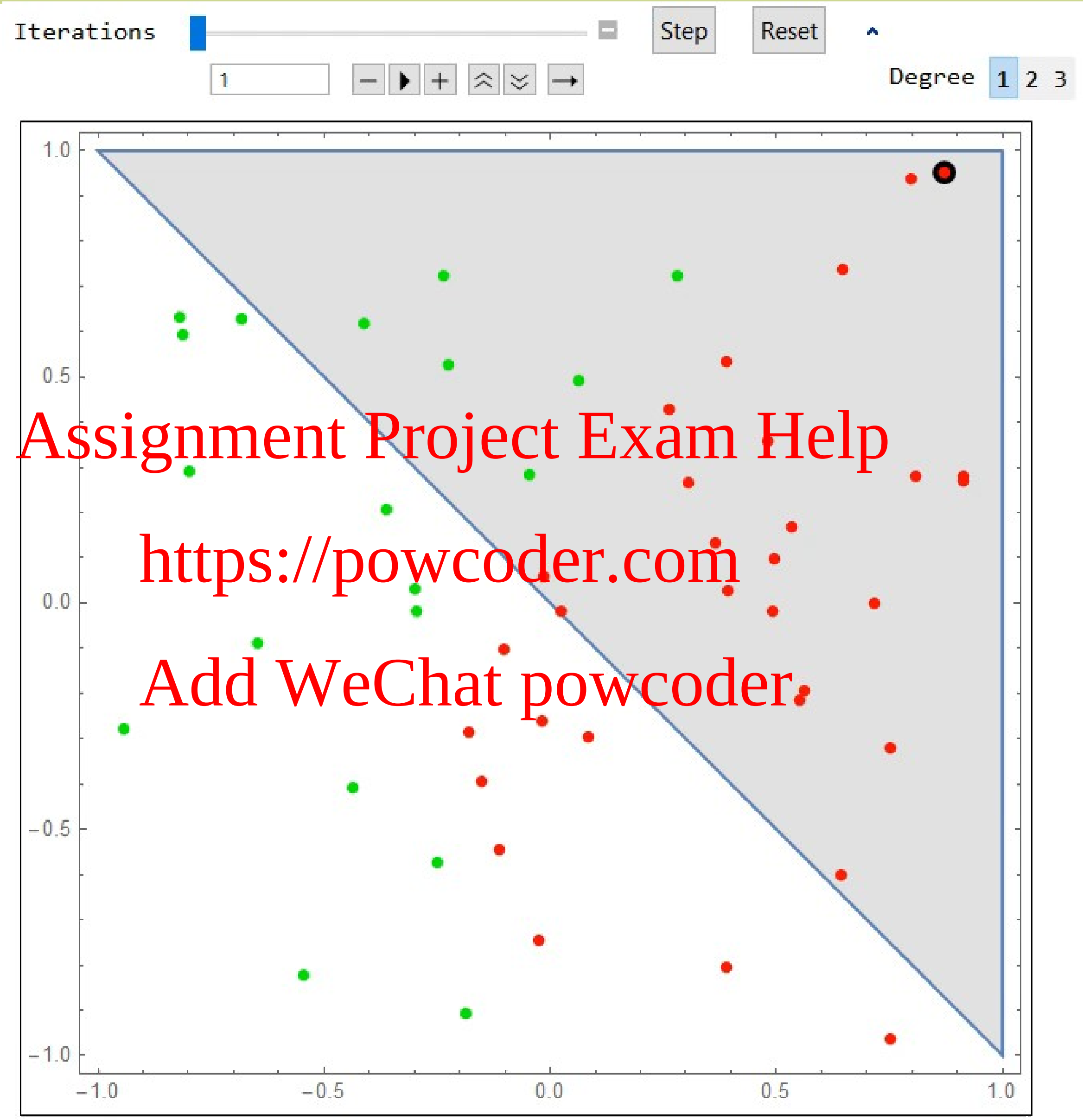


The above samples on the real line are not linearly separable. However, nonlinearly transformed versions of the very same samples are linearly separable.

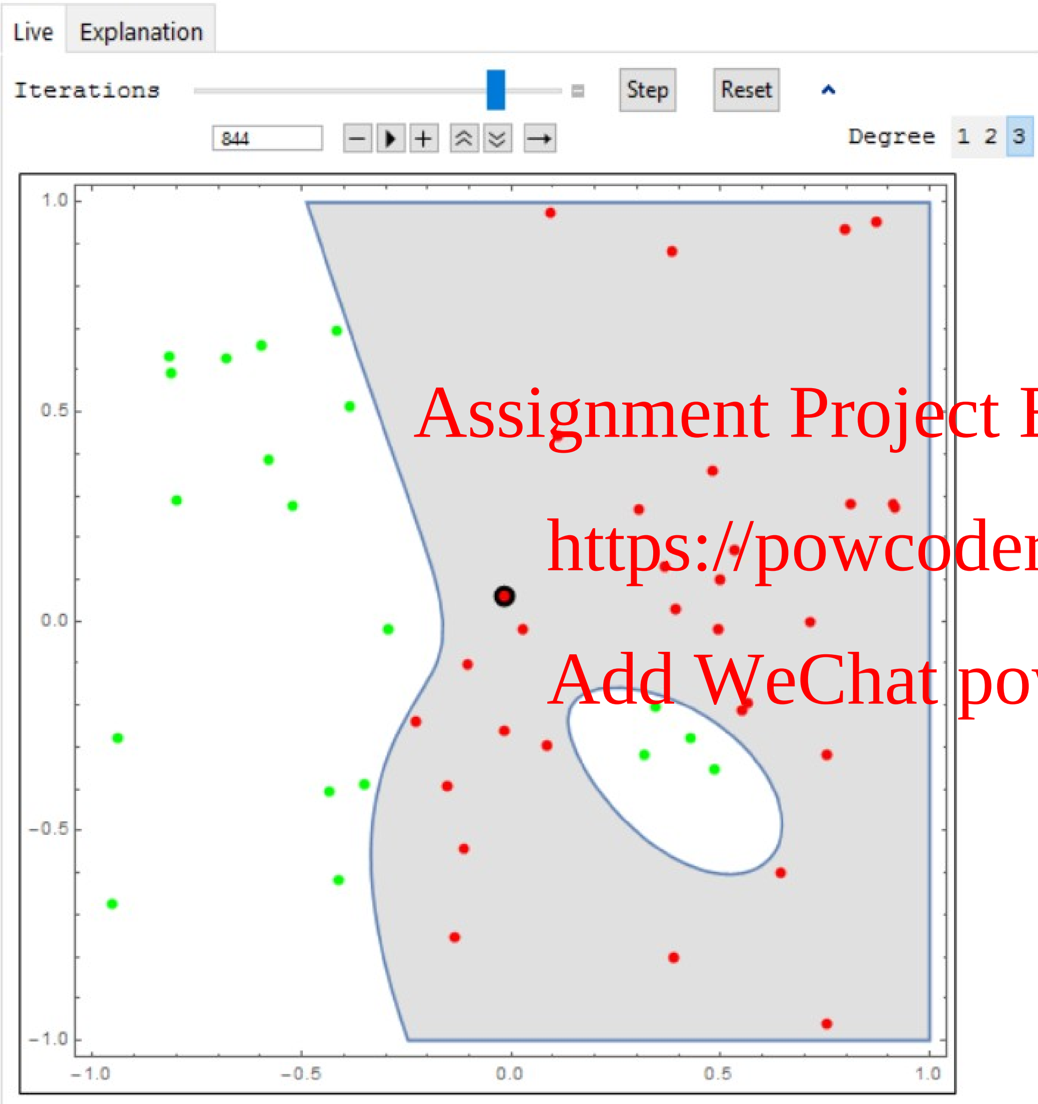
The nonlinear transformation used is

See: NonLinear_Classifier.cdf

Linear Classifier – Adding quadratic terms



Linear Classifier – Adding cubic terms



The 2D samples shown here are not linearly separable. However, nonlinearly transformed versions of the very same samples are linearly separable.

The nonlinear transformation used is

See: Linear_Classifier.cdf. Note: You can alter the training data by dragging the points. You can add nonlinear functions of features by setting the degree.

How to use nonlinear combinations of features

Variables used to build a simple linear classifier

x_0	x_1	x_2	x_3
1	...	...	...
1	...	...	...
1	...	...	...
1	...	...	...
1	...	...	...

Variables used to build an extended linear classifier with additional terms upto degree 2

	Linear terms			Quadratic terms					
x_0	x_1	x_2	x_3	x_1^2	$x_1\,x_2$	x_2^2	$x_1\,x_3$	$x_2\,x_3$	x_3^2
1	...	...	...	...	...	...	...	...	...
1	...	...	...	...	...	...	...	...	...
1	...	...	...	...	...	...	...	...	...
1	...	...	...	...	...	...	...	...	...
1	...	...	...	...	...	...	...	...	...

