

# Reducing Overfit

## Lecture 9

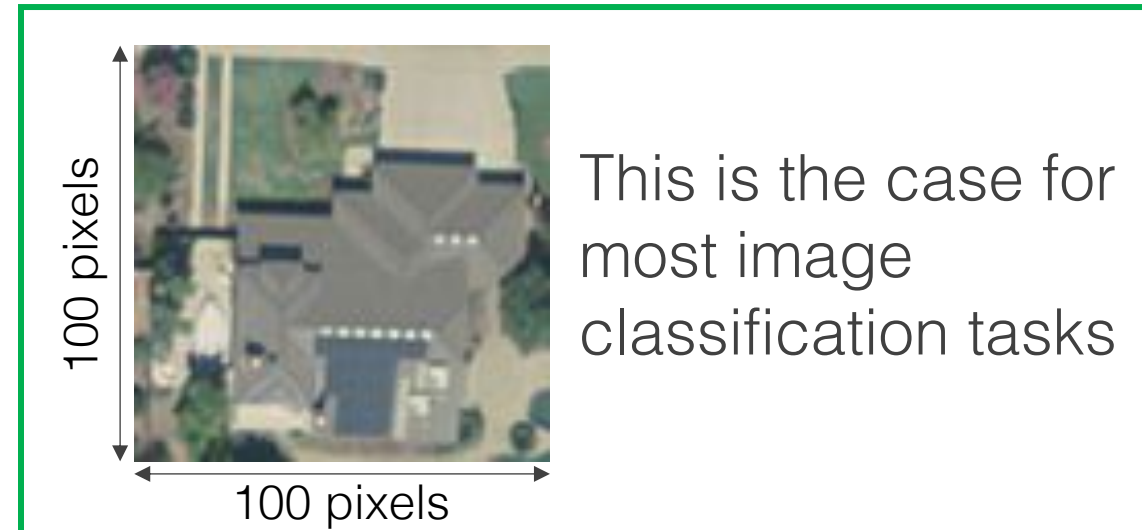
# Challenge

You have a dataset with  $n = 1,000$  samples (observations)

Each observation has  $p = 10,000$  predictors (features)

You're asked to develop a classifier for the data

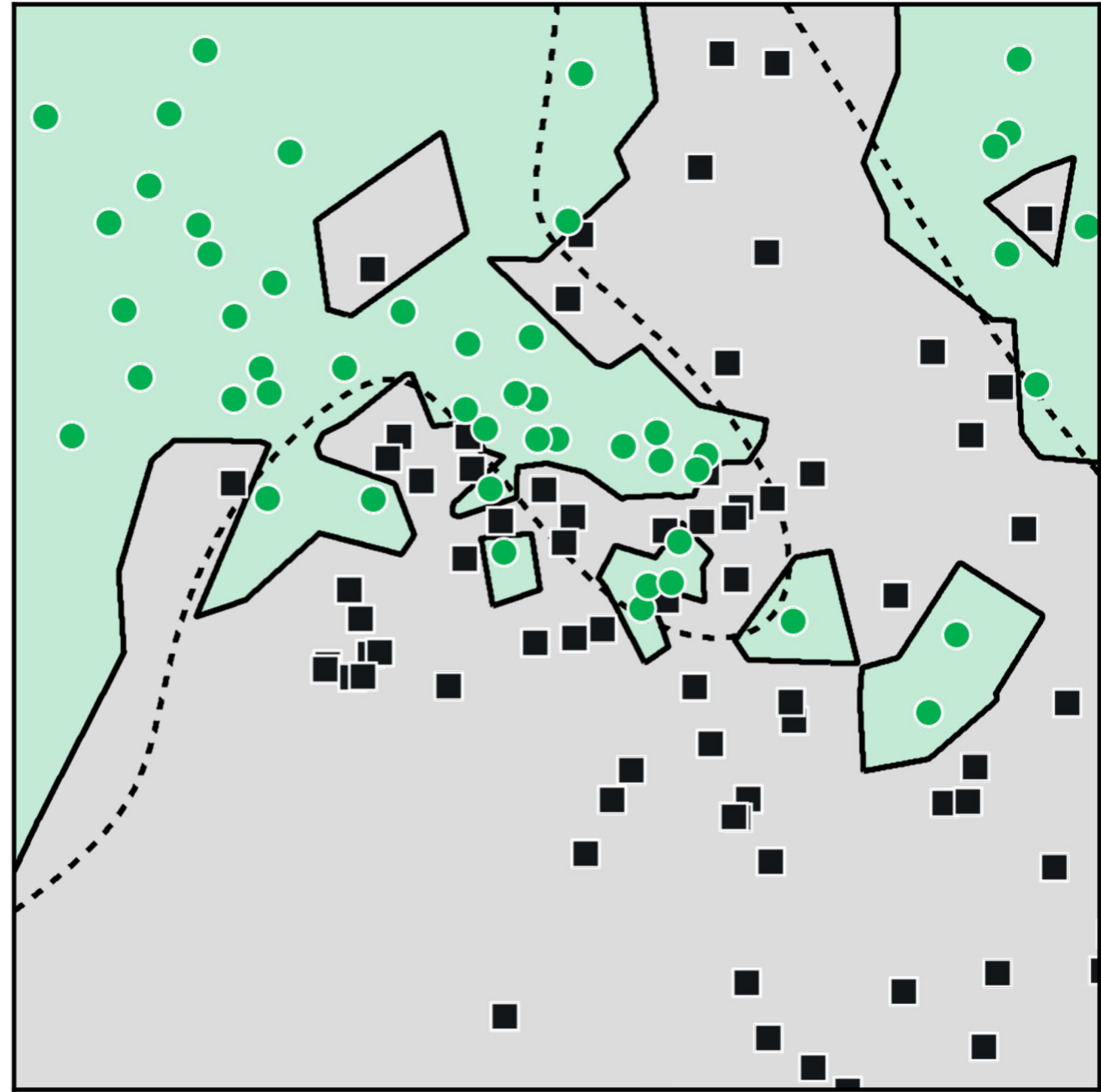
$p \gg n$  ....what do you do?



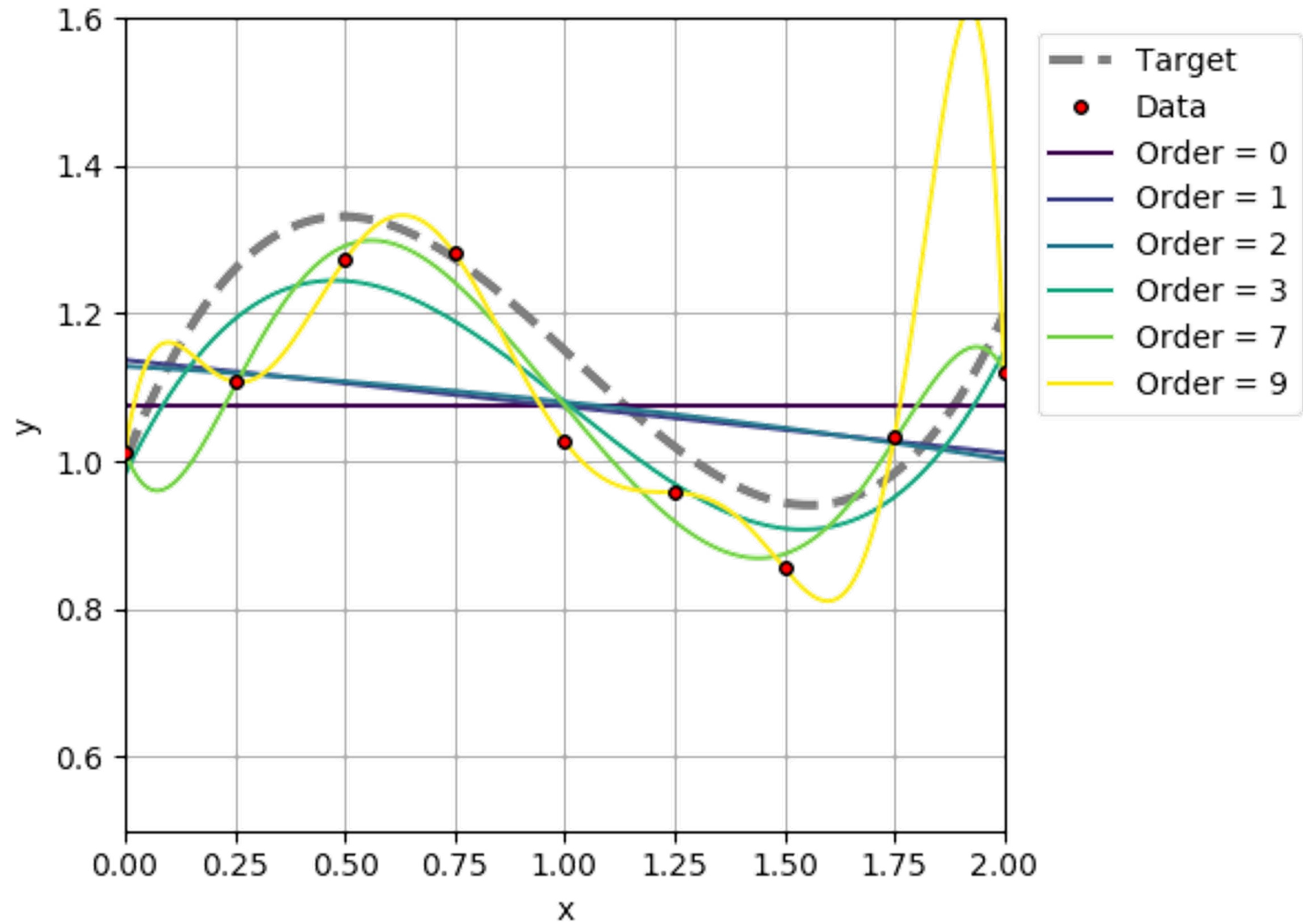
# A problem...

