LINEAR REGRESSION IV

11.11.2020

RECAP

* receptive fields

  * what kind of stuff in the world does a neuron (or voxel, or whatever) respond to?

  * we can estimate receptive fields by measuring responses in different conditions, and then fitting a model using **regression**

RECAP

* we need to find weights that minimize **squared error**

$$Err(\beta) = \sum_{t=1}^{T} (y_t - x_t \beta)^2$$

* one method is **gradient descent**

* another is the **analytic solution (np.linalg.lstsq)**

RECAP

* how do we measure *how good* a regression model is?

* ***R^2*** aka the **coefficient of determination** or **variance explained**

* ***R^2*** = 1 - (RSS / TSS)

    * RSS is “residual sum of squares” (=**squared error**)

    * TSS is “total sum of squares” (error with beta=0)

EVALUATING REGRESSION MODELS

* suppose that we are given a matrix of variables (aka regressors) **X**, and a vector of outputs **Y**

* we fit a linear model **Y_hat = X . beta**

* then we evaluate it by computing $\mathbf{R^2}$ using **X** and **Y**

* what are the possible values of $\mathbf{R^2}$?

IN-SET VS. OUT-OF-SET EVALUATION

* evaluating a regression model using the same data that we used to train/estimate/fit it is called *in-set evaluation*

* in-set evaluation is biased *upward*, and the amount of bias depends on the number of regressors in the model

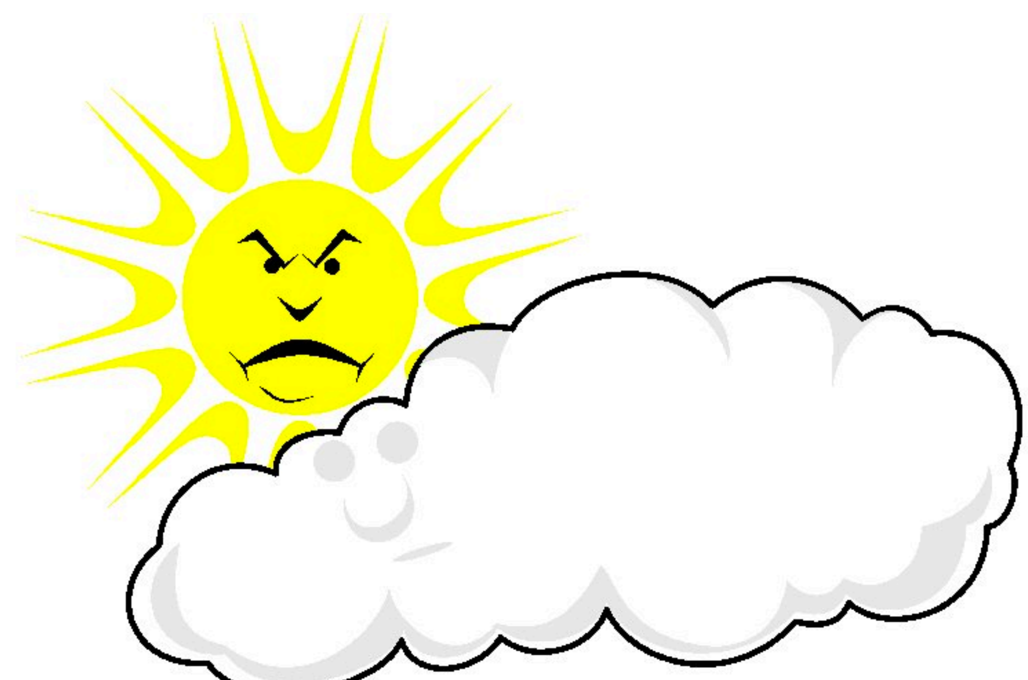
IN-SET VS. OUT-OF-SET EVALUATION

* for example: suppose we have *N* data points and *N* regressors that are pure noise—they have no relationship to the output whatsoever

* in-set variance explained is ***EXACTLY 1.0***

* *THE MODEL IS PERFECT*

* **  **



IN-SET VS. OUT-OF-SET EVALUATION

* instead, what if you split up your X and Y into “training” and “test” sets?