Variables used to build an extended linear classifier with additional terms upto degree 3

	Linear terms			Quadratic terms						Cubic terms									
x_0	x_1	x_2	x_3	x_1^2	$x_1\,x_2$	x_2^2	$x_1\,x_3$	$x_2\,x_3$	x_3^2	x_1^3	$x_1^2\,x_2$	$x_1\,x_2^2$	x_2^3	$x_1^2\,x_3$	$x_1\,x_2\,x_3$	$x_2^2\,x_3$	$x_1\,x_3^2$	$x_2\,x_3^2$	x_3^3
1	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Numeric Insight, Inc
Practical and exceptional number crunching and scientific programming

Linear Classifiers – For multiple classes (Kesler's construction)

Finding a 3 class classifier:

1 -1 -1

Example

$$\begin{bmatrix} x_{10} & x_{11} & x_{12} & x_{13} & \cdots \\ x_{20} & x_{21} & x_{22} & x_{23} & \cdots \\ x_{30} & x_{31} & x_{32} & x_{33} & \cdots \\ . & . & . & . & \\ x_{N0} & x_{N1} & x_{N2} & x_{N3} & \cdots \end{bmatrix}_{\{N\times(d+1)\}} \begin{bmatrix} w_{01} & w_{02} & w_{03} \\ w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ . & . & . \\ w_{d1} & w_{d2} & w_{d3} \end{bmatrix}_{\{(d+1)\times 3\}} = \begin{bmatrix} \boxed{t_{11} \; t_{12} \; t_{13}} \\ t_{21} \; t_{22} \; t_{23} \\ t_{31} \; t_{32} \; t_{33} \\ . \quad . \quad . \\ t_{N1} \; t_{N2} \; t_{N3} \end{bmatrix}_{\{N\times 3\}}$$

is again solved using the pseudoinverse as

To classify a given input, form and assign class based on the position of the maximum.

Real world example: Machine Failure

Temperature	x-acc	y-acc	z-acc	Pressure	Load	Flow			Nitrogen			Frequency			Failure Alert	Diagnostic Code
149	118	136	113	113	105	-1	-1	1	-1	-1	1	-1	-1	1	1	4
171	218	83	265	222	129	-1	1	-1	-1	1	-1	-1	1	-1	-1	0
295	267	177	160	317	216	-1	1	-1	1	-1	-1	-1	1	-1	-1	2
102	155	147	266	197	119	-1	1	-1	-1	1	-1	1	-1	-1	1	3
129	143	169	134	131	101	-1	-1	1	-1	-1	1	-1	-1	1	1	4
197	201	257	185	165	218	-1	1	-1	1	-1	-1	-1	1	-1	-1	0
143	109	185	183	180	101	-1	1	-1	-1	1	-1	1	-1	-1	1	3
258	197	262	218	255	253	-1	1	-1	1	-1	-1	-1	1	-1	-1	2
194	151	202	165	215	20[illegible]	-1	1	-1	-1	1	-1	1	-1	-1	1	5
117	103	133	60	80	88	-1	-1	1	-1	-1	1	-1	-1	1	1	4
125	157	192	167	142	109	-1	-1	1	-1	-1	1	-1	-1	1	1	4
233	94	259	148	172	288	-1	1	-1	-1	1	-1	1	-1	-1	1	5
138	126	155	116	104	84	-1	-1	1	-1	-1	1	-1	-1	1	1	4
122	99	106	137	178	148	-1	-1	1	-1	-1	1	-1	-1	1	1	4
238	149	176	269	236	223	-1	1	-1	-1	1	-1	-1	1	-1	-1	0
139	115	108	82	178	104	-1	-1	1	-1	-1	1	-1	-1	1	1	4
154	167	189	195	142	134	-1	1	-1	-1	1	-1	1	-1	-1	-1	1
209	176	170	253	212	226	-1	1	-1	1	-1	-1	1	-1	-1	-1	0
191	238	197	256	160	221	-1	1	-1	1	-1	-1	-1	1	-1	-1	2
138	165	153	258	195	220	-1	1	-1	-1	1	-1	1	-1	-1	1	3
152	156	164	223	191	241	-1	1	-1	-1	1	-1	1	-1	-1	-1	1
156	202	146	149	255	138	-1	1	-1	-1	1	-1	1	-1	-1	-1	1
226	215	179	210	218	225	-1	1	-1	1	-1	-1	-1	1	-1	-1	2
233	230	237	236	228	276	-1	1	-1	1	-1	-1	-1	1	-1	-1	2
175	190	179	224	192	195	-1	1	-1	1	-1	-1	-1	1	-1	-1	0
110	168	105	180	134	170	-1	-1	1	-1	-1	1	-1	-1	1	1	4

Use these targets for building a binary classifier to predict failure

Use these targets for building a multiclass classifier to predict a diagnostic code

Full dataset has 6600 items

Kesler's construction: Nominals to ordinals

Training and Testing Subsets

= Total sample size

= Training set size

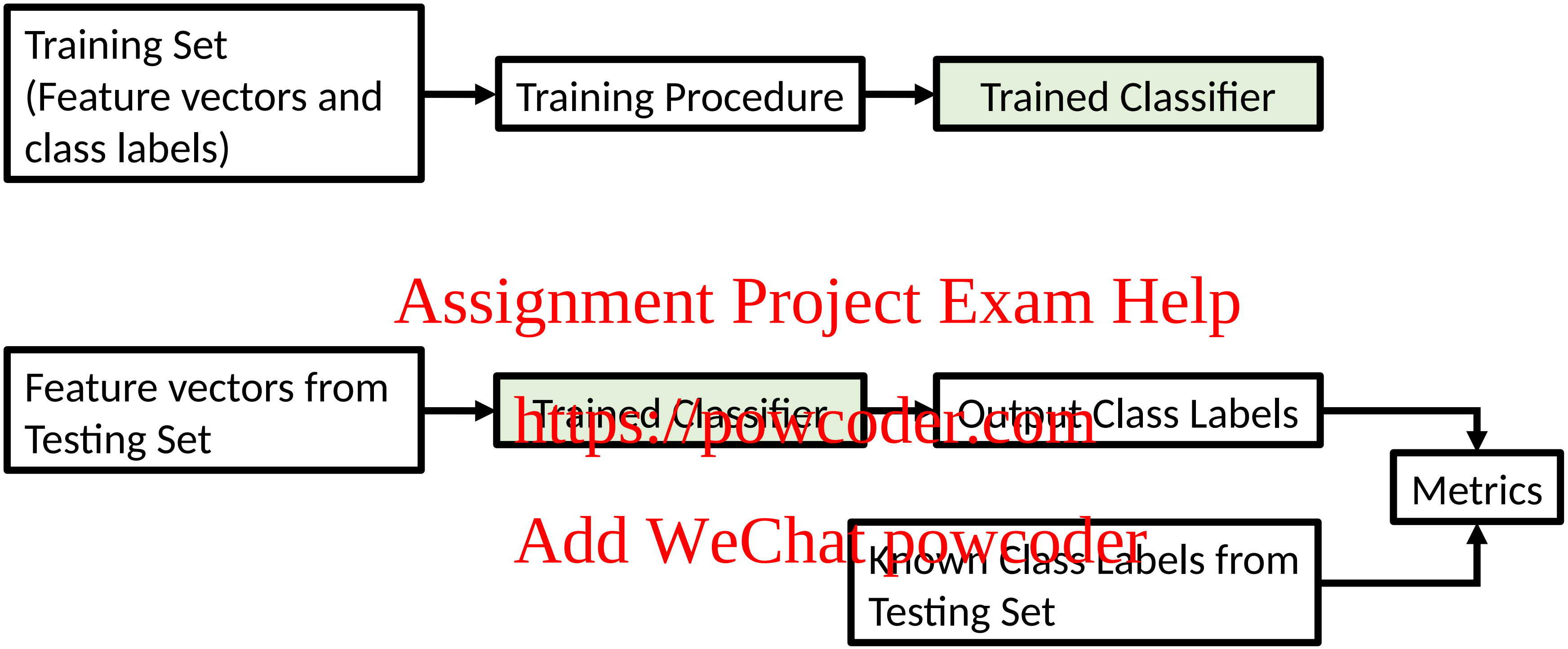
= Testing set size

Question: Given a set of 1000 items, how do I generate a random subset of 750 items?

