Psych 253
Advanced Statistical Modeling

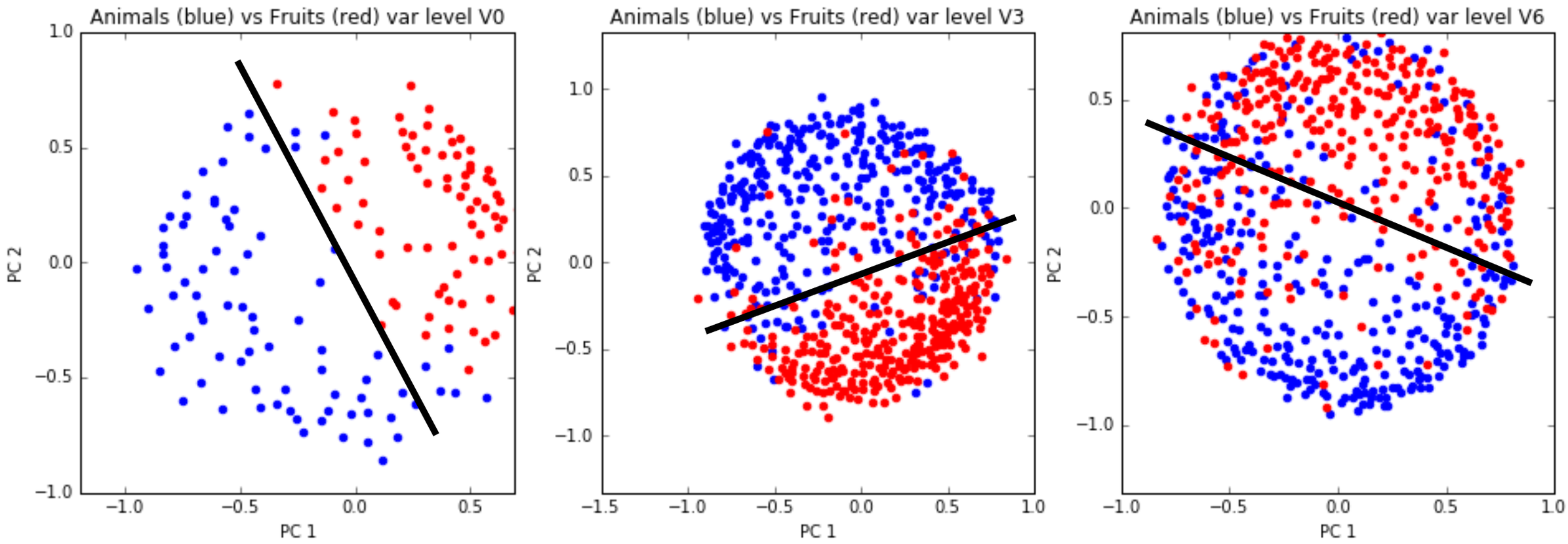
Regularization

Daniel Yamins
Wu Tsai Neurosciences Institute
Departments of Psychology and Computer Science
Stanford Artificial Intelligence Laboratory
Stanford University

Russ Poldrack
Department of Psychology
Stanford University

Regularization

Ok, recall:



Find ***w***'s and ***b*** that minimize:

$$\text{loss} = \max\left(0, 1 - \text{red}_j * \left[\sum_i w_i \cdot neuron_i(s_j) + b\right]\right)$$

averaged over training stimuli s_j

Regularization

Actually there are likely to be *many* hyperplanes with the same loss level:

Find ***w***'s and ***b*** that minimize:

$$\text{loss} = \max\left(0, 1 - \text{red}_j * \left[\sum_i w_i \cdot neuron_i(s_j) + b\right]\right)$$

averaged over training stimuli s_j

Regularization

Actually there are likely to be *many* hyperplanes with the same loss level:
so which one do we want??

Find w's and b that minimize:

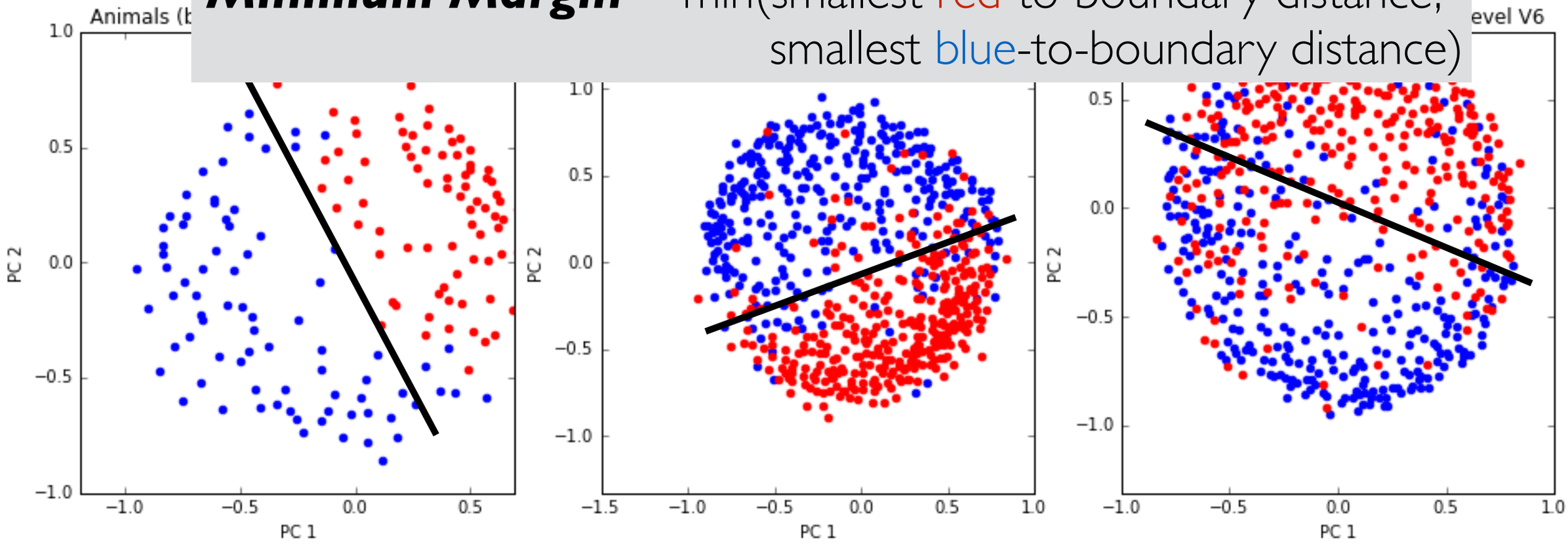
$$\text{loss} = \max\left(0, 1 - \text{red}_j * \left[\sum_i w_i \cdot neuron_i(s_j) + b\right]\right)$$

averaged over training stimuli s_j

Regularization

Which hyperplane do we want? The one with the **largest** minimum margin.

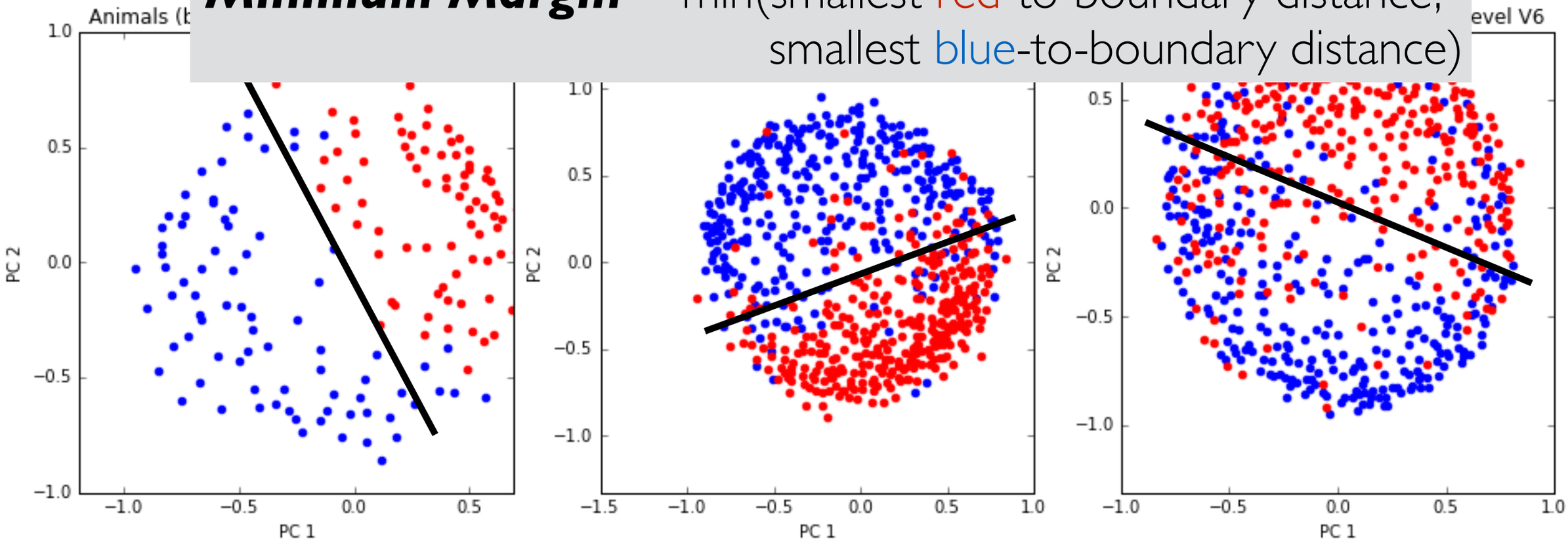
Minimum Margin = min(smallest red-to-boundary distance, smallest blue-to-boundary distance)



Regularization

Which hyperplane do we want? The one with the **largest** minimum margin.

Minimum Margin = min(smallest red-to-boundary distance, smallest blue-to-boundary distance)



Key math FACT: margin is well-measured by:

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

components of the slopes separating hyperplane

Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

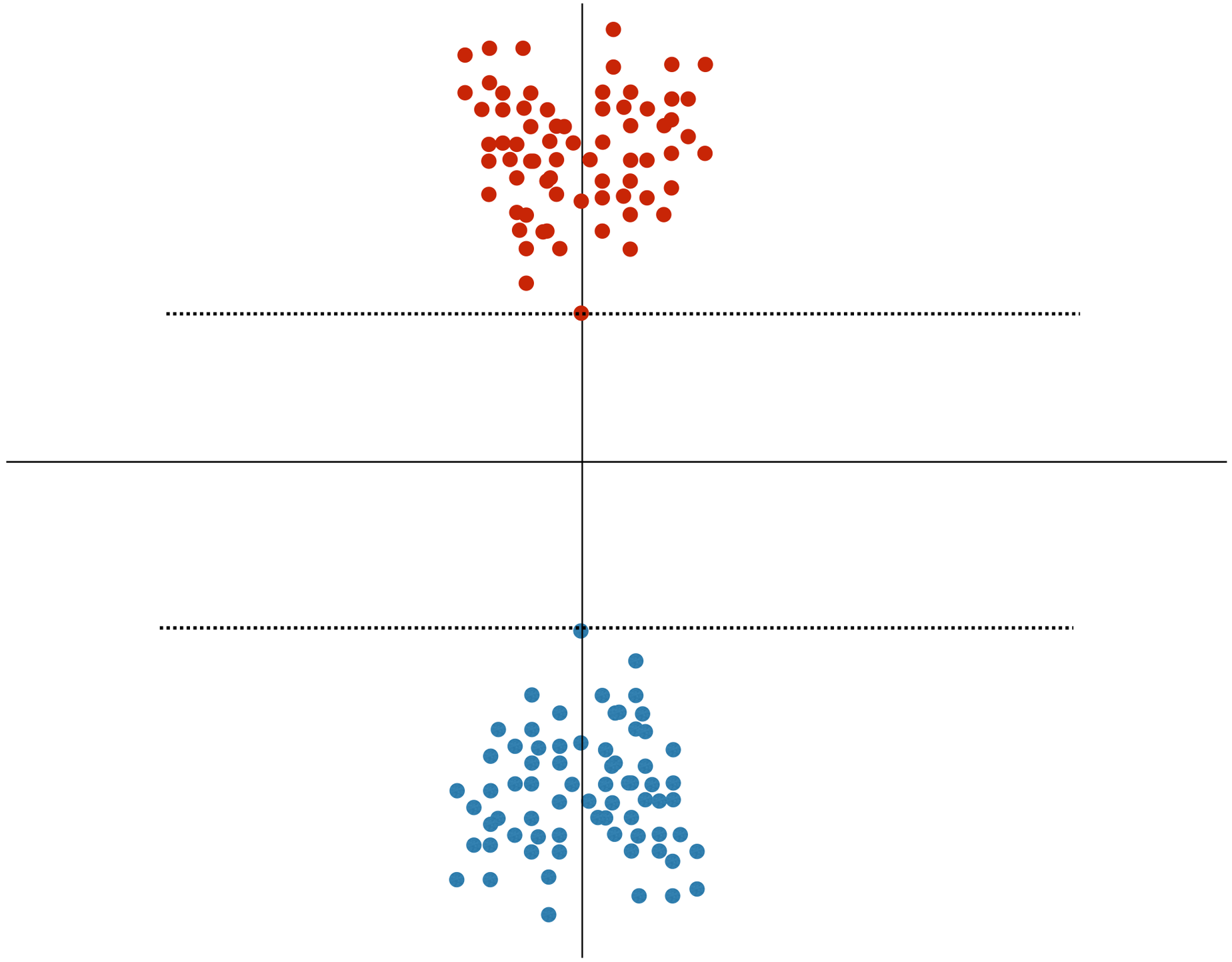
components of the slopes separating hyperplane

Why?

Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

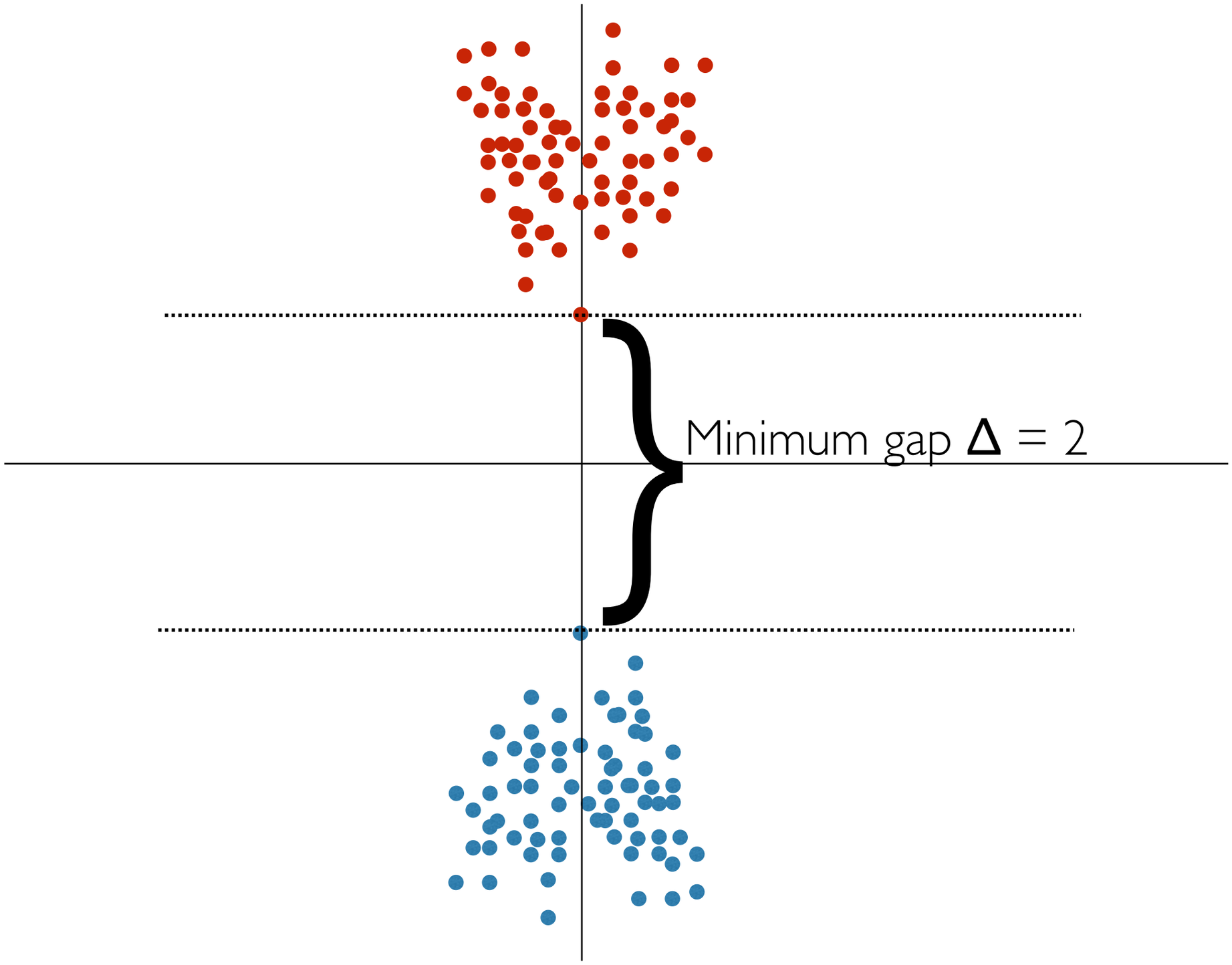
Why?



Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

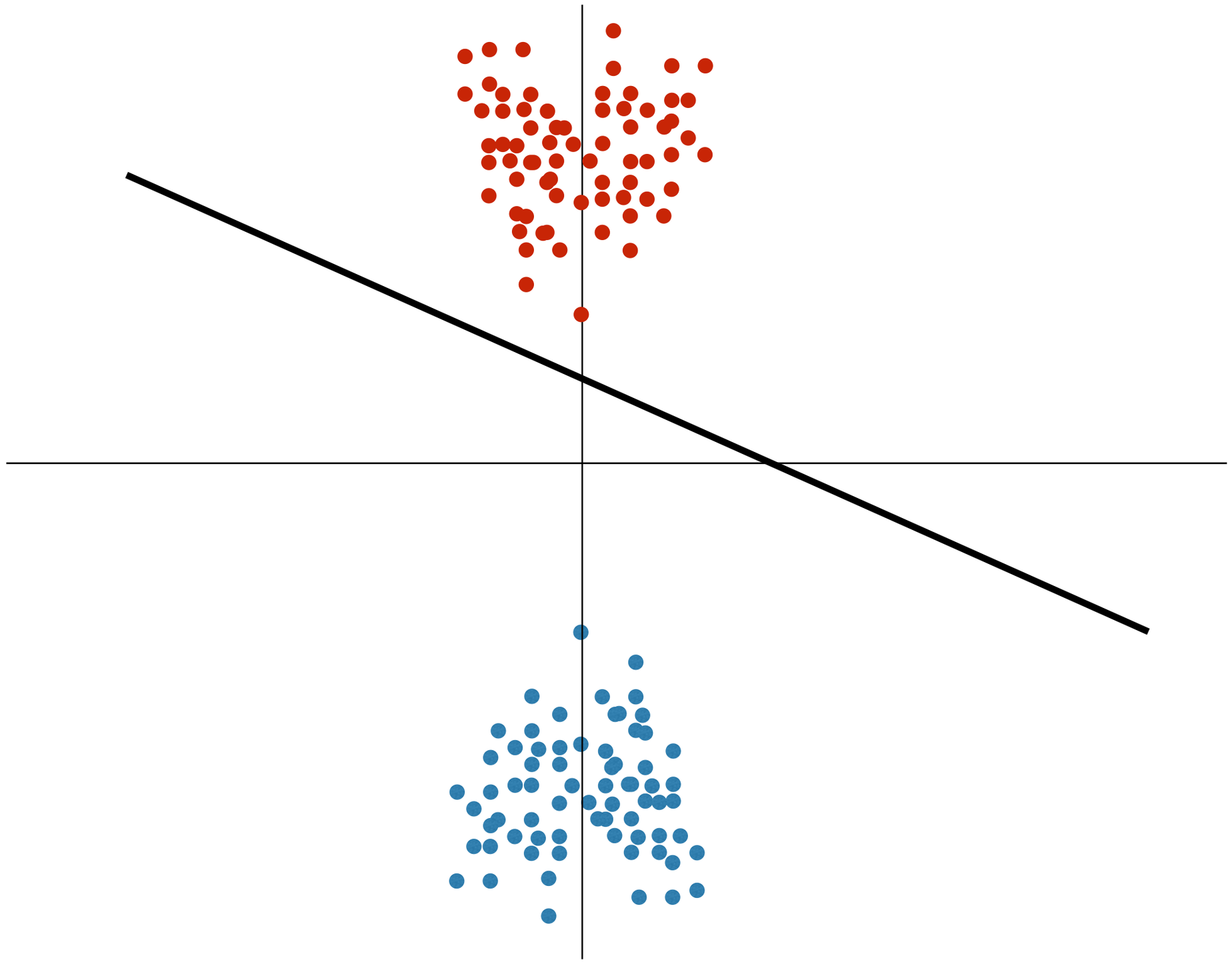
Why?



Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

Why?

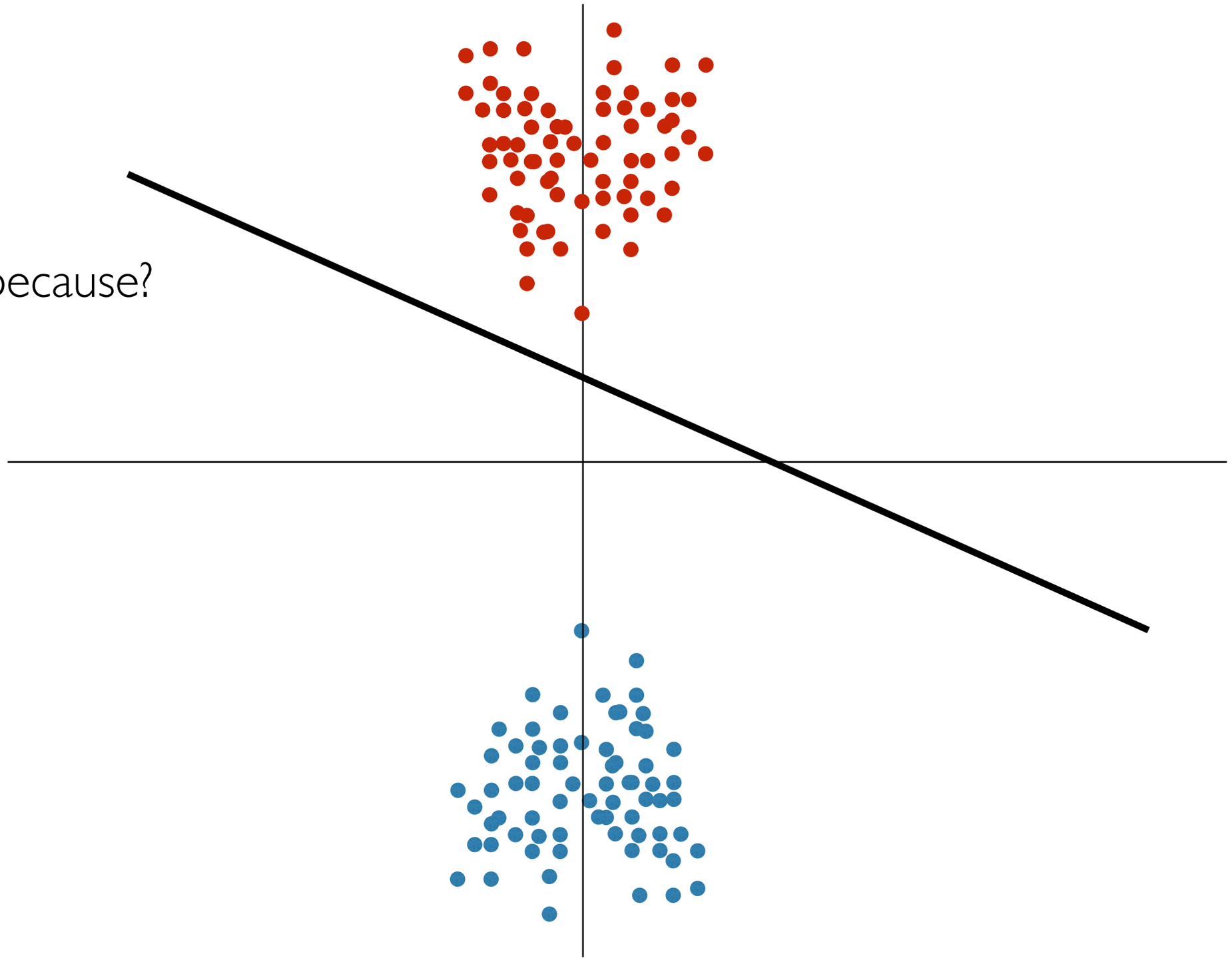


Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

Why?

