LECTURE 18

Cross-Validation and Regularization

Different methods for ensuring the generalizability of our models to unseen data.

Data 100/Data 200, Fall 2021 @ UC Berkeley

Fernando Pérez and Alvin Wan

(content by Joseph Gonzalez and Suraj Rampure)

Segment Roadmap

Sections 16.1 through 16.4 discuss train-test splits and cross-validation

- 16.1: why we need to split our data into train and test, and how cross-validation works
- 16.2/16.3: creating train-test split and evaluating models fit on training data using testing data
- 16.4: implementing cross-validation
  - Note: the `Pipeline` object in `scikit-learn` is **not** in scope for this class

Segment Roadmap

Sections 16.1 through 16.4 discuss train-test splits and cross-validation

- 16.1: why we need to split our data into train and test, and how cross-validation works
- 16.2/16.3: creating train-test split and evaluating models fit on training data using testing data
- 16.4: implementing cross-validation
    - Note: the `Pipeline` object in `scikit-learn` is **not** in scope for this class

Sections 16.5 and 16.6 discuss regularization.

- 16.5: why we need to regularize and how penalties on the norm of our parameter vector accomplish this goal
- 16.6: explicitly lists the optimal model parameter when using the L2 penalty on our linear model (called "Ridge Regression")

Segment Roadmap (Optional)

There are also three **supplementary** videos accompanying this lecture

- They do not introduce any new material, but may be helpful for your understanding
- They do not have accompanying Quick Checks

Sections 16.7 and 16.8 implement ridge and LASSO regression in a notebook

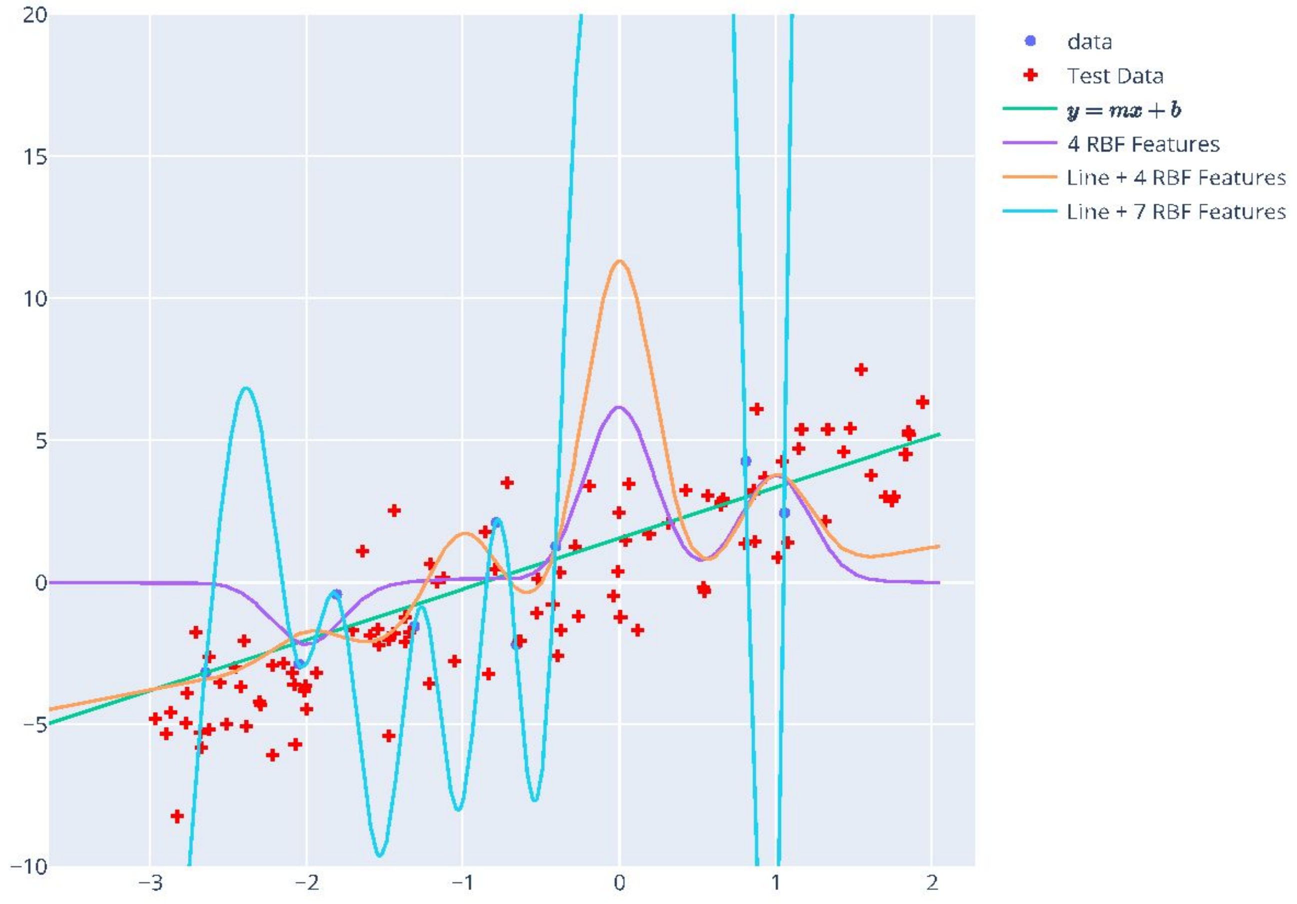
- These videos are helpful in explaining how regularization and cross-validation are used in practice
- These videos again use `Pipeline`, which is not in scope.

Section 16.9 is another **supplementary** video, created by Paul Shao (SP20 Data 100 TA)

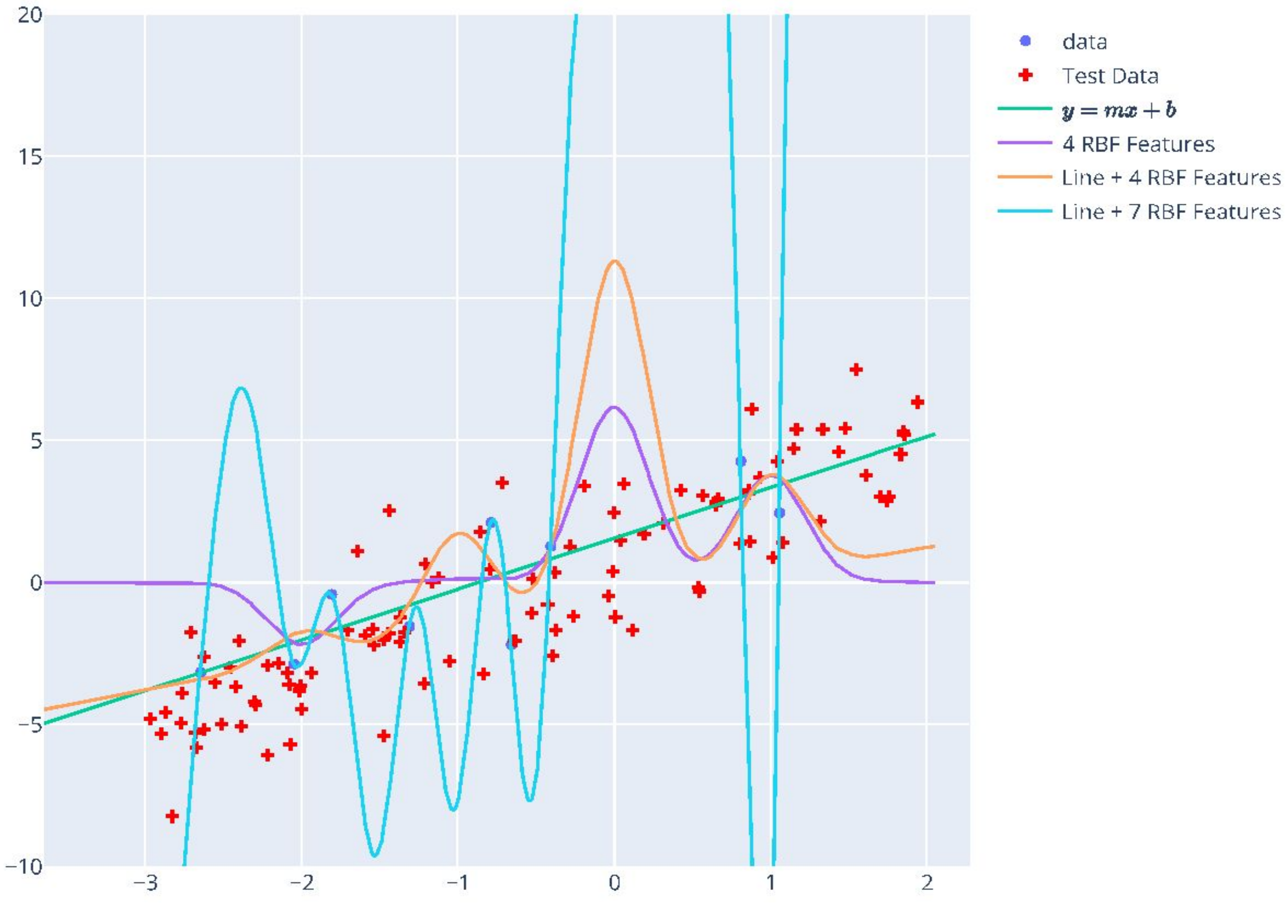
- Great high-level overview of both the bias-variance tradeoff and regularization