**Overfitting** to the  
training data

High model **variance**



# Overfitting to the training data



# How do we reduce model variance?

**Option #1: Add more data!**

**Option #2: Reduce model flexibility to  
reduce variance**

# Our conceptual tool...



Image from Speckyboy.com

# Occam's Razor / Law of Parsimony

All else being equal, choose the **simpler** solution



# Options for reducing variance

1. Variable/feature subset selection

2. Regularization/shrinkage

3. Dimensionality reduction  
(in a lecture coming soon!)

**These all reduce  
the number of  
features modeled  
and/or model  
flexibility**

# Benefits of reducing the number of predictors/features

Some algorithms scale poorly with increased dimensions (computationally)

Irrelevant and redundant features can confuse algorithms - removal of these features can increase generalization performance

Often reduces training data needs

# Feature (variable) selection

Filter methods

(e.g. remove correlated features)

Wrapper methods

(e.g. subset selection)

Embedded methods

(e.g. LASSO regularization)

# Variable subset selection: wrapper methods for feature selection

Search for subsets of features that perform well

- Exhaustive search
- Forward selection
- Backwards selection
- Simulated annealing
- Genetic algorithms
- Particle swarm optimization

**Challenge:** requires rerunning the training algorithm (computationally expensive)

# Forward selection

- Start with no features
- Greedily include the one feature that most improves performance
- Stop when a desired number of features is reached

# Backward selection

- Start with all features included
- Greedily remove the feature that decreases performance least
- Stop when a desired number of features is reached

Challenge: requires rerunning the training algorithm (computationally expensive)

# Regularization

## methods for variance reduction

Reduce the variance by simplifying the model during training

**Techniques that reduce generalization error, but NOT training error**

# Recall the model fitting process

1. Choose a **hypothesis set of models** to train  
(e.g. linear regression with  $p$  predictor variables)
2. Identify a **cost function** to measure the model fit to the training data  
(e.g. mean square error)
3. **Optimize** model **parameters** to minimize cost  
(e.g. ordinary least squares or gradient descent)

# Regularization

a.k.a. shrinkage

Adjust the **cost/loss function** to penalize larger parameters

$$L(\mathbf{w}) = \sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i - y_n)^2 + \lambda \sum_{i=1}^p w_i^2$$

↑  
Square error

↑  
**L<sub>2</sub> regularization penalty**

This term causes the estimated parameter values to “shrink”

More generally:  $L(\mathbf{w}) = C(\mathbf{w}, \mathbf{X}, \mathbf{y}) + \lambda R(\mathbf{w})$



# Norms



Images from Wikipedia, Norm MacDonald photo by playerx licensed under CC BY 2.0

# Norms

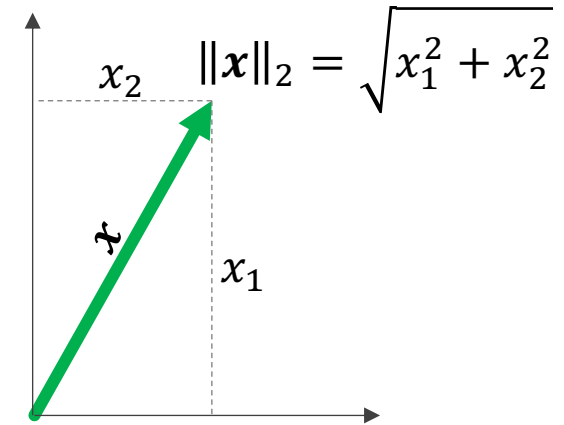
A function that assigns a positive **length or size** to a vector

The most familiar is likely the **Euclidean**, or  $L_2$  norm:

$$\|\mathbf{x}\|_2 \triangleq \sqrt{x_1^2 + \dots + x_n^2} = \left( \sum_{i=1}^n x_i^2 \right)^{\frac{1}{2}} = \sqrt{\mathbf{x}^T \mathbf{x}}$$

You'll often see this in its squared form:

$$\|\mathbf{x}\|_2^2 \triangleq x_1^2 + \dots + x_n^2 = \sum_{i=1}^n x_i^2 = \mathbf{x}^T \mathbf{x}$$

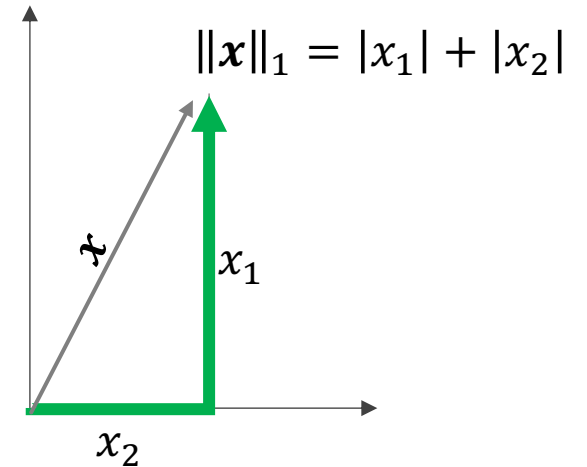


# Norms

There's also the  **$L_1$  norm**

(a.k.a taxicab or Manhattan distance)

$$\|\mathbf{x}\|_1 \triangleq |x_1| + \cdots + |x_n| = \sum_{i=1}^n |x_i|$$



The general  **$L_p$  norm**:

$$\|\mathbf{x}\|_p \triangleq \left( \sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}}$$

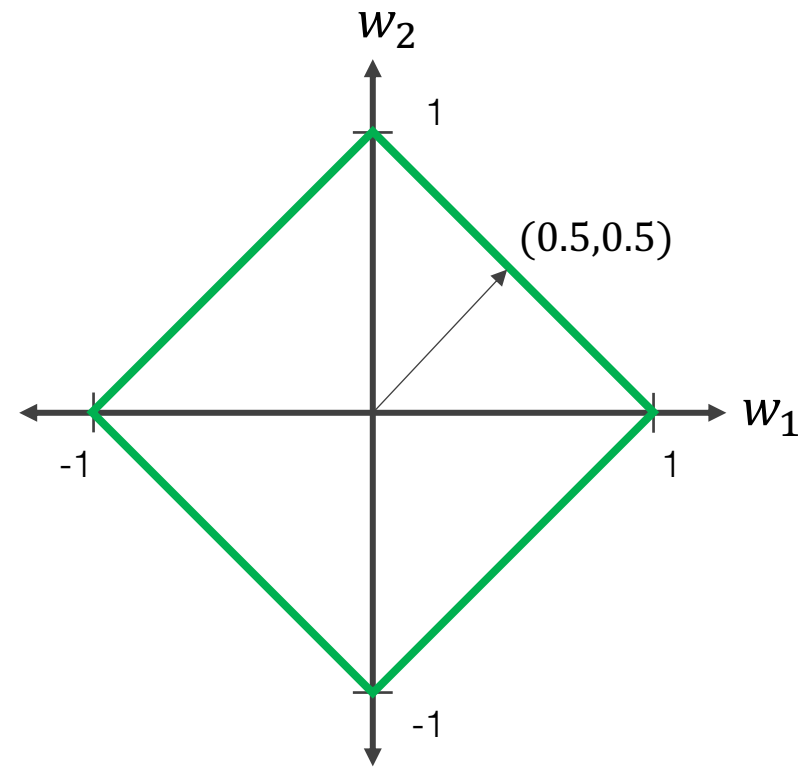
In the limit, the **infinity norm** is the maximum entry of the vector  $\mathbf{x}$ :

$$\|\mathbf{x}\|_\infty \triangleq \max_i |x_i|$$

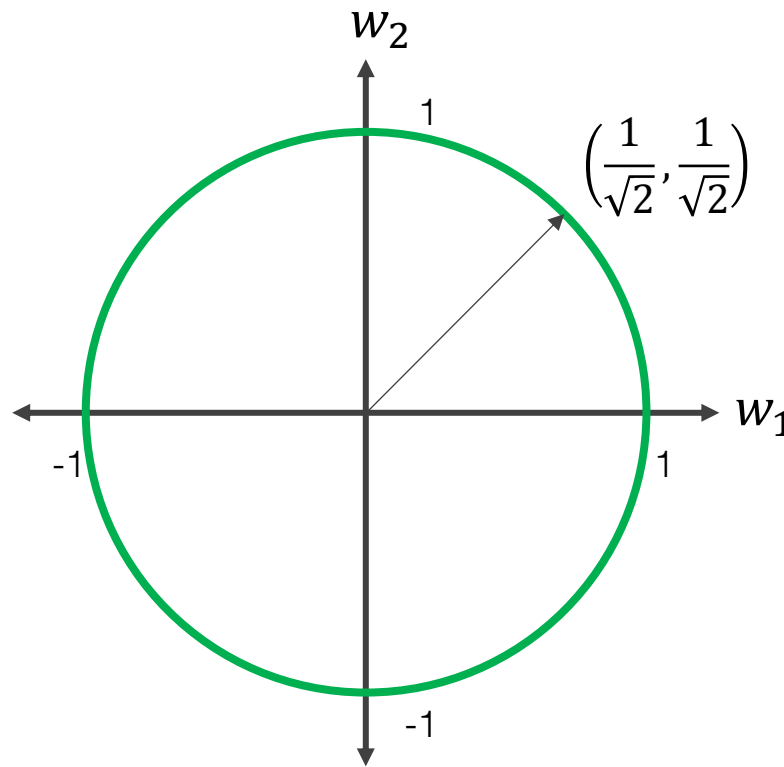
# Norms of length 1

Assume a 2-D vector whose origin is (0,0):  $\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$