Bad because?

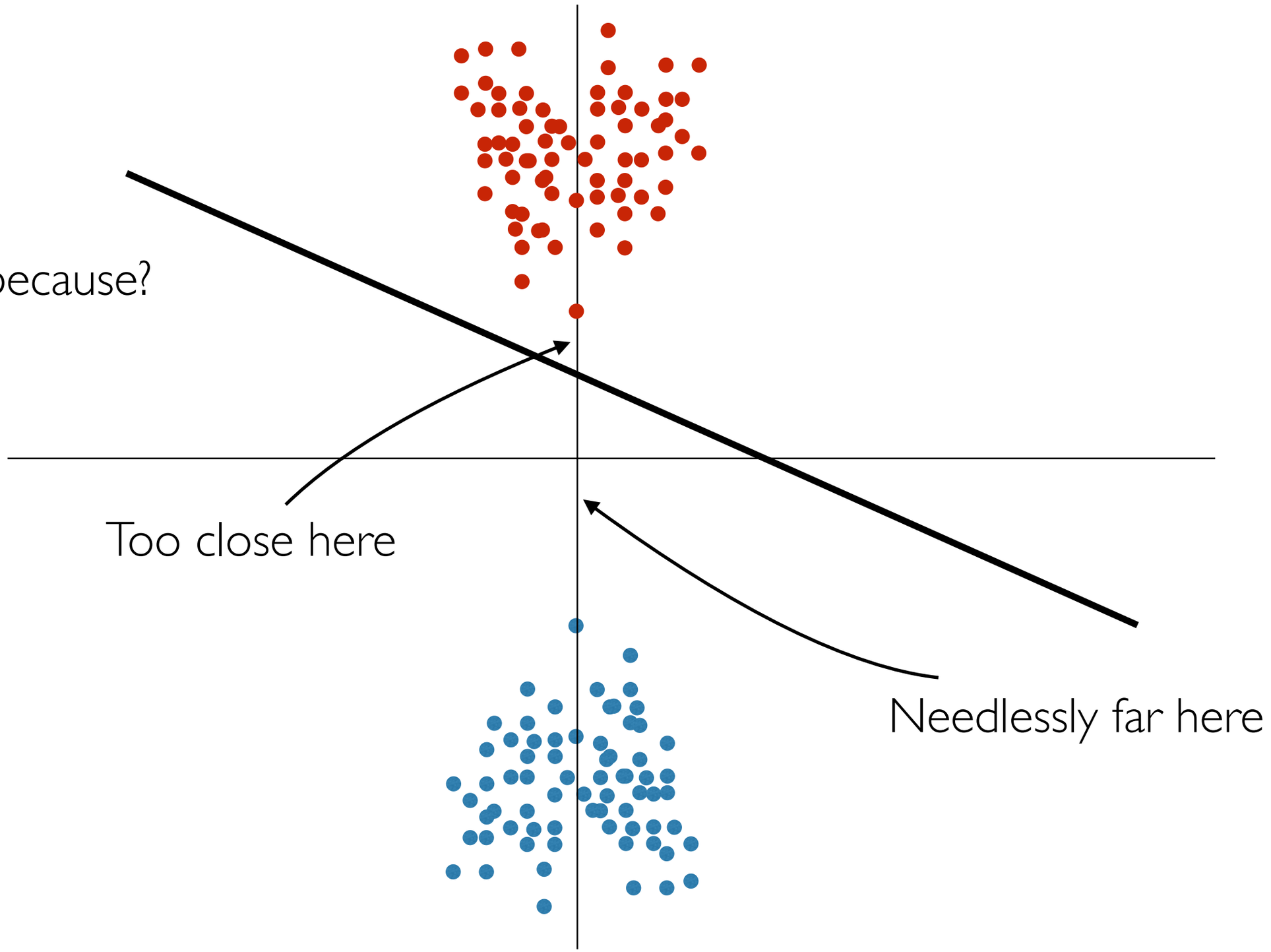


Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

Why?

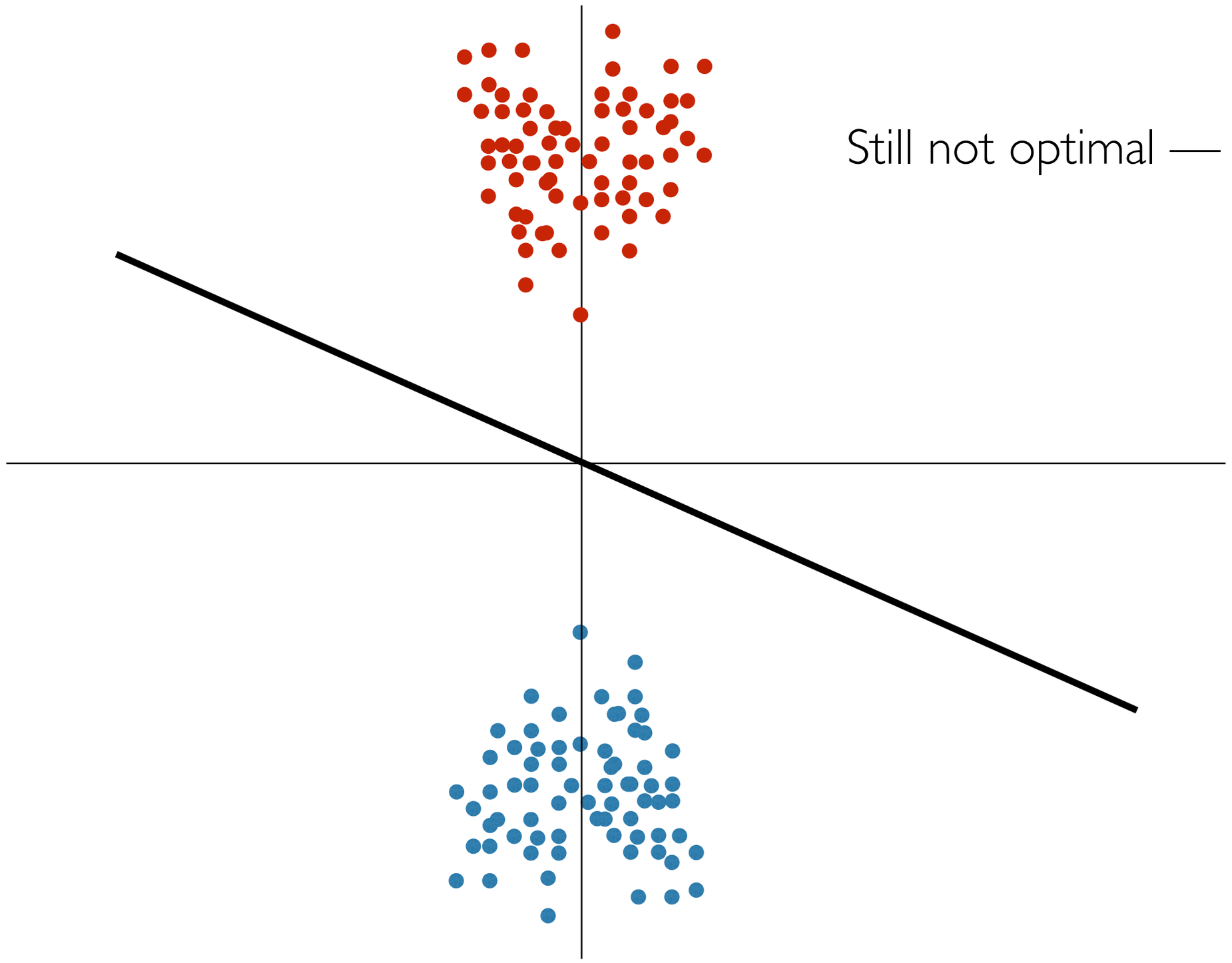
Bad because?



Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

Why?

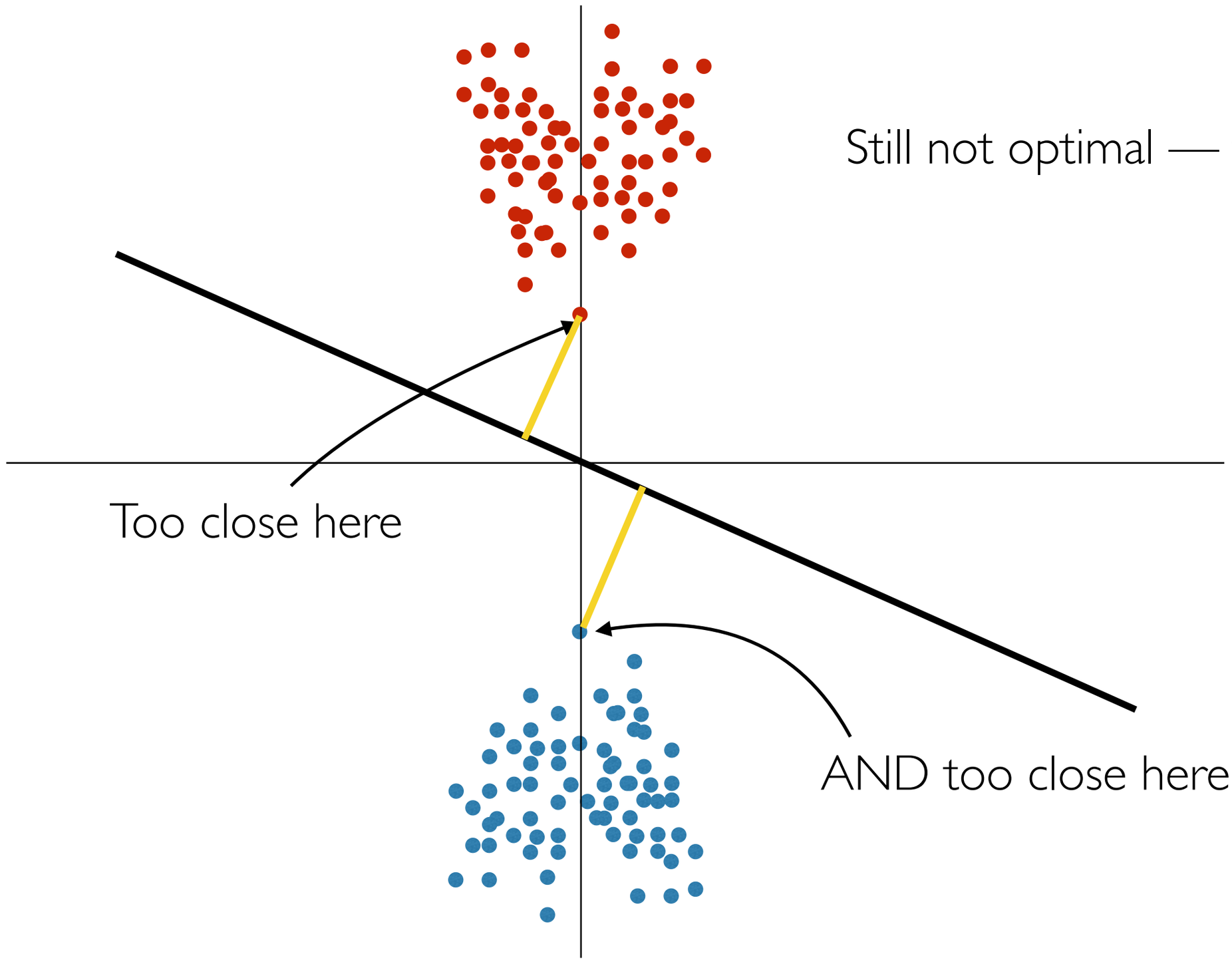


Still not optimal — why?

Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

Why?

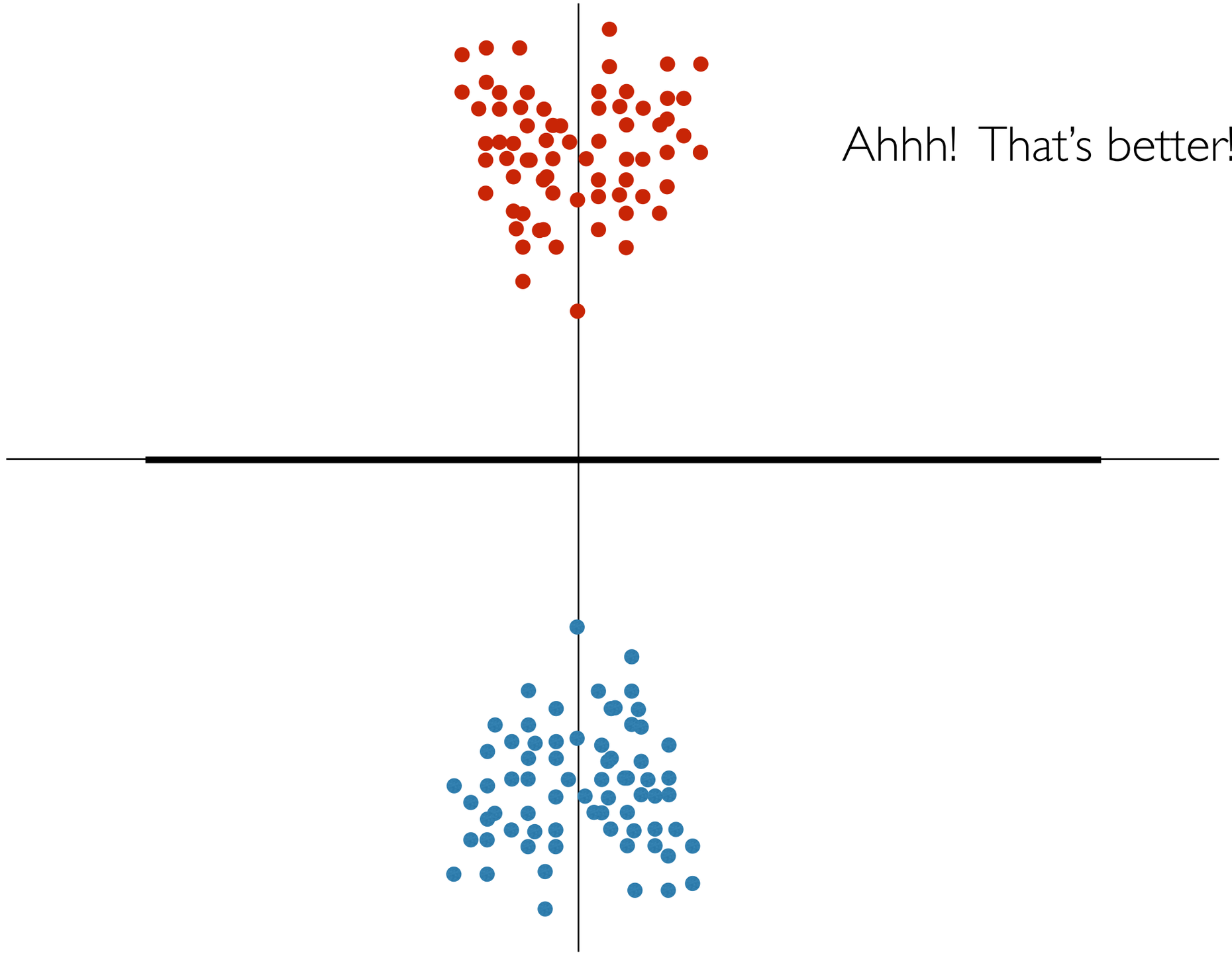


Still not optimal — why?

Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

Why?

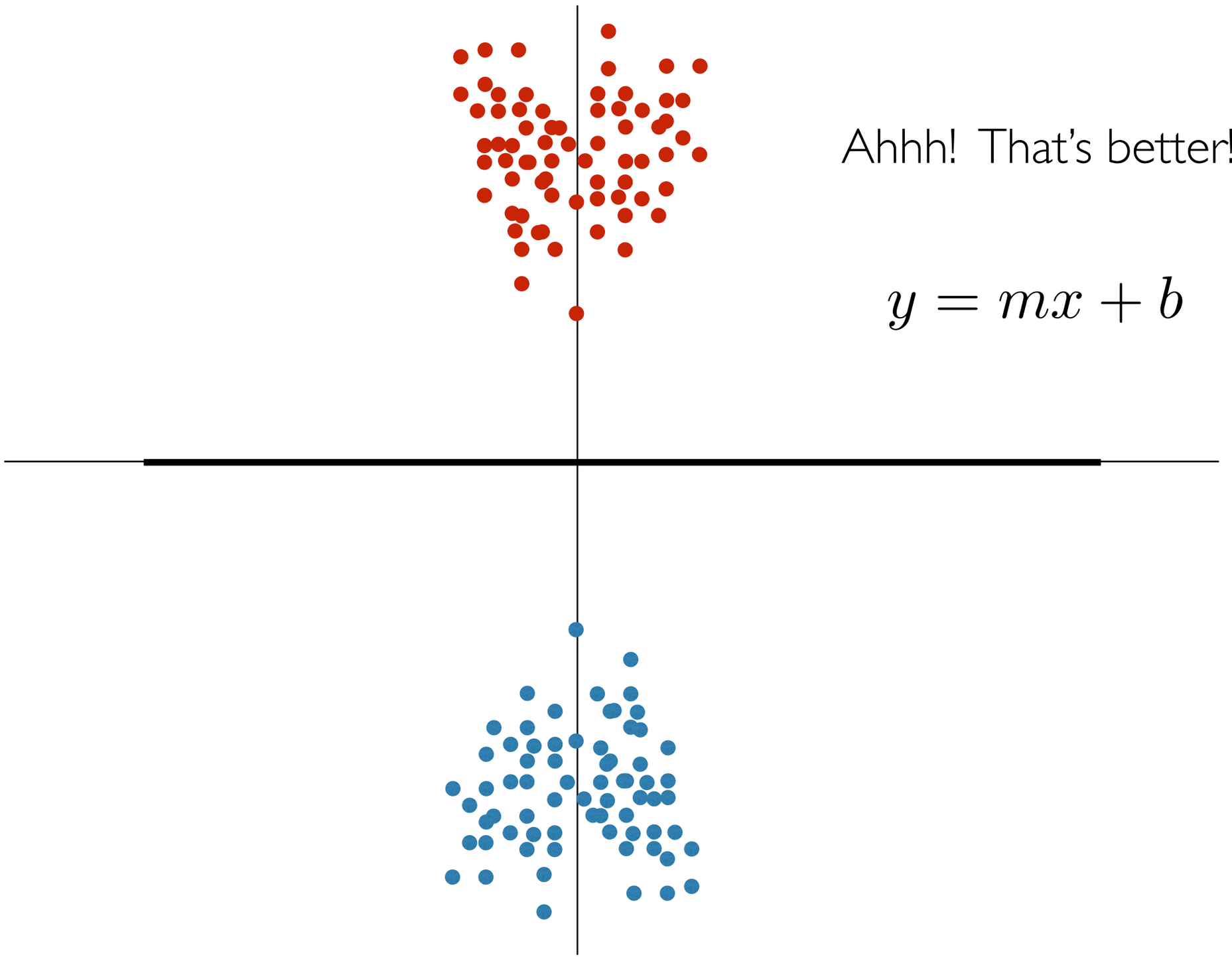


Ahhh! That's better!

Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

Why?



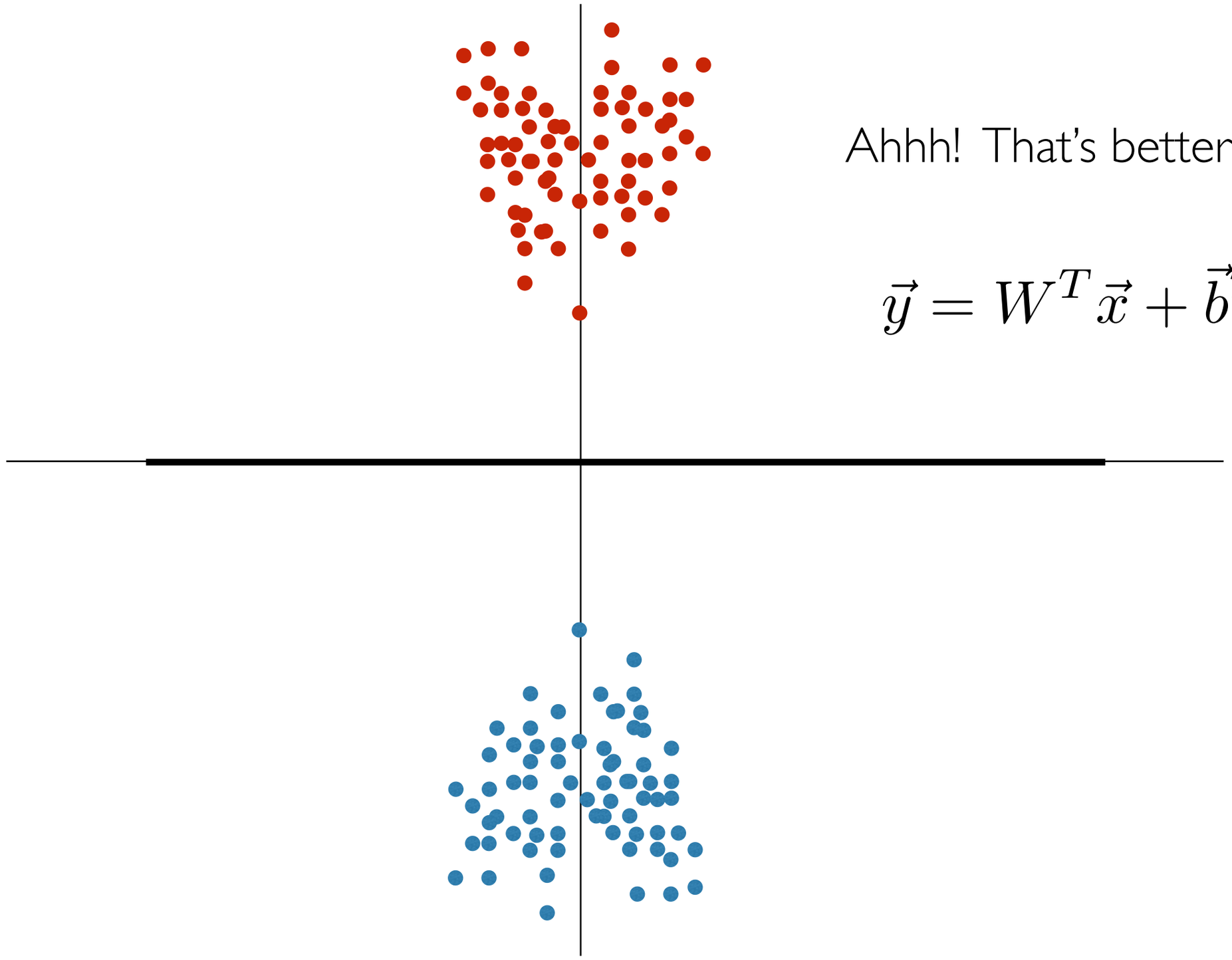
Ahhh! That's better!

$$y = mx + b$$

Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

Why?