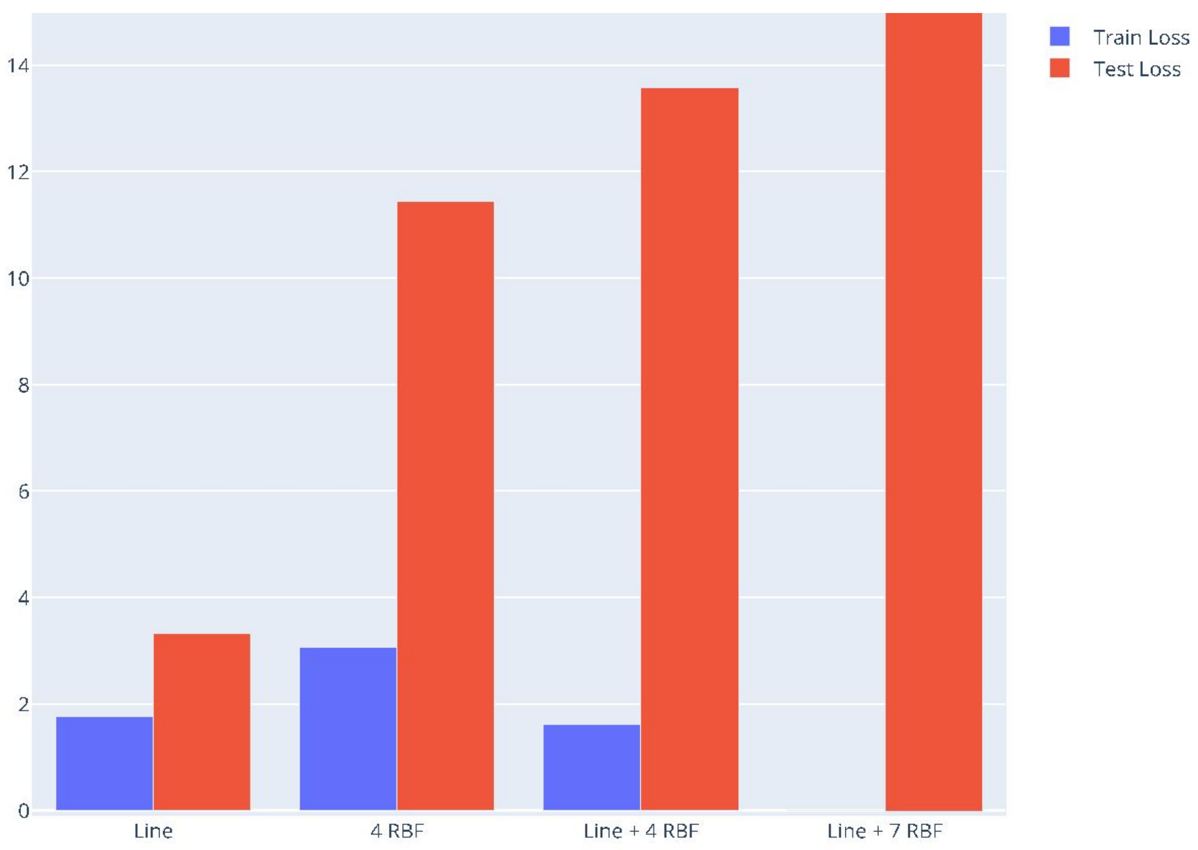
Cross Validation

Training Error vs **Test** Error



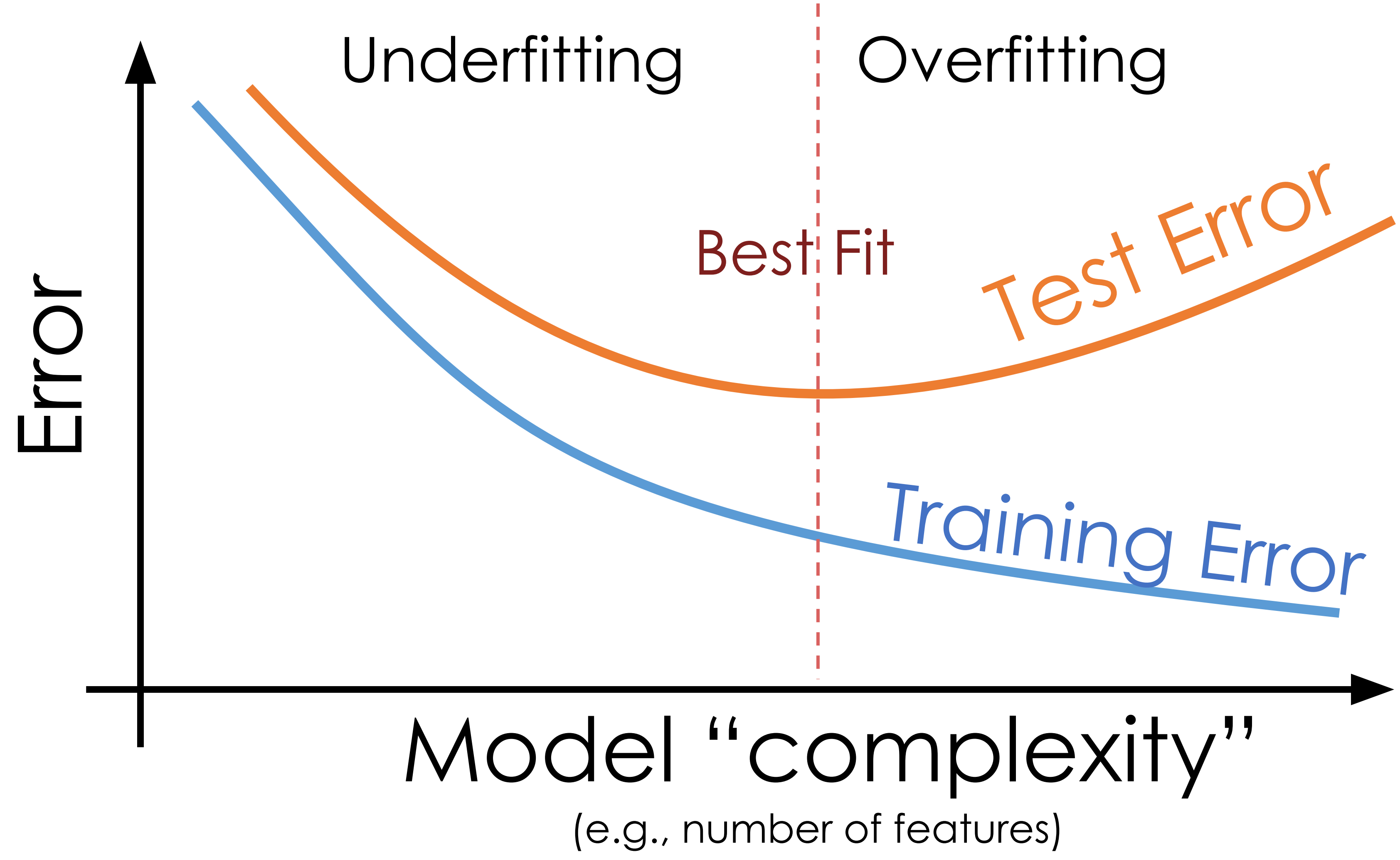
Training Error vs **Test** Error





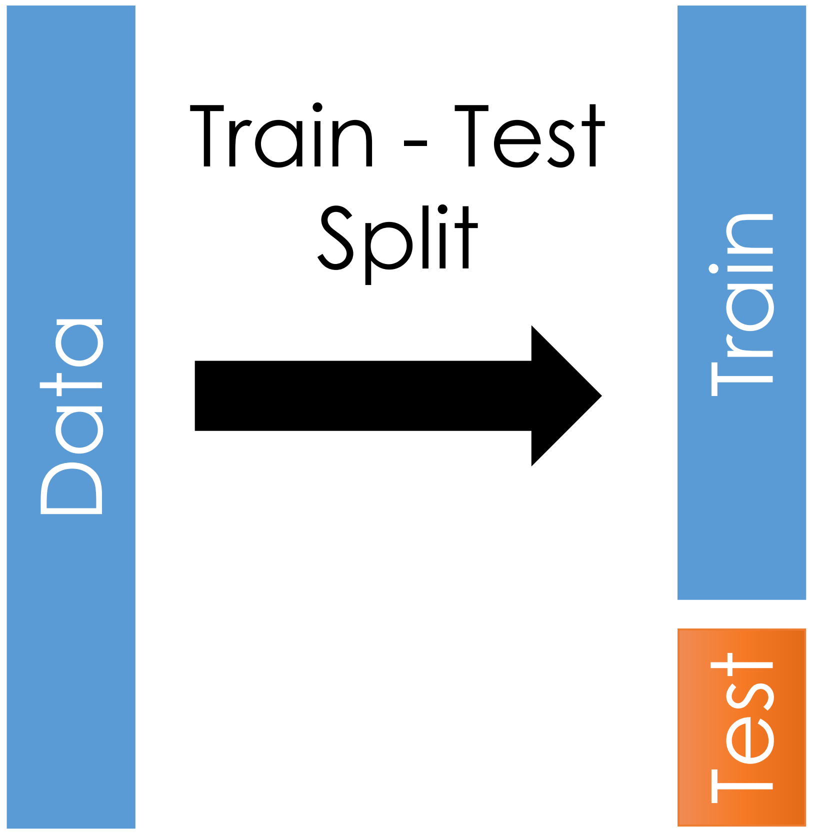
Training vs **Test** Error

Training error typically *<u>under estimates</u>* **test error**.



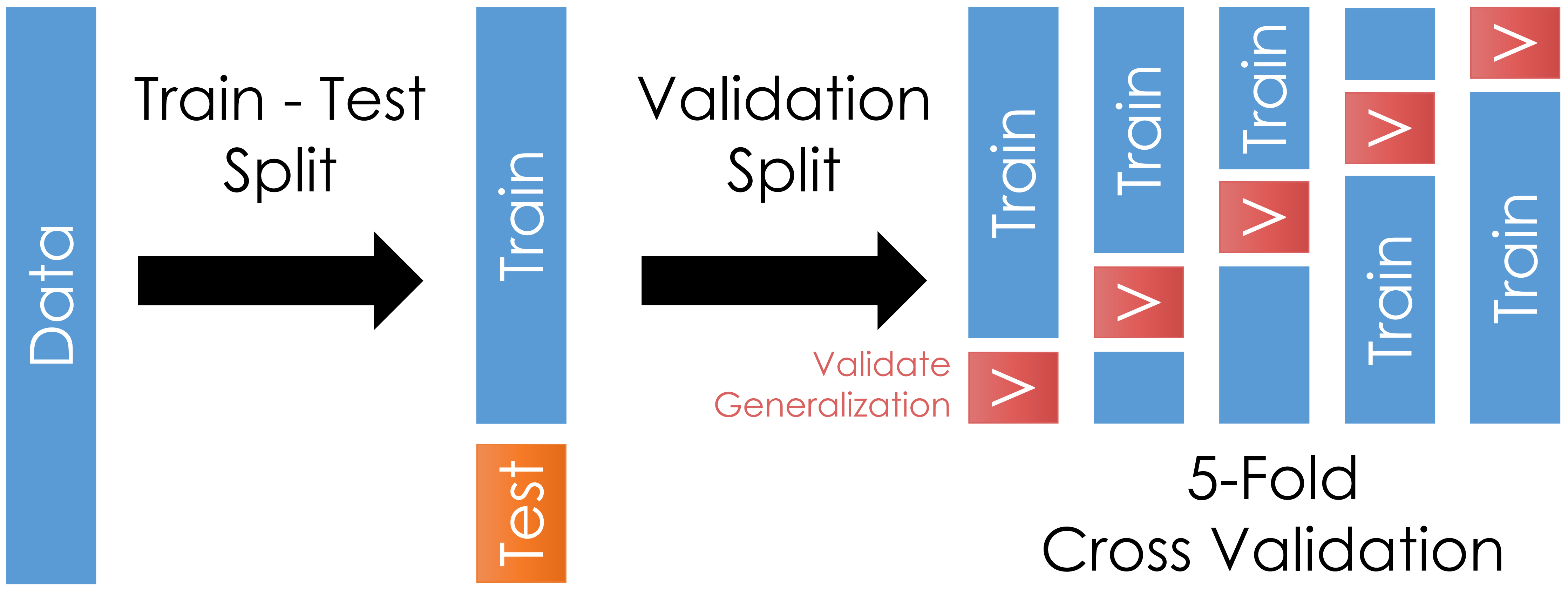
Generalization: *The Train-Test Split*

- **Training Data:** used to fit model
- **Test Data:** check generalization error
- How to split?
  - Randomly, Temporally, Geo…
  - Depends on application (usually randomly)
- What size? (90%-10%)
  - Larger training set – more complex models
  - Larger test set – better estimate of generalization error
  - Typically between 75%-25% and 90%-10%



You can only use the test dataset once after deciding on the model.