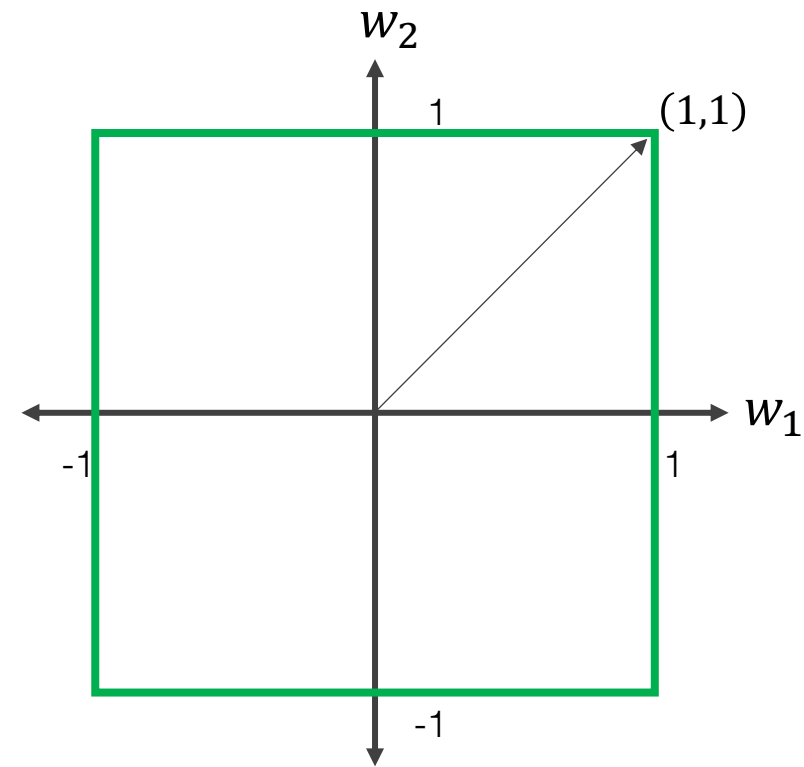
$$\|\mathbf{w}\|_1 = 1$$



$$\|\mathbf{w}\|_2 = 1$$



$$\|\mathbf{w}\|_\infty = 1$$



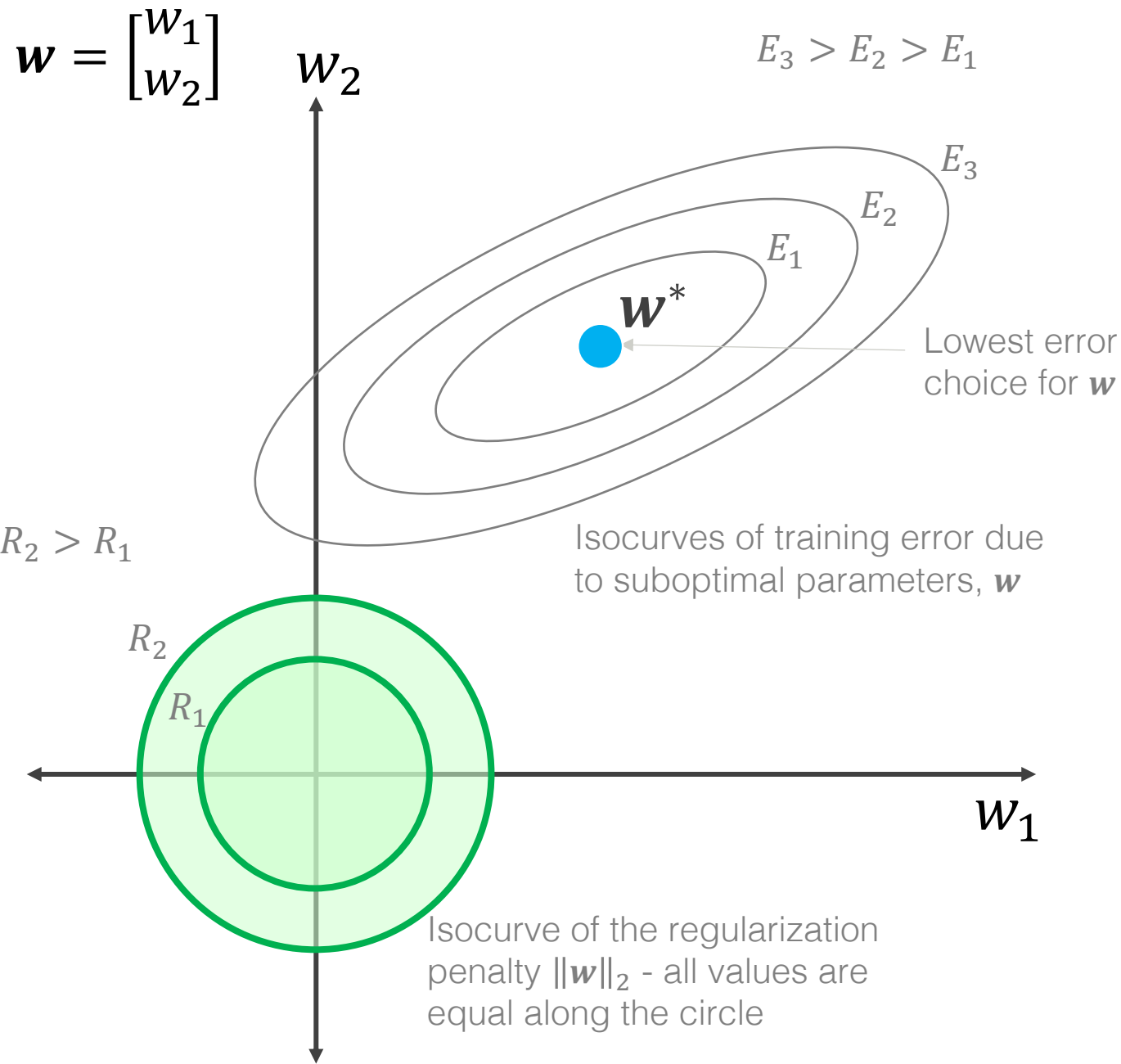
# Regularization

a.k.a. shrinkage

Adjust the **cost/loss function** to penalize larger parameter values

a.k.a....

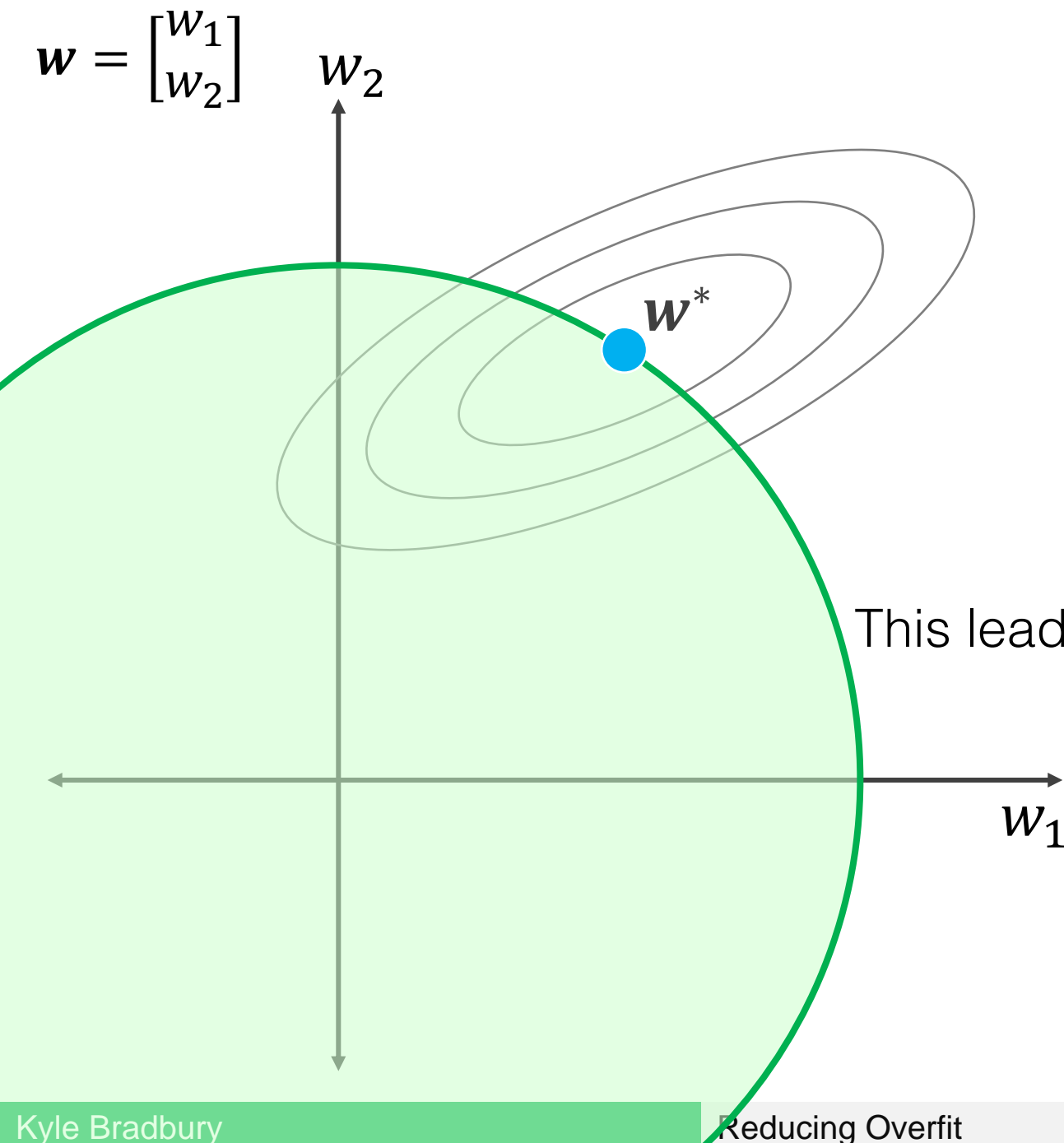
$L_2$ regularization	$L(\mathbf{w}) = \sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i - y_n)^2 + \lambda \sum_{i=1}^p w_i^2$	<b>ridge regression</b> or weight decay (Tikhonov regularization)
$L_1$ regularization	$L(\mathbf{w}) = \sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i - y_n)^2 + \lambda \sum_{i=1}^p  w_i $	least absolute shrinkage and selection operator ( <b>LASSO</b> )
$L_2$ & $L_1$ regularization	$L(\mathbf{w}) = \sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i - y_n)^2 + \lambda_1 \sum_{i=1}^p  w_i  + \lambda_2 \sum_{i=1}^p w_i^2$	<b>elastic net</b> regularization