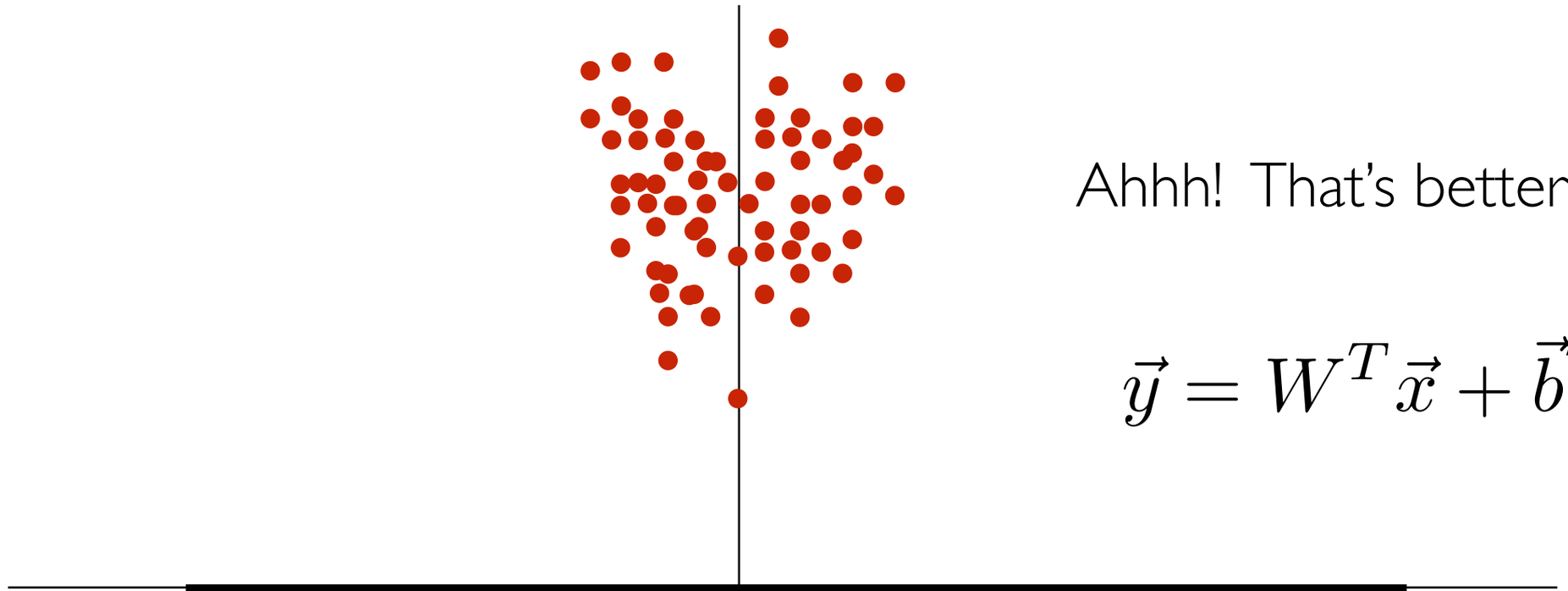
Ahhh! That's better!

$$\vec{y} = W^T\vec{x} + \vec{b}^T$$

Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

Why?

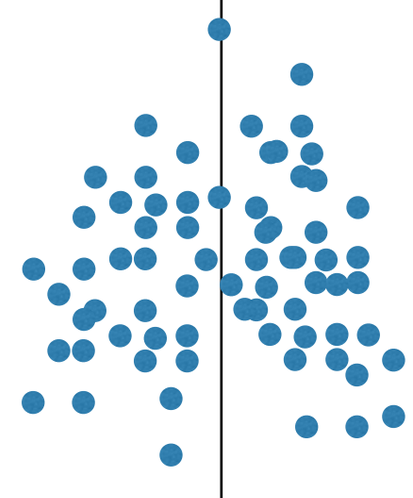


Ahhh! That's better!

$$\vec{y} = W^T\vec{x} + \vec{b}^T$$

In this case:

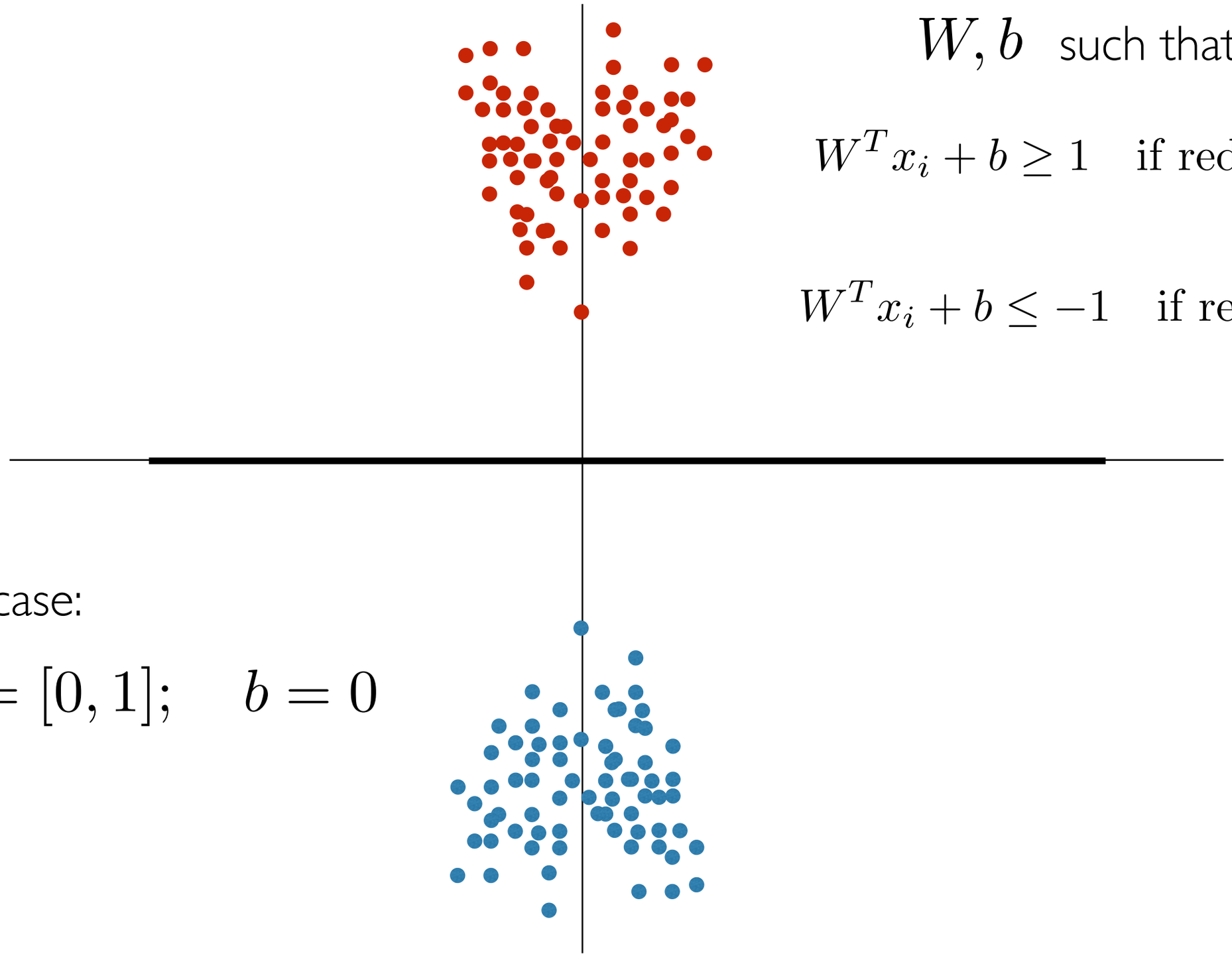
$$W = [0, 1]; \quad b = 0$$



Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

Why?



W, b such that

$$W^T x_i + b \geq 1 \quad \text{if } \text{red}_i = 1$$

$$W^T x_i + b \leq -1 \quad \text{if } \text{red}_i = -1$$

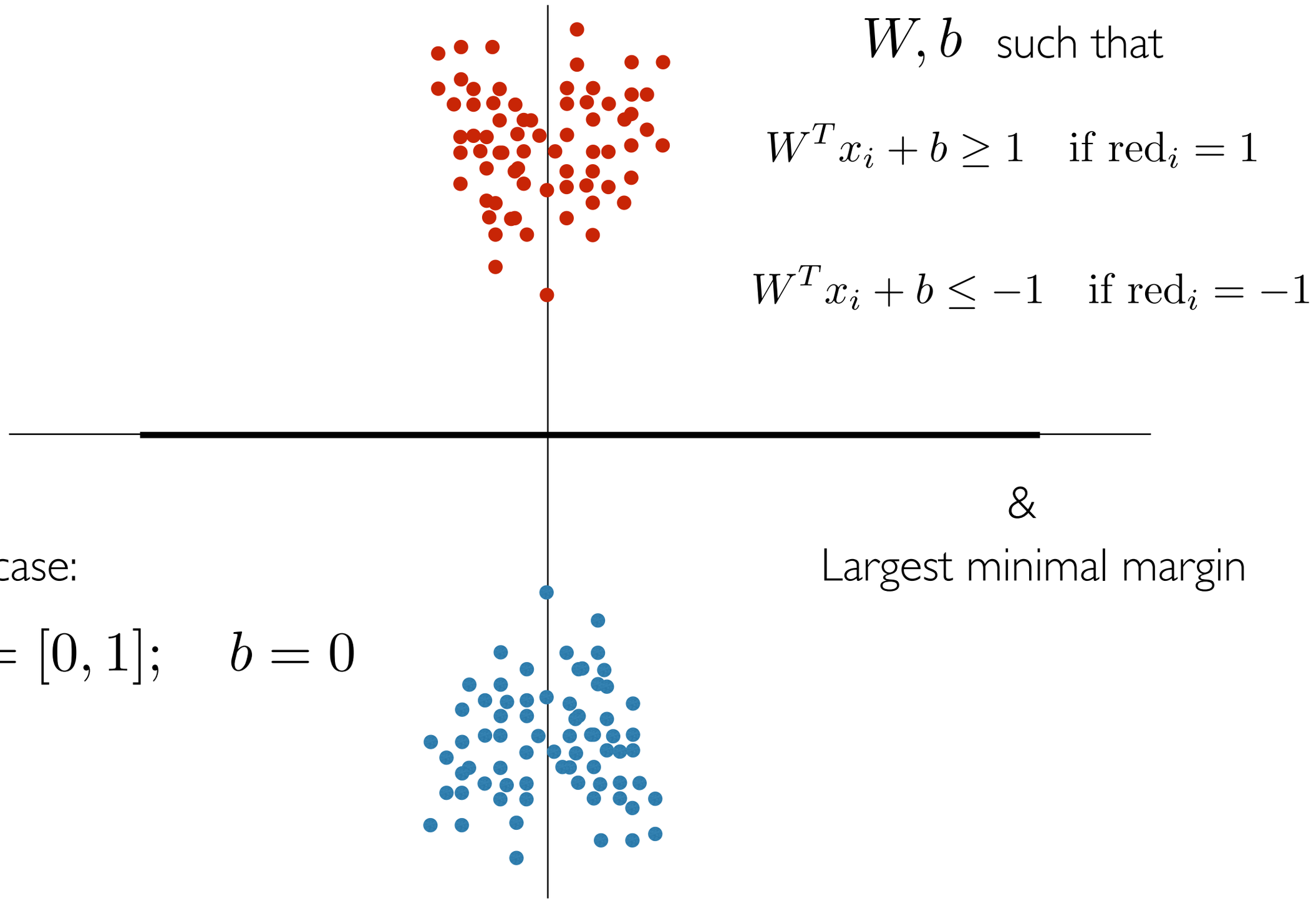
In this case:

$$W = [0, 1]; \quad b = 0$$

Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

Why?



W, b such that

$$W^T x_i + b \geq 1 \quad \text{if } \text{red}_i = 1$$

$$W^T x_i + b \leq -1 \quad \text{if } \text{red}_i = -1$$

&

Largest minimal margin

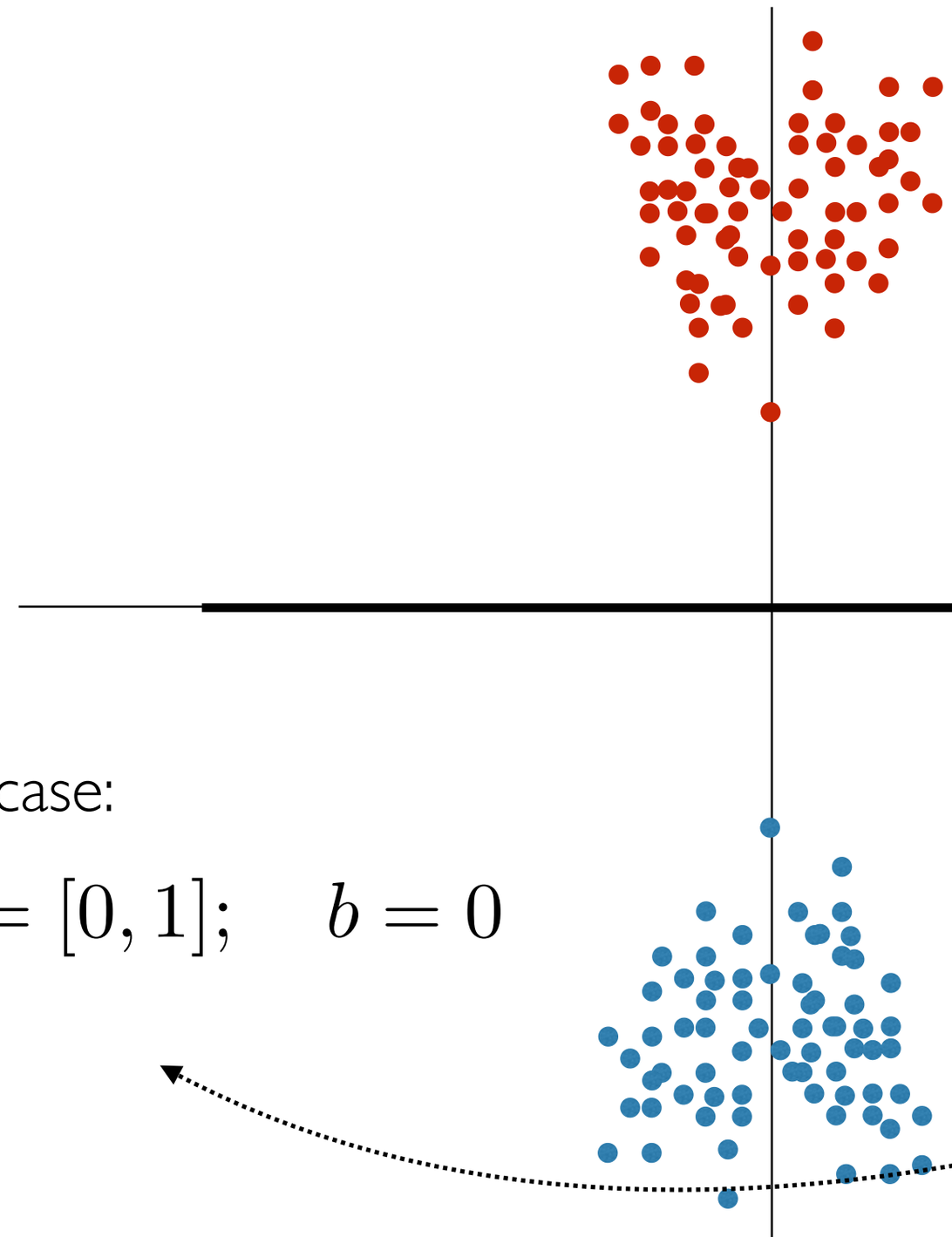
In this case:

$$W = [0, 1]; \quad b = 0$$

Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

Why?



W, b such that

$W^T x_i + b \geq 1 \quad \text{if red}_i = 1$

$W^T x_i + b \leq -1 \quad \text{if red}_i = -1$

&

Largest minimal margin

equivalent to

$||W||_2^2$ minimized

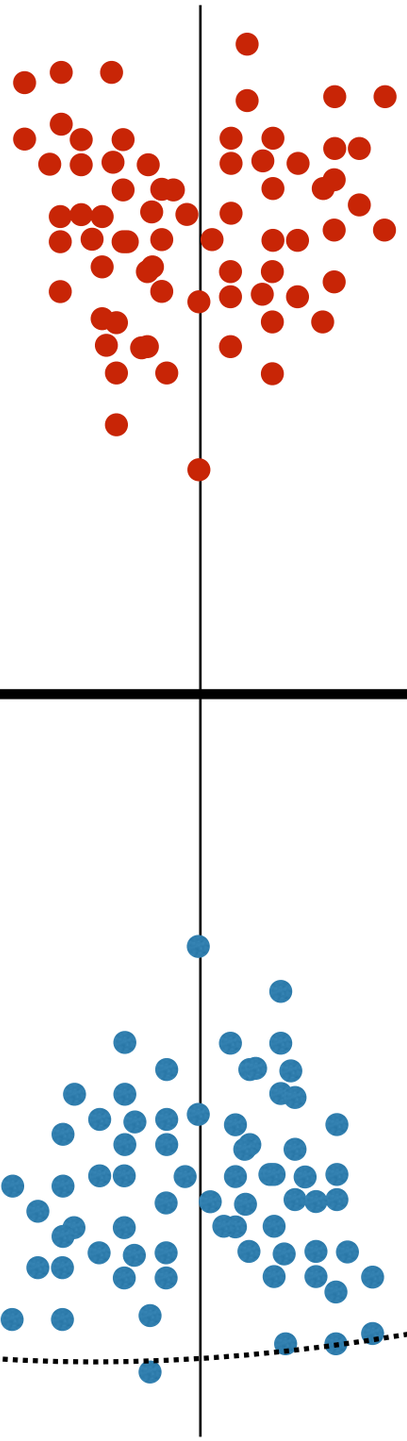
In this case:

$W = [0, 1]; \quad b = 0$

Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

Why?



W, b such that

$$\text{red}_i \cdot (W^T x_i + b) \geq 1$$

&

Largest minimal margin

equivalent to

$||W||_2^2$ minimized

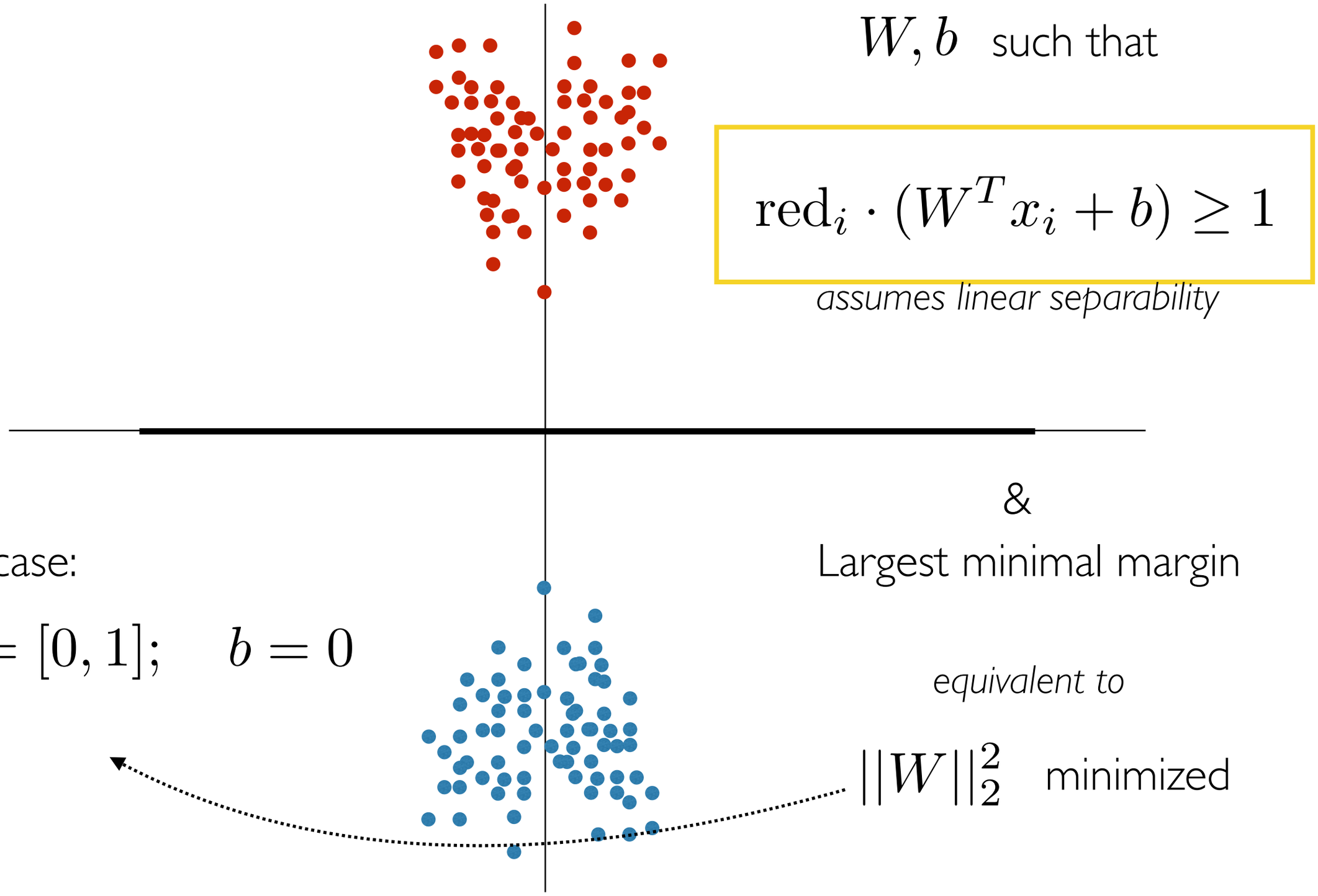
In this case:

$$W = [0, 1]; \quad b = 0$$

Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

Why?



W, b such that

$$\text{red}_i \cdot (W^T x_i + b) \geq 1$$

assumes linear separability

&

Largest minimal margin

equivalent to

$||W||_2^2$ minimized

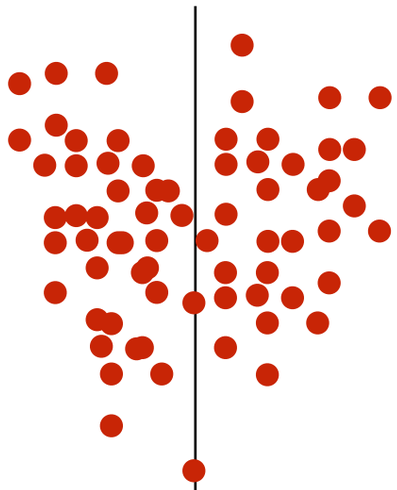
In this case:

$$W = [0, 1]; \quad b = 0$$

Key math FACT: margin is well-measured by (inverse of):

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

Why?