Generalization: *Validation Split*



Cross validation ***simulates multiple train test-splits*** *on the training data.*

Recipe for Successful Generalization

1. Split your data into **training** and **test** sets (90%, 10%)

2. Use **only the training data** when designing, training, and tuning the model
   - Use **cross validation** to test *generalization* during this phase
   - **Do not look at the test data**

3. Commit to your final model and train once more using **only the training data.**

4. Test the final model using the **test data.** If accuracy is not acceptable return to (2). (*Get more test data if possible*.)

5. Train **on all available data** and ship it! 

Regularization

Parametrically Controlling the *Model Complexity*

Basic Idea

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \mathbf{Loss}\left(y_i, f_\theta(x_i)\right)$$

<u>**Such that:**</u>

f_θ does not "overfit"

Can we make this more formal?

Basic Idea

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \mathbf{Loss}\left(y_i, f_\theta(x_i)\right)$$

<u>Such that:</u>

$$\text{Complexity}\left(f_\theta \right) \leq \beta$$

Regularization Parameter

How do we define this?

Idealized Notion of Complexity

$$\text{Complexity}(f_\theta) \leq \beta$$

- Focus on complexity of **linear models:**
  - Number and kinds of features
- Ideal definition:

$$\mathbf{Complexity}(f_\theta) = \sum_{j=1}^{d} \mathbb{I}\left[\theta_j \neq 0\right]$$

Number of non-zero parameters

- Why?

Ideal "Regularization"

Find the best value of θ which uses fewer than β features.

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \mathbf{Loss}\left(y_i, f_\theta(x_i)\right)$$

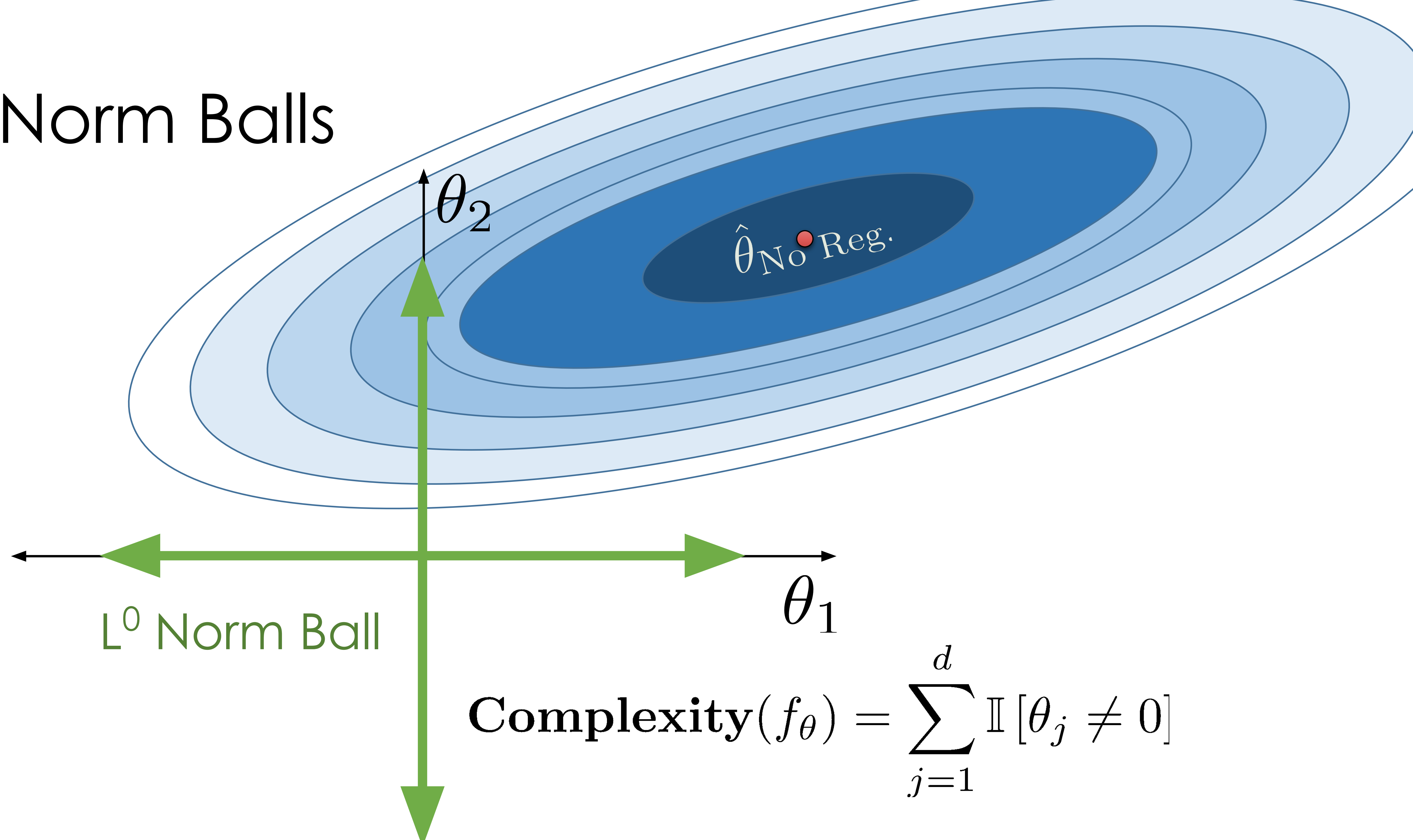
<u>**Such that:**</u>

Need an approximation!

$$\mathbf{Complexity}(f_\theta) = \sum_{j=1}^{d} \mathbb{I}\left[\theta_j \neq 0\right] \leq \beta$$

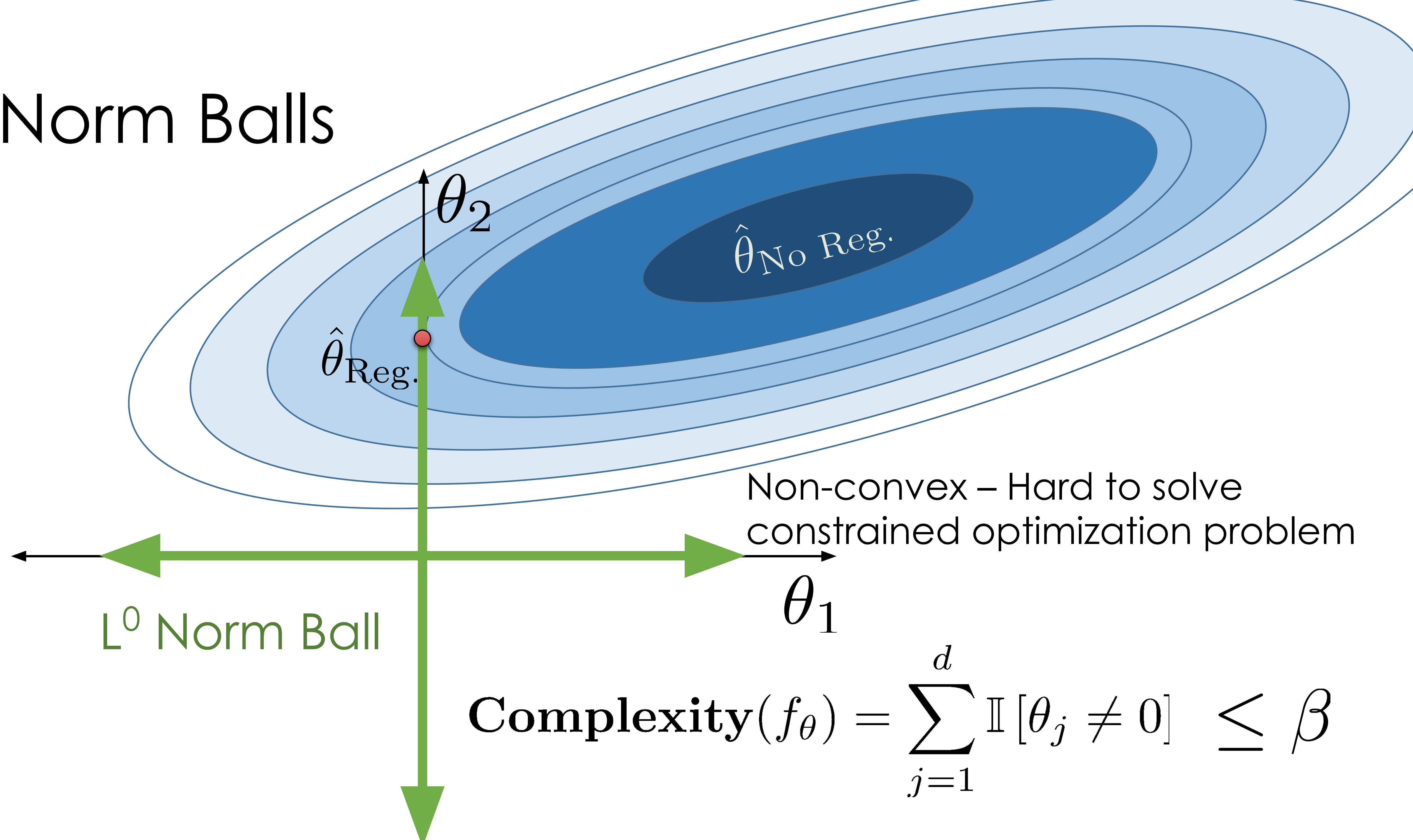
Combinatorial search problem – NP-hard to solve in general.