Trying to minimize my loss function:

$$L(\mathbf{w}) = \underbrace{\sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i - y_n)^2}_{\text{Error term (E)}} + \underbrace{\lambda \sum_{i=1}^p w_i^2}_{\text{Regularization penalty (R)}}$$

First attempt let's just minimize error

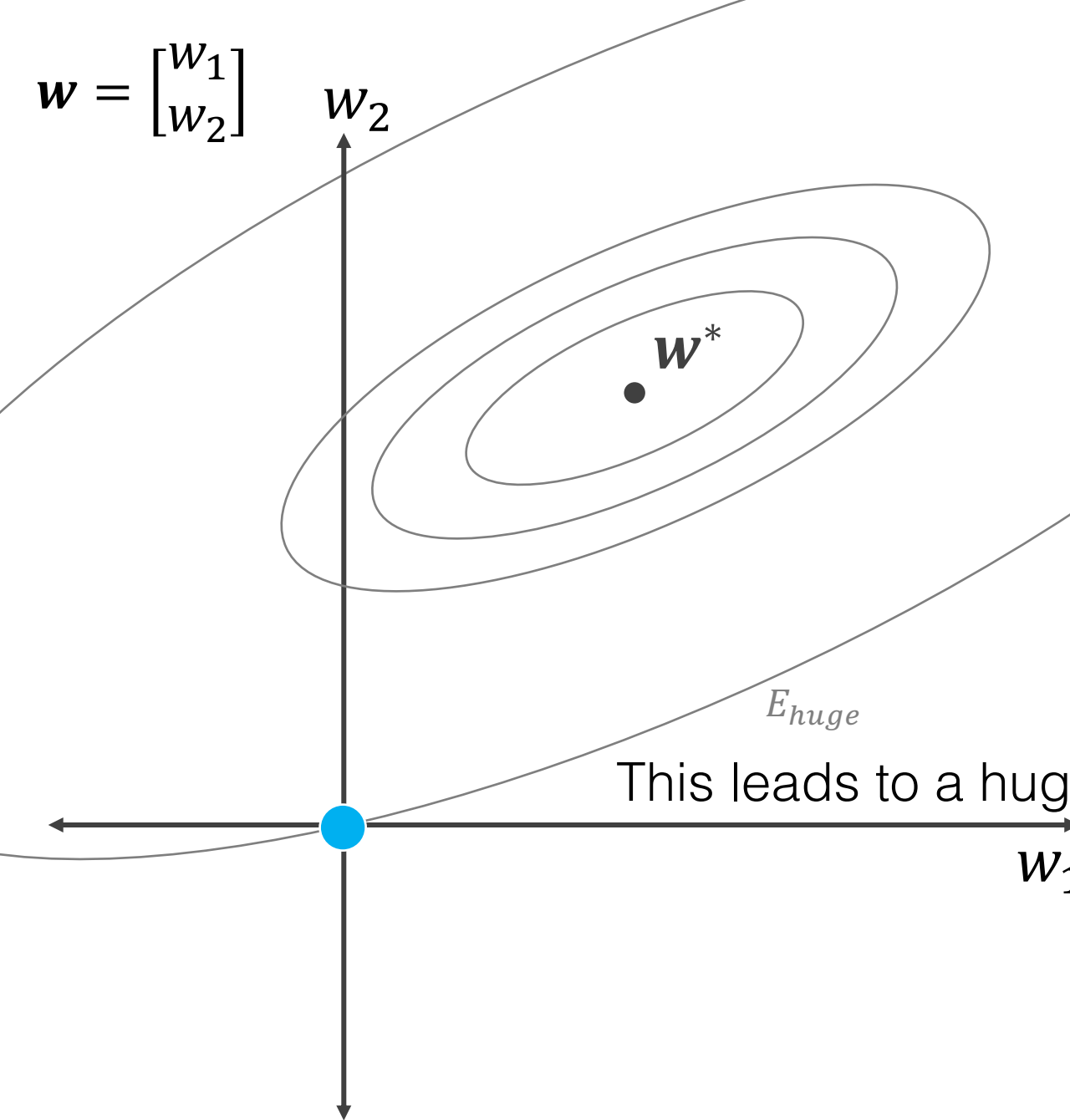


Trying to minimize my loss function:

$$L(\mathbf{w}) = \underbrace{\sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i - y_n)^2}_{\text{Error term}} + \underbrace{\lambda \sum_{i=1}^p w_i^2}_{\text{Regularization penalty}}$$

First attempt let's just minimize error

This leads to a huge regularization penalty



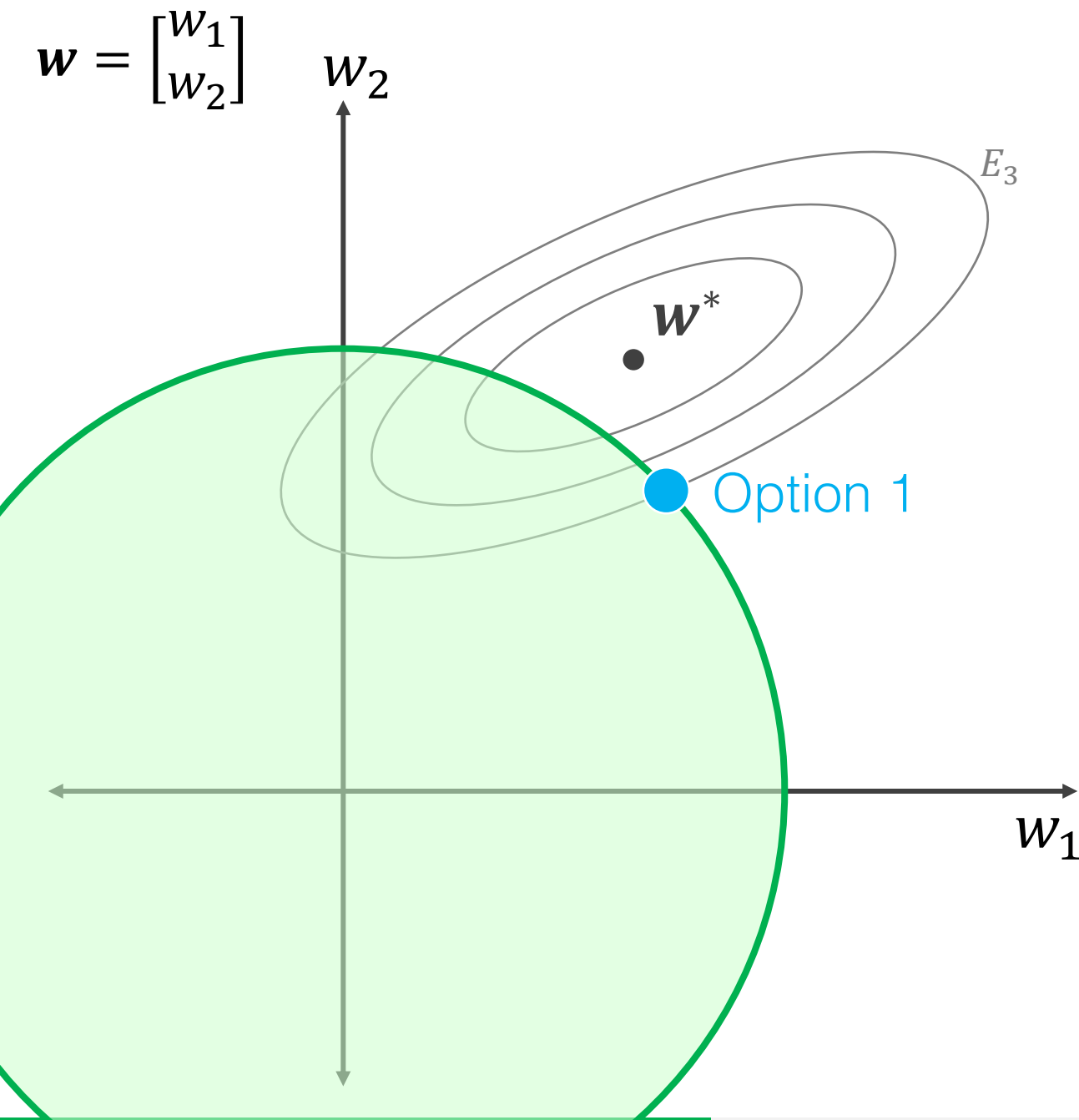
Trying to minimize my loss function:

$$L(\mathbf{w}) = \underbrace{\sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i - y_n)^2}_{\text{Error term}} + \underbrace{\lambda \sum_{i=1}^p w_i^2}_{\text{Regularization penalty}}$$

Instead, we could just minimize the regularization penalty

This leads to a huge error term...