W, b such that

$$\text{red}_i \cdot (W^T x_i + b) \geq 1$$

assumes linear separability

and really only speaks to the worst case

&

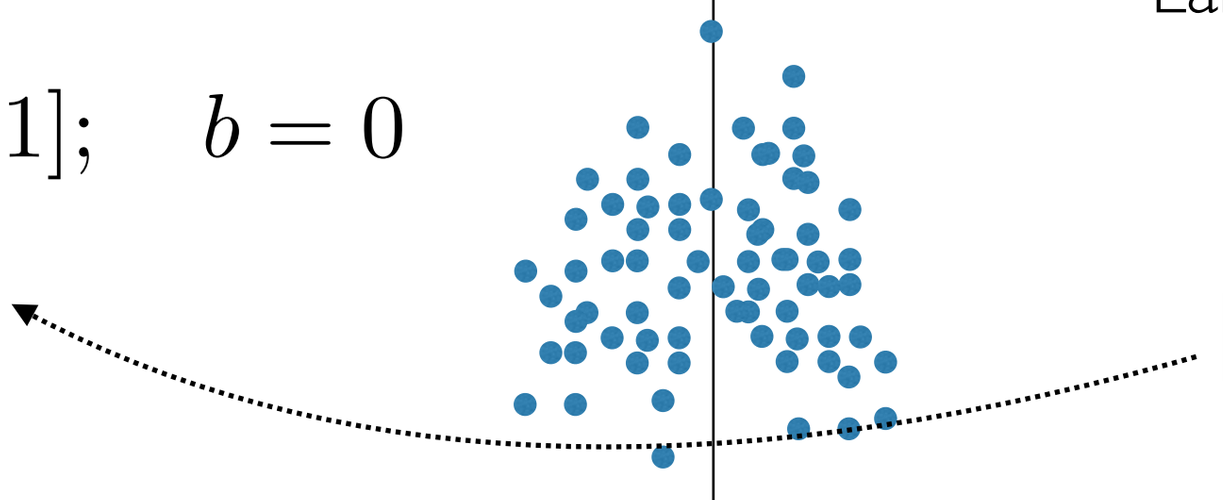
Largest minimal margin

equivalent to

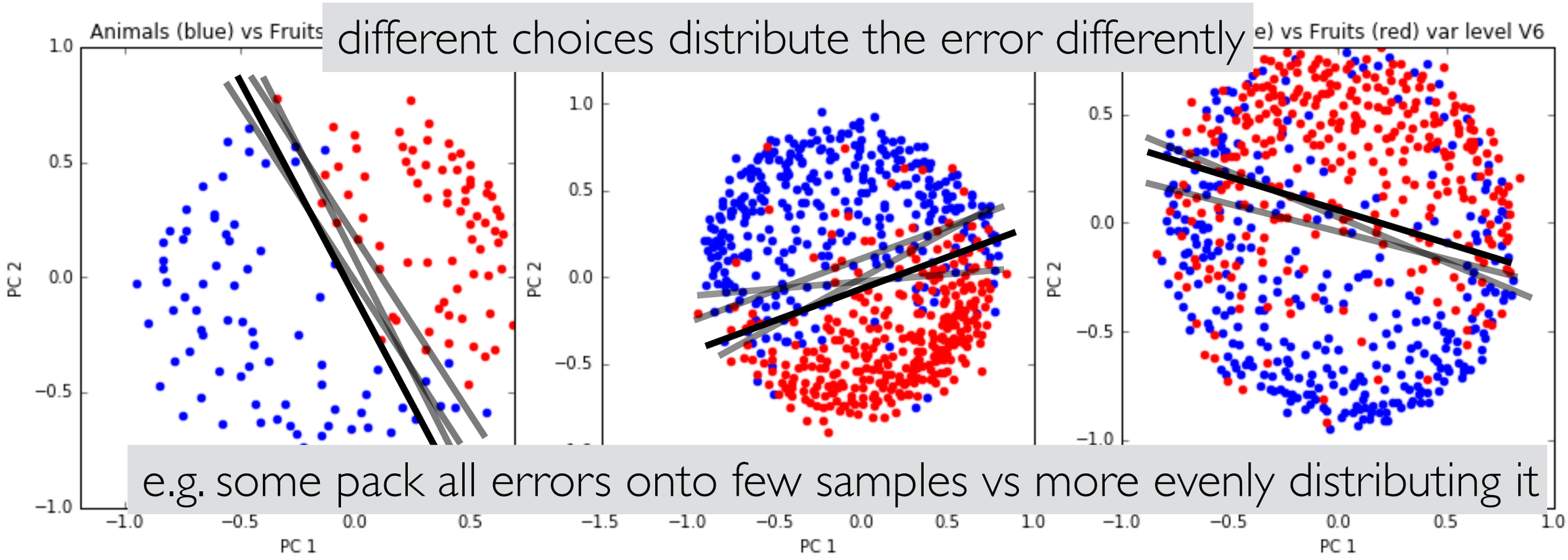
$||W||_2^2$ minimized

In this case:

$$W = [0, 1]; \quad b = 0$$



Regularization



New goal: find ***w***'s and ***b*** and minimize:

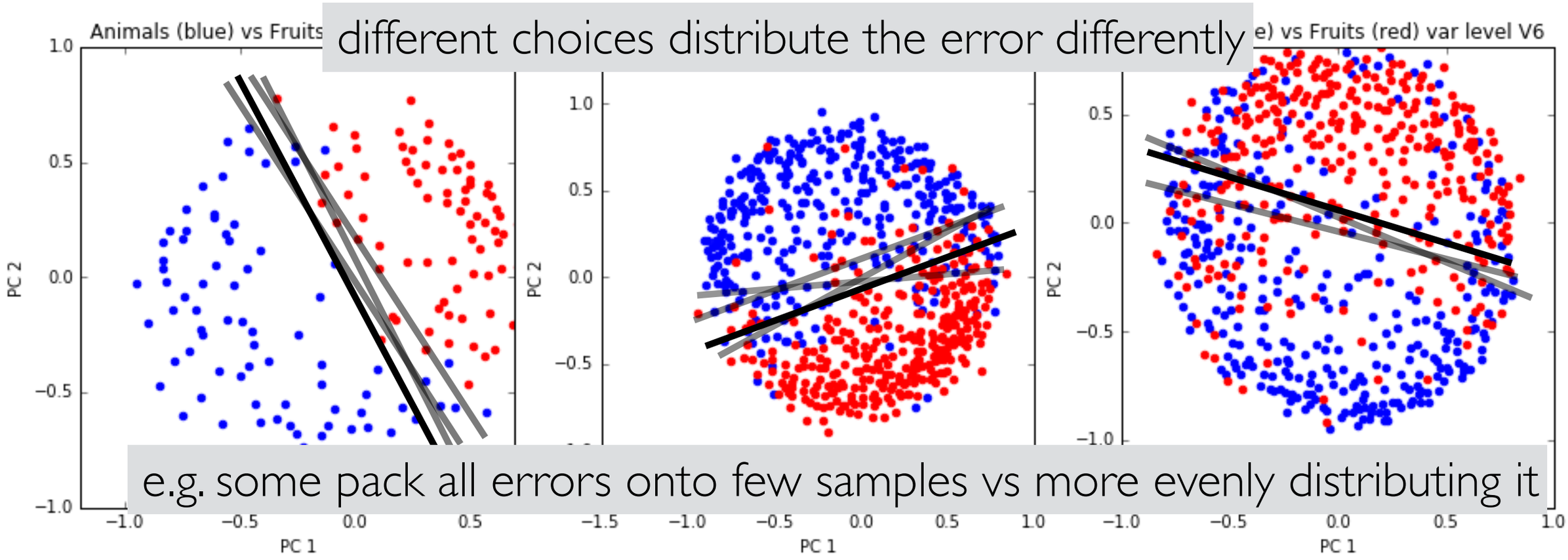
$$\text{loss} = max\left(0, 1 - \text{red}_i * \left[\sum_i w_i \cdot neuron_i(s_j) + b\right]\right) + \lambda \cdot ||w||_2^2$$

original hinge loss

small margin penalty

. . . averaged over training stimuli s_j

Regularization



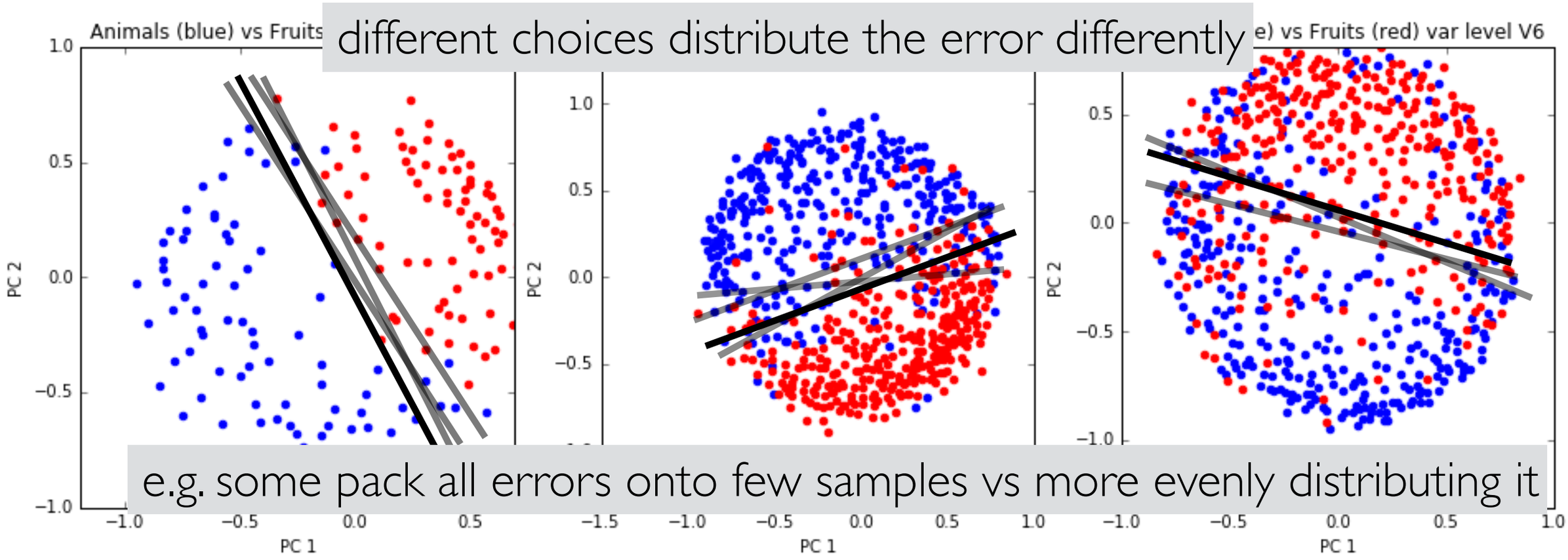
New goal: find ***w***'s and ***b*** and minimize:

$$\text{loss} = \underbrace{max\left(0, 1 - \text{red}_i * \left[\sum_i w_i \cdot neuron_i(s_j) + b\right]\right)}_{\text{original hinge loss}} + \underbrace{\lambda \cdot ||w||_2^2}_{\text{small margin penalty}}$$

average case | worst case

. . . averaged over training stimuli s_j

Regularization



New goal: find ***w***'s and ***b*** and minimize:

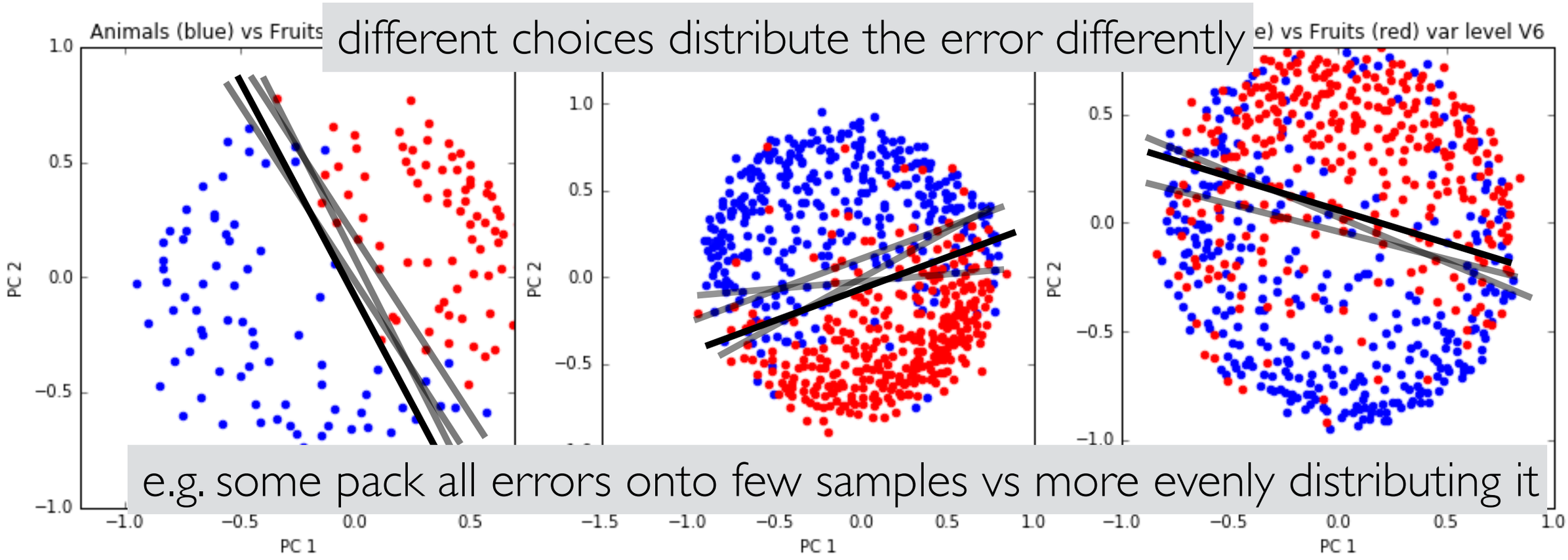
"L2 regularization"

$$\text{loss} = \underbrace{max\left(0, 1 - \text{red}_i * \left[\sum_i w_i \cdot neuron_i(s_j) + b\right]\right)}_{\text{original hinge loss}} + \underbrace{\lambda \cdot ||w||_2^2}_{\text{small margin penalty}}$$

average case | worst case

. . . averaged over training stimuli s_j

Regularization



New goal: find ***w***'s and ***b*** and minimize:

"L2 regularization"

$$\text{loss} = \underbrace{max\left(0, 1 - \text{red}_i * \left[\sum_i w_i \cdot neuron_i(s_j) + b\right]\right)}_{\text{original hinge loss}} + \underbrace{\lambda \cdot ||w||_2^2}_{\substack{\text{small margin} \\ \text{penalty}}}$$

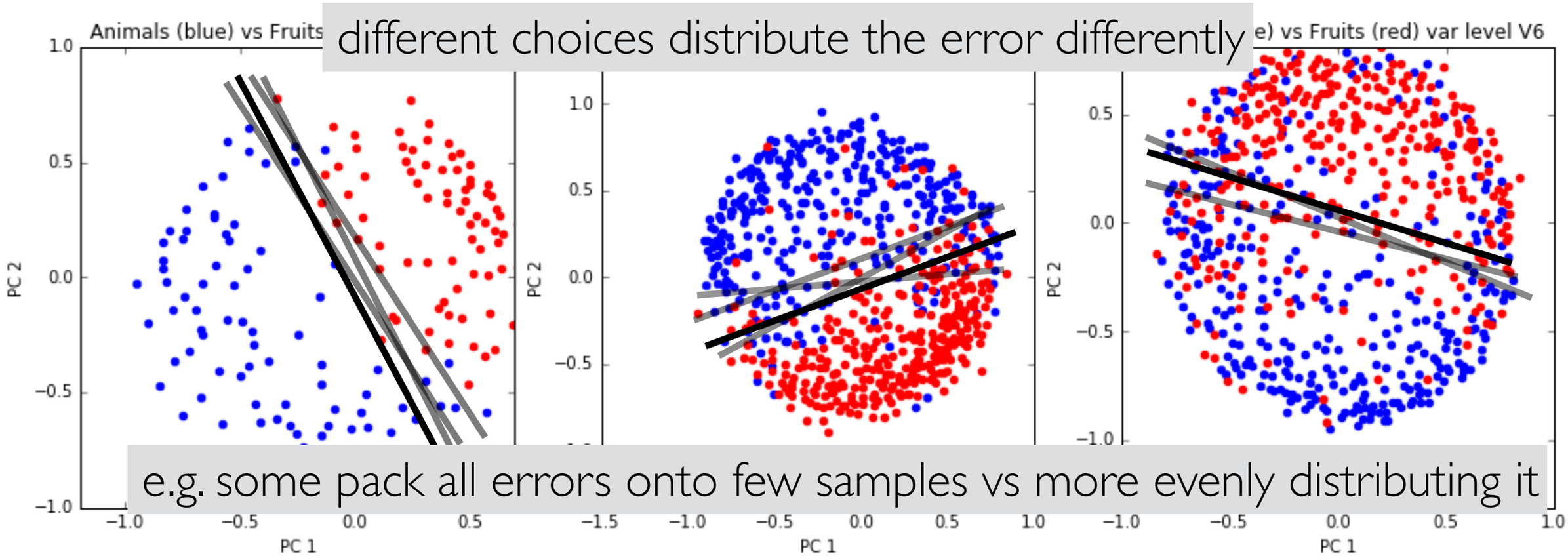
average case

worst case

. . . averaged over training stimuli s_j

… and no hard requirement for linear separability

Regularization



Actually, **`scikit-learn`** (like most SVM libraries) uses this parameterization:

$$\text{loss} = C \cdot \max\left(0, 1 - \text{red}_j * \left[\sum_i w_i \cdot neuron_i(s_j) + b\right]\right) + ||w||^2$$

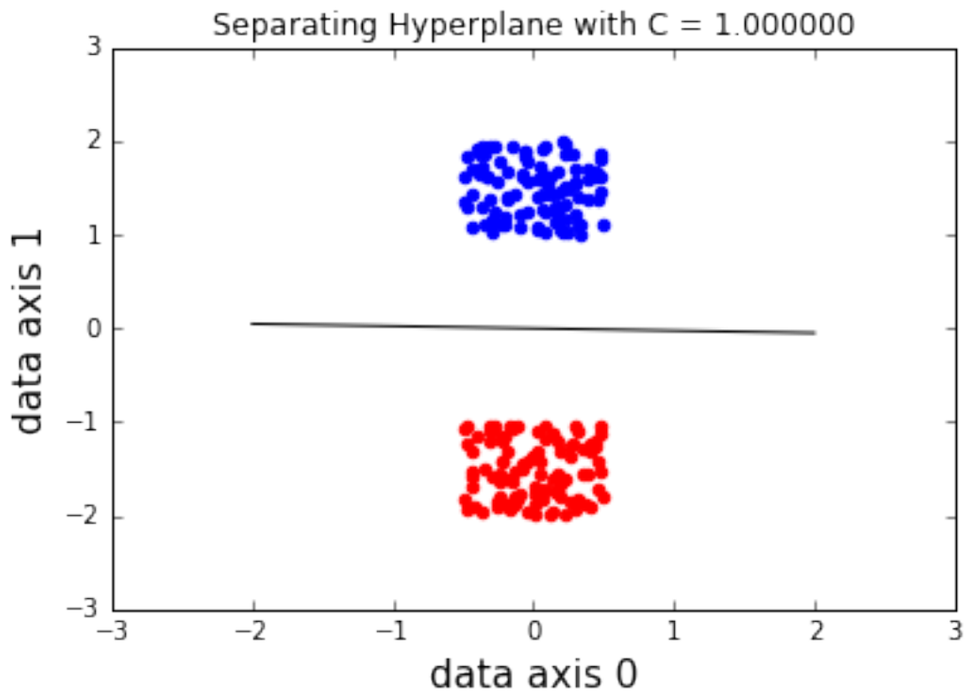
$$\text{with} \quad C = \frac{1}{\lambda}$$

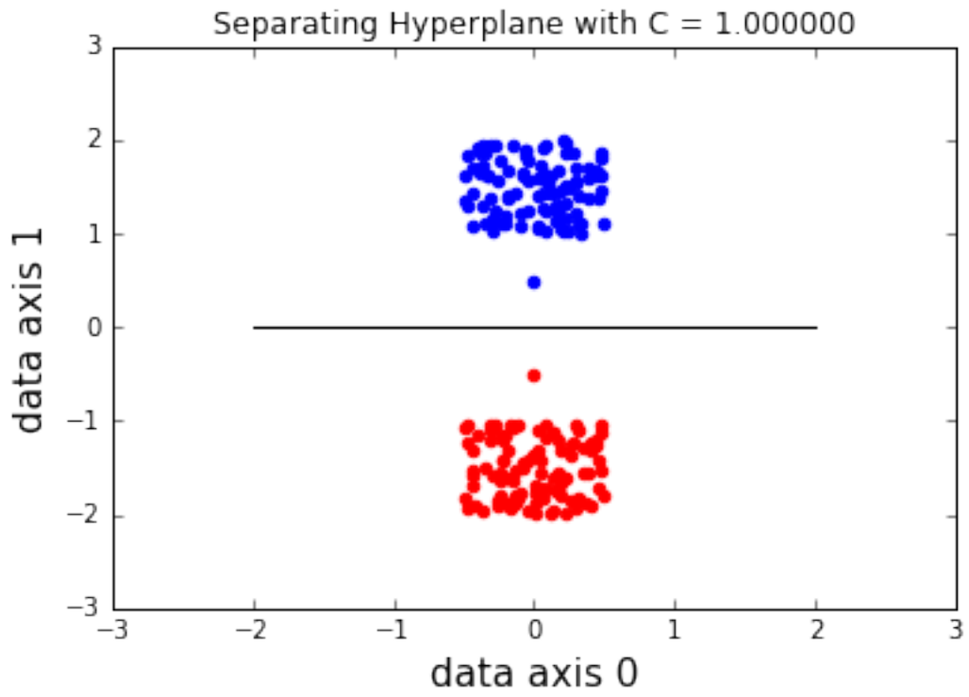
Regularization

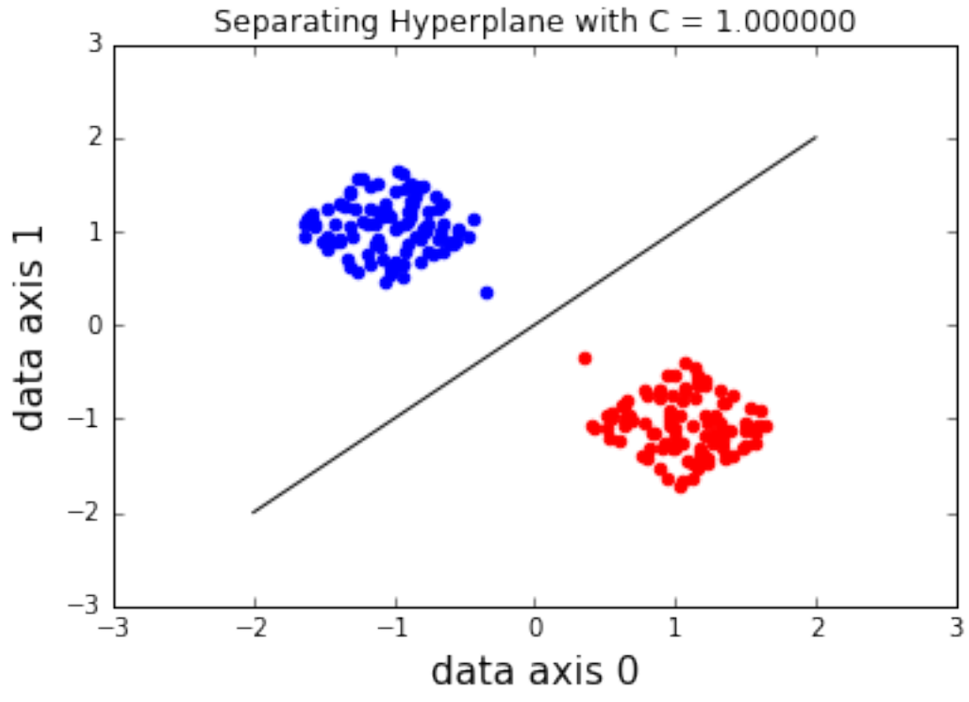
[Examples motivating L2 regularization]

Regularization

Some basic examples:

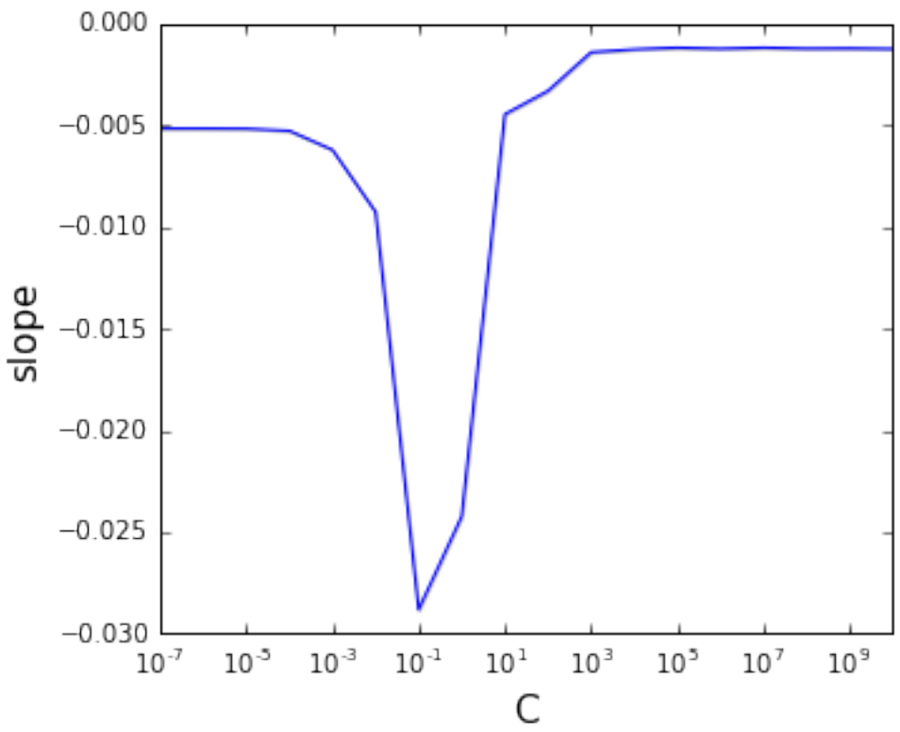


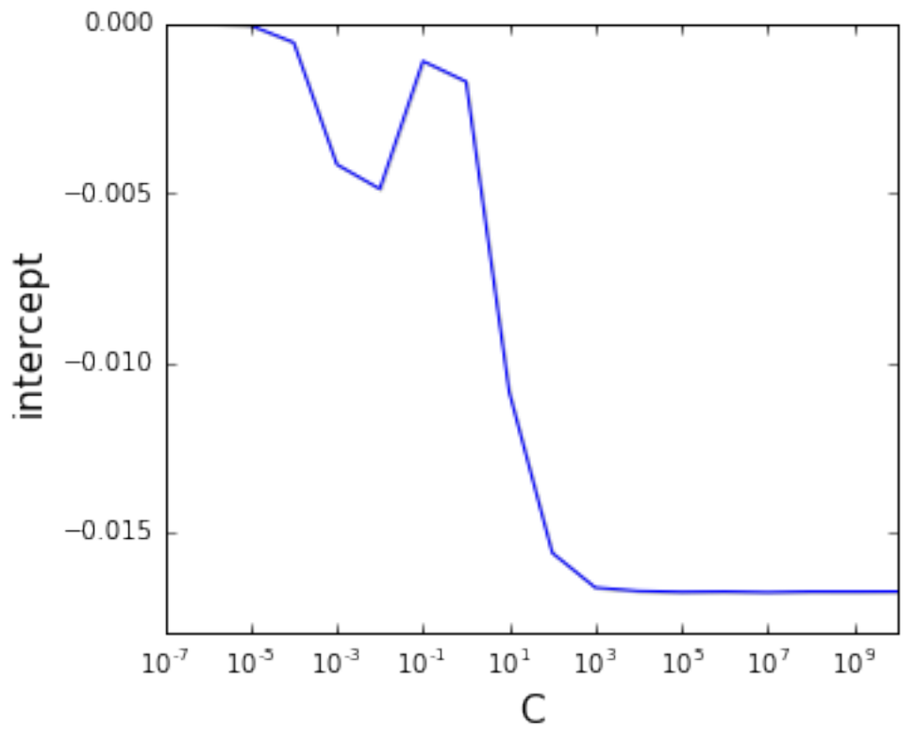


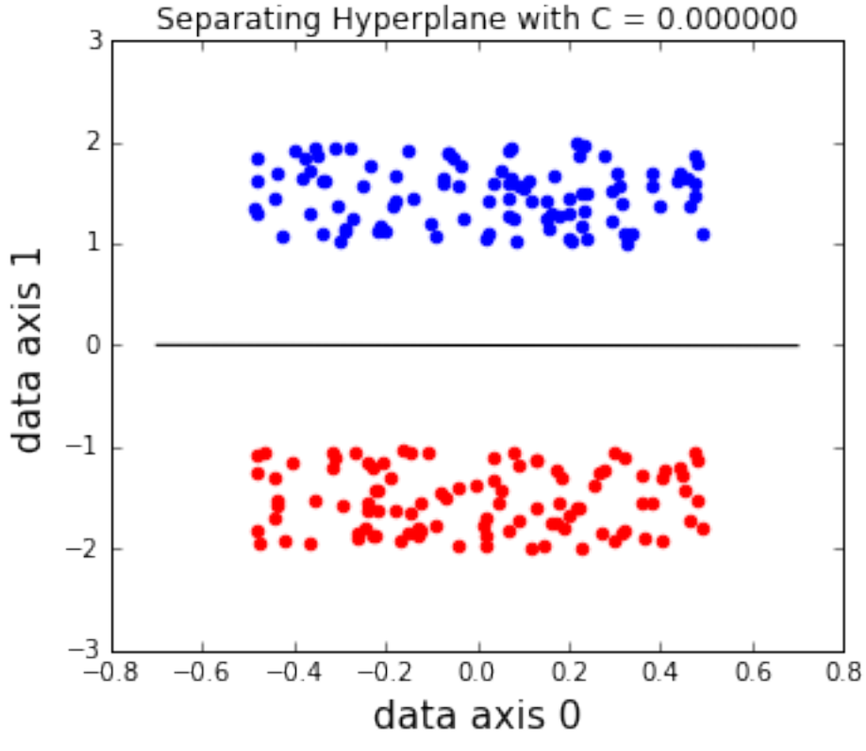


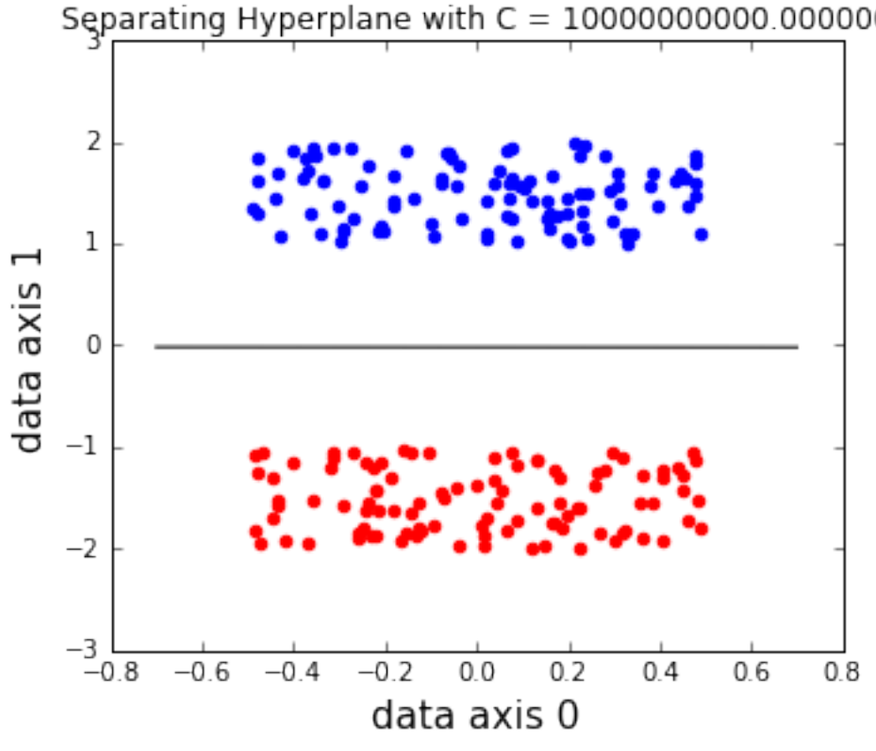
Regularization

L2-regularization doesn't matter if data is nicely separable:



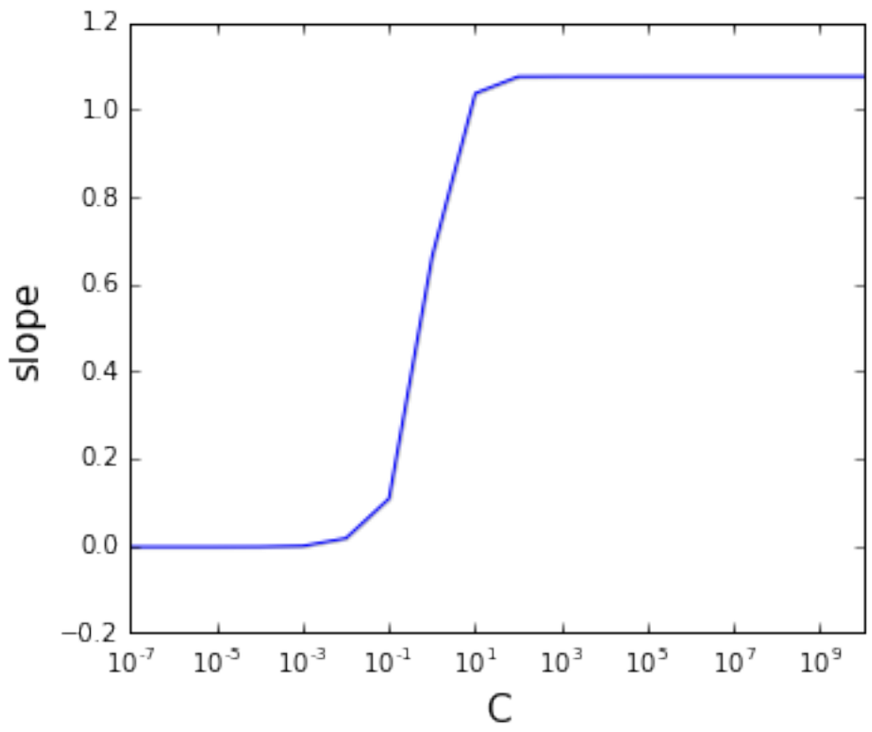


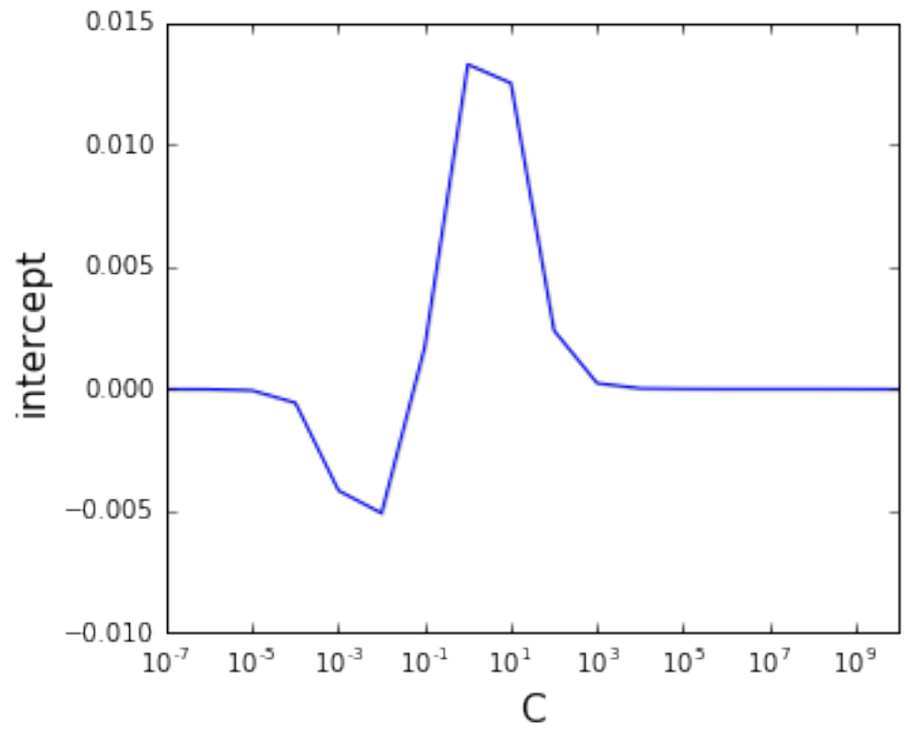




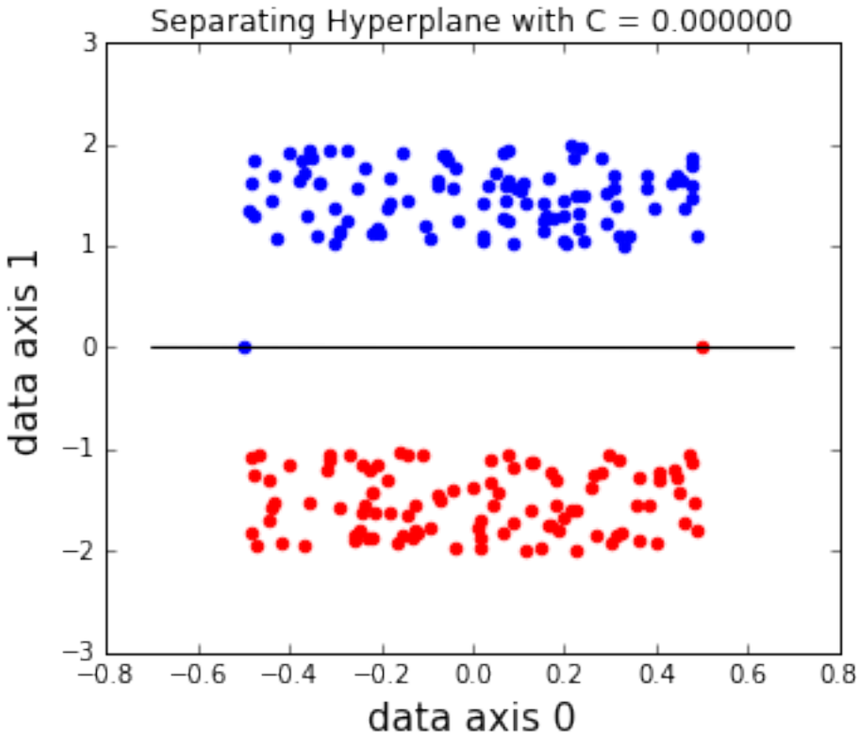
Regularization

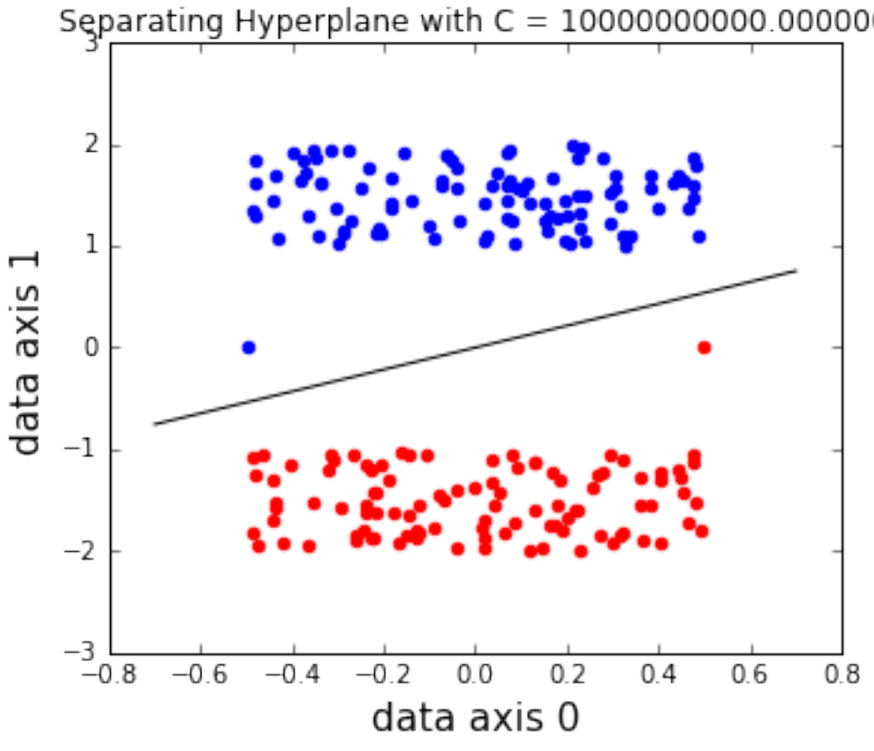
L2-regularization does matter if data **isn't** nicely separable:





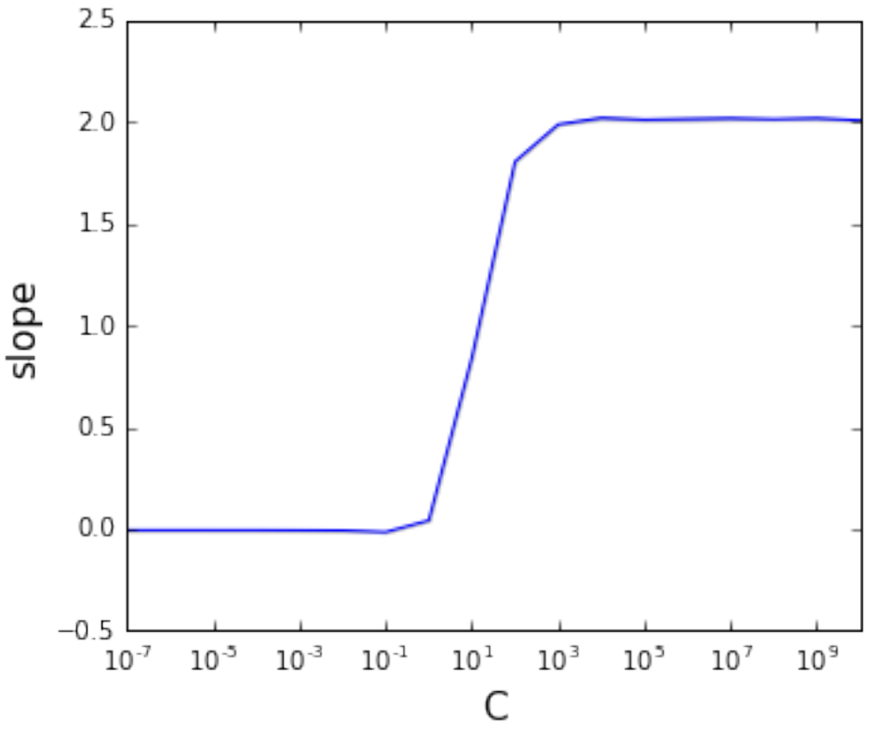
Effect of increasing l2-regularization for badly-separated data

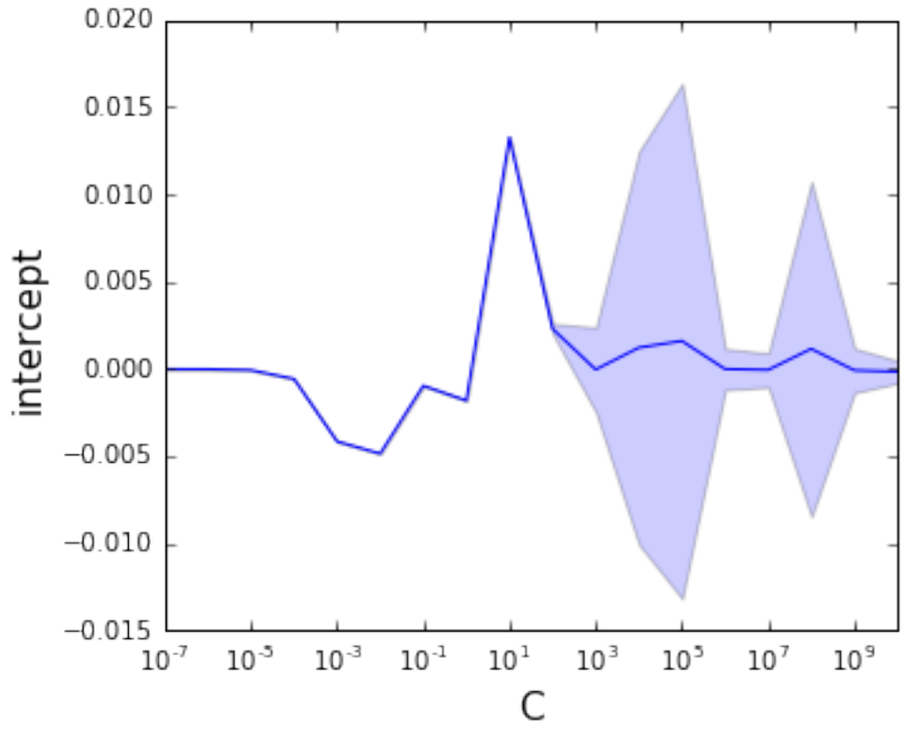




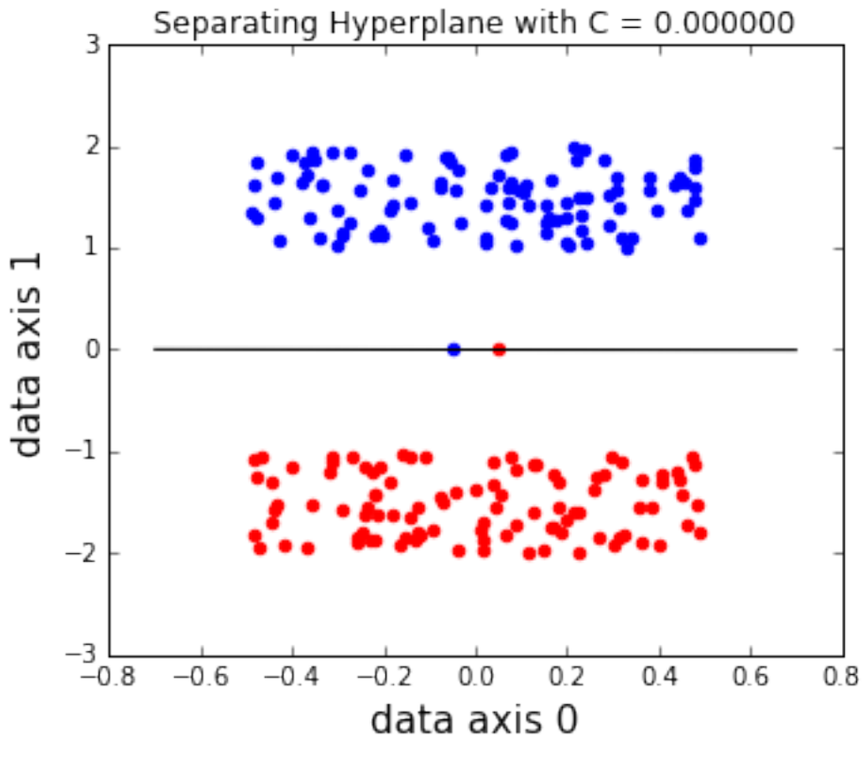
Regularization

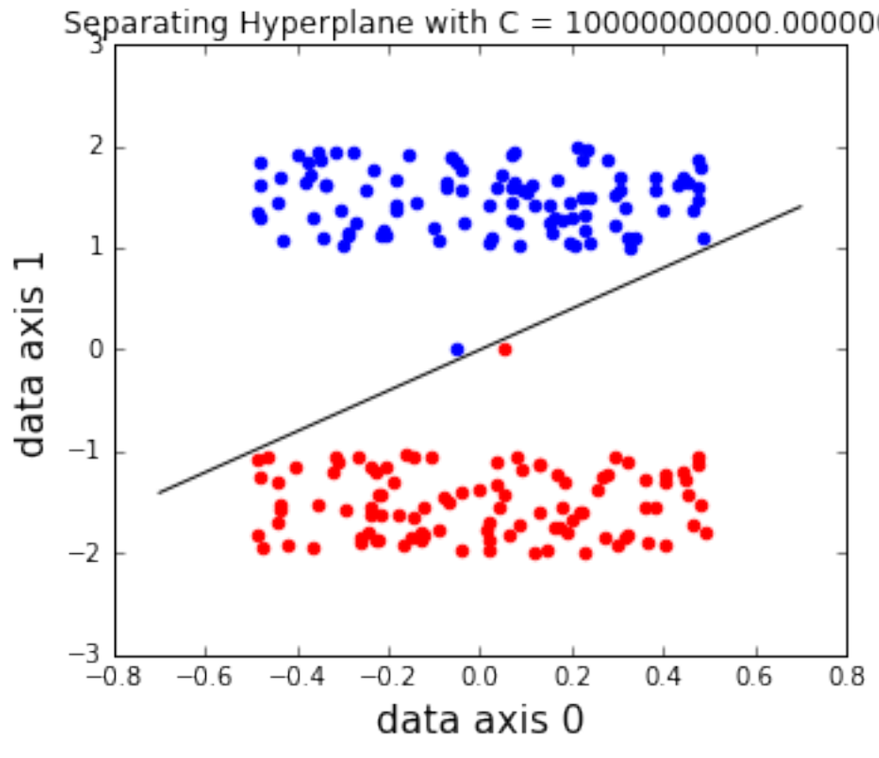
L2-regularization does matter if data **isn't** nicely separable:





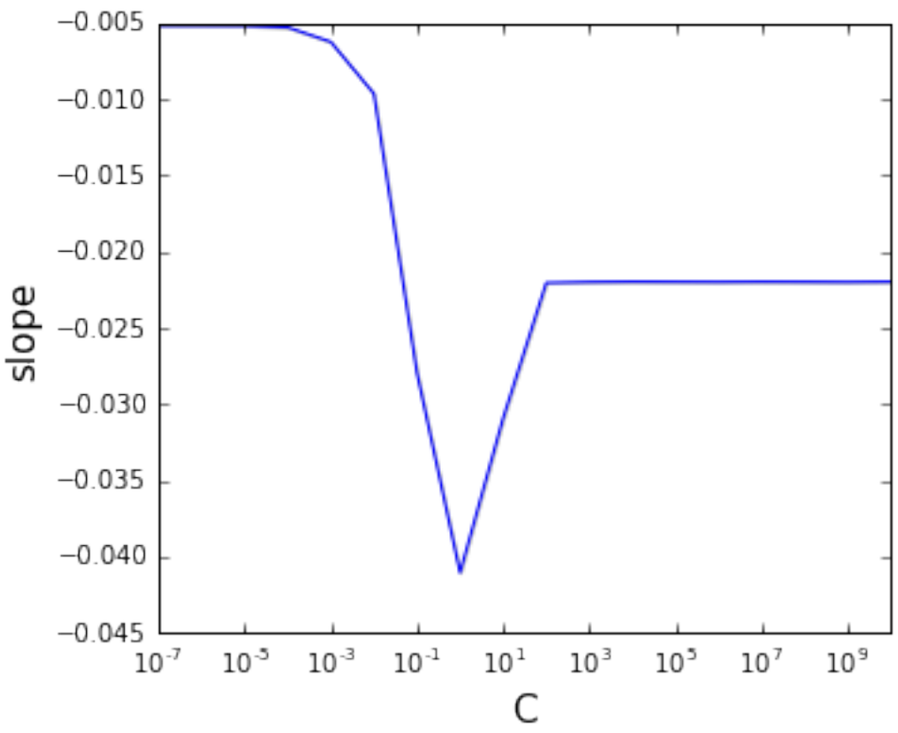
Effect of increasing l2-regularization for badly-separated data