Norm Balls



$$\textbf{Complexity}(f_\theta) = \sum_{j=1}^{d} \mathbb{I}\left[\theta_j \neq 0\right]$$

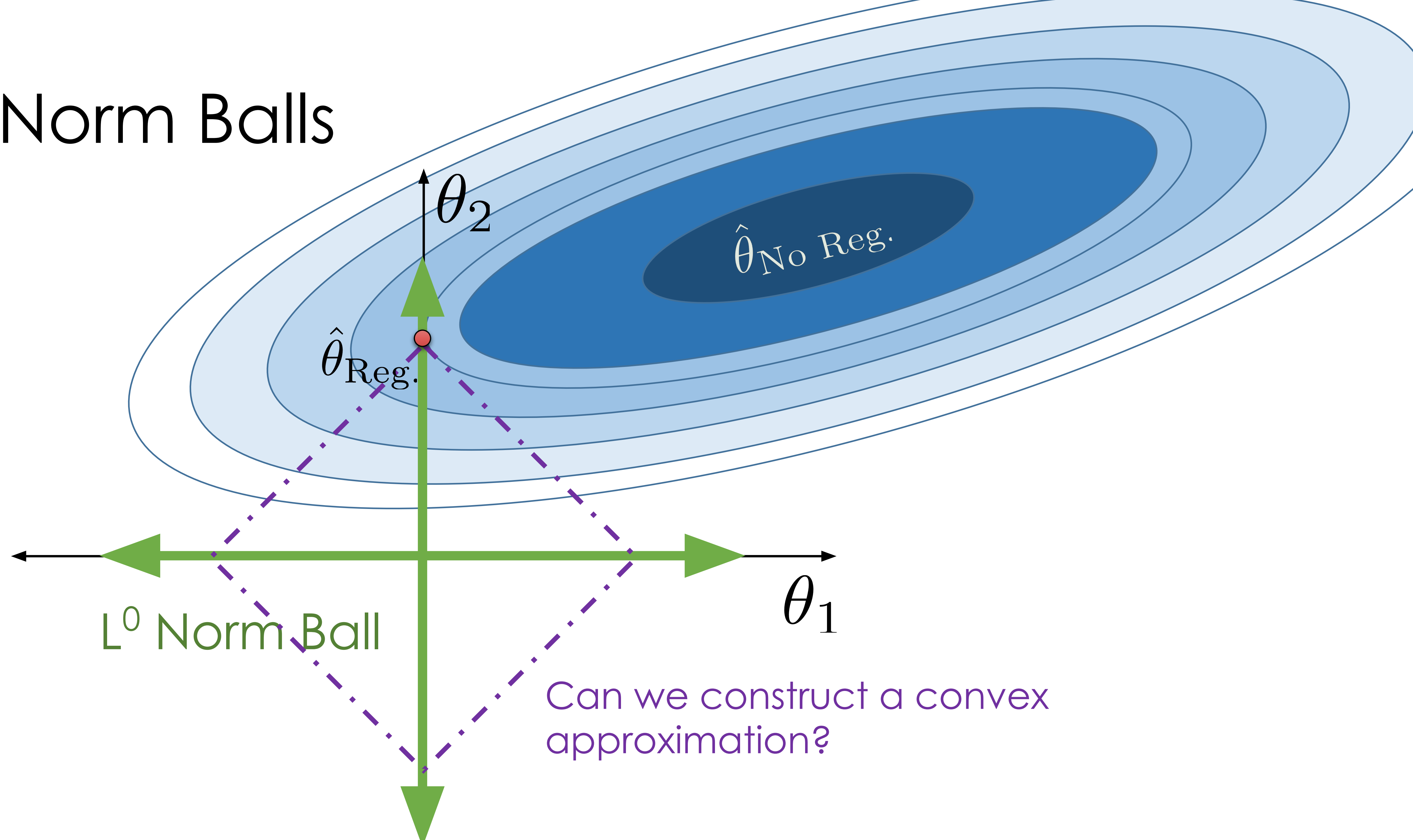
Norm Balls



Non-convex – Hard to solve constrained optimization problem

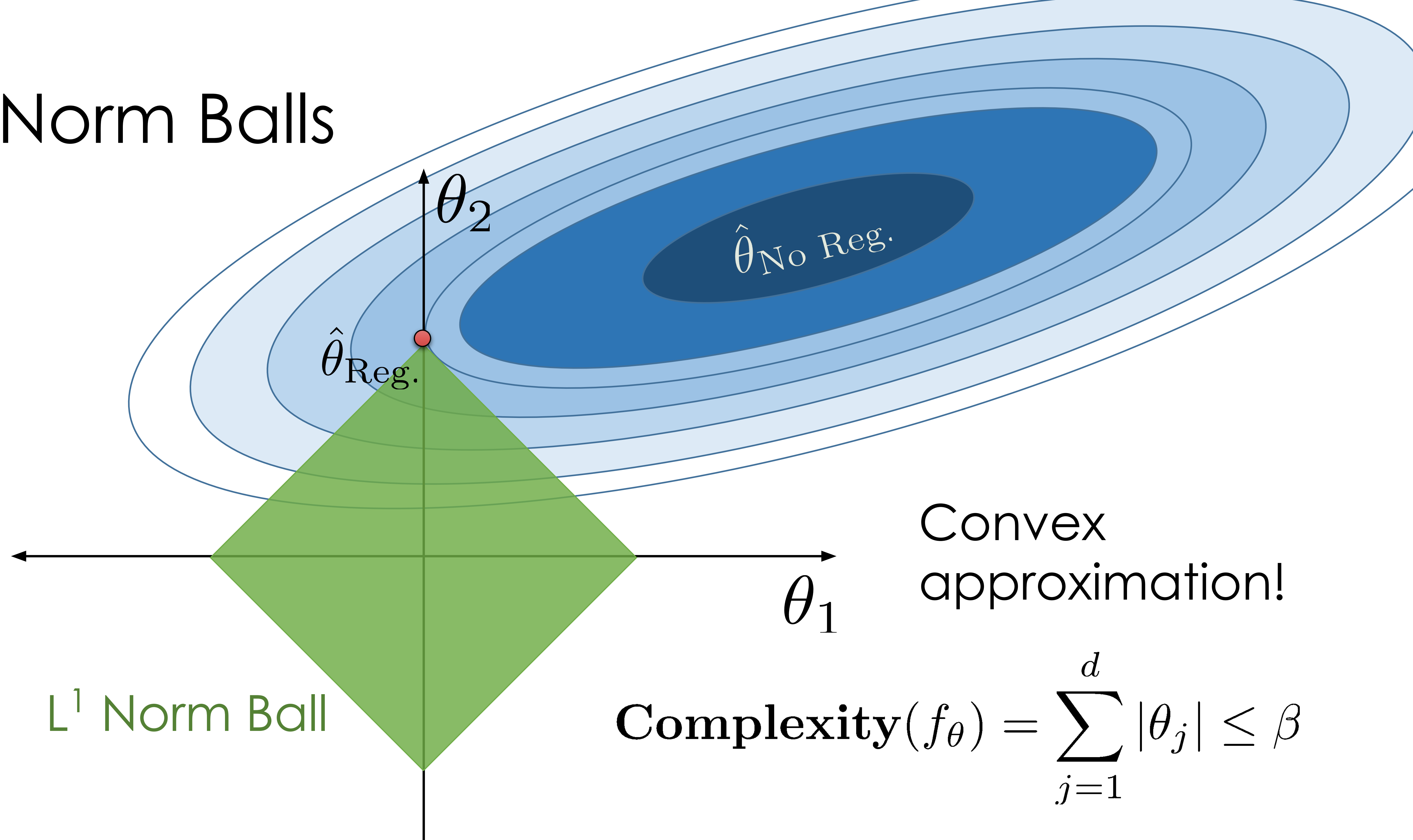
$$\textbf{Complexity}(f_\theta) = \sum_{j=1}^{d} \mathbb{I}\left[\theta_j \neq 0\right] \leq \beta$$

Norm Balls



Can we construct a convex approximation?

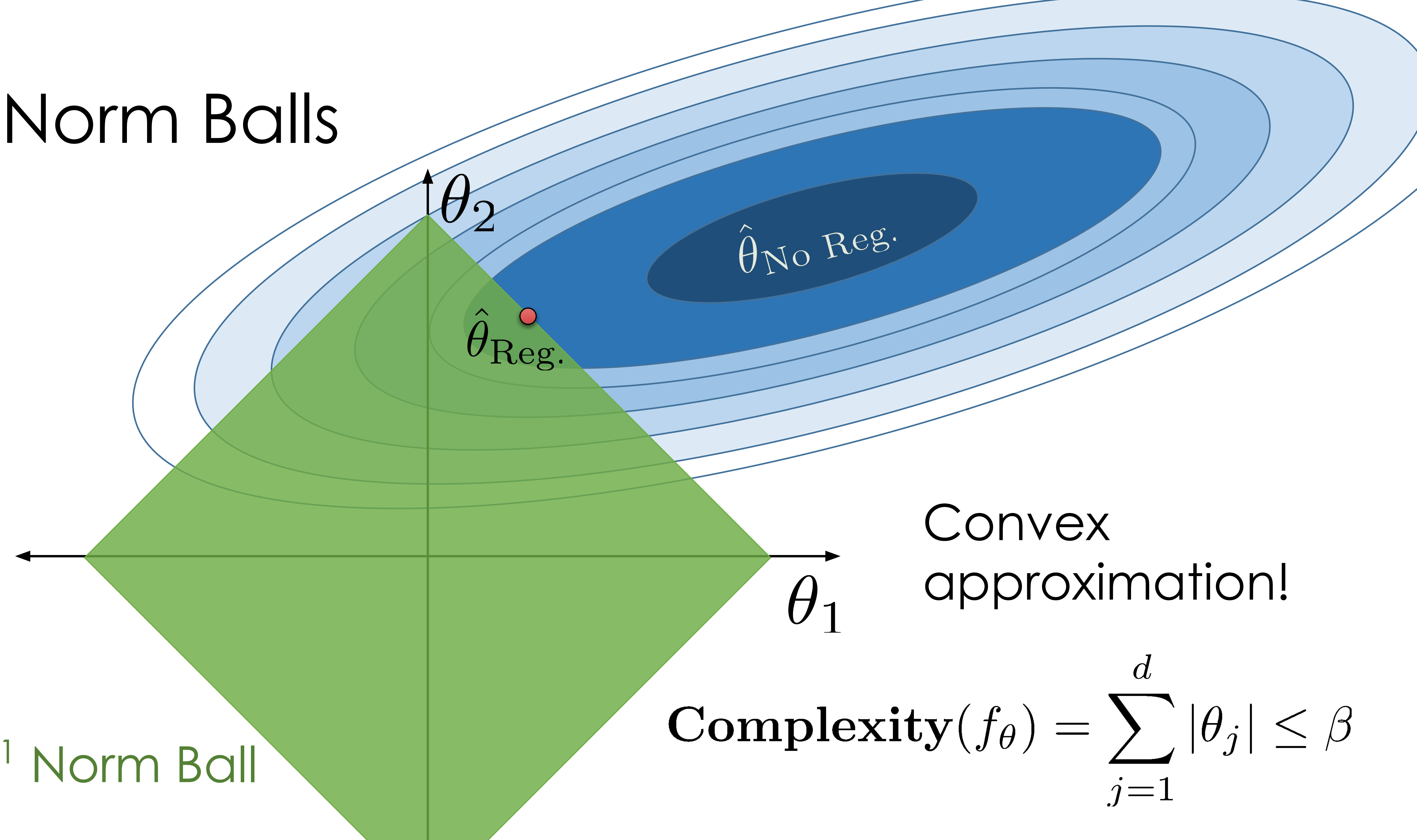
Norm Balls



Convex approximation!