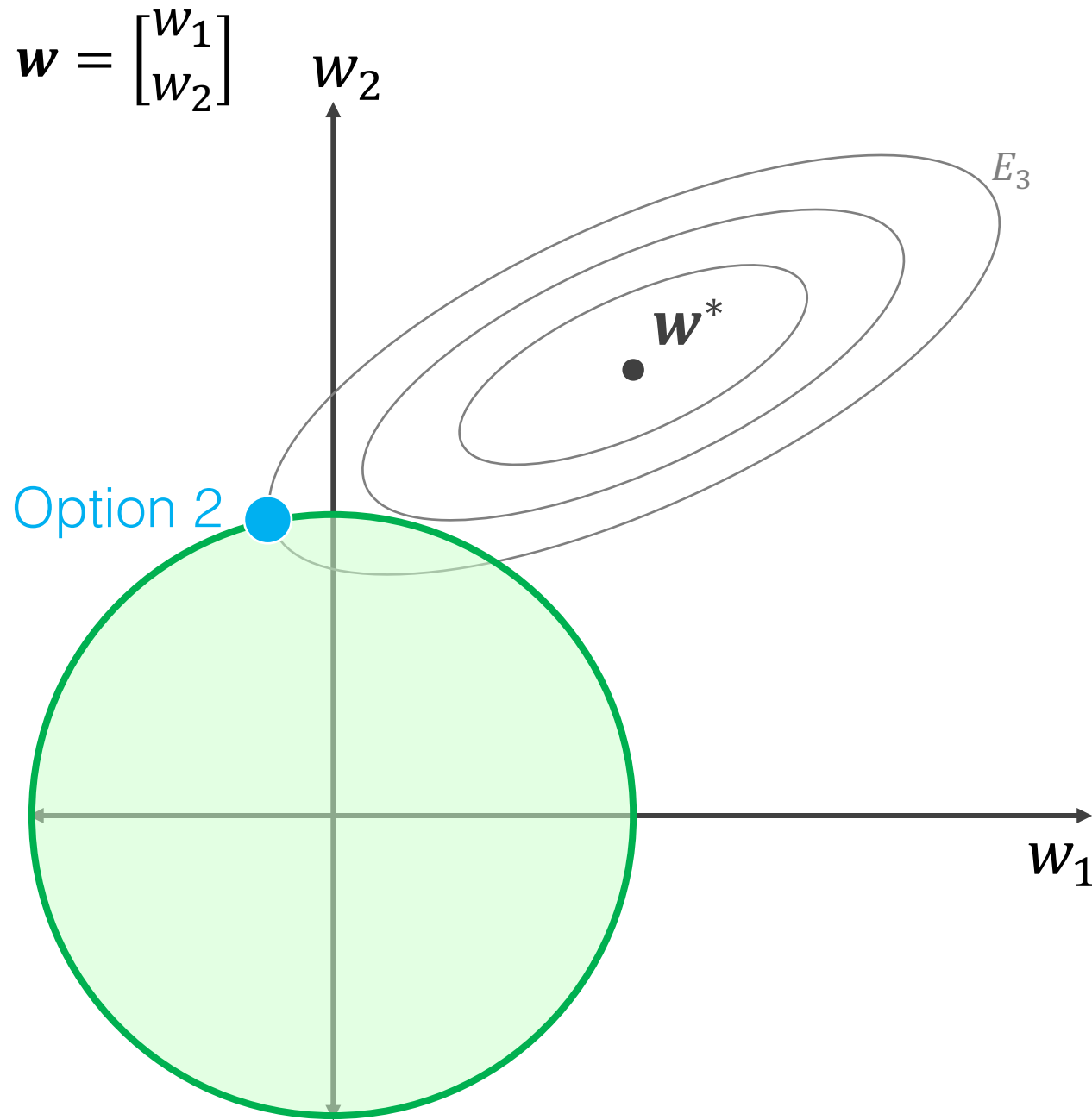




Trying to minimize my loss function:

$$L(\mathbf{w}) = \underbrace{\sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i - y_n)^2}_{\text{Error term}} + \underbrace{\lambda \sum_{i=1}^p w_i^2}_{\text{Regularization penalty}}$$

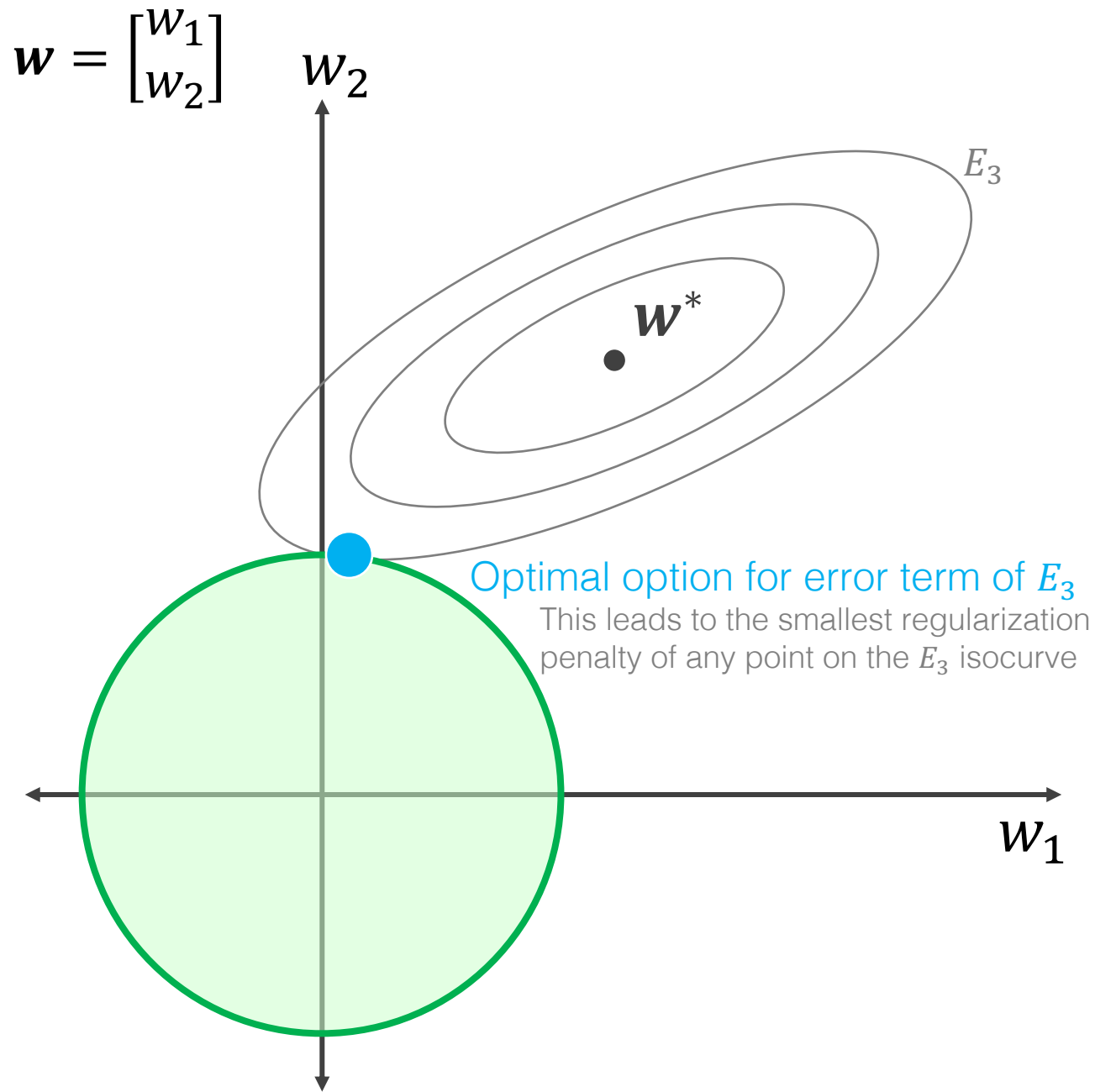
For any level of error (assume  $E_3$  here), there may be a number of parameter values that result in an equal error term