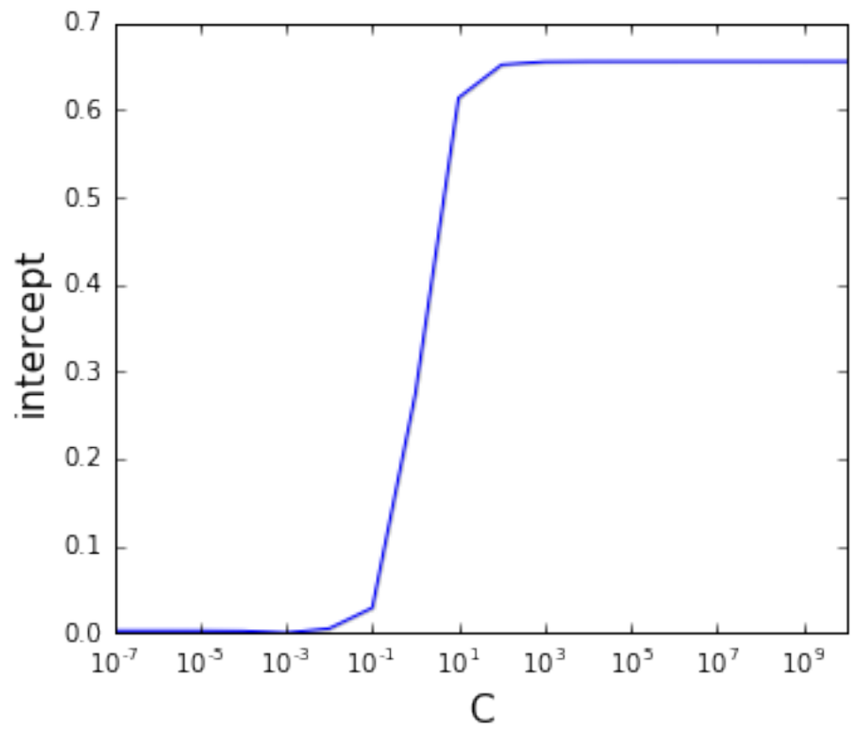




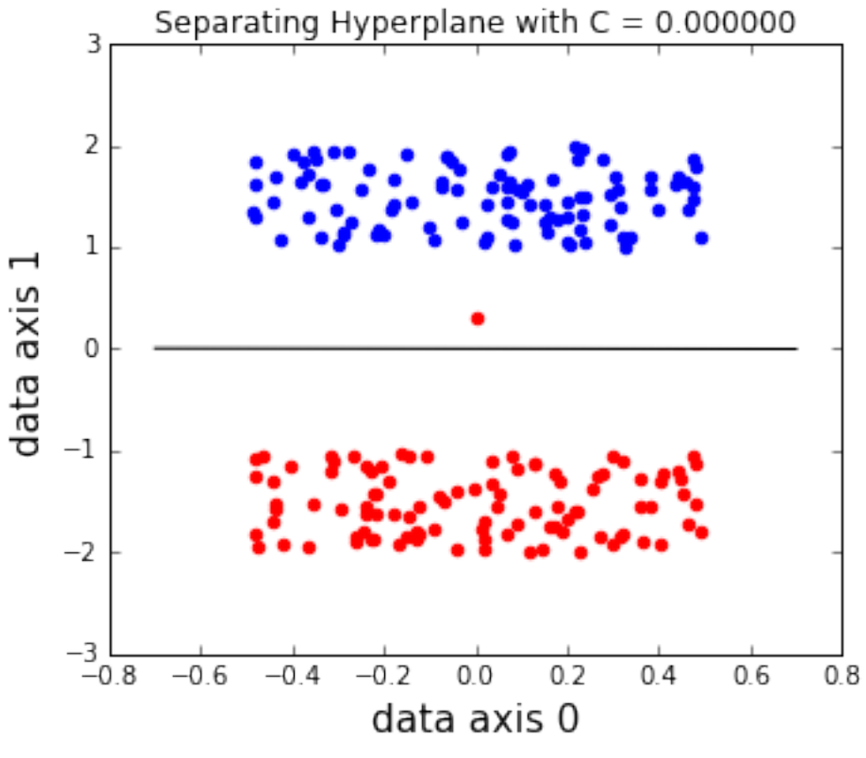
Regularization

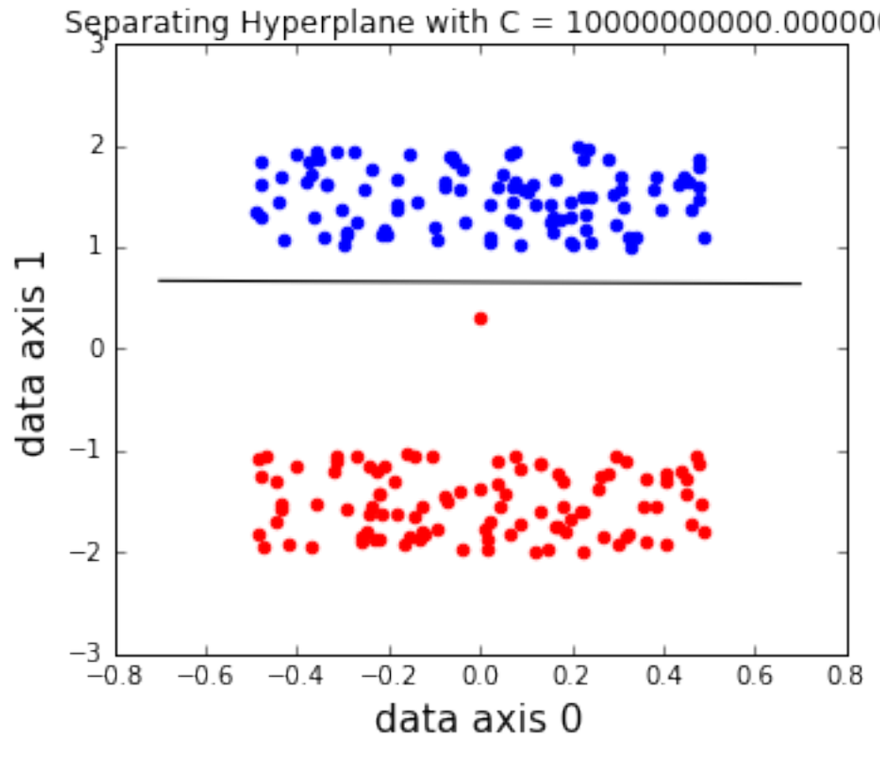
L2-regularization does matter if data **isn't** nicely separable:





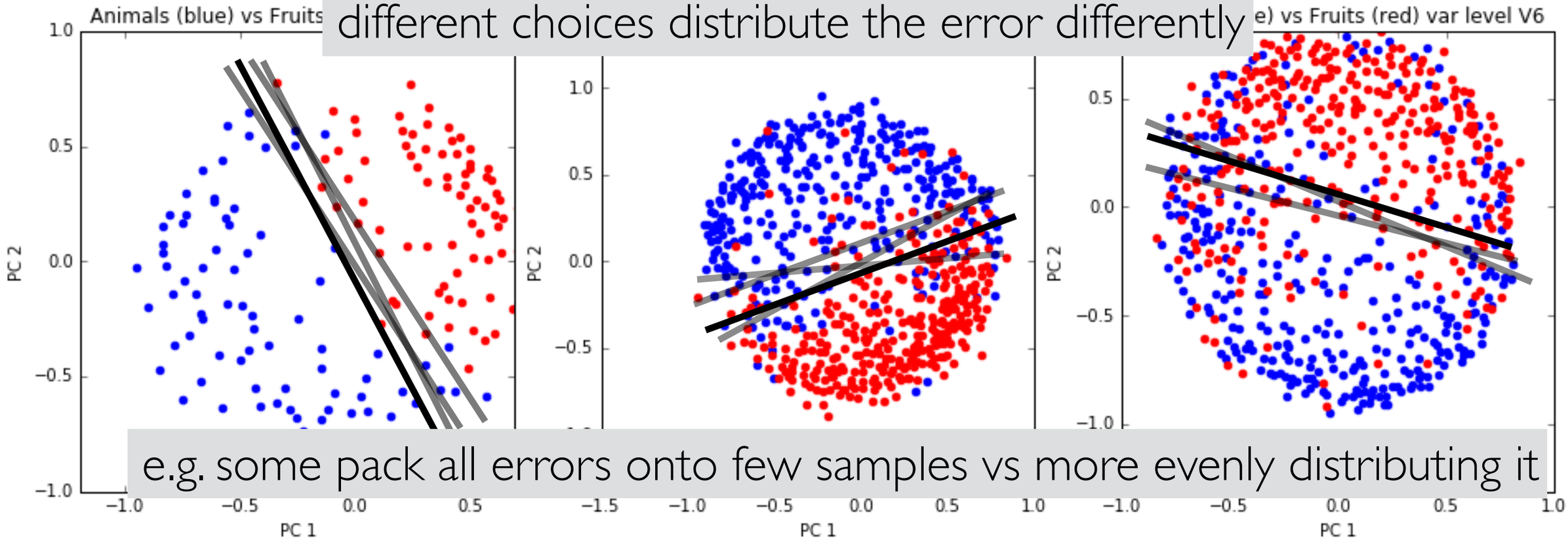
Effect of increasing l2-regularization for not-separable data





Regularization

different choices distribute the error differently

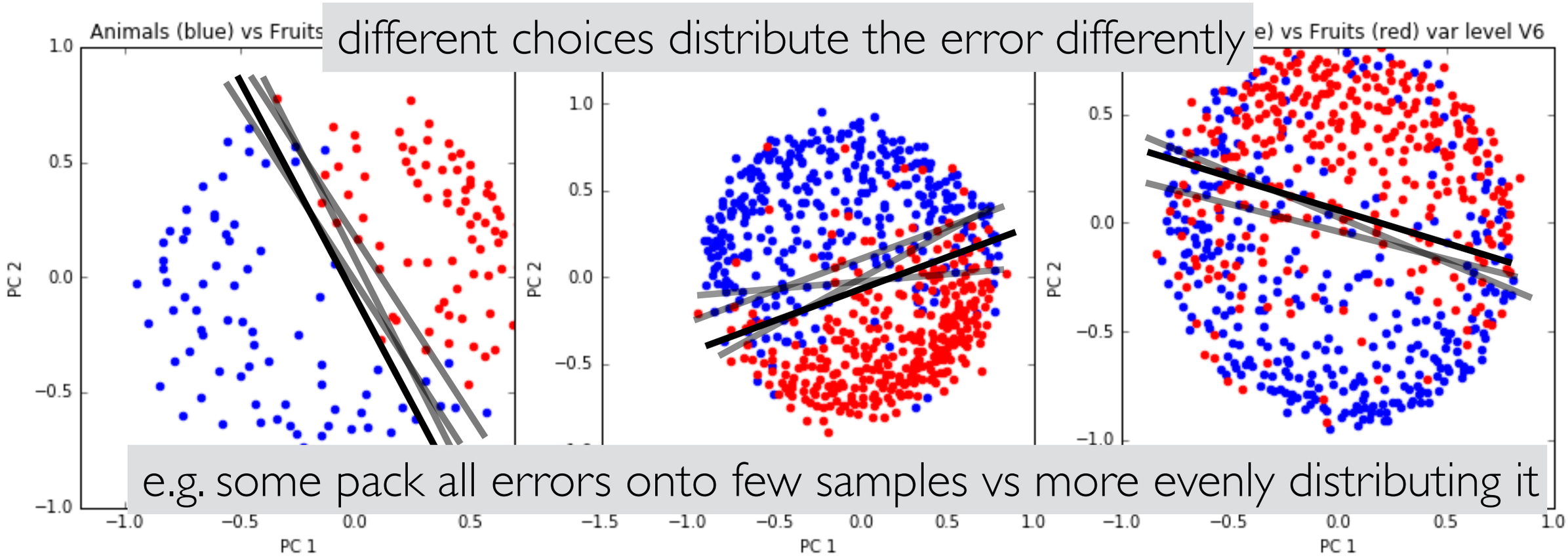


e.g. some pack all errors onto few samples vs more evenly distributing it

Hinge-loss + "L2 regularization"

$$\text{loss} = max\left(0, 1 - \text{red}_i * \left[\sum_i w_i \cdot neuron_i(s_j) + b\right]\right) + \lambda \cdot ||w||_2^2$$

Regularization



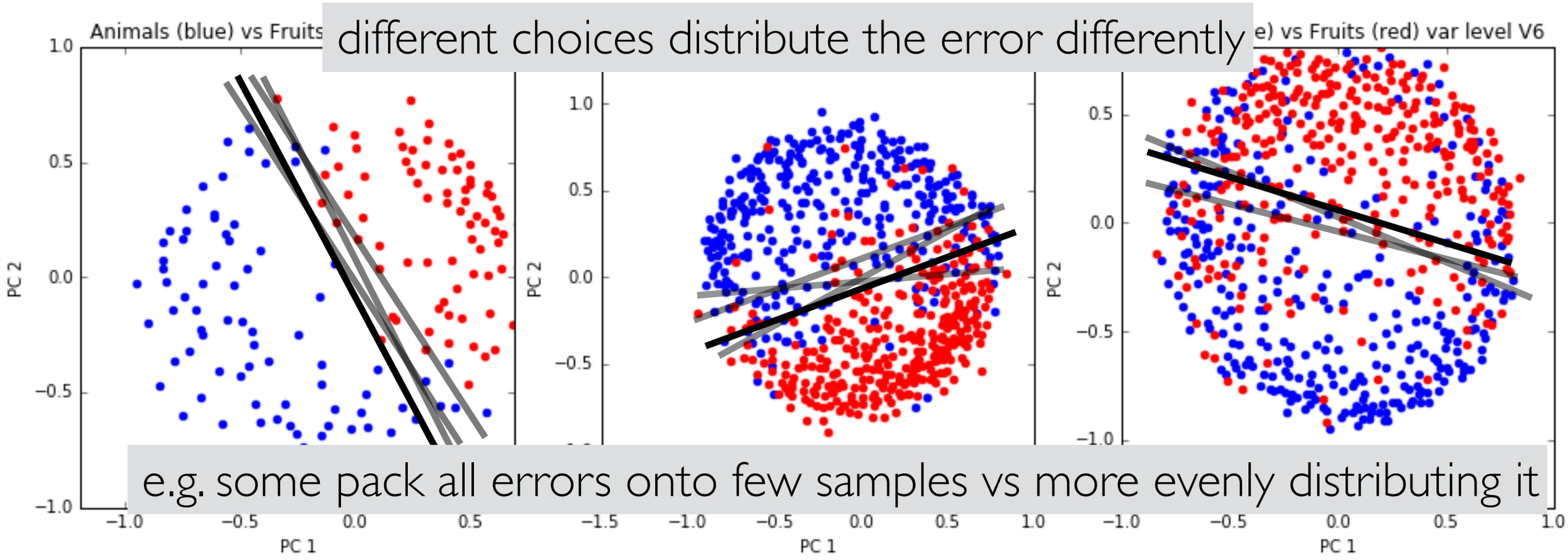
Hinge-loss + "L2 regularization"

$$\text{loss} = max\left(0, 1 - \text{red}_i * \left[\sum_i w_i \cdot neuron_i(s_j) + b\right]\right) + \lambda \cdot ||w||_2^2$$

general principle:

$$\text{loss} = \text{data-misfitting-penalty}[W, b, \text{data}] + \lambda \cdot \text{complexity-penalty}[W, b]$$

Regularization



Hinge-loss + "L1 regularization"

$$\text{loss} = max\left(0, 1 - \text{red}_i * \left[\sum_i w_i \cdot neuron_i(s_j) + b\right]\right) + \lambda \cdot ||w||_1$$

more sensitive
to equidistributed margin

general principle:

$$\text{loss} = \text{data-misfitting-penalty}[W, b, \text{data}] + \lambda \cdot \text{complexity-penalty}[W, b]$$

Regularization

$$\text{loss} = max\left(0, 1 - \text{red}_i * \left[\sum_i w_i \cdot neuron_i(s_j) + b\right]\right) + \lambda \cdot ||w||_1$$

L1 Norm:

$$||w||_1 = \frac{1}{n}\sum_{i=0}^{n-1} |w_i|$$

L2 Norm:

$$||w||_2^2 = \frac{1}{n}\sum_{i=0}^{n-1} w_i^2$$

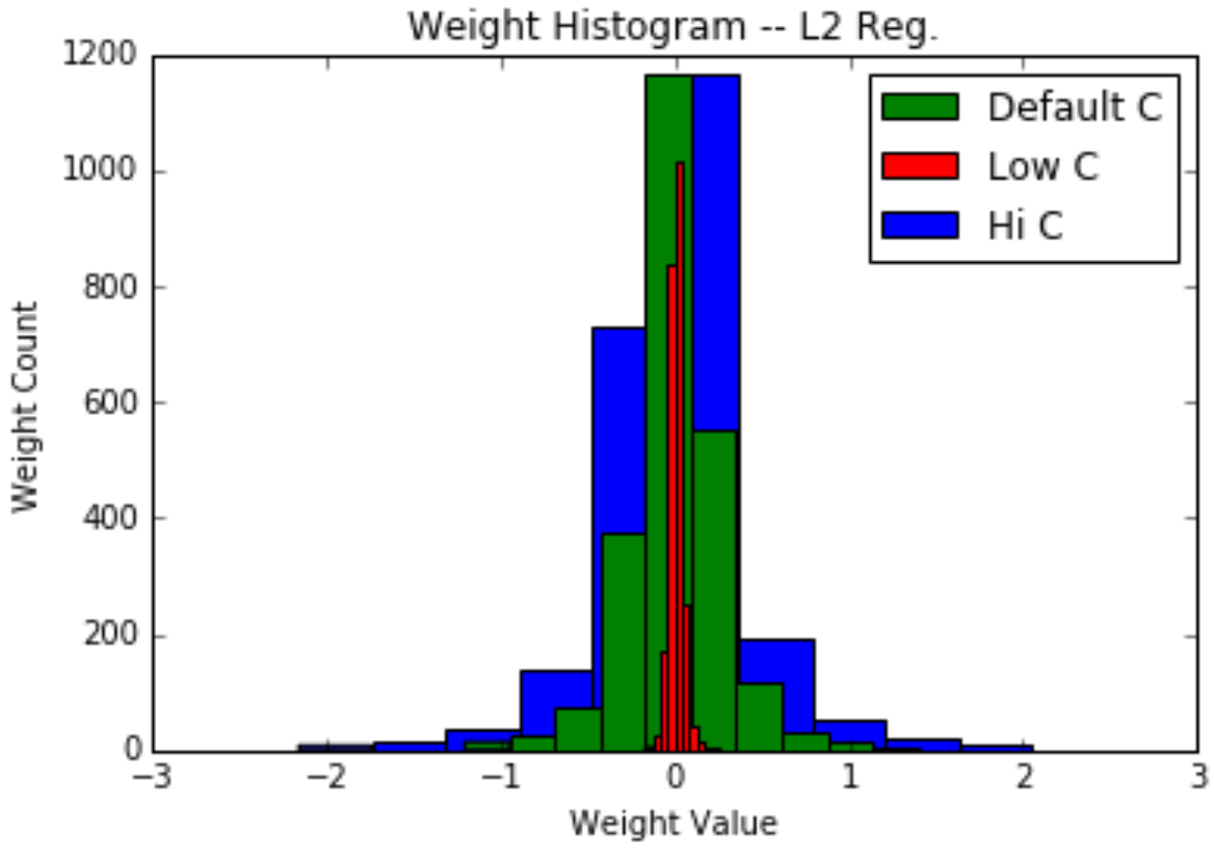
Regularization

[IPYNB: Regularization with real data]

Regularization

High C ⇒ larger weight values

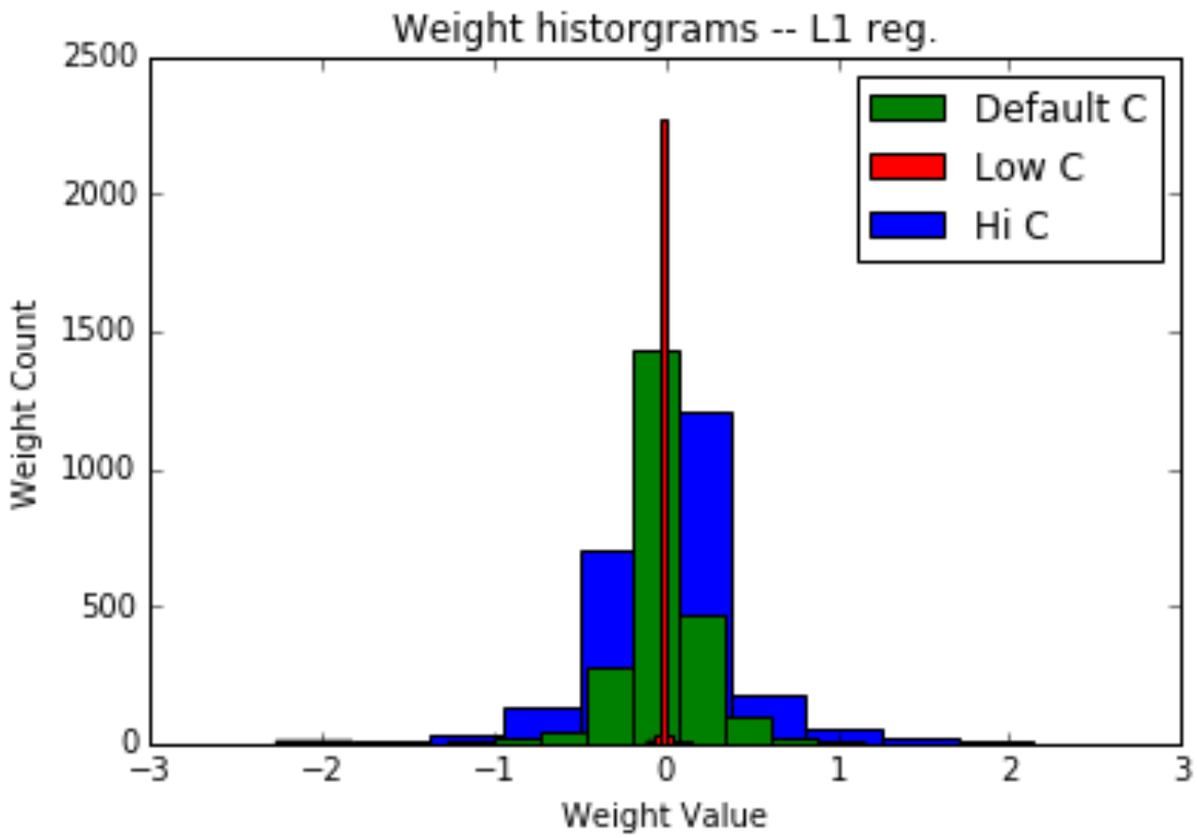
Low C ⇒ smaller weight values



. . . as expected

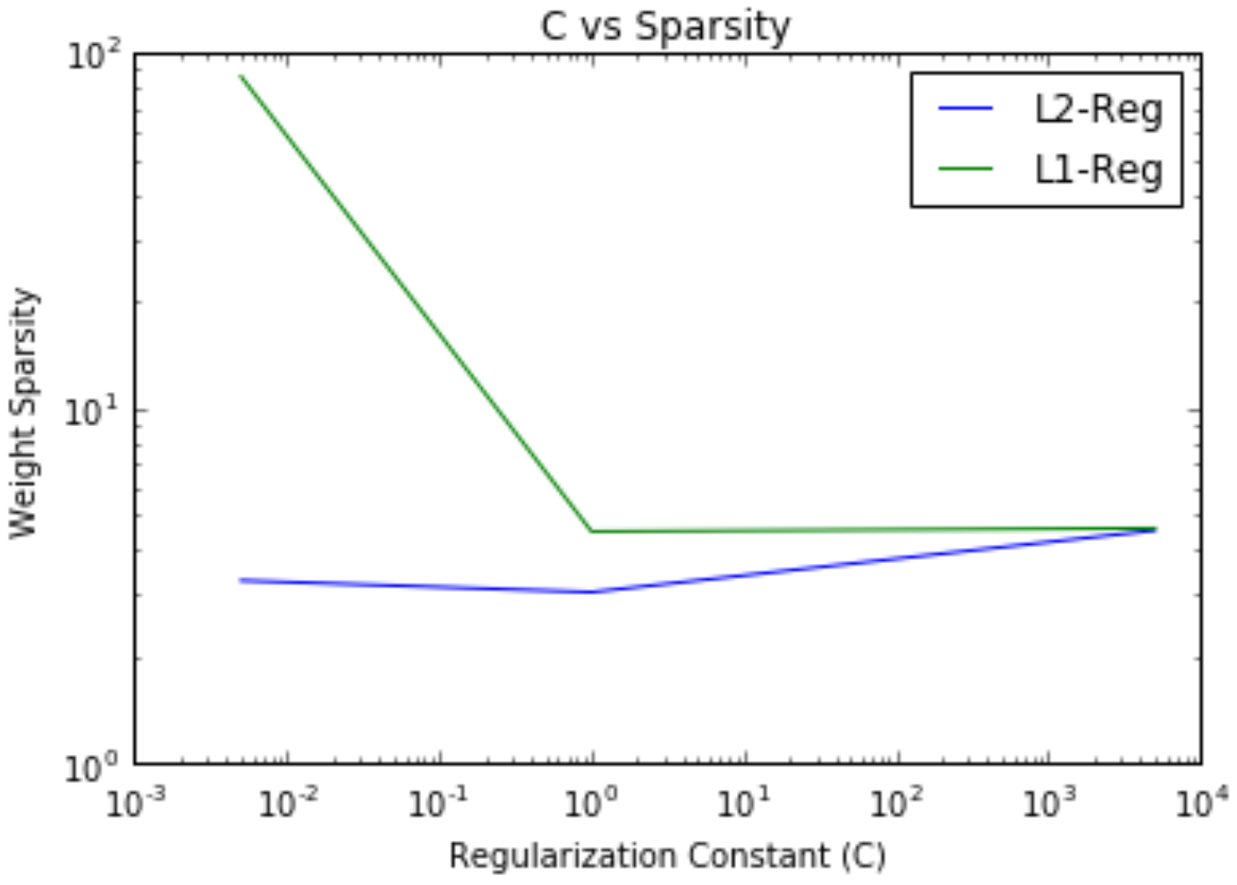
Regularization

L1 regularization leads to much sparser weights.