Answer:

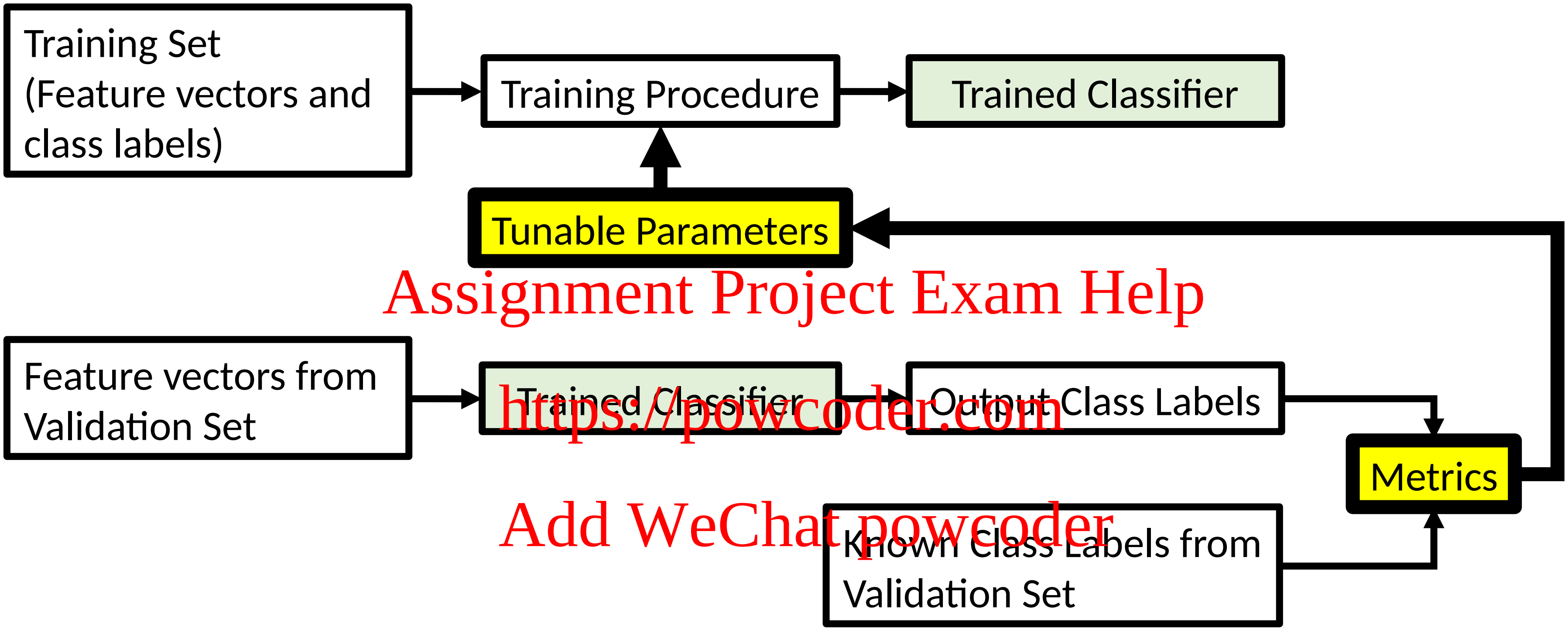

```
permuted_indices=np.random.permutation(1000)
training_indices=permuted_indices[:750]
testing_indices=permuted_indices[750:]
```

Training and Testing Subset



Answers the question “How good is my classifier?”

Validation Subset



Answers the question “How can I optimize my classifier?”
Finds use in setting histogram bin widths, for example.

Linear Classifier to answer "*Will the machine fail?*"

Given a training set, find set of weights such that
for positive points the value of is forced to 1
for negative points the value of is forced to -1

$$\begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & \cdots \\ 1 & x_{21} & x_{22} & x_{23} & \cdots \\ 1 & x_{31} & x_{32} & x_{33} & \cdots \\ . & . & . & . & \\ 1 & x_{N1} & x_{N2} & x_{N3} & \cdots \end{bmatrix} \begin{bmatrix} -0.79875 \\ -0.72152 \\ -0.64092 \\ 0.84074 \\ 0.22689 \\ -0.84919 \\ -0.63674 \\ -0.88744 \\ 0.27741 \\ 0.24347 \\ -0.82087 \\ -0.52954 \\ 0.69901 \\ -0.30353 \\ -0.10195 \\ -0.30998 \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ . \\ t_N \end{bmatrix}$$

is solved using the pseudoinverse as

Topic 4 TOC | Press ESC to go to Index

Step-by-step: Linear Classifier Design for 2 classes

Classifier Design

1. Collect and assemble matrix of feature vectors into matrix . Assemble targets into , a column vector of size containing class-labels -1 or 1. Note: If any of the original features are nominal, they must be converted to the numerical values using Kesler's construction.
2. Construct the augmented feature vector set by prepending a column of 1s in front of the matrix . Note that has dimensions .
3. Find the linear classifier . Here is the pseudoinverse of and has dimensions . The pseudoinverse is standard in linear algebra software. The classifier is a column vector of size

Classifier Application

To apply the linear classifier to an input vector , simply compute the augmented feature vector (by appending the element 1.0 in front of the list of the components of) and classify it by computing .

Classifier Performance

Feature Vector → Classifier → Output Class Label

Correct Wrong

Feature Vector						Output Class Label	Ground Truth
x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	c_1	t_1
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	c_2	t_2
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}	c_3	t_3
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	x_{46}	c_4	t_4
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}	x_{56}	c_5	t_5
x_{61}	x_{62}	x_{63}	x_{64}	x_{65}	x_{66}	c_6	t_6
x_{71}	x_{72}	x_{73}	x_{74}	x_{75}	x_{76}	c_7	t_7

Classifier Performance (Binary)

Feature Vector	Output Class Label	Ground Truth
$x_{11}\ x_{12}\ x_{13}\ x_{14}\ x_{15}\ x_{16}$	+	+
$x_{21}\ x_{22}\ x_{23}\ x_{24}\ x_{25}\ x_{26}$	-	-
$x_{31}\ x_{32}\ x_{33}\ x_{34}\ x_{35}\ x_{36}$	+	-
$x_{41}\ x_{42}\ x_{43}\ x_{44}\ x_{45}\ x_{46}$	+	+
$x_{51}\ x_{52}\ x_{53}\ x_{54}\ x_{55}\ x_{56}$	-	+
$x_{61}\ x_{62}\ x_{63}\ x_{64}\ x_{65}\ x_{66}$	+	+
$x_{71}\ x_{72}\ x_{73}\ x_{74}\ x_{75}\ x_{76}$		

Output Class Label →

Ground Truth ↓	Positive	Negative
Positive	True Positive	False Negative
Negative	False Positive	True Negative

Accuracy (Binary)

Output Class Label →

Ground Truth ↓	Positive	Negative
Positive	True Positives	False Negatives
Negative	False Positives	True Negatives

Accuracy of a classifier:
An estimate of the probability of correct classification

$$\frac{TP+TN}{TP+FN+FP+TN}$$

Why is this metric not universally satisfactory?

Sensitivity (Binary)

Output Class Label →

Ground Truth ↑	Positive	Negative
Positive	True Positives	False Negatives
Negative	False Positives	True Negatives

Sensitivity of a classifier:
An estimate of the probability of detecting a pattern as positive given that it is positive.

$$\frac{TP}{TP+FN}$$

Build a Disease Detector and take it to Disease City.
What fraction of the inhabitants are declared diseased?

Specificity (Binary)

	Positive	Negative ← Output Class Label
Positive	True Positives	False Negatives
Negative ↑ Ground Truth	False Positives	True Negatives

Specificity of a classifier:
An estimate of the probability of detecting a pattern as negative given that it is negative.

$$\frac{TN}{FP + TN}$$

Take the same Disease Detector to Healthy City. What fraction of the inhabitants are declared healthy?

Positive Predictive Value (Binary)

	Positive	Negative ← Output Class Label
Positive	True Positives	False Negatives
Negative ↑ Ground Truth	False Positives	True Negatives

PPV of a classifier:
An estimate of the probability that a pattern detected as positive is in fact positive.

$$\frac{TP}{FP+TP}$$

Take the same Disease Detector to Real City. How believable is the device?

Confusion Matrix

Detected Class

Actual Class	0	1	2	3	4	5
0	25	28	73	38	68	0
1	0	0	57	74	0	68
2	28	0	0	66	97	61
3	46	0	29	64	53	41
4	70	88	39	0	30	86
5	0	40	83	0	64	58

Accuracy of detecting class 3:

Q: Is this any good?

A: Compare against accuracy of "random classifier"

Positive Predictive Value

PPV of detecting class 3:

Q: Is this any good?