Trying to minimize my loss function:

$$L(\mathbf{w}) = \underbrace{\sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i - y_n)^2}_{\text{Error term}} + \underbrace{\lambda \sum_{i=1}^p w_i^2}_{\text{Regularization penalty}}$$

For any level of error (assume  $E_3$  here), there may be a number of parameter values that result in an equal error term

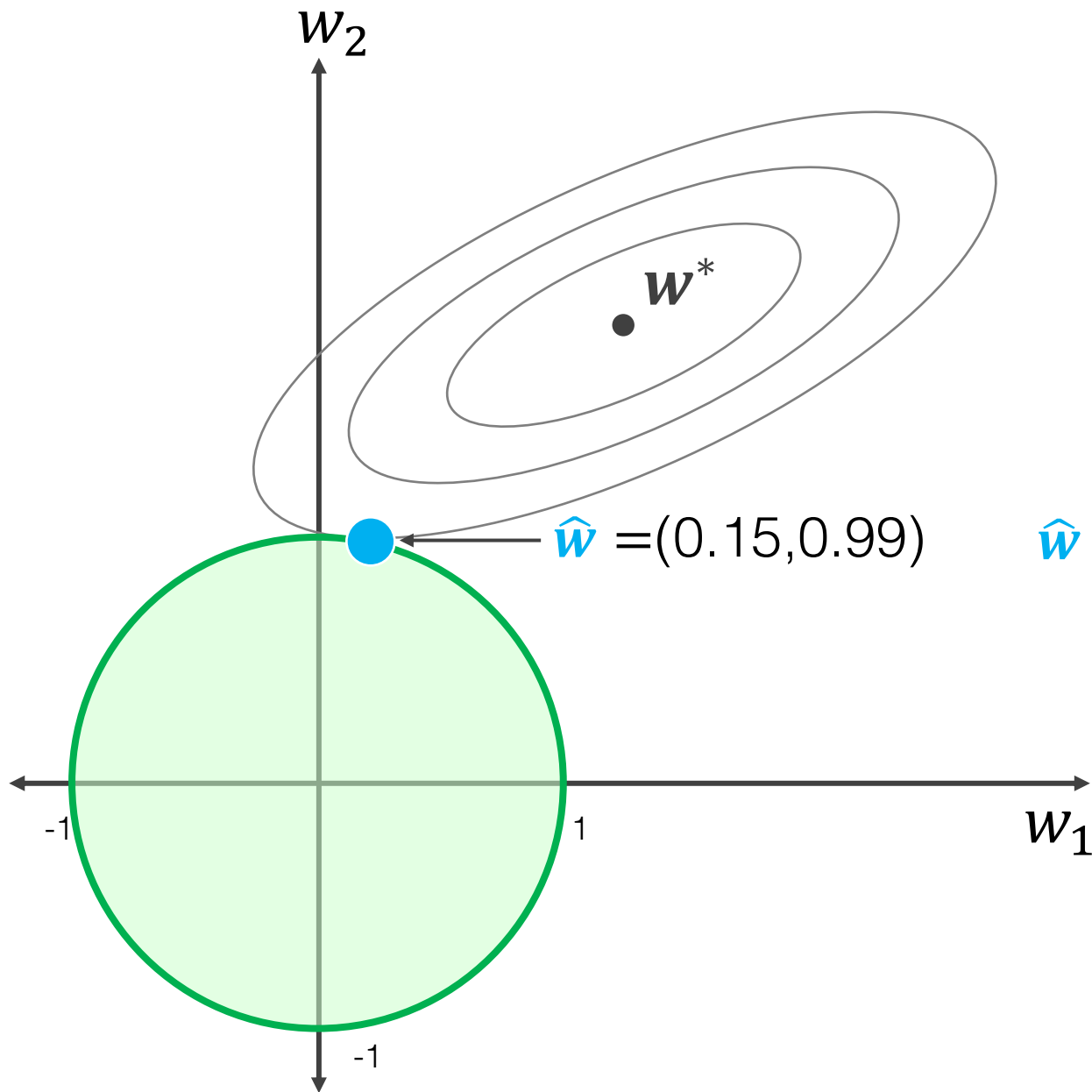


Trying to minimize my loss function:

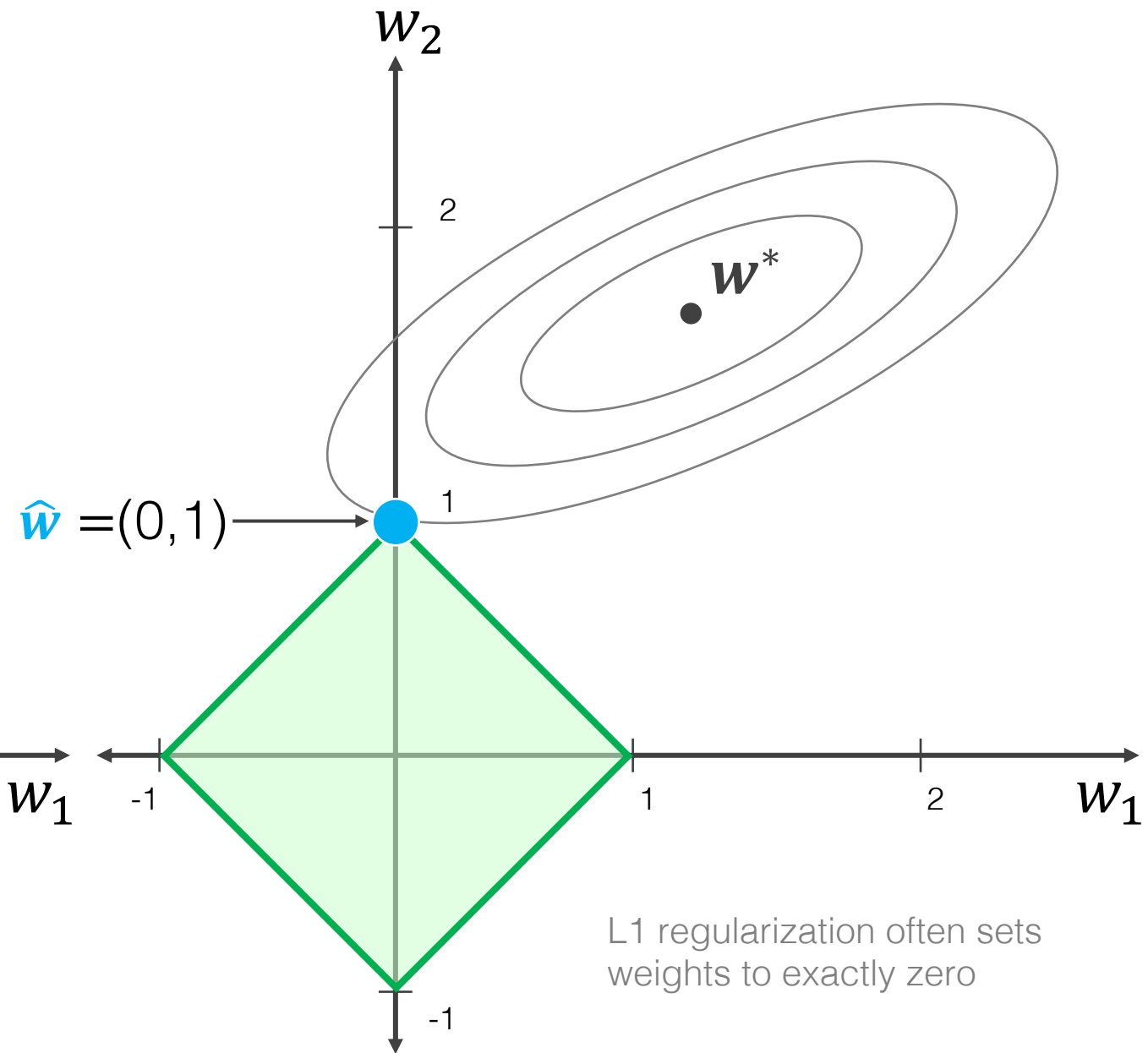
$$L(\mathbf{w}) = \underbrace{\sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i - y_n)^2}_{\text{Error term}} + \underbrace{\lambda \sum_{i=1}^p w_i^2}_{\text{Regularization penalty}}$$

However, we can choose between the options by minimizing the regularization penalty