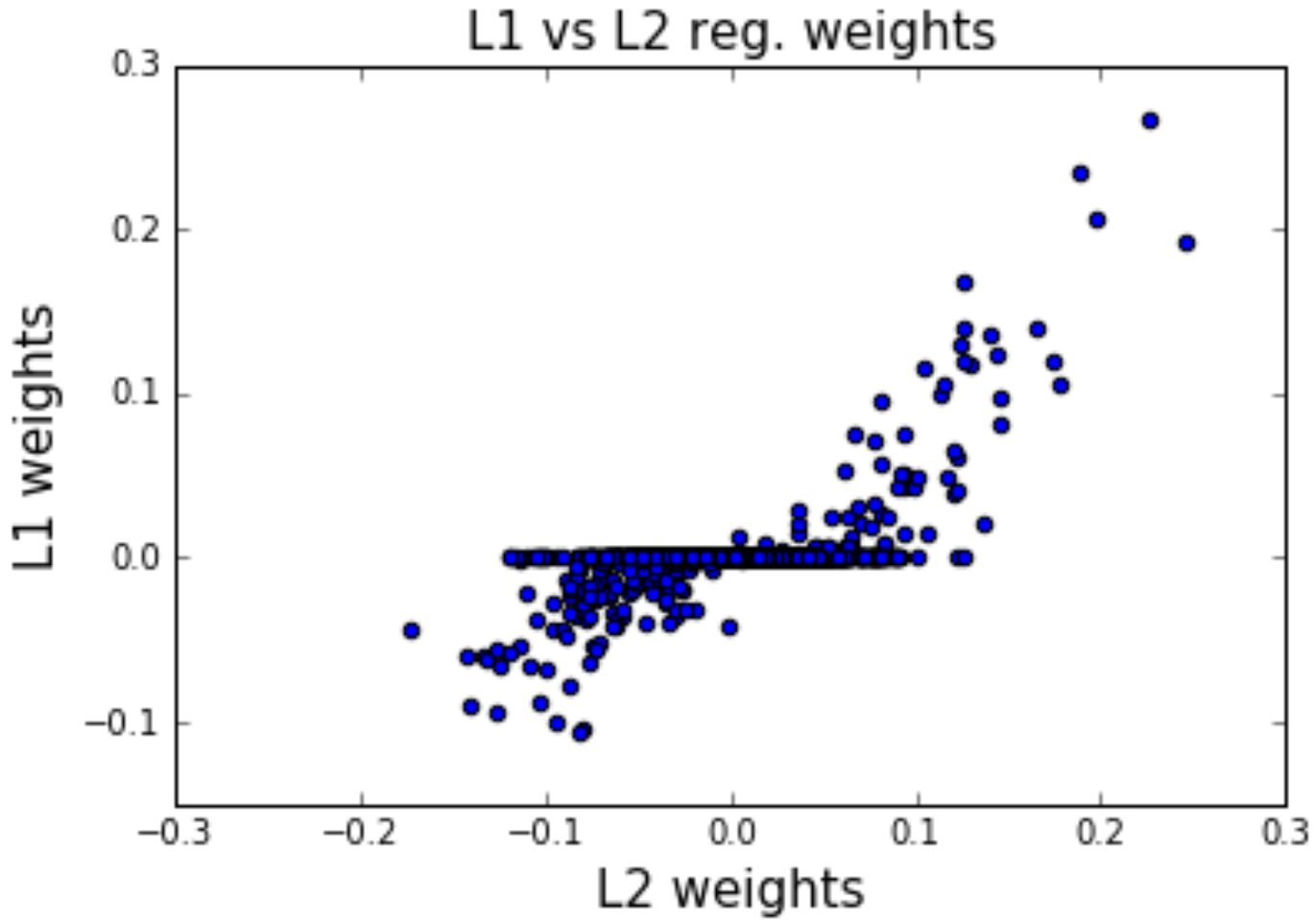
Regularization

L1 regularization leads to much sparser weights.



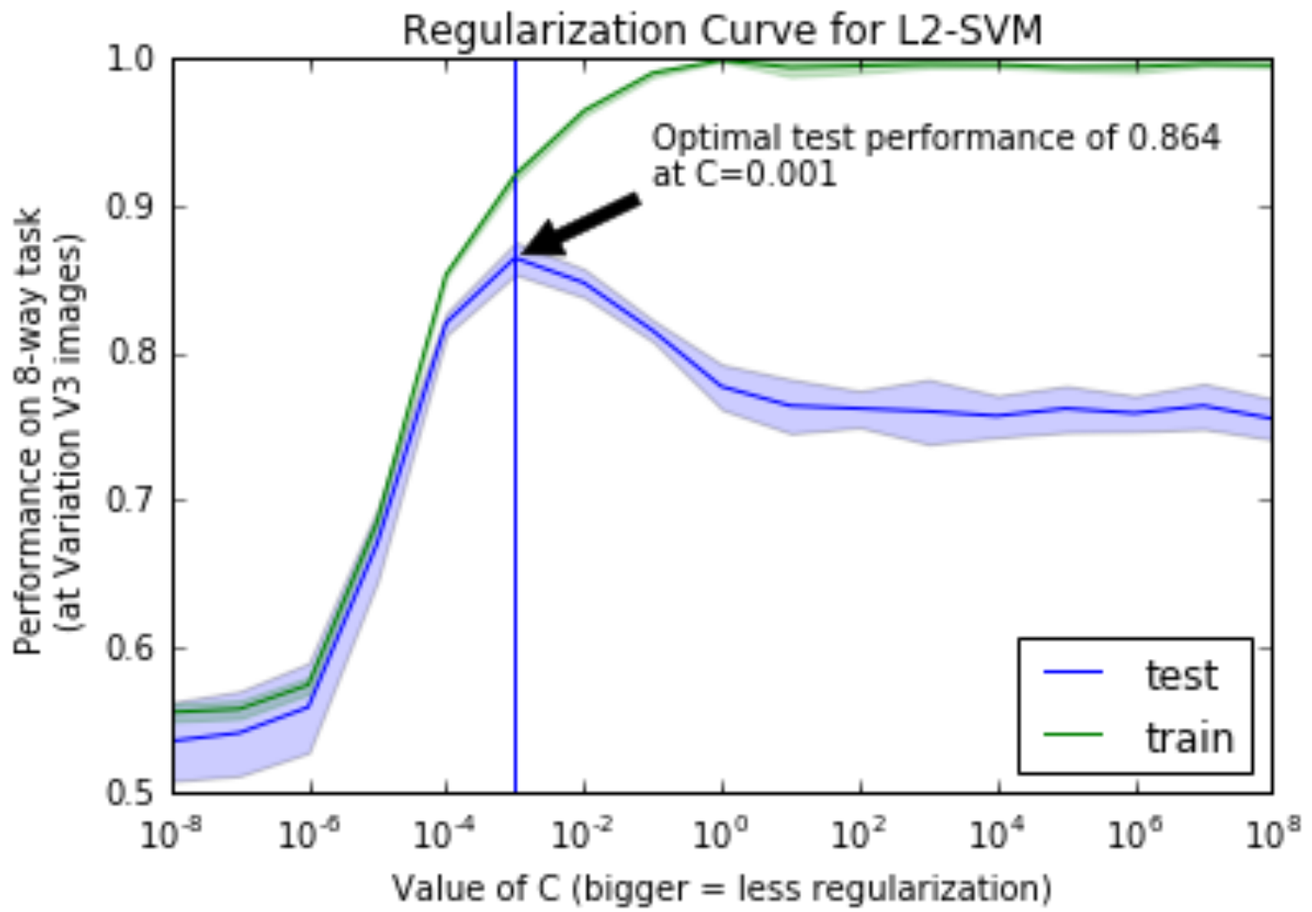
Regularization

L1 regularization leads to much sparser weights.



. . . because it encourages weights to be 0 if they're not high.
A form of **interpretable feature selection**.

Regularization



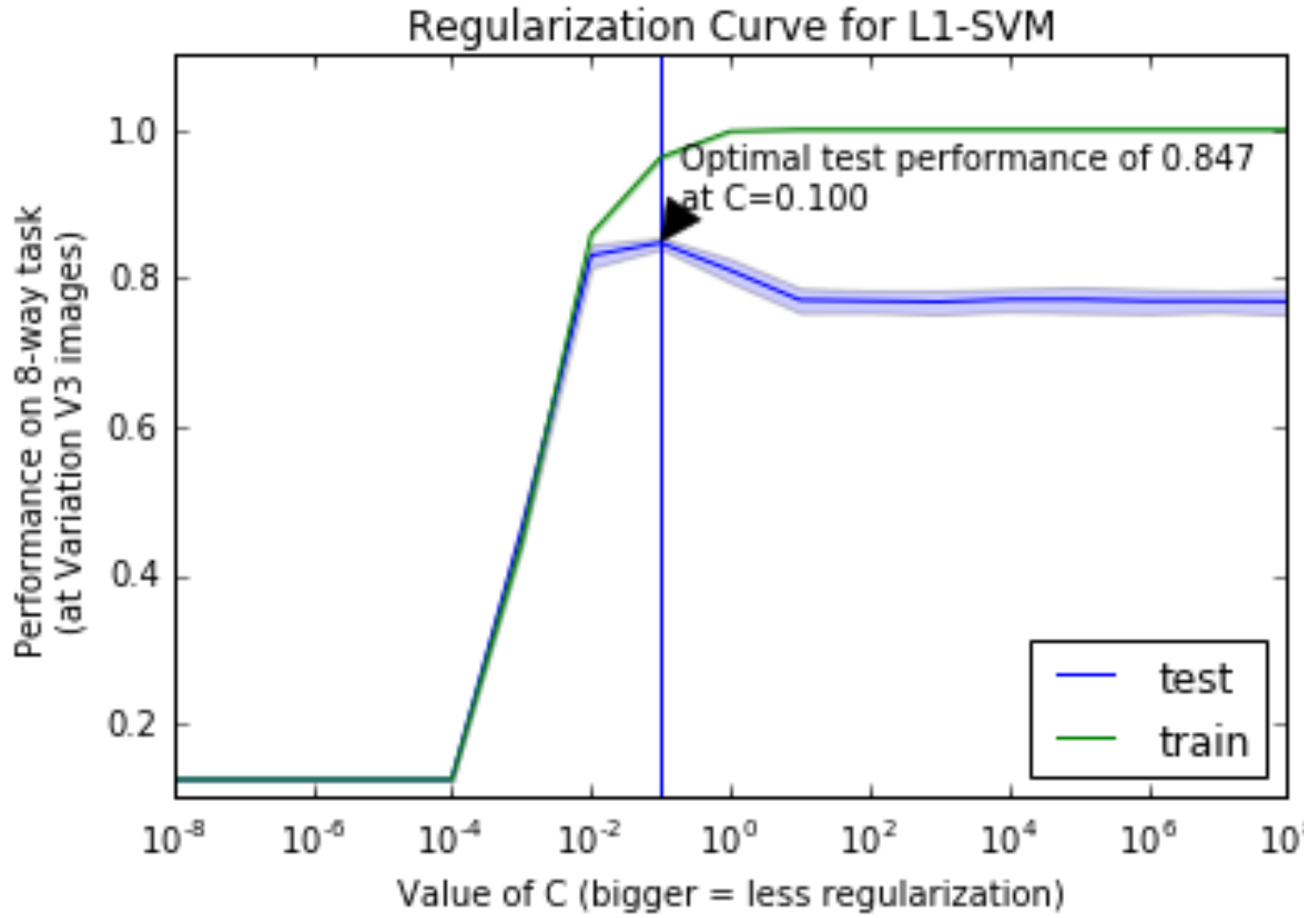
(1) training and testing are less good at very high levels of regularization (low C) than at very low levels of regularization (high C)

(2) training performance increases monotonically as regularization decreases, converging to ceiling levels at very high values of C

(3) overfitting (gap between green and blue) increases as regularization decreases

(4) **most important**: there is an ***optimal*** level of regularization for test performance

Regularization



specific to L1:

(5) performance of L1 criterion by itself (low C) is terrible

(6) optimal performance is little less good than L2 (often)

(7) overfitting is little less bad than for L2

(1) training and testing are less good at very high levels of regularization (low C) than at very low levels of regularization (high C)

(2) training performance increases monotonically as regularization decreases, converging to ceiling levels at very high values of C

(3) overfitting (gap between green and blue) increases as regularization decreases

(4) **most important**: there is an ***optimal*** level of regularization for test performance

Regularization

. . . of course when you actually do regularization you **must** cross validate your choice of **C**, since it is a free parameter of the model.

[IPYNB: Cross-Validating your regularization parameter]

Linear Regression

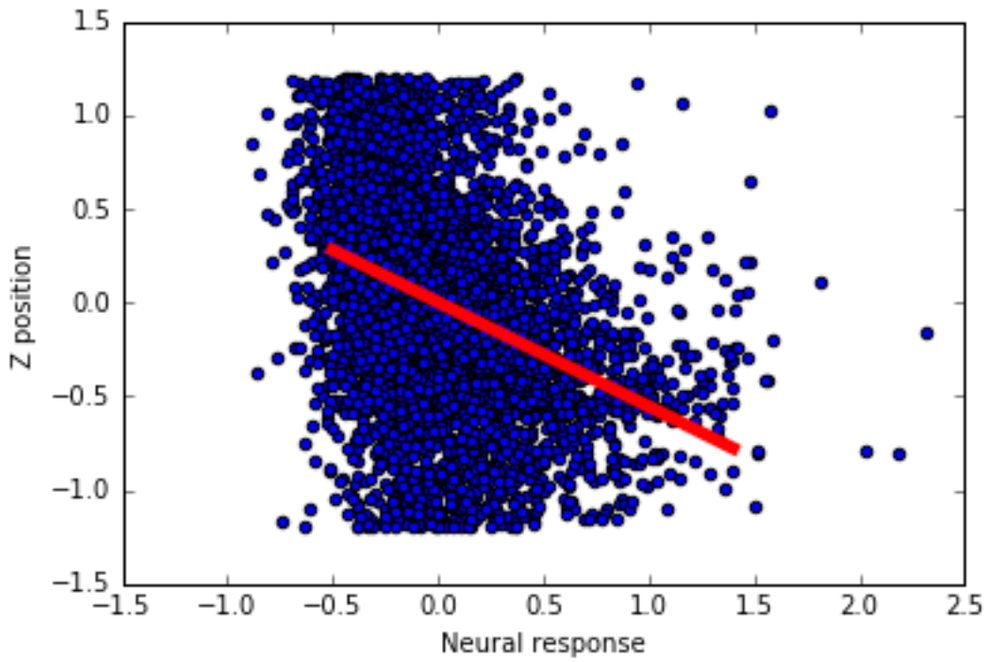
Simple linear regression is case when there's one regressor.

$$y \sim mx + b$$

Linear Regression

Simple linear regression is case when there's one regressor.

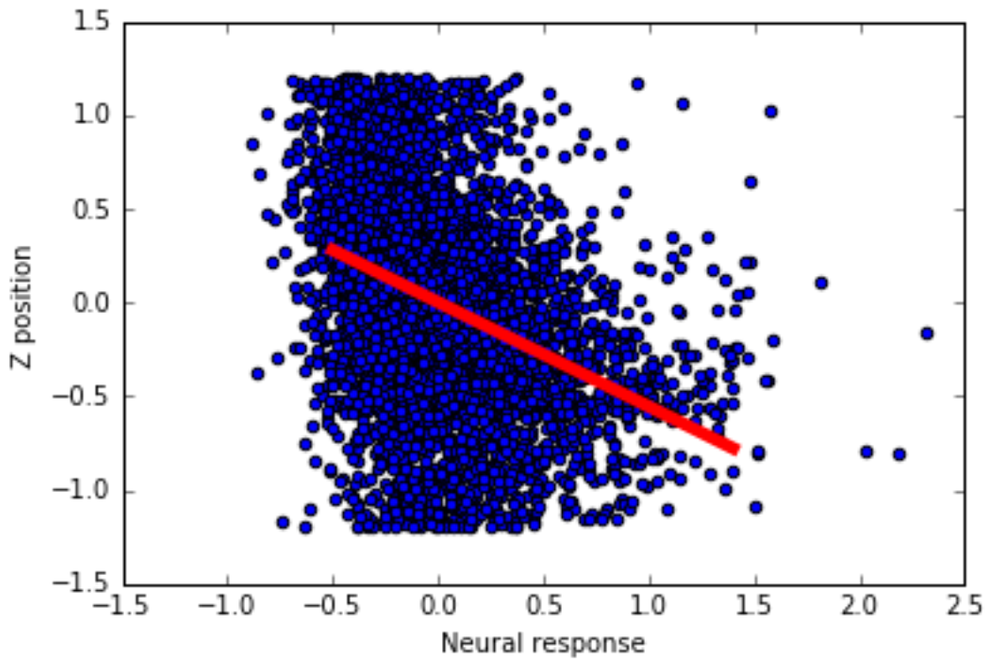
$$y \sim mx + b$$



Linear Regression

Simple linear regression is case when there's one regressor.

$$y \sim mx + b$$



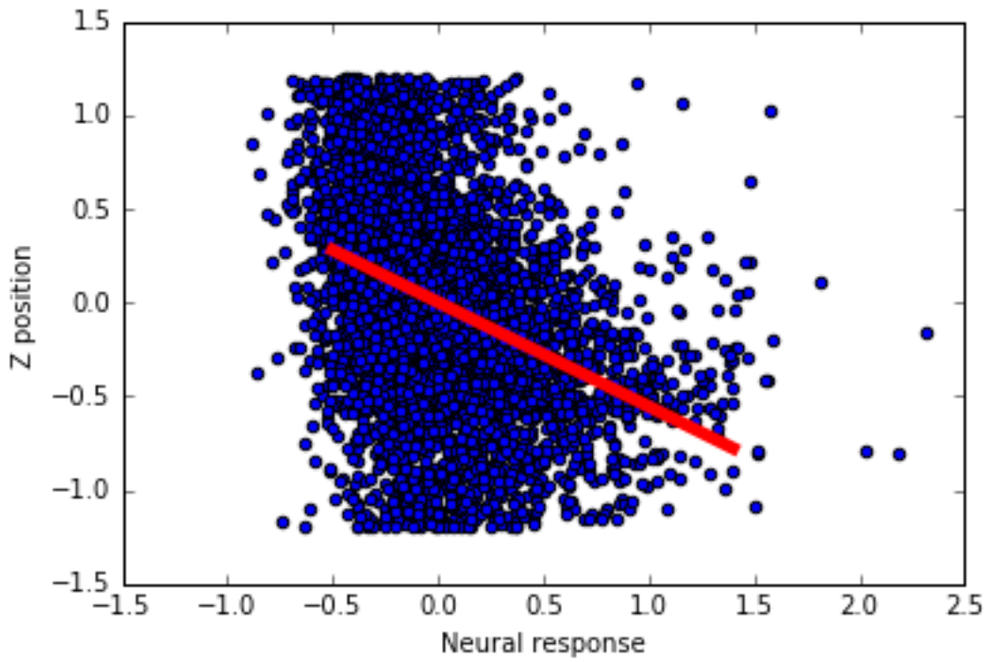
N = number of training points

Find m, b that minimize sum-of-squares error: $\sum_{i=0}^{N-1} (y_i - mx_i - b)^2$

Linear Regression

Simple linear regression is case when there's one regressor.

$$y \sim mx + b$$



N = number of training points

Find m, b that minimize sum-of-squares error: $\sum_{i=0}^{N-1} (y_i - mx_i - b)^2$

$$m = \frac{Cov(x,y)}{Var(x)} = corr(x,y) \cdot \frac{\sigma_y}{\sigma_x}$$

Linear Regression

Multivariate linear regression is very similar:

$$\mathbf{Y} = W \cdot \mathbf{X}^T + b$$

(n, m) (m, k) (n, k)

k = number of regressors

m = number of dimensions to regress

n = number of data points

… and accordingly, has a simple explicit formula solution:

Of course, actually computing the pseudo-inverse is computationally intensive.

Linear Regression

Multivariate linear regression is very similar:

$$\underset{(n,\,m)}{\mathbf{Y}} = \underset{(m,\,k)}{W} \cdot \underset{(n,\,k)}{\mathbf{X}^T} + b$$

k = number of regressors

m = number of dimensions to regress

n = number of data points

… and accordingly, has a simple explicit formula solution:

$$\underbrace{(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T}\mathbf{Y}$$

"pseudo-inverse" of X

Of course, actually computing the pseudo-inverse is computationally intensive.

Linear Regression

Multivariate linear regression is very similar:

$$\mathbf{Y} = W \cdot \mathbf{X}^T + b$$

(n, m) (m, k) (n, k)

k = number of regressors

m = number of dimensions to regress

n = number of data points

… and accordingly, has a simple explicit formula solution:

$$(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \sim \frac{cov(\mathbf{X}, \mathbf{Y})}{Var(\mathbf{X})}$$

"pseudo-inverse" of X

effectively generalizing one-d case

Of course, actually computing the pseudo-inverse is computationally intensive.