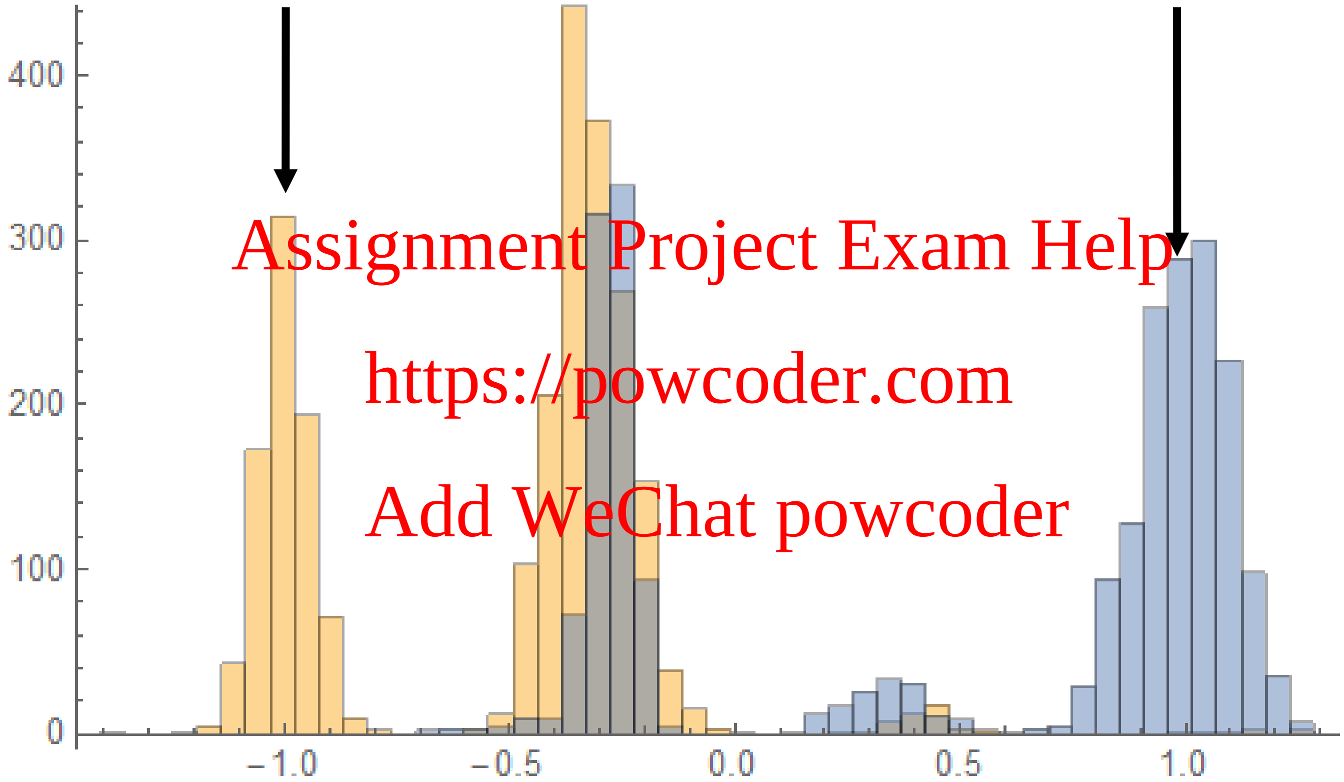
A: Compare against PPV of “random classifier”

Type	0	1	2	3	4	5
PPV	0.15	0.	0.	0.26	0.1	0.18

How good is this classifier?

Histogram of negative samples projected along

Histogram of positive samples projected along



Training Accuracy = 82%

Testing Accuracy = 81%

Tweaking the threshold

The default threshold is 0

Linear Classifier

Class 1

$$w_0+w_1x_1+w_2x_2+\ldots+w_dx_d \;¿¿\; 0 \quad \text{OR} \quad \sum_{j=0}^{d} w_j x_j \;¿¿\; 0 \;(x_0 \stackrel{\text{def}}{=} 1)$$

Class 2

Linear Classifier

Class 1

$$w_0+w_1x_1+w_2x_2+\ldots+w_dx_d \;¿¿\; \tau \quad \text{OR} \quad \sum_{j=0}^{d} w_j x_j \;¿¿\; \tau \;(x_0 \stackrel{\text{def}}{=} 1)$$

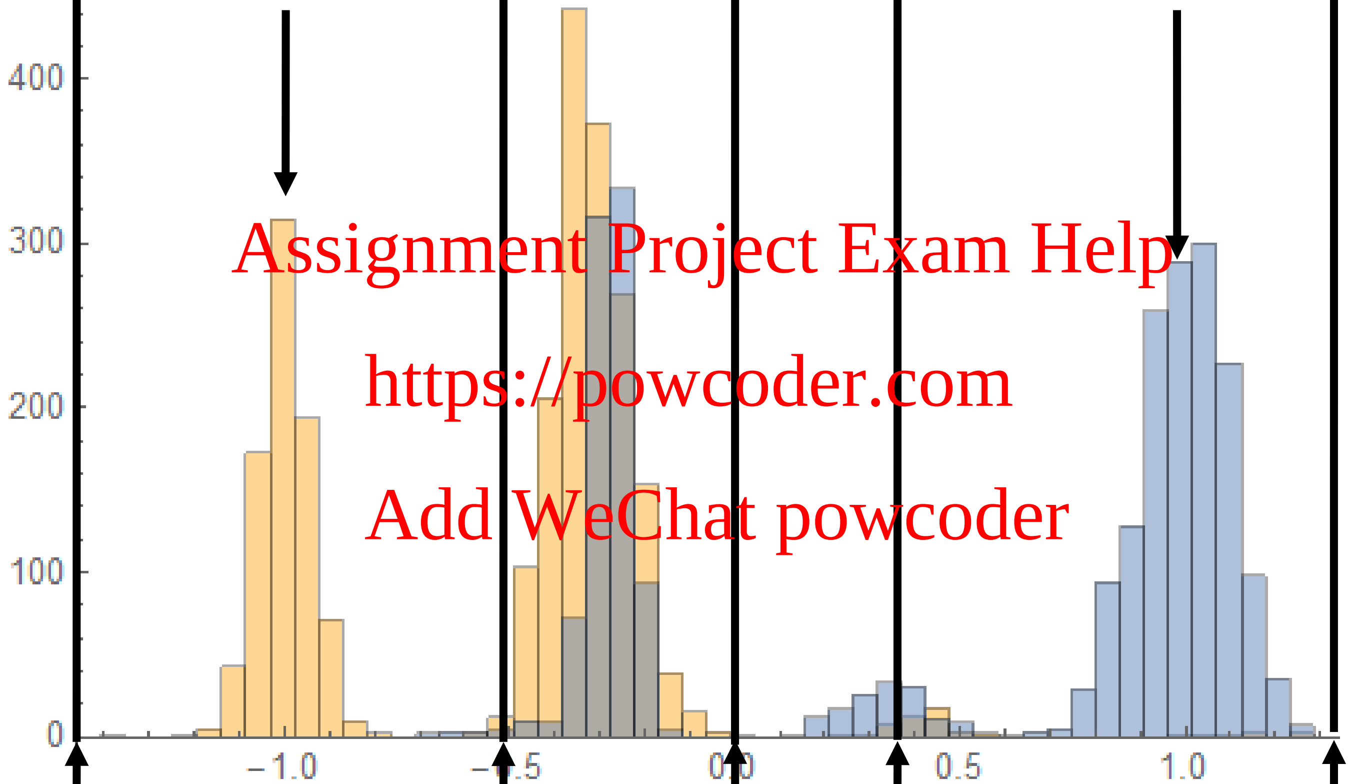
Class 2

The threshold can be tweaked to favor the sensitivity or specificity.