$$\textbf{Complexity}(f_\theta) = \sum_{j=1}^{d} |\theta_j| \leq \beta$$

Norm Balls

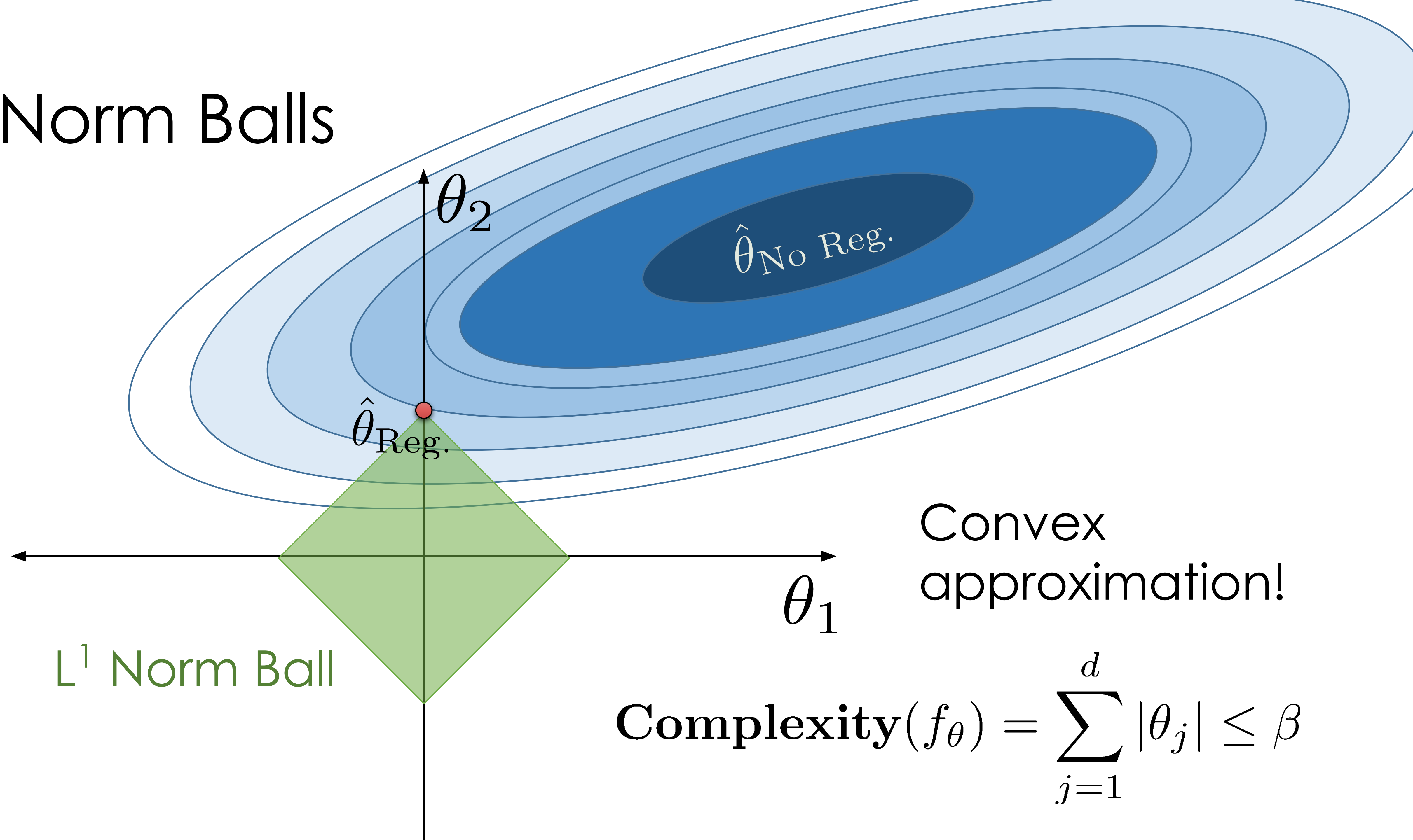


Convex approximation!

$$\textbf{Complexity}(f_\theta) = \sum_{j=1}^{d} |\theta_j| \leq \beta$$

L1 Norm Ball

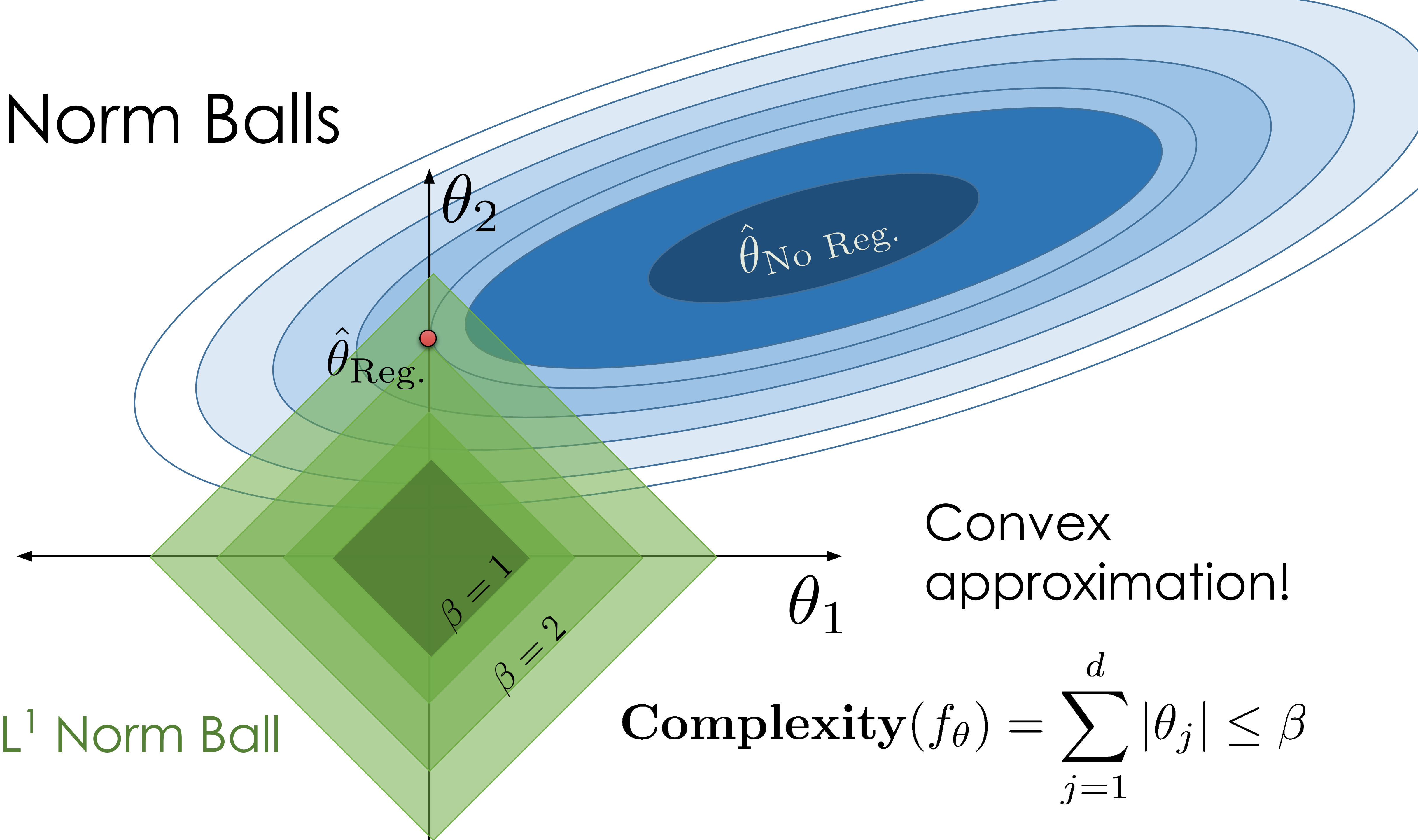
Norm Balls



Convex approximation!