* you could fit your regression model using (X_trn, Y_trn), and then test how well it works on (X_test, Y_test)!

* is $\mathbf{R}^2$ biased in this case? What possible values can it take?

REGRESSION STABILITY

* so ordinary least squares regression has a problem:

* if two regressors are *very similar* (i.e. correlated), then there are many weight combinations that would give ~the same answer!

* which of these combinations is “best” ends up being totally determined by *noise* (example)

REGRESSION STABILITY

* this is bad: if your weights are essentially random, they are ~impossible to interpret, and model performance can suffer, so:

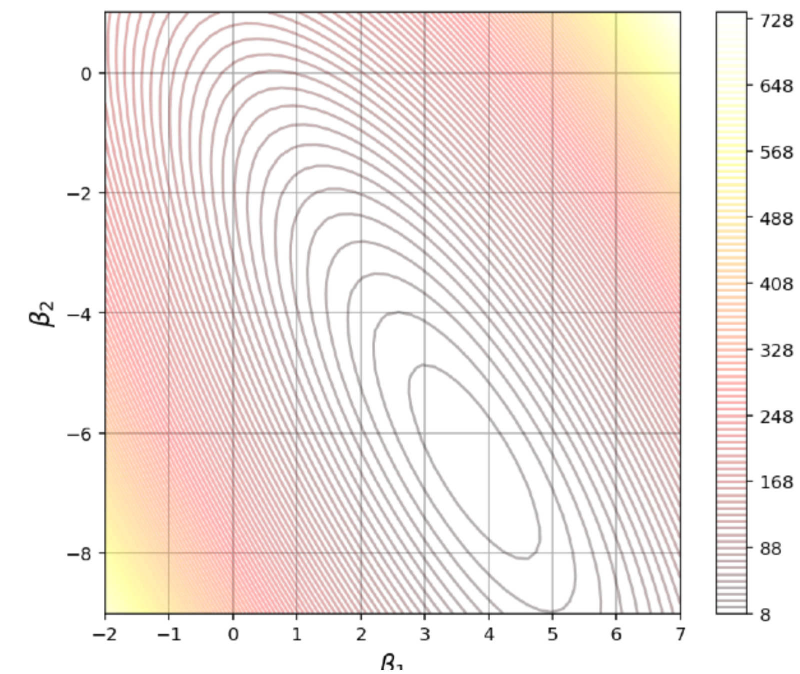
* (1) let’s figure out when this is happening, and