Tweaking the threshold

Histogram of negative samples projected along

Histogram of positive samples projected along



100% Sensitiv
0% Specifit

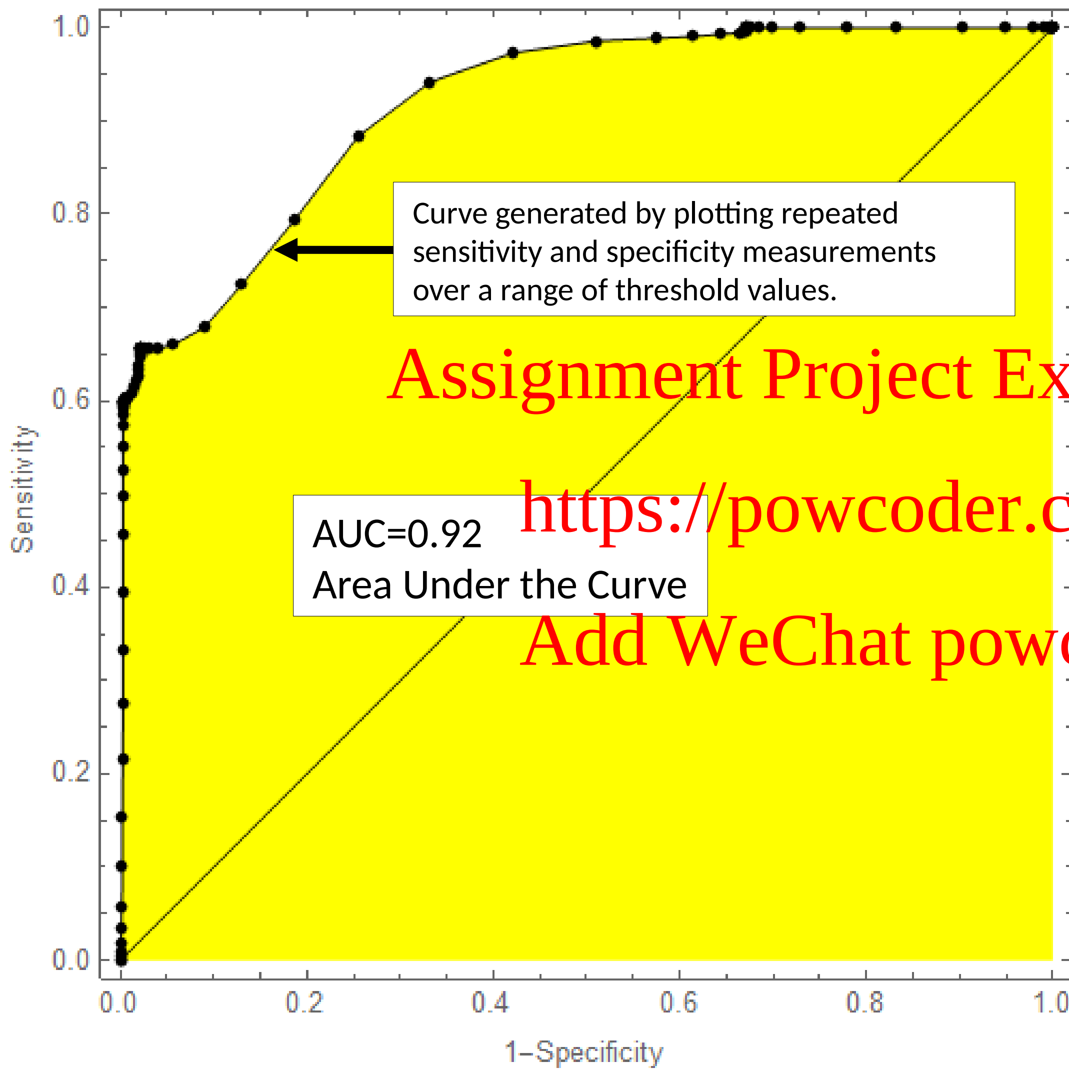
Adjust threshold to desired sensitivity /

Adjust threshold to get desired sensitivity / specificity

)% Specifity
Sensitivity

ress ESC to TOC go to Index

Receiver Operating Curve (ROC)



AUC interpretation

AUC=1	for a perfect classifier
AUC=0.5	for a classifier performing no better than random decisions
AUC>0.8	Generally considered to be a satisfactory classifier

Multiclass Linear Classifier - Recap

Finding a classifier for classes:

1	-1	... -1

Example

$$
\begin{bmatrix}
x_{10} & x_{11} & x_{12} & x_{13} & \cdots \\
x_{20} & x_{21} & x_{22} & x_{23} & \cdots \\
x_{30} & x_{31} & x_{32} & x_{33} & \cdots \\
. & . & . & . \\
x_{N0} & x_{N1} & x_{N2} & x_{N3} & \cdots
\end{bmatrix}
\begin{bmatrix}
w_{01} & w_{02} & \cdots \\
w_{11} & w_{12} & \cdots \\
w_{21} & w_{22} & \cdots \\
w_{31} & w_{32} & \cdots \\
. & . & . \\
w_{d1} & w_{d2} & \cdots
\end{bmatrix}
=
\begin{bmatrix}
\boxed{t_{11} \quad t_{12} \quad \cdots} \\
t_{21} \quad t_{22} \quad \cdots \\
t_{31} \quad t_{32} \quad \cdots \\
. \quad . \quad . \\
t_{N1} \quad t_{N2} \quad \cdots
\end{bmatrix}
$$

$\{N\times(d+1)\}$ $\{N\times C\}$

is again solved using the

To classify a given input, form and assign class base

Nupur Dixit, you have been honored with the Eagle Eye Award for spotting errors on the previous slide. Congratulations and thank you for your help in improving this presentation.

CLICK TO DISMISS

Press ESC to go to Index

Step-by-step: Linear Classifier Design for C classes

Classifier Design

1. Collect and assemble matrix of feature vectors into matrix . Assemble targets into , a matrix of size containing Keslerized class-labels. Note: If any of the original features and/or class labels are nominal, they must be Keslerized as well.
2. Construct the augmented feature vector set by prepending a column of 1s in front of the matrix . Note that has dimensions .
3. Find the linear classifier . Here is the pseudoinverse of and has dimensions . The pseudoinverse is standard in linear algebra software. The classifier is a matrix of size

Classifier Application

To apply the linear classifier to an input vector , simply compute the augmented feature vector (by appending the element 1.0 in front of the list of the components of) and classify as the index of the maximal component of .

Linear Classifier to answer "*<u>How</u> will the machine fail?*"

Binary Classifier | 6-class Classifier

-0.79875	0.46764	0.86452	0.95398	-0.19715	0.39389	-0.34451
-0.72152	0.64304	0.00204	0.21375	-0.18968	0.63631	0.39103
-0.64092	-0.01114	0.72715	0.75406	0.20382	0.17809	-0.37686
0.84074	0.06236	-0.44749	-0.32788	-0.86882	-0.51815	-0.69778
0.22689	-0.57577	-0.93963	-0.20831	-0.8682	-0.08857	0.09765
-0.84919	0.0235[illegible]	-0.59679	-0.71866	0.56215	0.63277	-0.87498
-0.63674	-0.25439	-0.50861	-0.05918	-0.23052	0.75178	-0.88669
-0.88744	-0.18551	-0.95396	-0.41446	0.37999	0.91929	0.09951
0.27741	-0.19792	0.82271	0.78985	0.75868	0.11323	-0.24436
0.24347	-0.54545	0.32876	-0.48256	0.39447	-0.93019	0.67873
-0.82087	0.68672	-0.53742	-0.65024	0.29985	-0.17256	0.44519
-0.52954	-0.8868	-0.46615	0.04039	0.606	0.14894	-0.68804
0.69901	0.49147	0.80087	0.5031	-0.93141	0.89929	-0.69476
-0.30353	-0.52576	0.1406	-0.45299	-0.24506	-0.26347	0.12544
-0.10195	0.31402	0.70919	0.51812	0.35231	-0.89767	0.16193
-0.30998	-0.01845	-0.32724	0.2938	0.06672	-0.29876	0.60873

Linear Regression – Mean-square error minimizer, again

Given a training set consisting of a feature vector matrix and a real valued target vector , find a set of weights such that

These are real numbers, not class labels.

$$\begin{bmatrix} x_{10}¿1 & x_{11} & x_{12} & x_{13} & \cdots \\ x_{20}¿1 & x_{21} & x_{22} & x_{23} & \cdots \\ x_{30}¿1 & x_{31} & x_{32} & & \cdots \\ & . & . & . & . \\ x_{N0}¿1 & x_{N1} & x_{N2} & x_{N3} & \cdots \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}$$

Matrix

Low dim vector

High dim vector

is a column vector

is a column vector

$\{N \times (d+1)\}$ $\{(d+1) \times 1\}$ $\{N \times 1\}$

is again solved using the pseudoinverse as

Example: Estimation of house price using various features

Linear classifiers: Moving on to Logistic Regression

General form of a linear classifier

$$w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d \quad \overset{\text{Class 1}}{\underset{\text{Class 2}}{¿¿}} \quad 0$$

With math swag

OR

$$\sum_{j=0}^{d} w_j x_j \quad \overset{\text{Class 1}}{\underset{\text{Class 2}}{¿¿}} \quad 0 \left(x_0 \overset{\text{def}}{=} 1 \right)$$

Linear Classifier Design Given a training set , find an optimal set of weights .

$$\mathbf{xaw} \quad \overset{\text{Class 1}}{\underset{\text{Class 2}}{¿¿}} \quad 0$$

is a column vector

- We would like our classifier to give us class labels and posterior probabilities
- That seems impossible to arrange because the dot product potentially runs from to , whereas probabilities can only be in the range from 0 to 1
- The dot product is called the **linear discriminant**

Connecting the discriminant to the probability

Probability is only one measure of the chances of an event happening. Instead of measuring the **probability** (the ratio of number of occurrences of an event to the number of trials), you can measure the **odds** (the ratio of number of occurrences of an event to the number of non- occurrences), or the **logit** (the logarithm of the ratio of number of occurrences of an event to the number of non- occurrences). The advantage of using the logit is that it's range is the same as the range of the linear discriminant .