$$\textbf{Complexity}(f_\theta) = \sum_{j=1}^{d} |\theta_j| \leq \beta$$

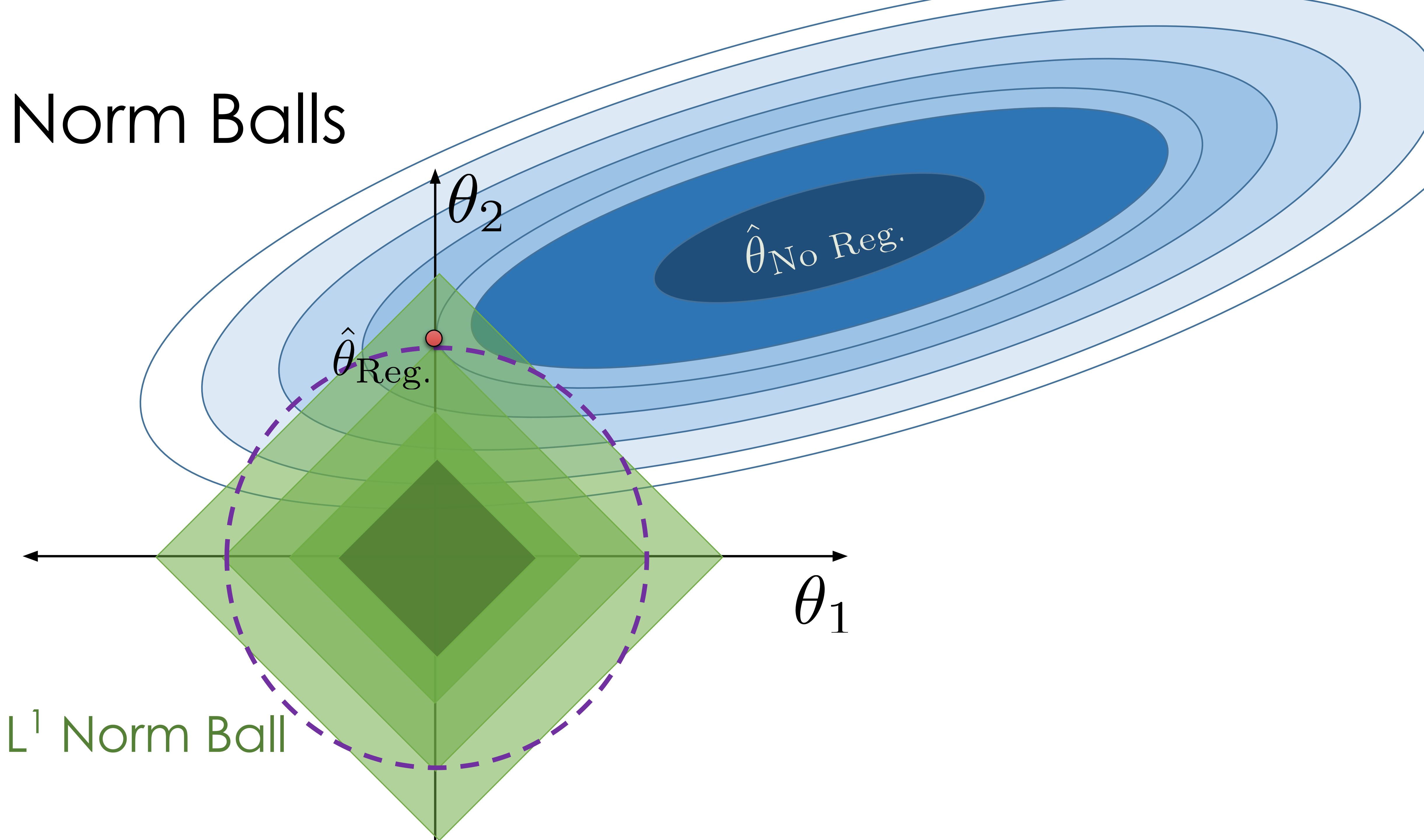
Norm Balls



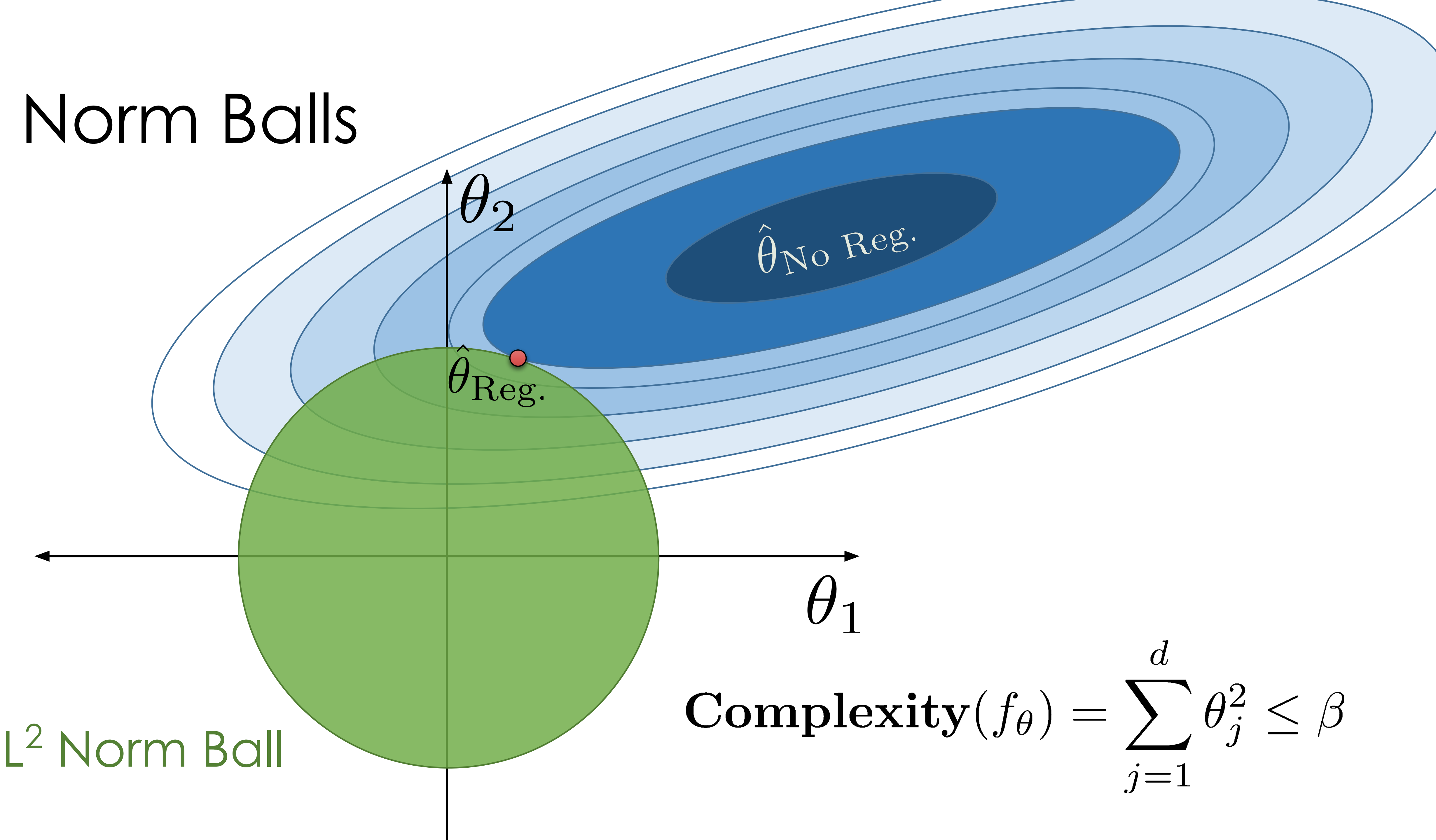
Convex approximation!

$$\textbf{Complexity}(f_\theta) = \sum_{j=1}^{d} |\theta_j| \leq \beta$$

Norm Balls

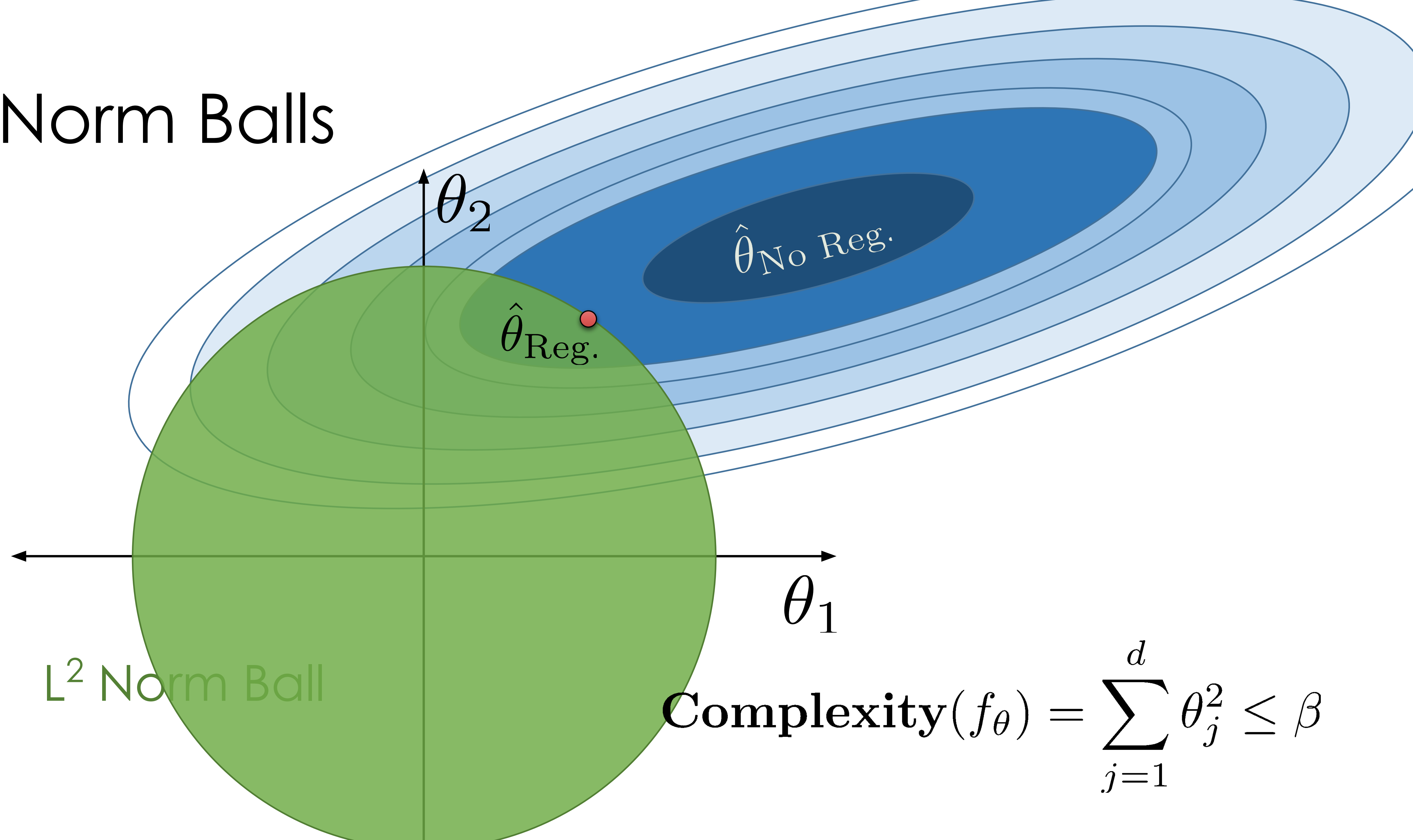

θ_2
$\hat{\theta}_{\text{No Reg.}}$
$\hat{\theta}_{\text{Reg.}}$
θ_1
L^1 Norm Ball

Norm Balls



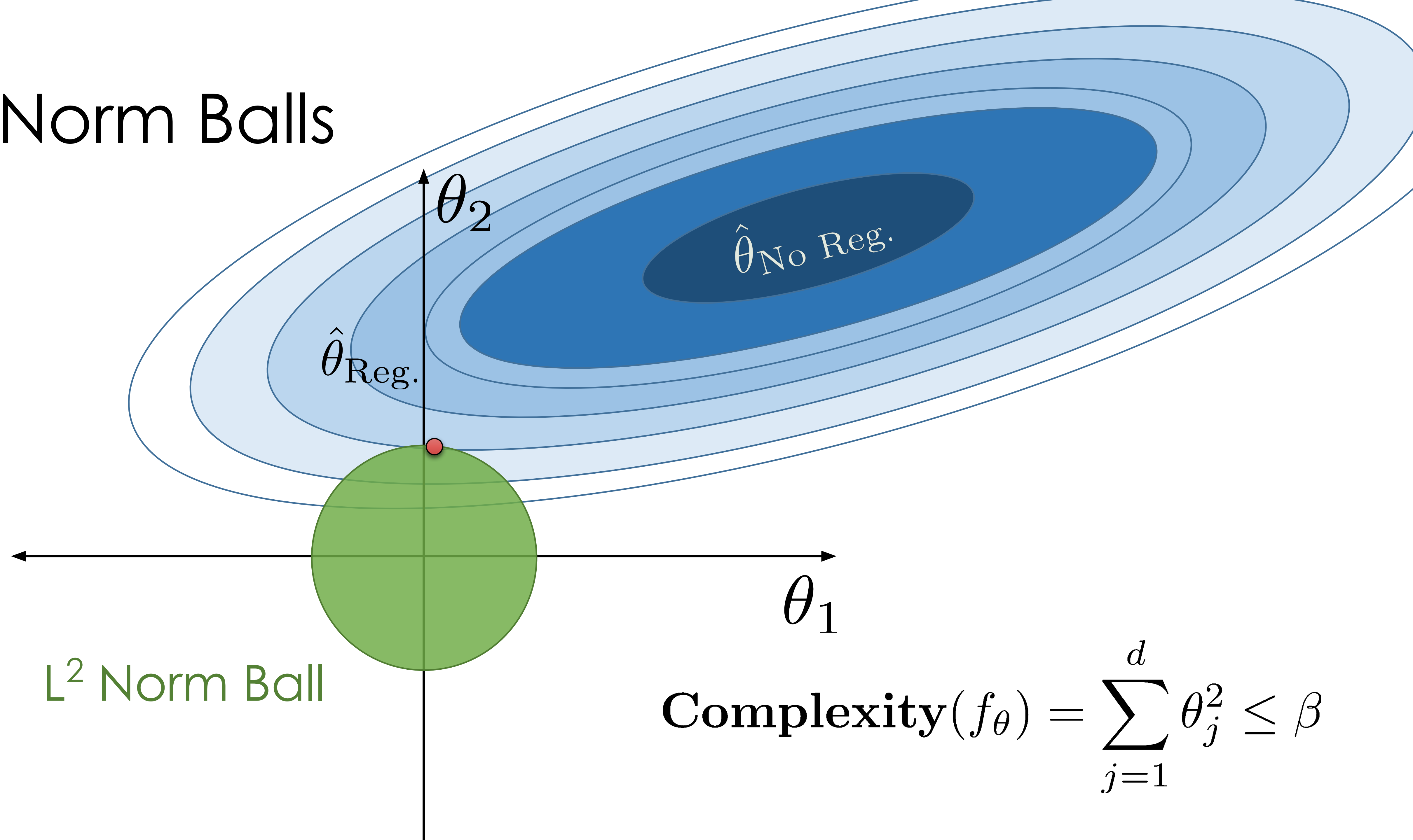
$$\textbf{Complexity}(f_\theta) = \sum_{j=1}^{d} \theta_j^2 \leq \beta$$