## Ridge: $L_2$ regularization



## LASSO: $L_1$ regularization



# Regularization reduces variance

Leads to smaller model parameters

$L_1$  regularization also performs variable selection

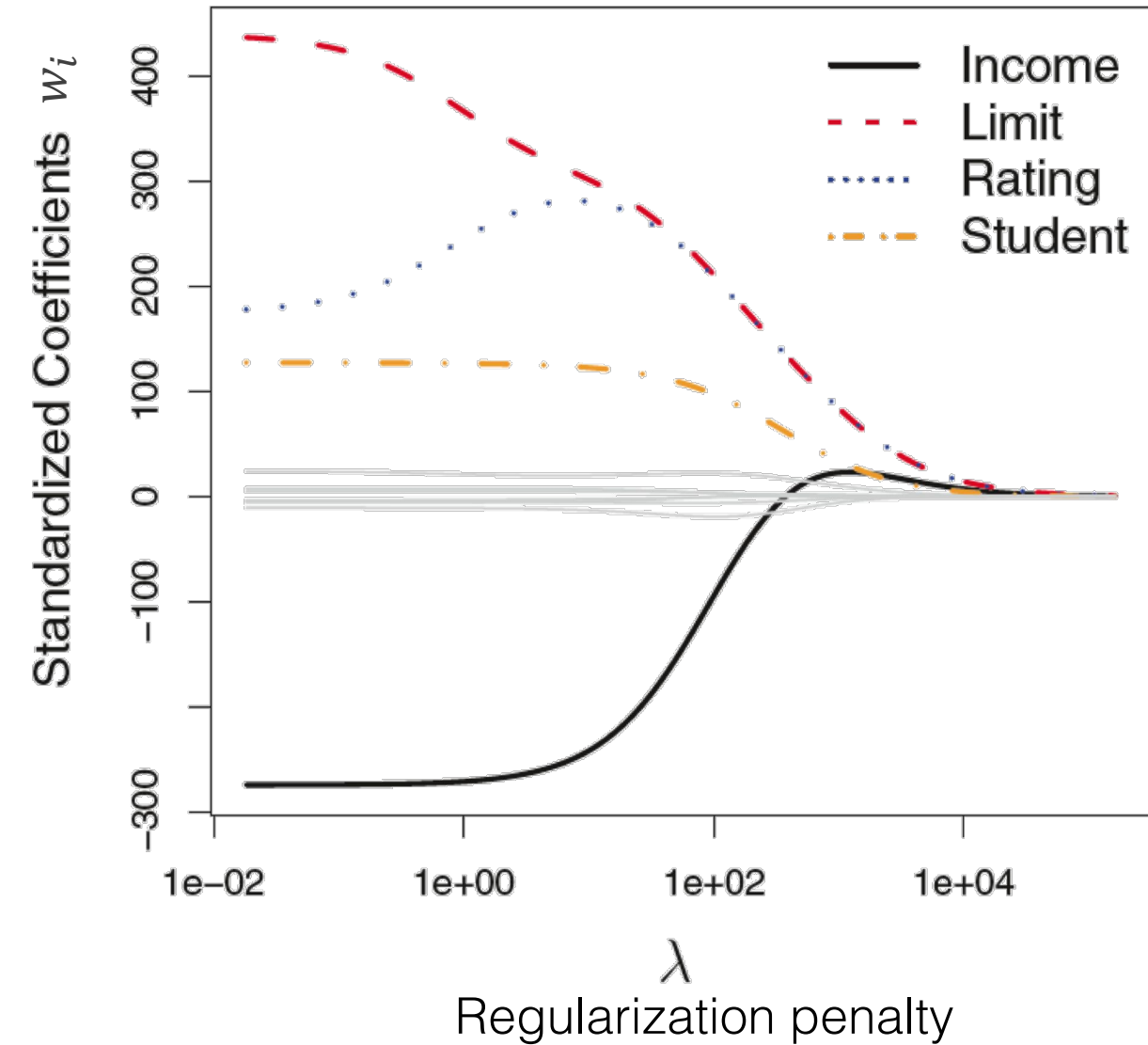
# Example: predicting credit default

11 features to use to predict default:

- Income
- Credit limit
- Credit rating
- Credit balance
- Number of credit cards
- Age
- Education
- Gender
- Student status
- Ethnicity
- Marriage status