Probability

Incompatible with the linear discriminant

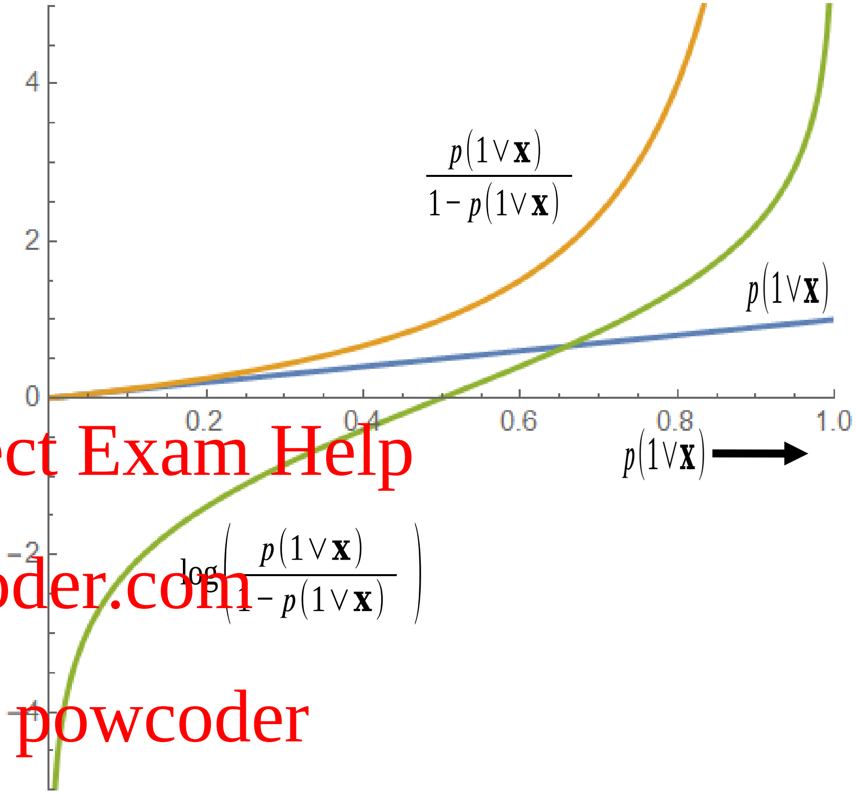
Odds

Incompatible with the linear discriminant

Logit

Compatible with the linear discriminant

Probability	goes from	0	to	1
Odds	goes from	0	to	$+\infty$
Logit	goes from	$-\infty$	to	$+\infty$



The definition of the logit as gives us a method of converting a value of probability to a logit . As examples, a probability of translates to a logit of . A probability of translates to a logit of .

In the reverse direction, a given value of logit can be converted to probability using the relation . As examples, a logit of translates to a probability of . A logit of translates to a probability of .

Ordinary linear regression: Try to find a weight vector that makes the linear discriminant evaluate to for positive examples and for negative examples. can be found using the pseudoinverse of the data matrix.

Logistic regression: Try to find a weight vector that makes the probability evaluate to for positive examples and for negative examples.

Equivalently, try to find a weight vector that makes the probability evaluate to for positive examples and the probability evaluate to (also) for negative examples.

Taking the target to be for positive examples and for negative examples (Note: this represents a change in convention), we can write the above compactly as: Try to find a weight vector that maximizes the following expression

Equivalently, after taking the logarithm, try to find a weight vector that maximizes the following expression

This maximization must happen for every pair of feature-vector and target in the training set. So, we can finally write logistic regression as the search for a weight vector that maximizes the following objective function

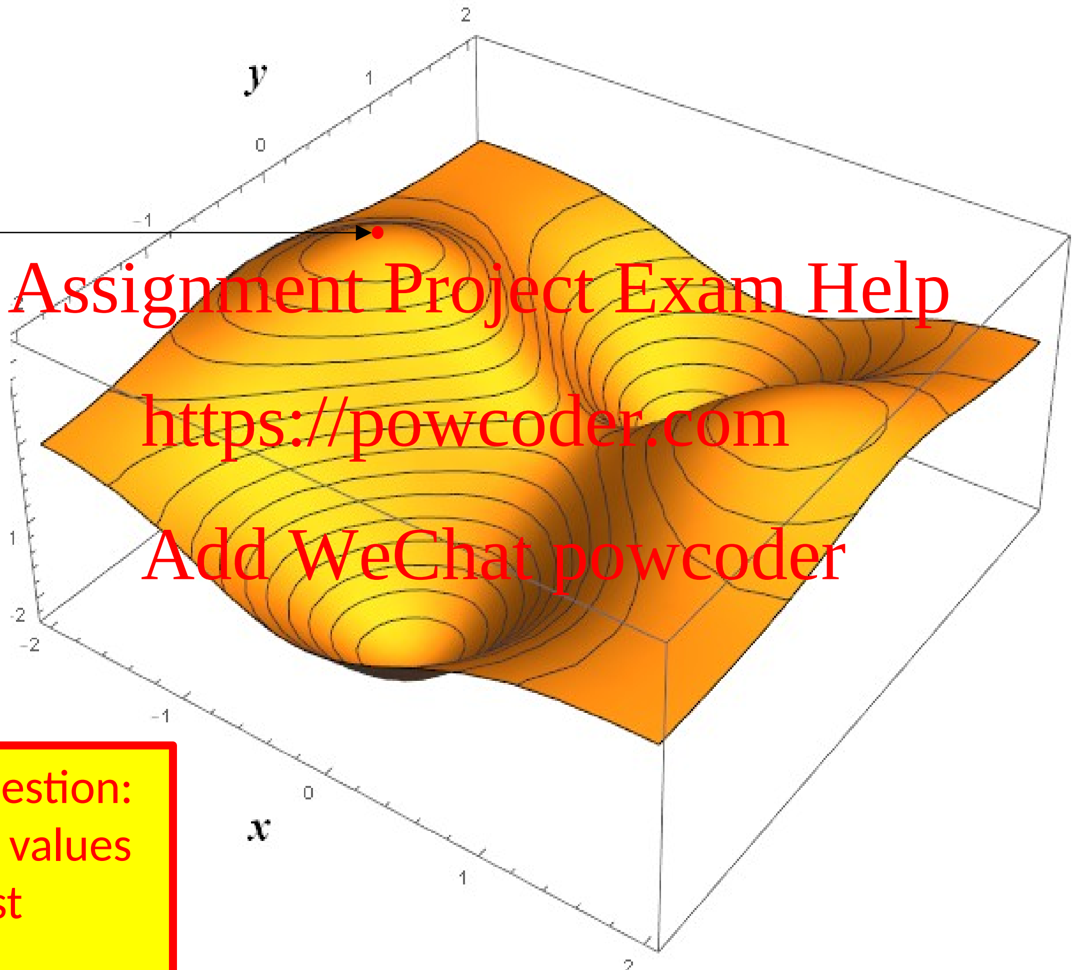
This is called the **logistic log likelihood**

The which makes the logistic log likelihood maximum cannot be found by analytical methods