Norm Balls



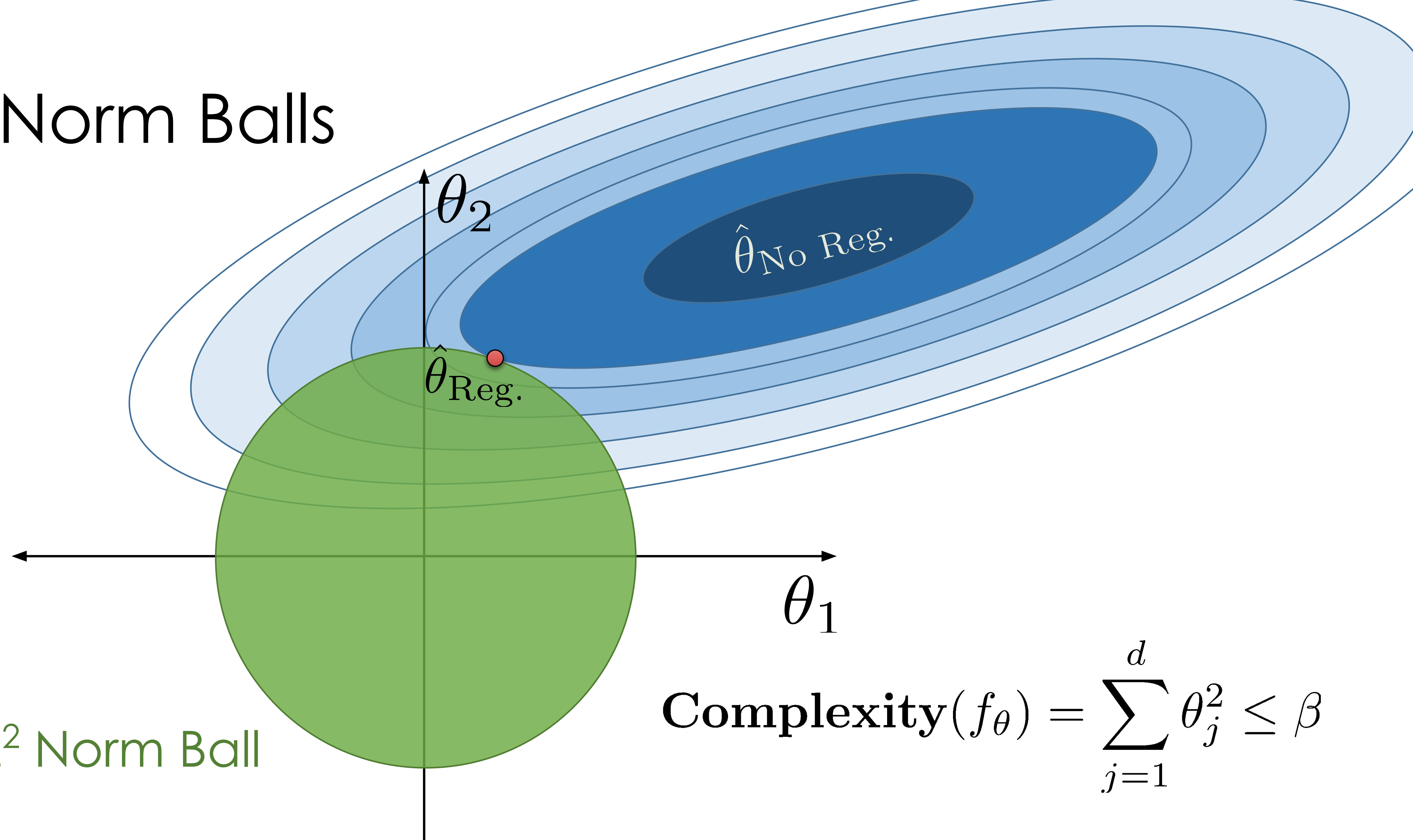
$$\textbf{Complexity}(f_\theta) = \sum_{j=1}^{d} \theta_j^2 \leq \beta$$

Norm Balls



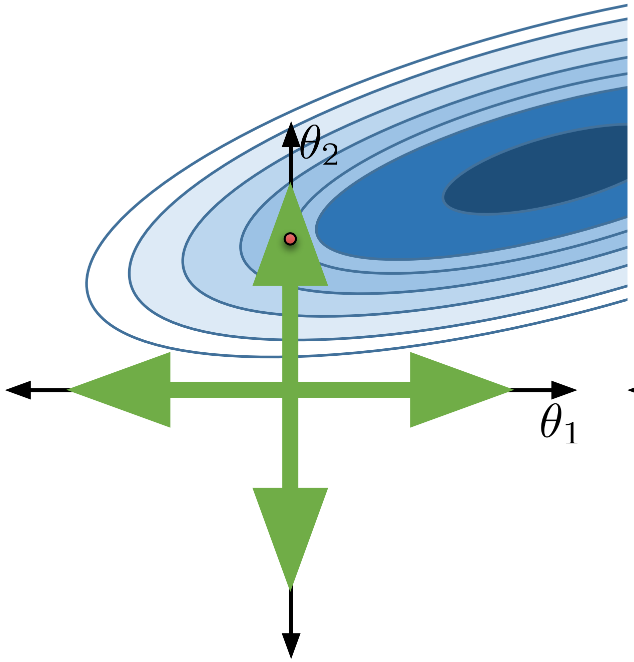
$$\textbf{Complexity}(f_\theta) = \sum_{j=1}^{d} \theta_j^2 \leq \beta$$

Norm Balls



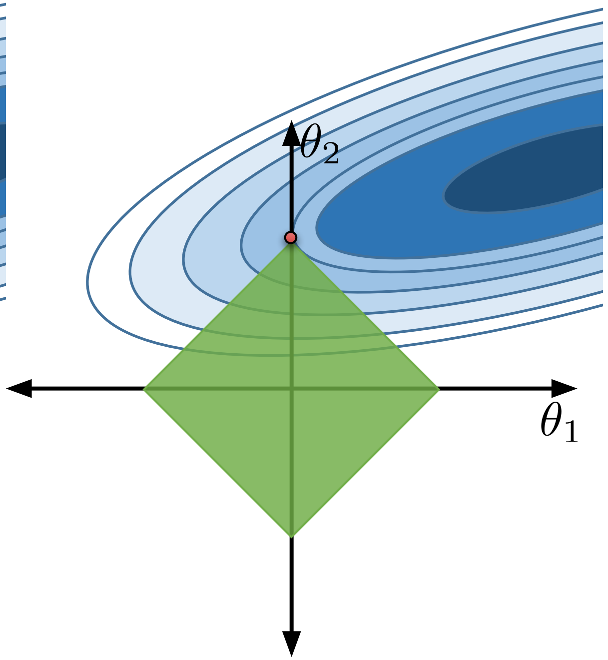
$$\textbf{Complexity}(f_\theta) = \sum_{j=1}^{d} \theta_j^2 \leq \beta$$

L^0 Norm Ball



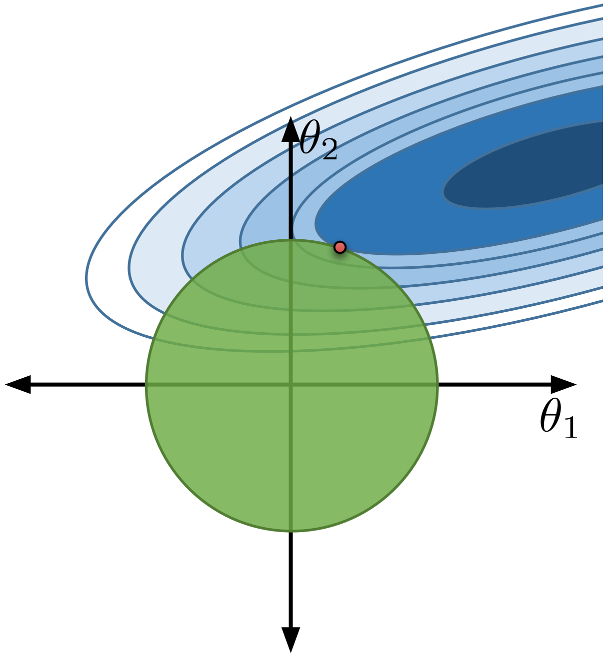
Ideal for
Feature Selection
but combinatorically
difficult to optimize

L^1 Norm Ball



Encourages
Sparse Solutions
Convex!