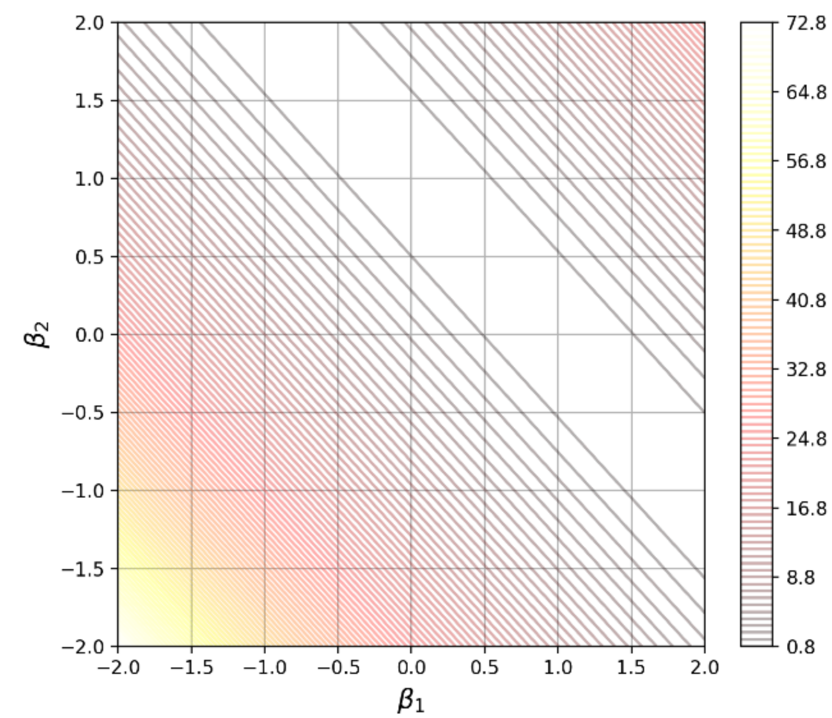
* (2) let’s stop it from happening

REGRESSION STABILITY

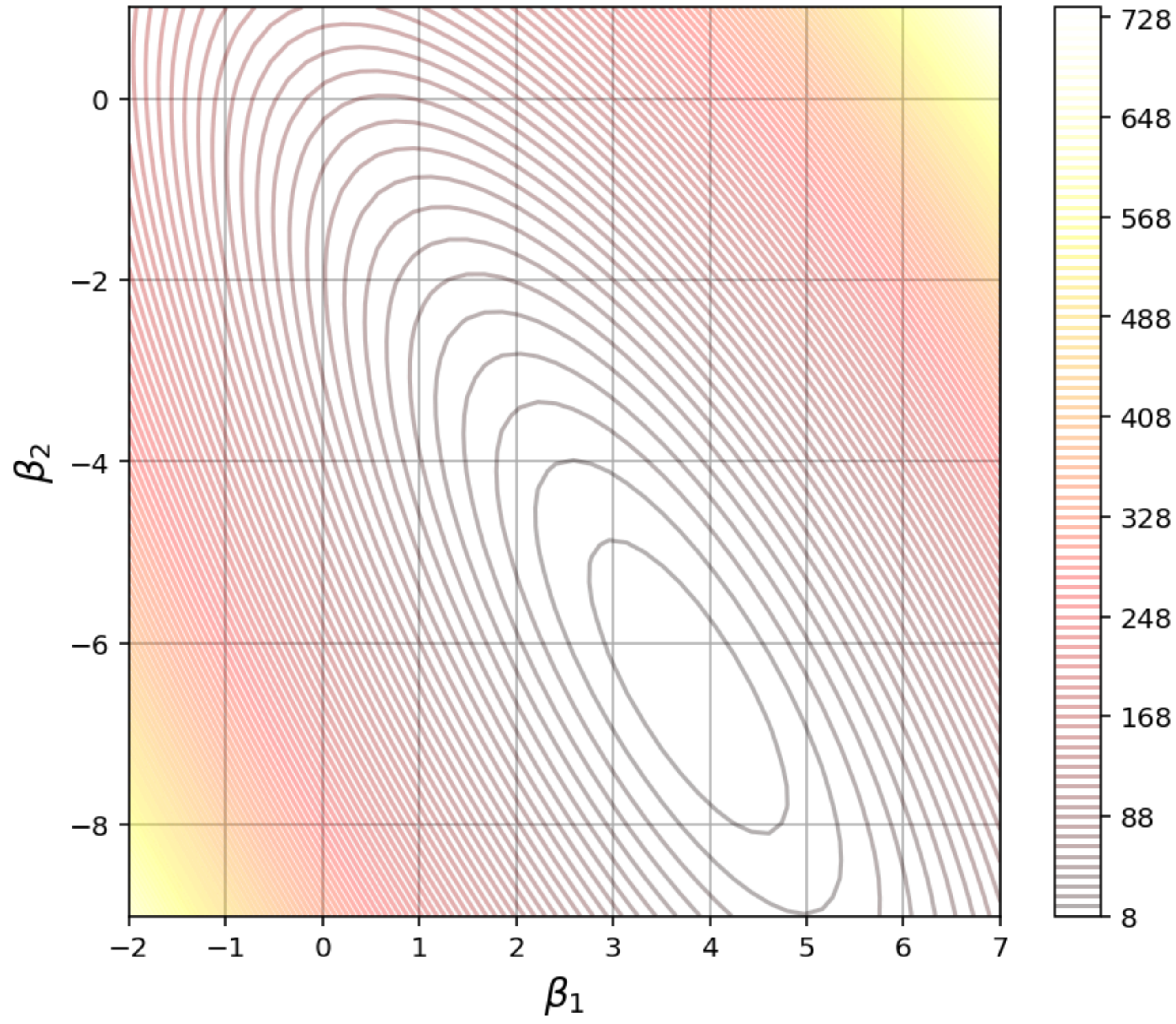
* how do we know when regression is unstable?