The Optimization Question

A function of two variables and

The highest point on the landscape is shown here

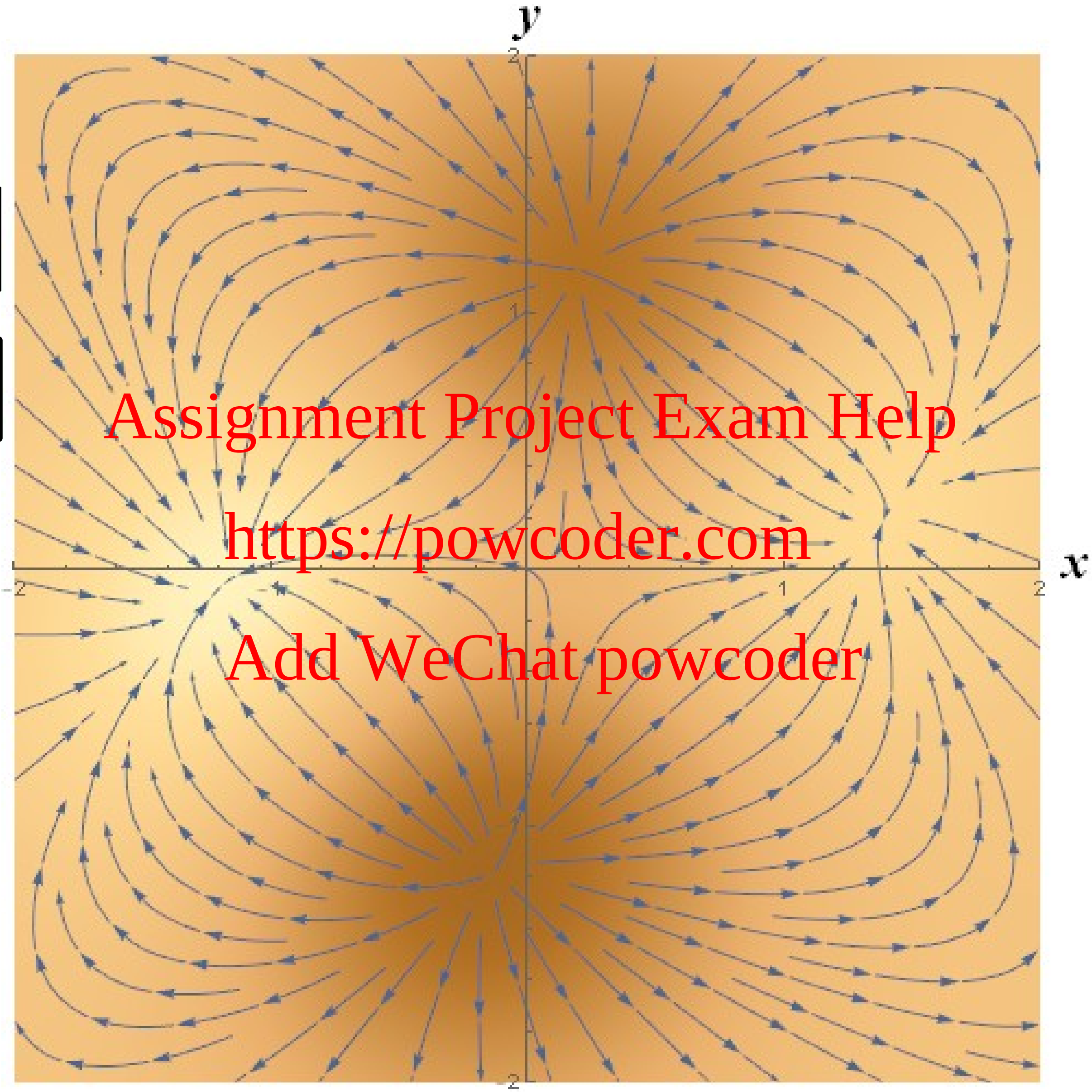


The Optimization Question: How can we find the values of and at the highest point?