Ridge Regression

Multivariate linear regression is very similar:

$$\mathbf{Y} = W \cdot \mathbf{X}^T + b$$

(n, m) (m, k) (n, k)

k = number of regressors

m = number of dimensions to regress

n = number of data points

The goal was to minimize:

$$\sum_{i=0}^{n-1} (Y_i - x_i^T W - b)^2 = ||Y - \mathbf{X}^T \cdot W - b||$$

… and accordingly, had the simple explicit formula solution:

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

Ridge Regression

Multivariate linear regression is very similar:

$$\mathbf{Y} = W \cdot \mathbf{X}^T + b$$

(n, m) (m, k) (n, k)

k = number of regressors

m = number of dimensions to regress

n = number of data points

For reasons just like in categorization, can help prevent overfitting to minimize:

$$||Y - \mathbf{X}^T \cdot W - b|| + \alpha||W||^2$$

fitting error regularization

Ridge Regression

Multivariate linear regression is very similar:

$$\mathbf{Y} = W \cdot \mathbf{X}^T + b$$

(n, m) (m, k) (n, k)

k = number of regressors

m = number of dimensions to regress

n = number of data points

For reasons just like in categorization, can help prevent overfitting to minimize:

$$||Y - \mathbf{X}^T \cdot W - b|| + \alpha||W||^2$$

fitting error regularization

Again, has a simple explicit solution:

$$W = (\mathbf{X}^T\mathbf{X} + \alpha\mathbb{I})^{-1}\mathbf{X}^T\mathbf{Y}$$

Ridge Regression

Multivariate linear regression is very similar:

$$\mathbf{Y} = W \cdot \mathbf{X}^T + b$$

(n, m) (m, k) (n, k)

k = number of regressors

m = number of dimensions to regress

n = number of data points

For reasons just like in categorization, can help prevent overfitting to minimize:

$$||Y - \mathbf{X}^T \cdot W - b|| + \alpha||W||^2$$

fitting error regularization

Again, has a simple explicit solution:

Ridge Regression
(aka Tikhonov regression)

$$W = (\mathbf{X}^T\mathbf{X} + \alpha\mathbb{I})^{-1}\mathbf{X}^T\mathbf{Y}$$

Ridge Regression

Multivariate linear regression is very similar:

$$\mathbf{Y} = W \cdot \mathbf{X}^T + b$$

(n, m) (m, k) (n, k)

k = number of regressors

m = number of dimensions to regress

n = number of data points

For reasons just like in categorization, can help prevent overfitting to minimize:

$$||Y - \mathbf{X}^T \cdot W - b|| + \alpha||W||^2$$

fitting error

cross validating the regularization strength

Again, has a simple explicit solution:

RidgeCV Regression

(aka Tikhonov regression)

$$W = (\mathbf{X}^T\mathbf{X} + \alpha\mathbb{I})^{-1}\mathbf{X}^T\mathbf{Y}$$

Ridge Regression

$$W = (\mathbf{X}^T\mathbf{X} + \alpha\mathbb{I})^{-1}\mathbf{X}^T\mathbf{Y}$$

effectively like adding uncorrelated noise of variance α

makes 0 contribution to the numerator term

like adding a "ridge" along the diagonal of the regressor covariance matrix — that's why it's called "ridge"

Lasso Regression

Multivariate linear regression is very similar:

$$\mathbf{Y} = W \cdot \mathbf{X}^T + b$$

(n, m) (m, k) (n, k)

k = number of regressors

m = number of dimensions to regress

n = number of data points

For reasons just like in categorization, can help prevent overfitting to minimize:

$$||Y - \mathbf{X}^T \cdot W - b|| + \alpha||W||$$

fitting error regularization

Lasso Regression

Multivariate linear regression is very similar:

$$\mathbf{Y} = W \cdot \mathbf{X}^T + b$$

(n, m) (m, k) (n, k)

k = number of regressors

m = number of dimensions to regress

n = number of data points

For reasons just like in categorization, can help prevent overfitting to minimize:

$$||Y - \mathbf{X}^T \cdot W - b|| + \alpha||W||$$

fitting error regularization

Does not have a simple solution.

Lasso Regression

Lasso leads to *sparse* weights.

Lasso Regression

Multivariate linear regression is very similar:

$$\mathbf{Y} = W \cdot \mathbf{X}^T + b$$

(n, m) (m, k) (n, k)

k = number of regressors

m = number of dimensions to regress

n = number of data points

For reasons just like in categorization, can help prevent overfitting to minimize:

$$||Y - \mathbf{X}^T \cdot W - b|| + \alpha||W||$$

fitting error regularization

Does not have a simple solution.

Lasso leads to *sparse* weights.

Lasso Regression

LassoCV Regression

Lasso Regression

Multivariate linear regression is very similar:

$$\mathbf{Y} = W \cdot \mathbf{X}^T + b$$

(n, m) (m, k) (n, k)

k = number of regressors

m = number of dimensions to regress

n = number of data points

For reasons just like in categorization, can help prevent overfitting to minimize:

$$||Y - \mathbf{X}^T \cdot W - b|| + \alpha||W||$$

fitting error regularization

Does not have a simple solution.

Lasso leads to *sparse* weights.

which matters if you care about interpretability of coefficients — not so relevant in neuroscience, but relevant in (e.g.) psychology or economics

Lasso Regression

LassoCV Regression

ElasticNet Regression

Multivariate linear regression is very similar:

$$\mathbf{Y} = W \cdot \mathbf{X}^T + b$$

(n, m) (m, k) (n, k)

k = number of regressors

m = number of dimensions to regress

n = number of data points

For reasons just like in categorization, can help prevent overfitting to minimize:

$$||Y - \mathbf{X}^T \cdot W - b|| + \alpha||W||^2 + \beta||W||$$

fitting error two regularization terms

Potential advantage:
Can be as good at fitting as Ridge while still allowing some sparse interpretability

ElasticNet regression

ElasticNetCV regression

Kernel Ridge Regression

Multivariate linear regression is very similar:

$$\mathbf{Y} = W \cdot \mathbf{X}^T + b$$

(n, m) (m, k) (n, k)

k = number of regressors

m = number of dimensions to regress

n = number of data points

In Ridge Regression we had:

$$||Y - \mathbf{X}^T \cdot W - b|| + \alpha ||W||^2$$

Kernel Ridge Regression

Multivariate linear regression is very similar:

$$\mathbf{Y} = W \cdot \mathbf{X}^T + b$$

(n, m) (m, k) (n, k)

k = number of regressors

m = number of dimensions to regress

n = number of data points

In Ridge Regression we had — equivalently,

$$d(Y, \mathbf{X}^T \cdot W + b) + \alpha ||W||^2$$

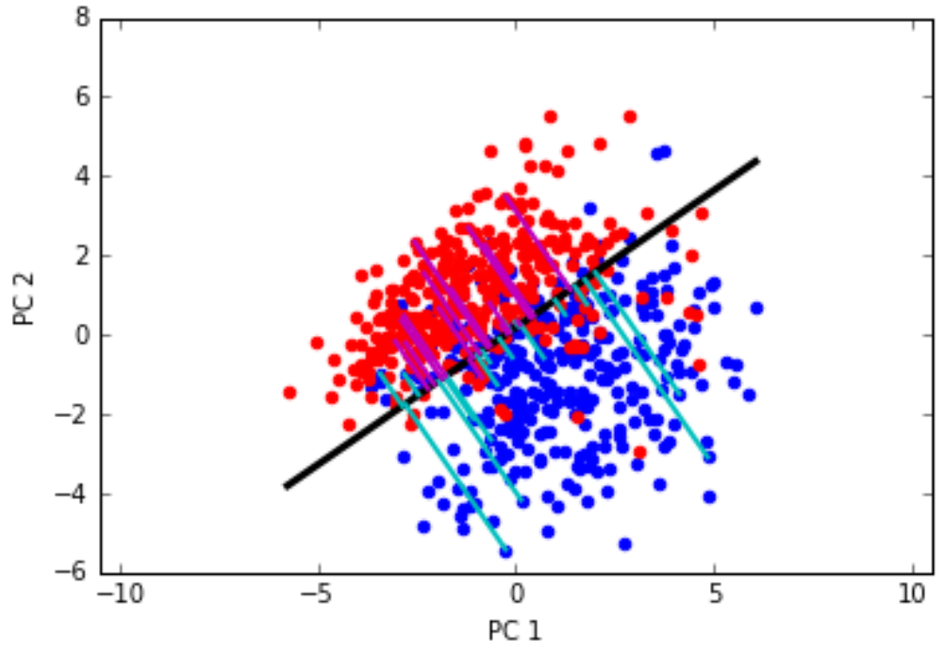
where d = euclidean distance

Recall — Other Distances in the SVM Classifier (lecture 5)

Recall out hyperplane-based classifiers are based on

$$\text{margins}(s_j) = \sum_i w_i \cdot \text{neuron}_i(s_j) + b$$

Results only depend on matrix of *pairwise similarities* (inverse distances) between training points (s_j's).*



Regular SVM uses euclidean similiarity.

`sklearn.svm.LinearSVC`

only works with regular euclidean similarity

→

Can actually use *any* similarity measure you want.

`sklearn.svm.SVC`

you can input any similarity function, even one that's *nonlinear* fn of its inputs

Most common is the "radial basis function" (RBF) similarity:

$$\mathbf{RBF}_\gamma(s_j, s_k) = e^{-\gamma||s_j - s_k||^2}$$

* matrix of pairwise similarities is sometimes called a "kernel" or a "gram matrix"

Kernel Ridge Regression

Multivariate linear regression is very similar:

$$\mathbf{Y} = W \cdot \mathbf{X}^T + b$$

(n, m) (m, k) (n, k)

k = number of regressors

m = number of dimensions to regress

n = number of data points

In Kernel Ridge Regression we have

$$d(Y, \mathbf{X}^T \cdot W + b) + \alpha ||W||^2$$

where, for example:

$$d_\gamma(x, y) = 1 - e^{-\gamma ||x-y||^2}$$

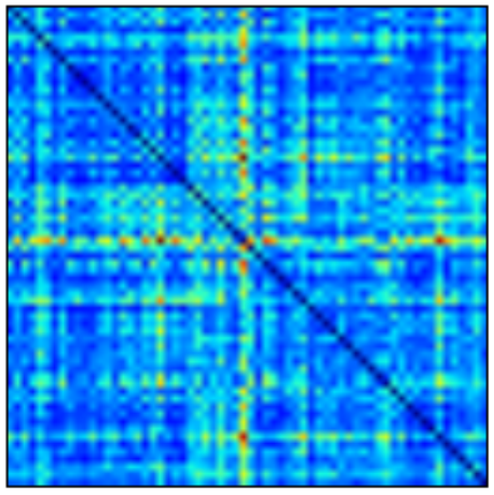
Linear Regression

[IPYNB: Multivariate regression]

Linking Hypotheses

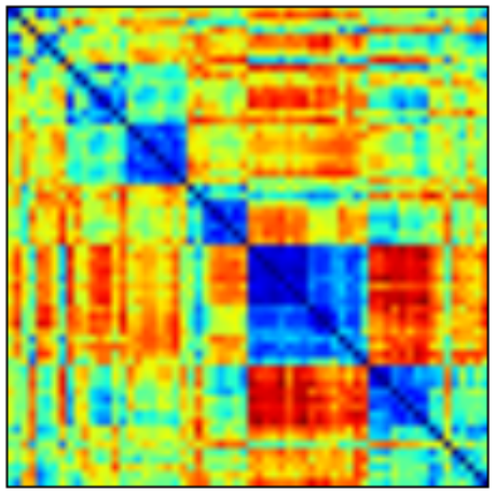
Models are linking hypotheses:

Pixel representation



3.
encoding

IT neural representation



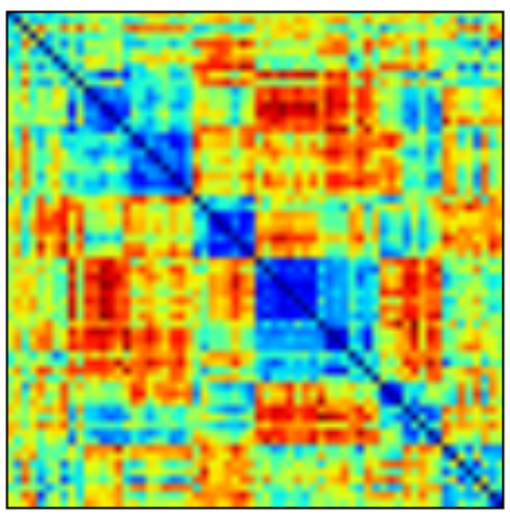
1.
decoding

categorization

Behavioral patterns

"animal"
"boat"
"car"
.
.
.

2.
reliability

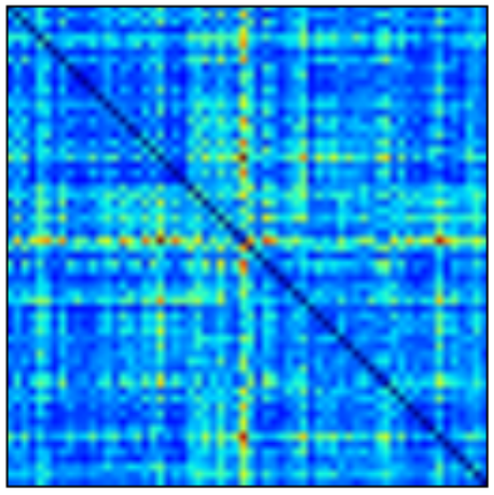


IT representation, animal 2

Linking Hypotheses

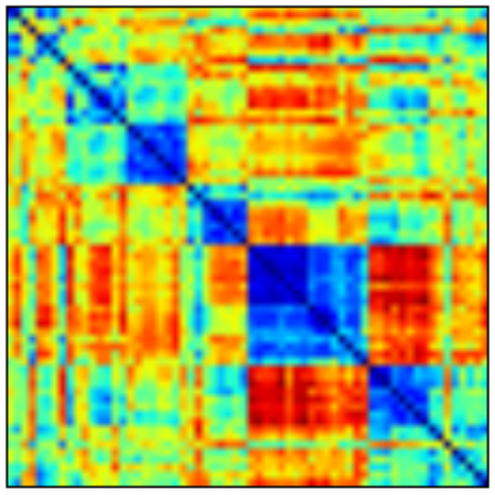
Models are linking hypotheses:

Pixel representation



3.
encoding

IT neural representation



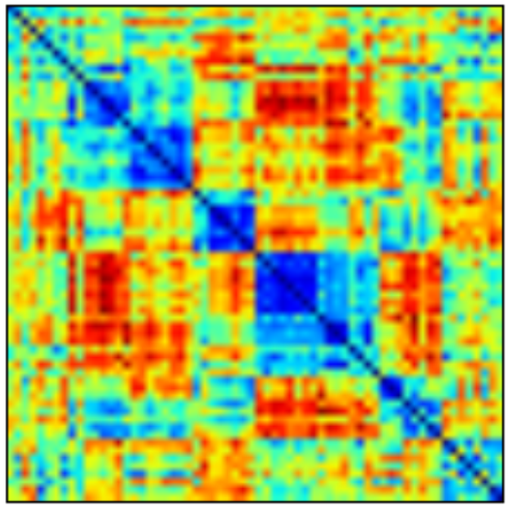
1.
decoding

regression

Behavioral patterns

x-position
y-position
size
rotation
mass
.
.
.

2.
reliability

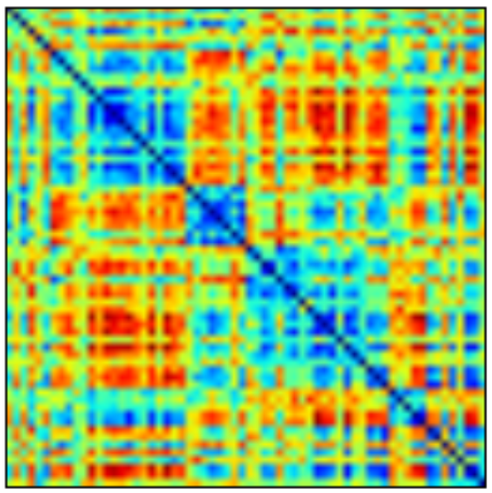


IT representation, animal 2

Linking Hypotheses

Models are linking hypotheses:

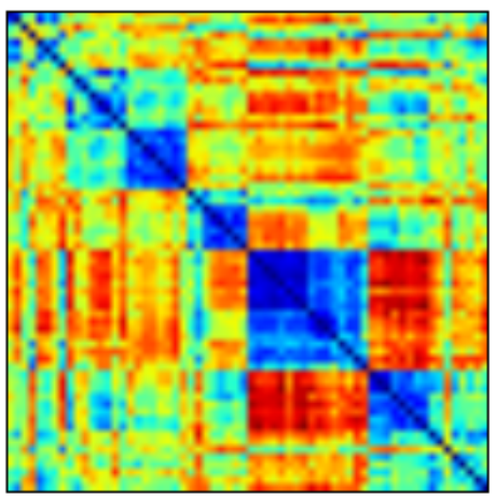
V4 representation



3. encoding

regression

IT neural representation

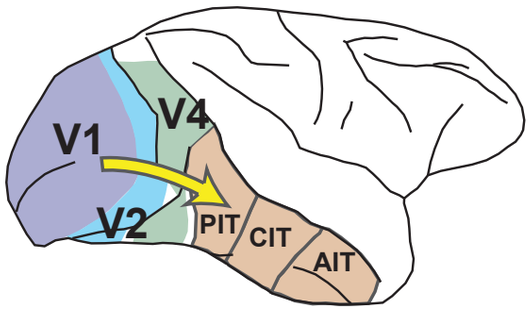


1. decoding

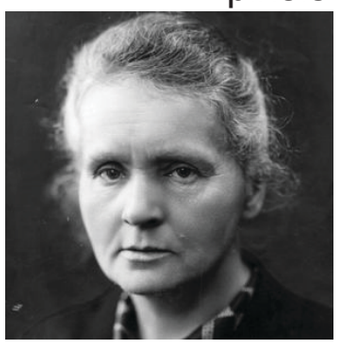
regression

Behavioral patterns

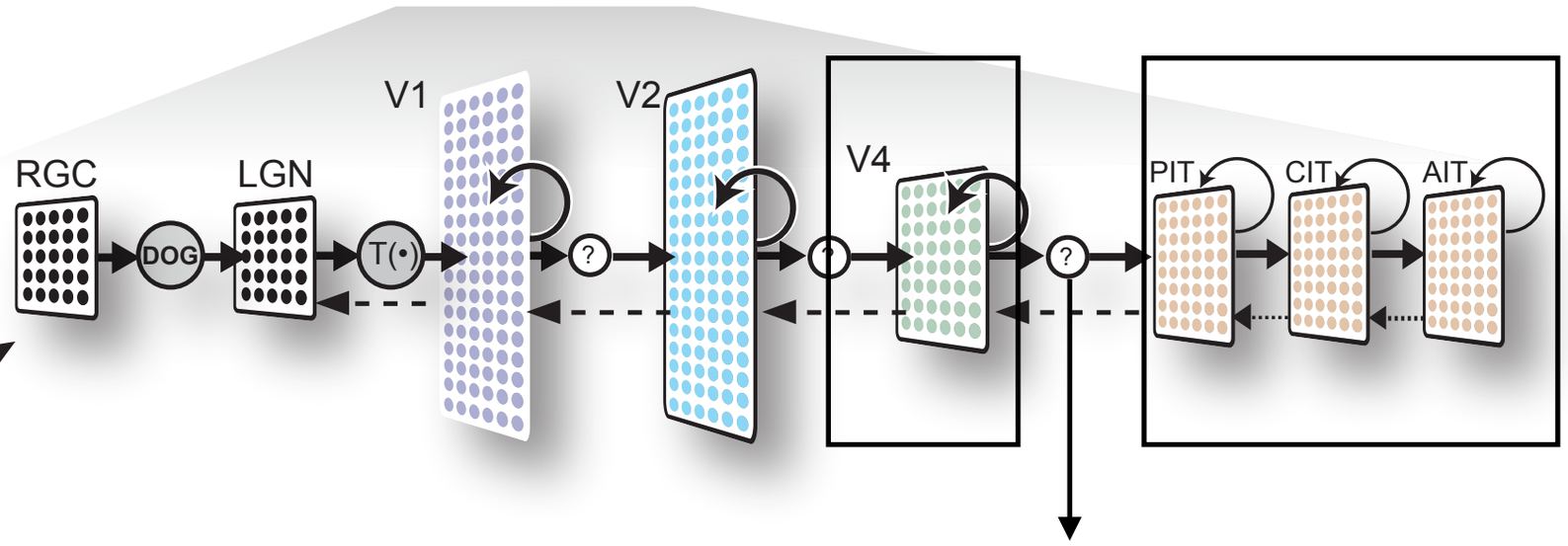
x-position
y-position
size
rotation
mass
.
.
.



pixels



100ms
Visual
Presentation



Linear transform?

Linear Regression

[IPYNB: Regressing IT from V4]

PLS Regression

Partial Least Squares (PLS) Regression

$$\mathbf{X} = TP^T \qquad \mathbf{Y} = UQ^T$$

(n, k) (n, m)

k = number of regressors

m = number of dimensions to regress

n = number of data points

PLS Regression

Partial Least Squares (PLS) Regression

k = number of regressors

$$\mathbf{X} = TP^T \qquad \mathbf{Y} = UQ^T$$

(n, k) (n, m)

n = number of data points

T (n, L) matrix

U (n, L) matrix

PLS Regression

Partial Least Squares (PLS) Regression

k = number of regressors

m = number of dimensions to regress

n = number of data points

$$\mathbf{X} = TP^T \qquad \mathbf{Y} = UQ^T$$

(n, k) (n, m)

T (n, L) matrix

U (n, L) matrix

P (k, L) orthogonal matrix

Q (m, L) orthogonal matrix

PLS Regression

Partial Least Squares (PLS) Regression

k = number of regressors

m = number of dimensions to regress

n = number of data points

$$\mathbf{X} = TP^T \qquad \mathbf{Y} = UQ^T$$

(n, k) (n, m)

T	(n, L) matrix	P	(k, L) orthogonal matrix
U	(n, L) matrix	Q	(m, L) orthogonal matrix

goal is to choose T, U, P and Q to maximimize: $corr(T, U)$

PLS Regression

Partial Least Squares (PLS) Regression

k = number of regressors

m = number of dimensions to regress

n = number of data points

$$\mathbf{X} = TP^T \qquad \mathbf{Y} = UQ^T$$

(n, k) (n, m)

T	(n, L) matrix	P	(k, L) orthogonal matrix
U	(n, L) matrix	Q	(m, L) orthogonal matrix

"trying to put X and Y in a common space"

goal is to choose T, U, P and Q to maximimize: $corr(T, U)$

PLS Regression

Partial Least Squares (PLS) Regression

k = number of regressors

m = number of dimensions to regress

n = number of data points

$$\mathbf{X} = TP^T + E \qquad \mathbf{Y} = UQ^T + F$$

(n, k) (n, m)

T (n, L) matrix P (k, L) orthogonal matrix

U (n, L) matrix Q (m, L) orthogonal matrix

"trying to put X and Y in a common space"

goal is to choose T, U, P and Q to maximimize: $corr(T, U)$

…. and minimize residuals E and F.

PLS Regression

Partial Least Squares (PLS) Regression

k = number of regressors

m = number of dimensions to regress

n = number of data points

$$\underset{(n,k)}{\mathbf{X}} = TP^T + E \qquad \underset{(n,m)}{\mathbf{Y}} = UQ^T + F$$

T (n, L) matrix

P (k, L) orthogonal matrix

U (n, L) matrix

Q (m, L) orthogonal matrix

"trying to put X and Y in a common space"

goal is to choose T, U, P and Q to maximimize: $corr(T, U)$

…. and minimize residuals E and F.

L = how many common dimensions

has to be cross-validated

PLS Regression

Partial Least Squares (PLS) Regression

k = number of regressors

m = number of dimensions to regress

n = number of data points

$$\mathbf{X} = TP^T + E \qquad \mathbf{Y} = UQ^T + F$$

(n, k) (n, m)

T (n, L) matrix P (k, L) orthogonal matrix

U (n, L) matrix Q (m, L) orthogonal matrix

"trying to put X and Y in a common space"

goal is to choose T, U, P and Q to maximimize: $corr(T, U)$

…. and minimize residuals E and F. $\mathbf{Y}$

L = how many common dimensions

has to be cross-validated

PLS Regression

Partial Least Squares (PLS) Regression

k = number of regressors

m = number of dimensions to regress

n = number of data points

$$\underset{(n,k)}{\mathbf{X}} = TP^T + E \qquad \underset{(n,m)}{\mathbf{Y}} = UQ^T + F$$

T (n, L) matrix — P (k, L) orthogonal matrix

U (n, L) matrix — Q (m, L) orthogonal matrix

"trying to put X and Y in a common space"

goal is to choose T, U, P and Q to maximimize: $corr(T, U)$

…. and minimize residuals E and F. $\mathbf{Y} \sim \mathbf{X}Q^T(P^T)^{-1}$

L = how many common dimensions

has to be cross-validated

PLS Regression

Partial Least Squares (PLS) Regression

k = number of regressors

m = number of dimensions to regress

n = number of data points

$$\mathbf{X} = TP^T + E \qquad \mathbf{Y} = UQ^T + F$$

(n, k) (n, m)

T (n, L) matrix P (k, L) orthogonal matrix

U (n, L) matrix Q (m, L) orthogonal matrix

"trying to put X and Y in a common space"

goal is to choose T, U, P and Q to maximimize: $corr(T, U)$

…. and minimize residuals E and F.

$$\mathbf{Y} \sim \mathbf{X}Q^T(P^T)^{-1}$$

the effective regression weight matrix — really needs pseudo-inverse of P^T

L = how many common dimensions

has to be cross-validated

PLS Regression

[IPYNB: PLS Regression]