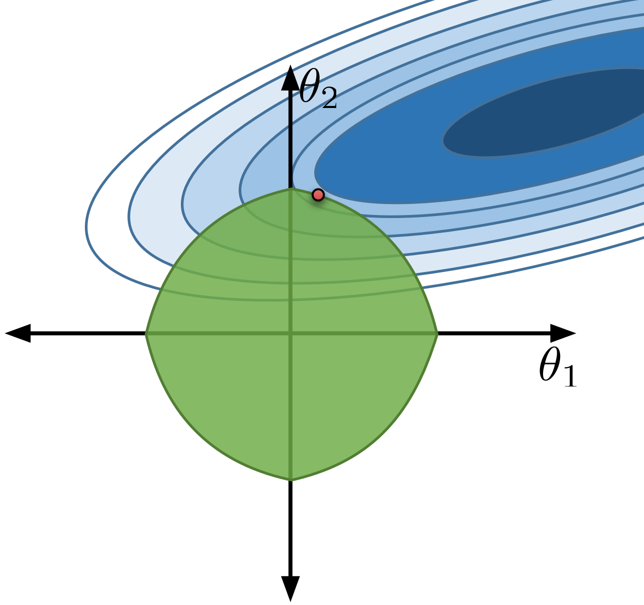
L^2 Norm Ball



Spreads weight
over features **(robust)**
does not
encourage sparsity

$L^1 + L^2$ Norm

Elastic Net



Compromise
Need to tune
two regularization
parameters

Generic Regularization (Constrained)

Defining $\textbf{Complexity}(f_\theta) = R(\theta)$

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \textbf{Loss}\left(y_i, f_\theta(x_i)\right)$$

<u>Such that:</u> $R(\theta) \leq \beta$

There is an equivalent unconstrained formulation (obtained by Lagrangian duality)

Generic Regularization (Constrained)

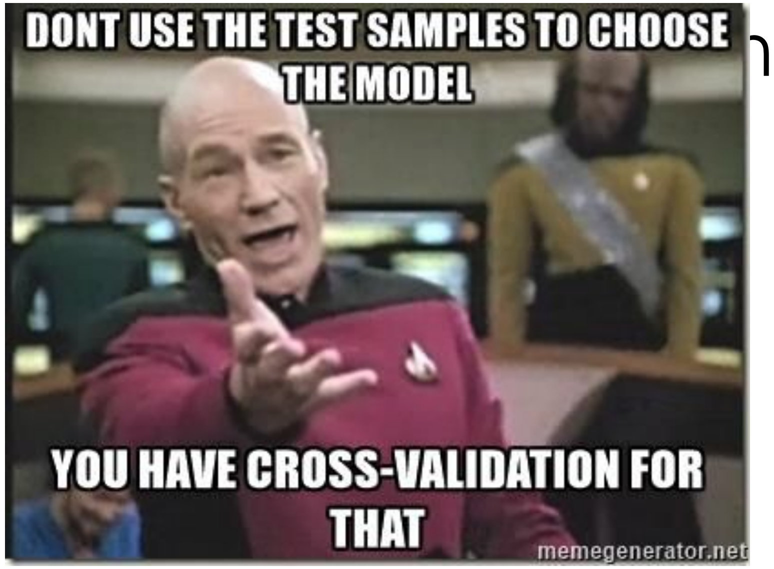
Defining $$\mathbf{Complexity}(f_\theta) = R(\theta)$$

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \mathbf{Loss}\left(y_i, f_\theta(x_i)\right) + \lambda R(\theta)$$

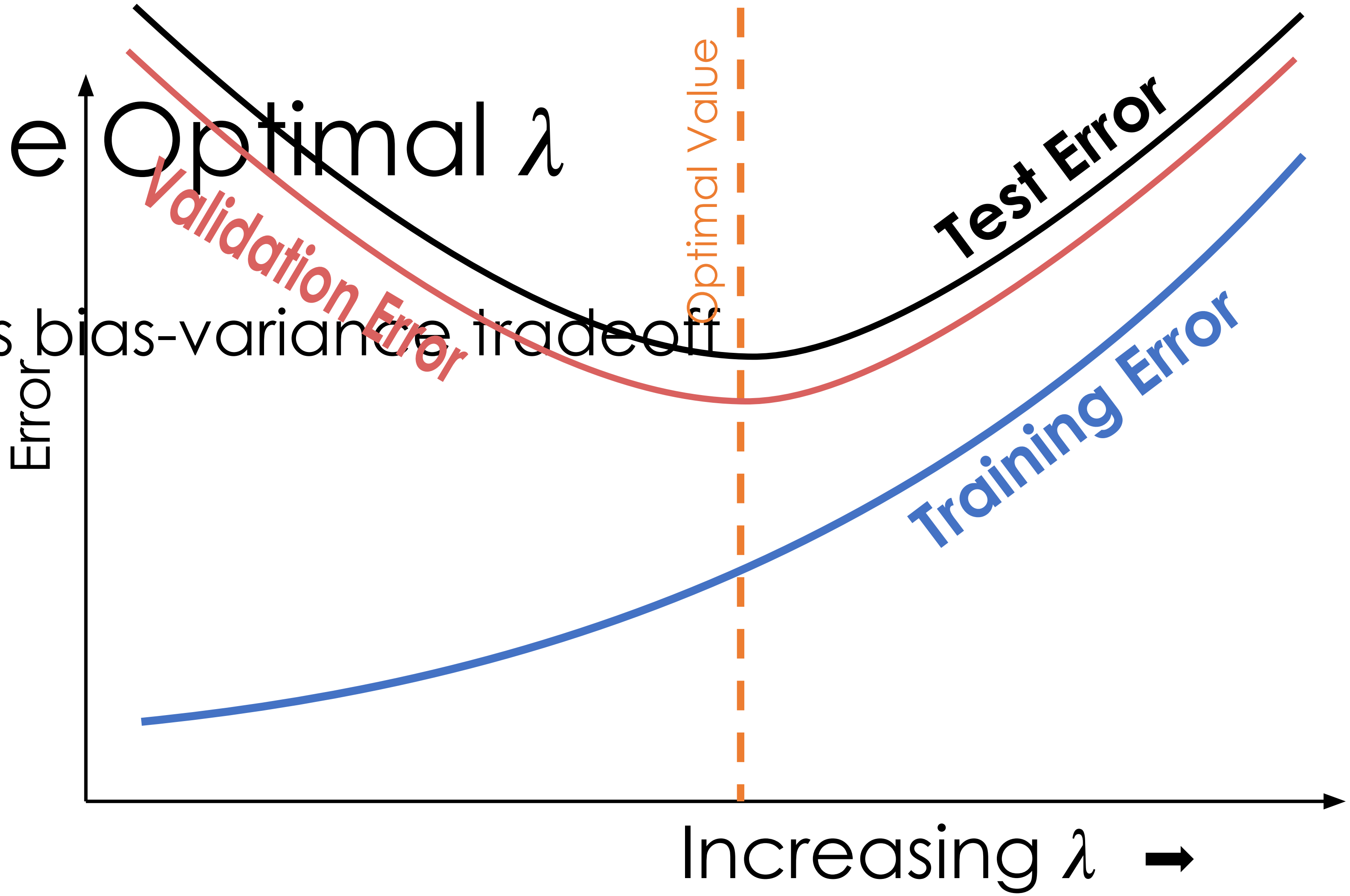
Regularization Parameter

There is an equivalent unconstrained formulation (obtained by Lagrangian duality)

Determining the Optimal λ



hes bias-variance tradeoff



Determined through cross validation

Standardization and the **Intercept** Term

- `Height =` θ_1 Small `n_se` Large `+` θ_2 `weight_in_tons`

- Regularization penalized dimensions equally

- **Standardization**
  - Ensure that each dimensions has the same scale
  - centered around zero

- **Intercept Terms**
  - Typically don't regularize intercept term

Standardization

For each dimension *k*:

$$z_k = \frac{x_k - \mu_k}{\sigma_k}$$

Ridge Regression

"Ridge Regression" is a term for the following specific combination of model, loss, and regularization:

- Model: $\hat{\mathbb{Y}} = \mathbb{X}\theta$
- Loss: Squared loss
- Regularization: L2 regularization

The **objective function** we minimize for Ridge Regression is average squared loss, plus an added penalty:

$$\hat{\theta}_{\text{ridge}} = \arg\min_{\theta} \frac{1}{n} ||\mathbb{Y} - \mathbb{X}\theta||_2^2 + \lambda \sum_{j=1}^{d} \theta_i^2$$

Ridge Regression

We can also express this objective slightly differently:

$$\hat{\theta}_{\text{ridge}} = \arg\min_{\theta} \frac{1}{n} ||\mathbb{Y} - \mathbb{X}\theta||_2^2 + \lambda \sum_{j=1}^{d} \theta_i^2$$

$$\hat{\theta}_{\text{ridge}} = \arg\min_{\theta} \frac{1}{n} ||\mathbb{Y} - \mathbb{X}\theta||_2^2 + \lambda ||\theta||_2^2$$

L2 norm of theta (hence, L2 regularization)

The latter representation ignores the fact that we don't regularize the intercept term.

Ridge Regression

Ridge Regression has a closed form solution, conveniently:

$$\hat{\theta}_{\text{ridge}} = (\mathbb{X}^T\mathbb{X} + n\lambda I)^{-1}\mathbb{X}^T\mathbb{Y}$$

Identity matrix

Unlike OLS, there always exists a unique optimal parameter vector for Ridge Regression.

This is important, you should remember it!

LASSO Regression

"LASSO Regression" is a term for the following specific combination of model, loss, and regularization:

- Model: $\hat{\mathbb{Y}} = \mathbb{X}\theta$
- Loss: Squared loss
- Regularization: L1 regularization

The **objective function** we minimize for LASSO Regression is average squared loss, plus an added penalty:

$$\hat{\theta}_{\text{LASSO}} = \arg\min_{\theta} \frac{1}{n}||\mathbb{Y} - \mathbb{X}\theta||_2^2 + \lambda \sum_{j=1}^{d} |\theta_i|$$

LASSO Regression

We can also express this objective slightly differently:

$$\hat{\theta}_{\text{LASSO}} = \arg\min_{\theta} \frac{1}{n}||\mathbb{Y} - \mathbb{X}\theta||_2^2 + \lambda \sum_{j=1}^{d} |\theta_i|$$

$$\hat{\theta}_{\text{LASSO}} = \arg\min_{\theta} \frac{1}{n}||\mathbb{Y} - \mathbb{X}\theta||_2^2 + \lambda||\theta||_1$$

Unfortunately, there is no closed-form solution for the optimal parameter vector for LASSO. We must use numerical methods (like gradient descent).

Summary of Regression Methods

Name	Model	Loss	Reg.	Objective	Solution
OLS	$\hat{\mathbb{Y}} = \mathbb{X}\theta$	Squared loss	None	$\frac{1}{n}\|\|\mathbb{Y} - \mathbb{X}\theta\|\|_2^2$	$\hat{\theta}_{\text{OLS}} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}$
Ridge Regression	$\hat{\mathbb{Y}} = \mathbb{X}\theta$	Squared loss	L2	$\frac{1}{n}\|\|\mathbb{Y} - \mathbb{X}\theta\|\|_2^2 + \lambda\sum_{j=1}^{d}\theta_i^2$	$\hat{\theta}_{\text{ridge}} = (\mathbb{X}^T\mathbb{X} + n\lambda I)^{-1}\mathbb{X}^T\mathbb{Y}$
LASSO	$\hat{\mathbb{Y}} = \mathbb{X}\theta$	Squared loss	L1	$\frac{1}{n}\|\|\mathbb{Y} - \mathbb{X}\theta\|\|_2^2 + \lambda\sum_{j=1}^{d}\|\theta_i\|$	No closed form

Fitting vs. Evaluating

While we may use a regularized objective function to determine our model's parameters, we still look at **(root) mean squared error** to evaluate our model's performance.

$$\hat{\theta}_{\text{ridge}} = \arg\min_{\theta} \frac{1}{n}||\mathbb{Y} - \mathbb{X}\theta||_2^2 + \lambda \sum_{j=1}^{d} \theta_i^2$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbb{X}_i^T \hat{\theta}_{\text{ridge}})^2} = \sqrt{\frac{1}{n}||\mathbb{Y} - \mathbb{X}\hat{\theta}_{\text{ridge}}||_2^2}$$

The regularization penalty is there for the purposes of model fitting only.