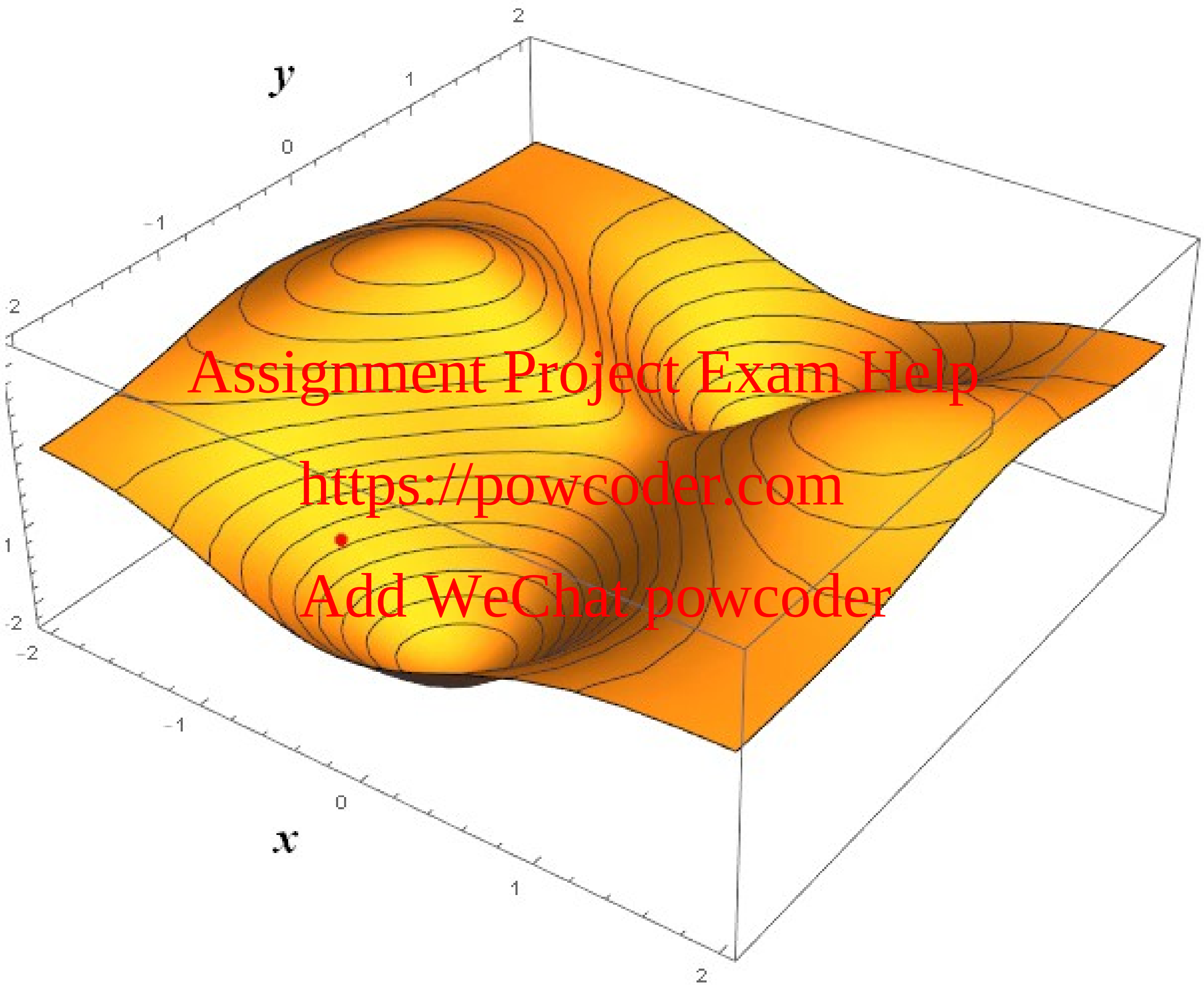
Contour and gradient plots

A contour plot of the function

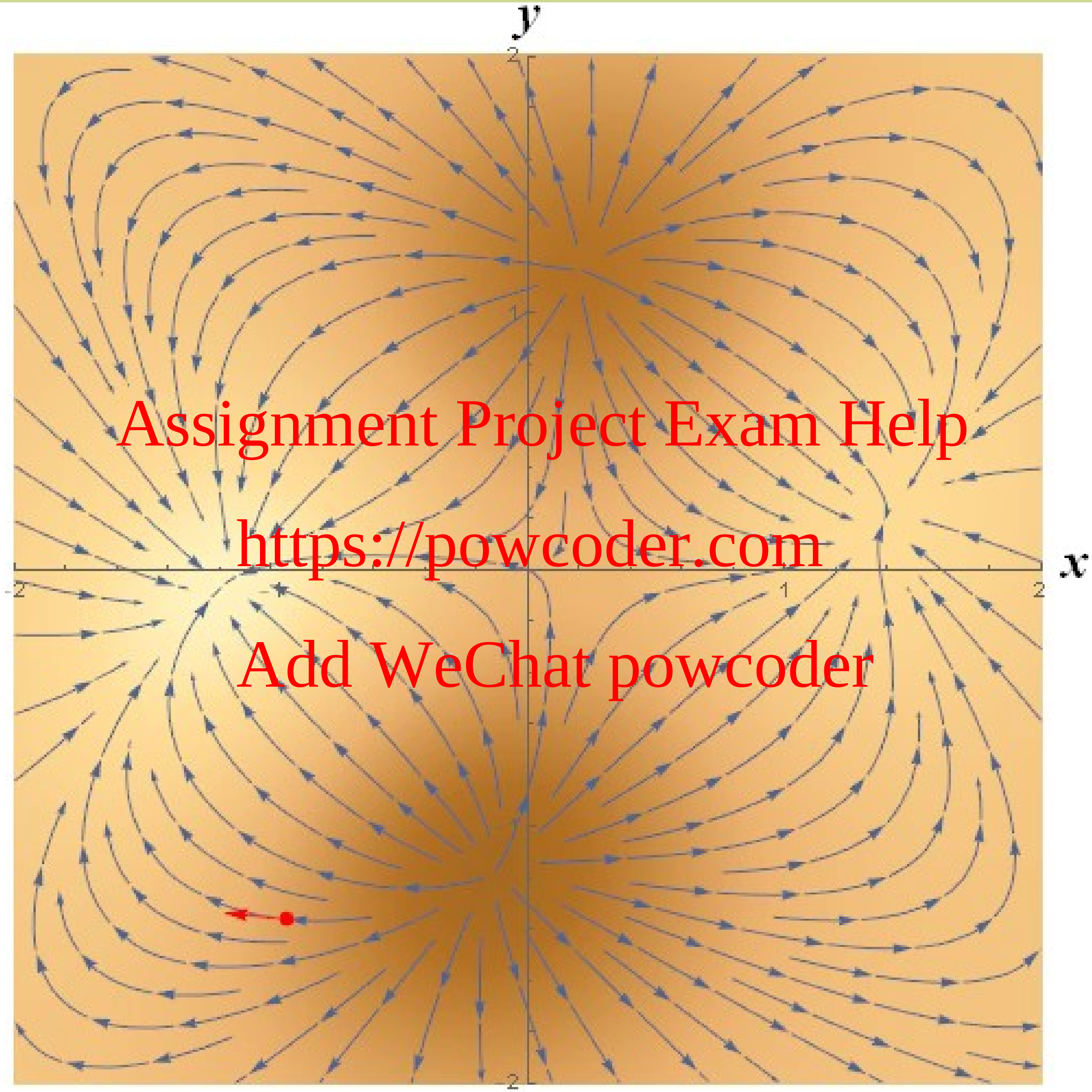
A gradient plot of the function



Steepest gradient ascent



Steepest gradient ascent (multiple maxima)



Maximization of the logistic log likelihood by steepest gradient ascent

Note that, for a given problem (i.e. are fixed), the logistic log likelihood is a function of the weight vector . Since the weight vector which makes the logistic log likelihood maximum cannot be found by analytical methods, the only recourse is to start with an initial weight vector and do computations to replace it with a weight vector which improves the logistic log likelihood. The final solution is obtained by repeated refinement of the weight vector till the logistic log likelihood cannot be further improved.

Let us assume that the problem is two-dimensional. That is   and . Extending this to the -dimensional case is straightforward. The logistic log likelihood, now a function of , and , can be written out as

The method of steepest gradient ascent (or one of its many accelerated variants) is generally used to find the weight vector that maximizes the logistic log likelihood. This method, based on calculus, calls for starting with initially assigned values of , and and tweaking them by adding , and . The tweaks must, of course, be carefully chosen such that the logistic log likelihood is improved. Calculus tells us that this can be arranged by making sure that the tweaks are small numbers proportional to the derivatives and . Fortunately, these derivatives can easily be derived, again thanks to calculus.

Maximization of the logistic log likelihood by steepest gradient ascent (contd.)

The derivatives and can be shown to be

The above set of derivatives regarded as a vector is called the gradient of the logistic log likelihood. For a given problem (i.e. are fixed), the gradient vector, denoted by , is also a function of the weight vector . Remember that the tweaks (the set of which can also be regarded as a vector) must be computed as small numbers proportional to the gradient vector .

Armed with the above, the method of steepest gradient ascent can be used to iteratively find a new weight vector given the old weight vector :

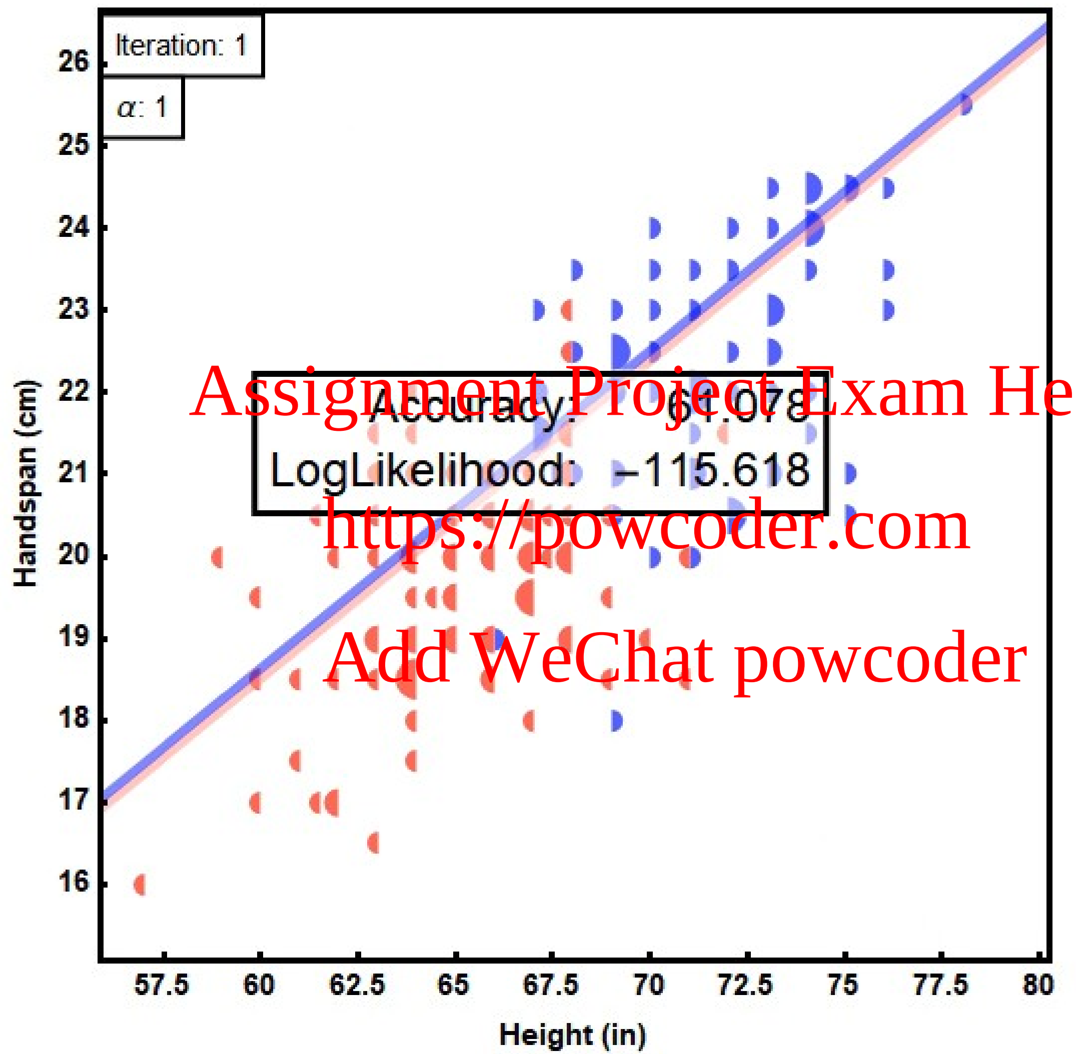
Compute the gradient vector:

Calculate the new weight vector: (is a small number e.g.)

Note: The logistic log likelihood function has only one maximum.

Nick Dargahi, you have been honored with the Eagle Eye Award for spotting errors on the previous slide. Congratulations and thank you for your help in improving this presentation.

CLICK TO DISMISS


Iteration: 1
α: 1
Accuracy: 61.078
LogLikelihood: −115.618
Handspan (cm)
Height (in)
16
17
18
19
20
21
22
23
24
25
26
57.5
60
62.5
65
67.5
70
72.5
75
77.5
80

Linear Classifiers in sklearn applied to MNIST data

```
from sklearn import linear_model
```

```
percept = linear_model.Perceptron()
```

```
LR = linear_model.LogisticRegression()
```

```
percept.fit(P, T).score(Ptest, Ttest)
```

```
0.6132596685082872
```



```
LR.fit(P, T).score(Ptest, Ttest)
```

```
0.6464088397790055
```



```
percept.coef_
```

```
array([[-864.10953638,  468.25259324]])
```



```
percept.intercept_
```

```
array([-706.])
```

```
%timeit -n1 -r1 (percept.fit(X, T).score(Xtest, Ttest))
```

```
1 loop, best of 1: 117 ms per loop
```

```
%timeit -n1 -r1 (LR.fit(X, T).score(Xtest, Ttest))
```

```
1 loop, best of 1: 22.3 s per loop
```

Topic 4 Summary

Here is what you learned in Topic 4

- A linear classifier is the linear inequality.

$$w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d \underset{\text{Class 2}}{\overset{\text{Class 1}}{\gtrless}} 0$$

- The weights can be optimized using various criteria
- The simple perceptron attempts to maximize the training accuracy.
- The pseudoinverse classifier minimizes the sum-of-squares error.
- Nonlinear classification can be achieved by extending the feature set using nonlinear derived features
- Multiclass classifiers are built using Kesler's construction
- AUC, the area under the ROC curve measures the intrinsic accuracy of a linear classifier