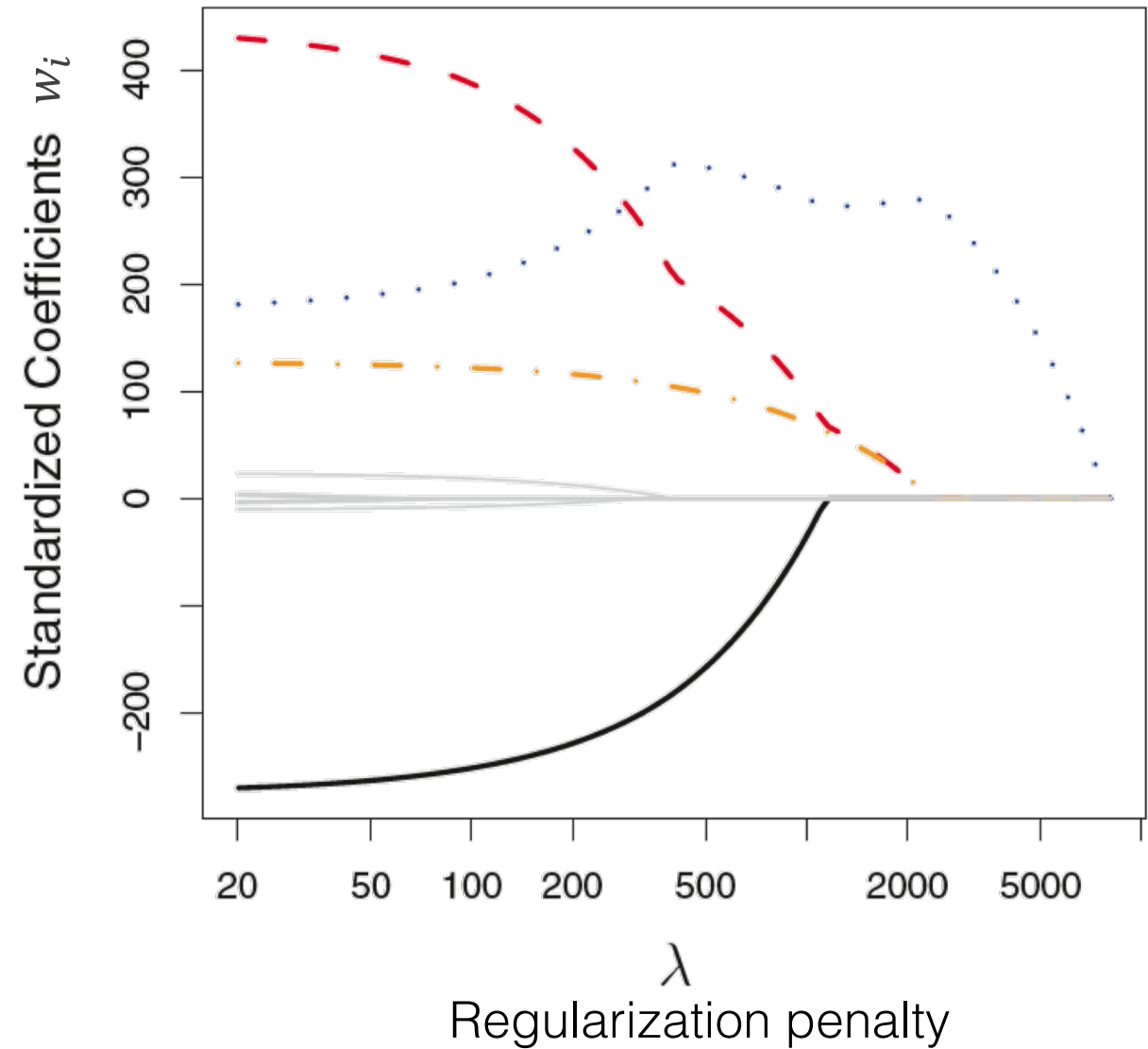
## $L_2$ regularization

Ridge regression

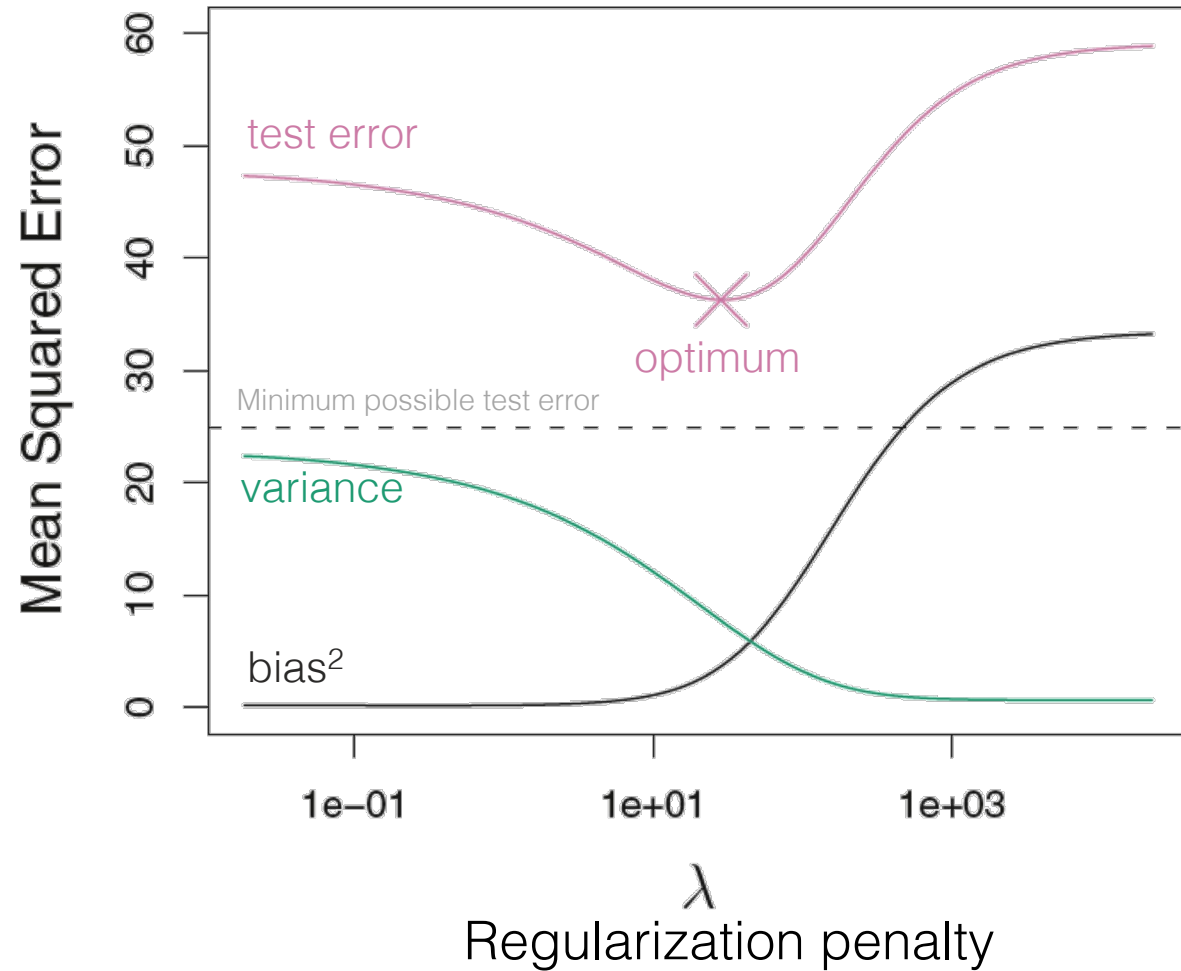


## $L_1$ regularization

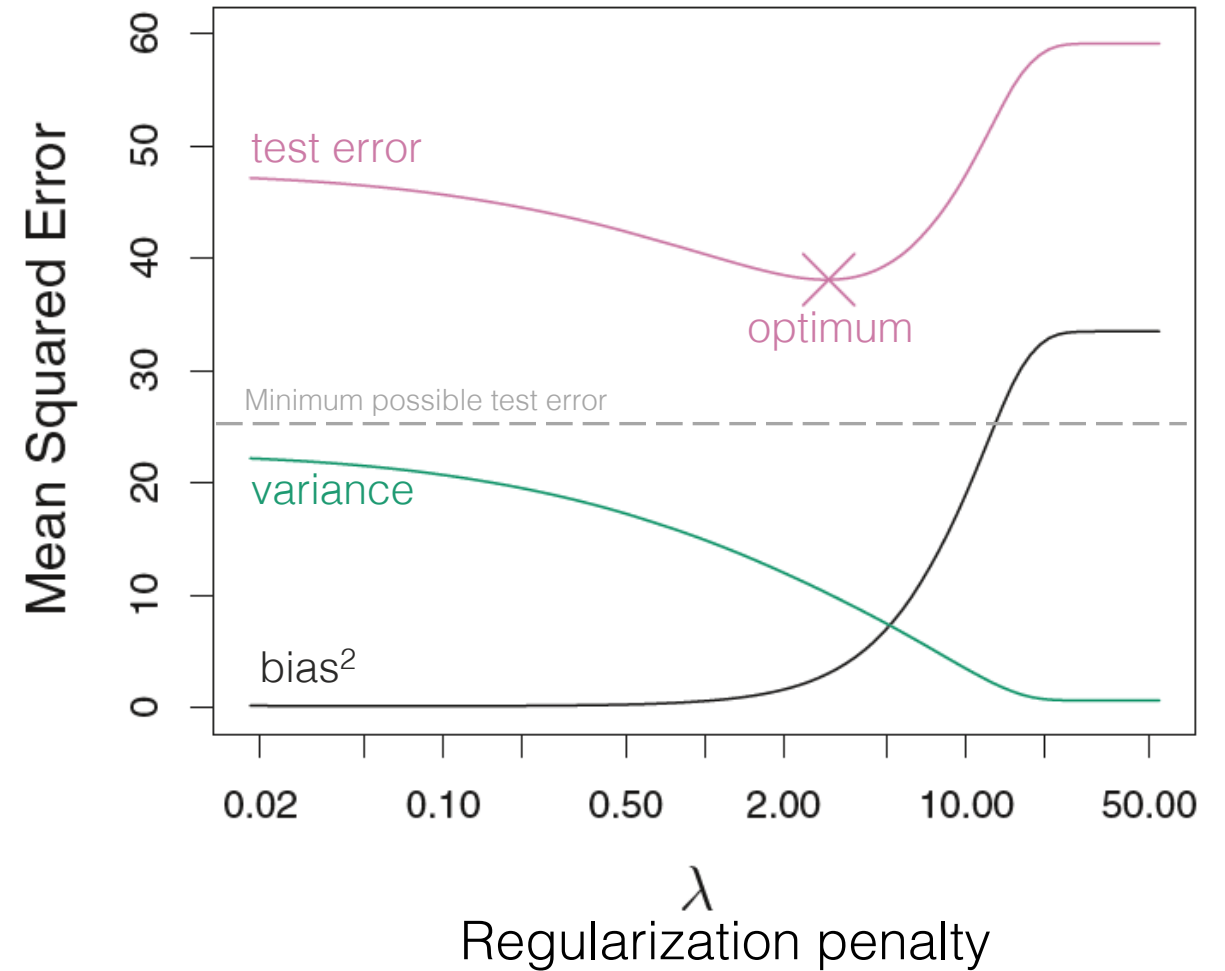
LASSO regularization



## $L_2$ regularization



## $L_1$ regularization





# Underdetermined systems and OLS

$$X = \begin{bmatrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}$$

Number of features  $p$

$N$   
Number of samples

If  $p > N$ , then the system is **underdetermined**

Often means there are infinitely many solutions

Ridge regression makes this problem solvable

# Choosing the regularization parameter $\lambda$

- $\lambda$  is a hyperparameter
- Use a training, validation, and test set
- Can also apply nested cross validation

**Train**

Used for model training / fitting

**Validation**

Used to  
approximate  
generalization  
performance and  
optimize  
hyperparameters

**Test**

Used to evaluate  
generalization  
performance of the  
final model(s)

# Strengths of $L_1$ and $L_2$ regularization

Ridge regression ( $L_2$  regularization) handles **multicollinearity** well

LASSO regularization ( $L_1$  regularization) reduces the number of predictors in a model (yields **sparse** models)

LASSO selects among redundant features

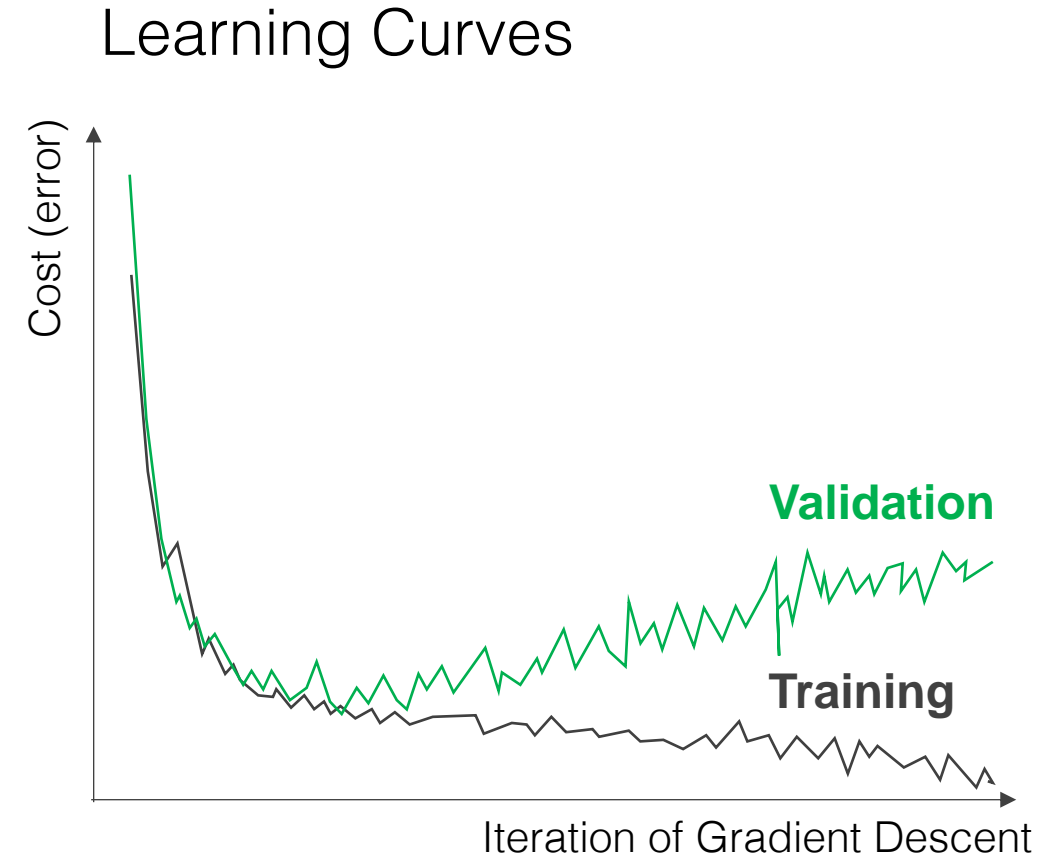
You can use a little of both via elastic net regularization

# One more approach: Early Stopping

Iterative learning (training) methods, (e.g. gradient descent) tend to learn more complex models over time

Stop the fitting process earlier, before overfit has occurred

Common in neural network training



# Takeaways

Reducing the number of features in a model may improve generalization error by reducing overfit

Overly flexible models can be regularized to reduce overfit (reducing variance)

$L_1$  and  $L_2$  regularization are effective tools for battling overfit