* it’s related to the shape of the error function!

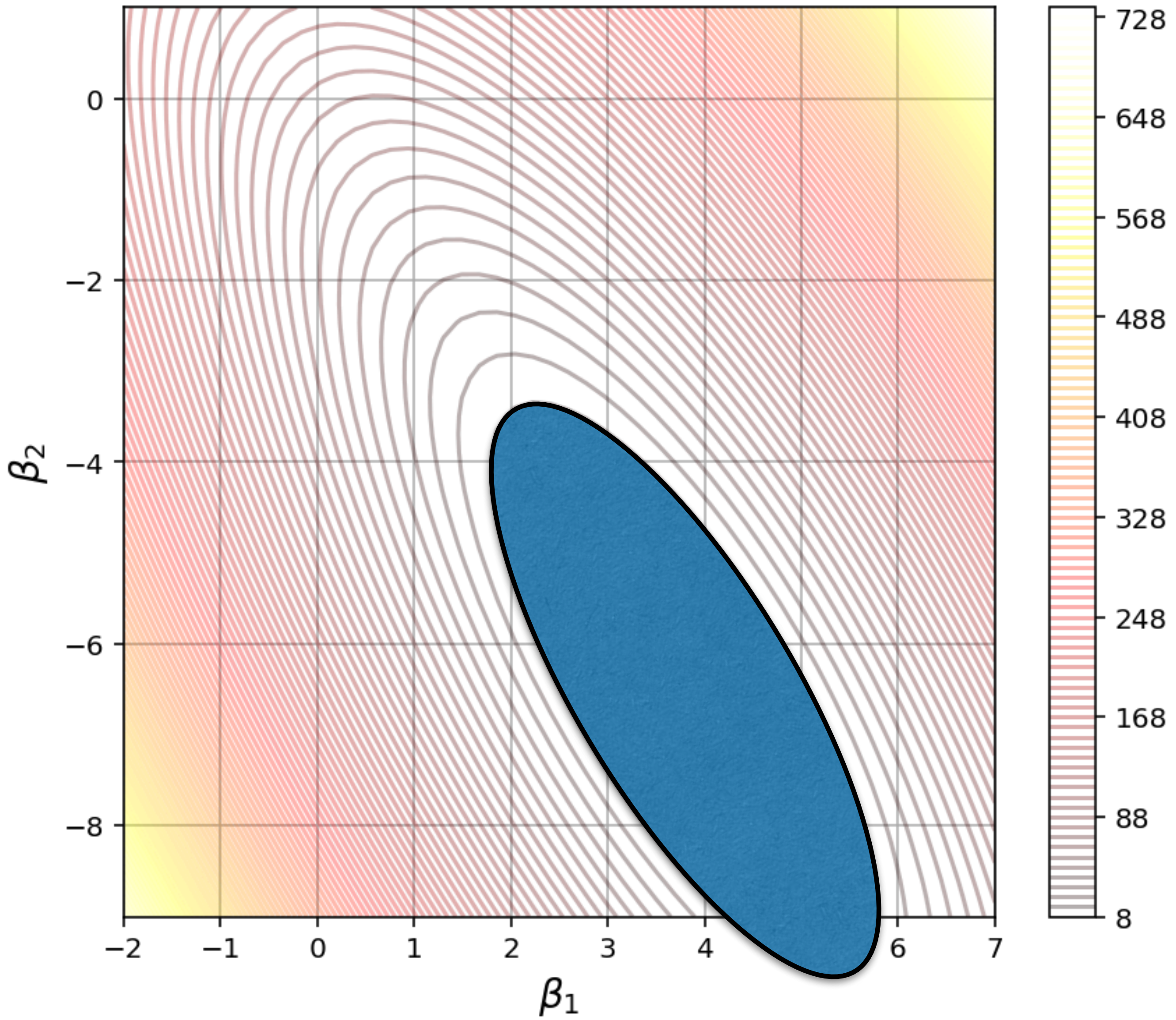




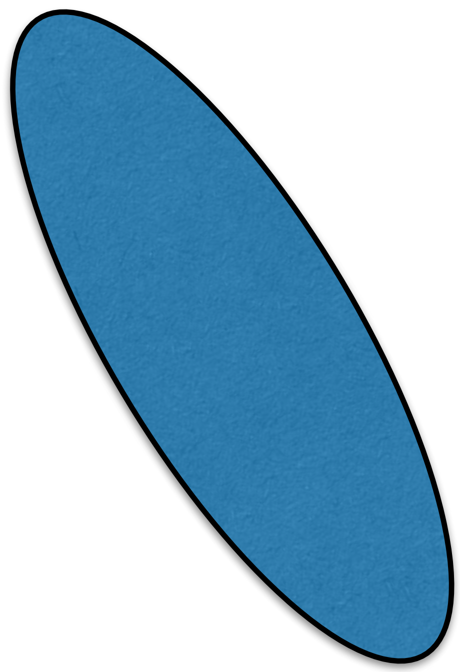
REGRESSION STABILITY



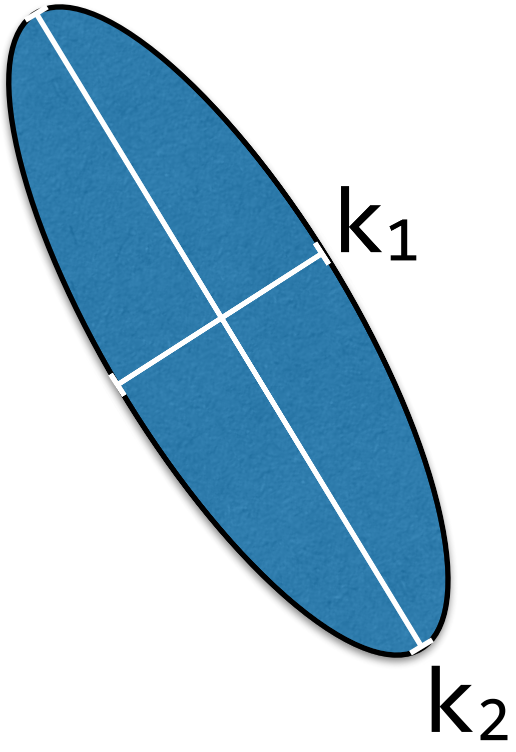
REGRESSION STABILITY



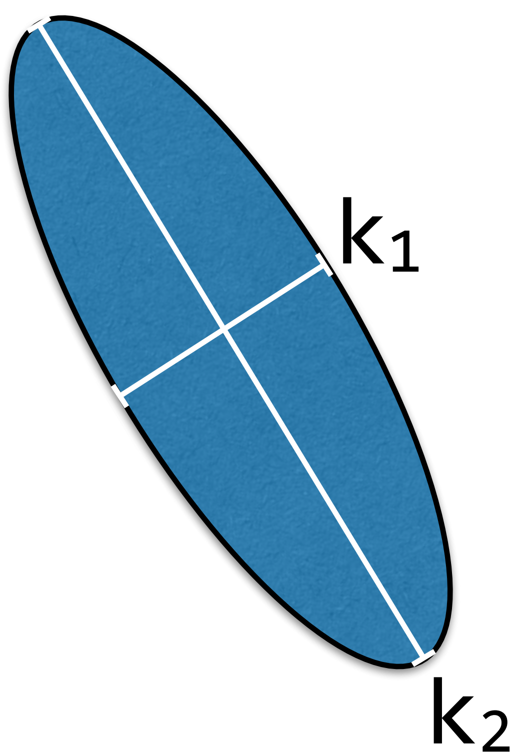
REGRESSION STABILITY

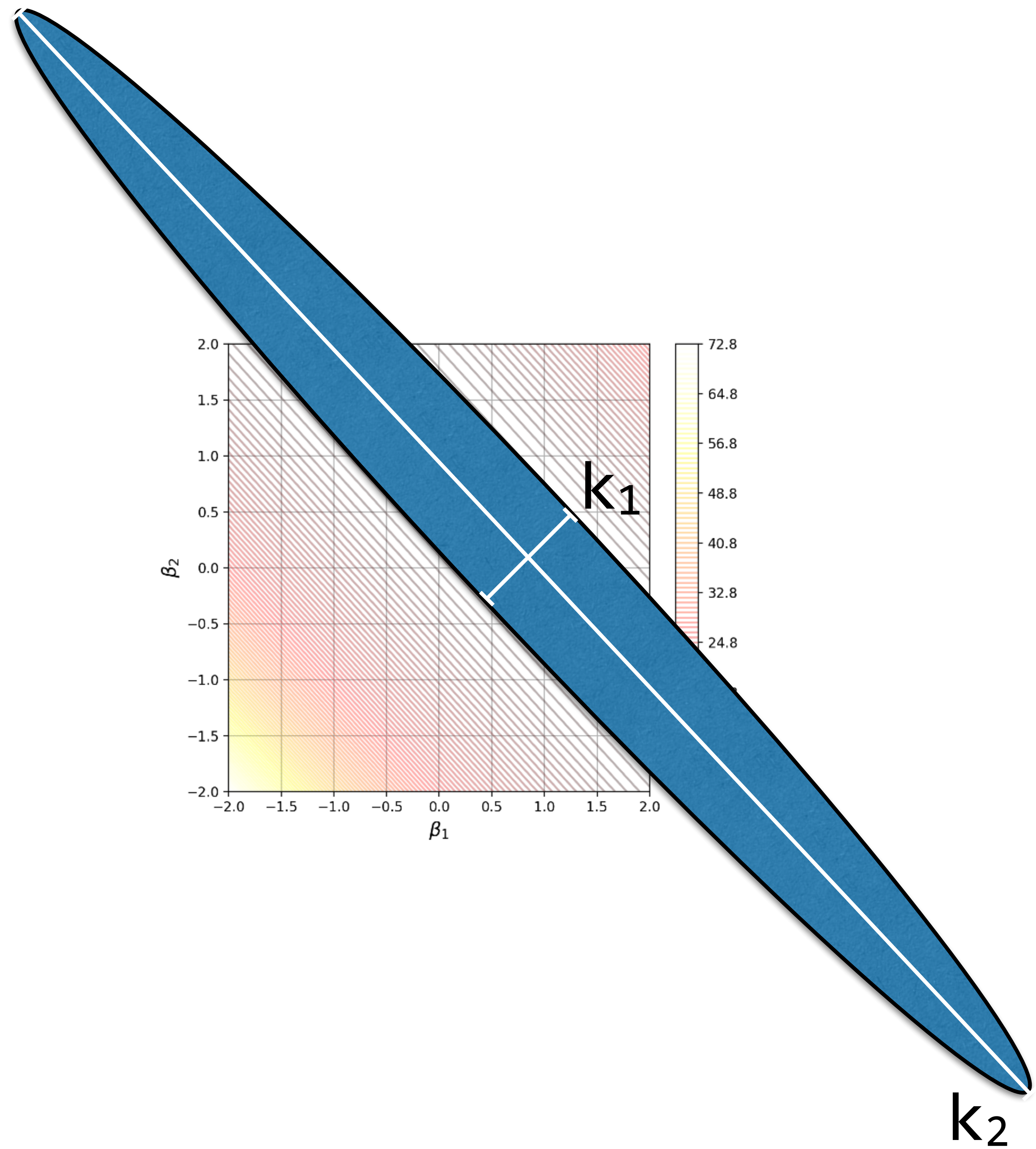


REGRESSION STABILITY



REGRESSION STABILITY





REGRESSION STABILITY

* the dimensions of the error ellipse, k_1 and k_2, are related to the **singular values** returned by np.linalg.lstsq!

* if the singular values (ordered from largest to smallest) are s_1, s_2, etc.,

* then $k_1 \propto s_1^{-1}$, $k_2 \propto s_2^{-1}$, etc.

REGRESSION STABILITY

* so it’s easy to detect when a regression is unstable: look for tiny singular values! (example)

* (or at least, tiny relative to the largest singular value)

REGRESSION STABILITY

* but what do you do if the regression is unstable?

* how could you possibly solve this problem?

END