Reducing Overfit

Lecture 11

Our quest to generalize...

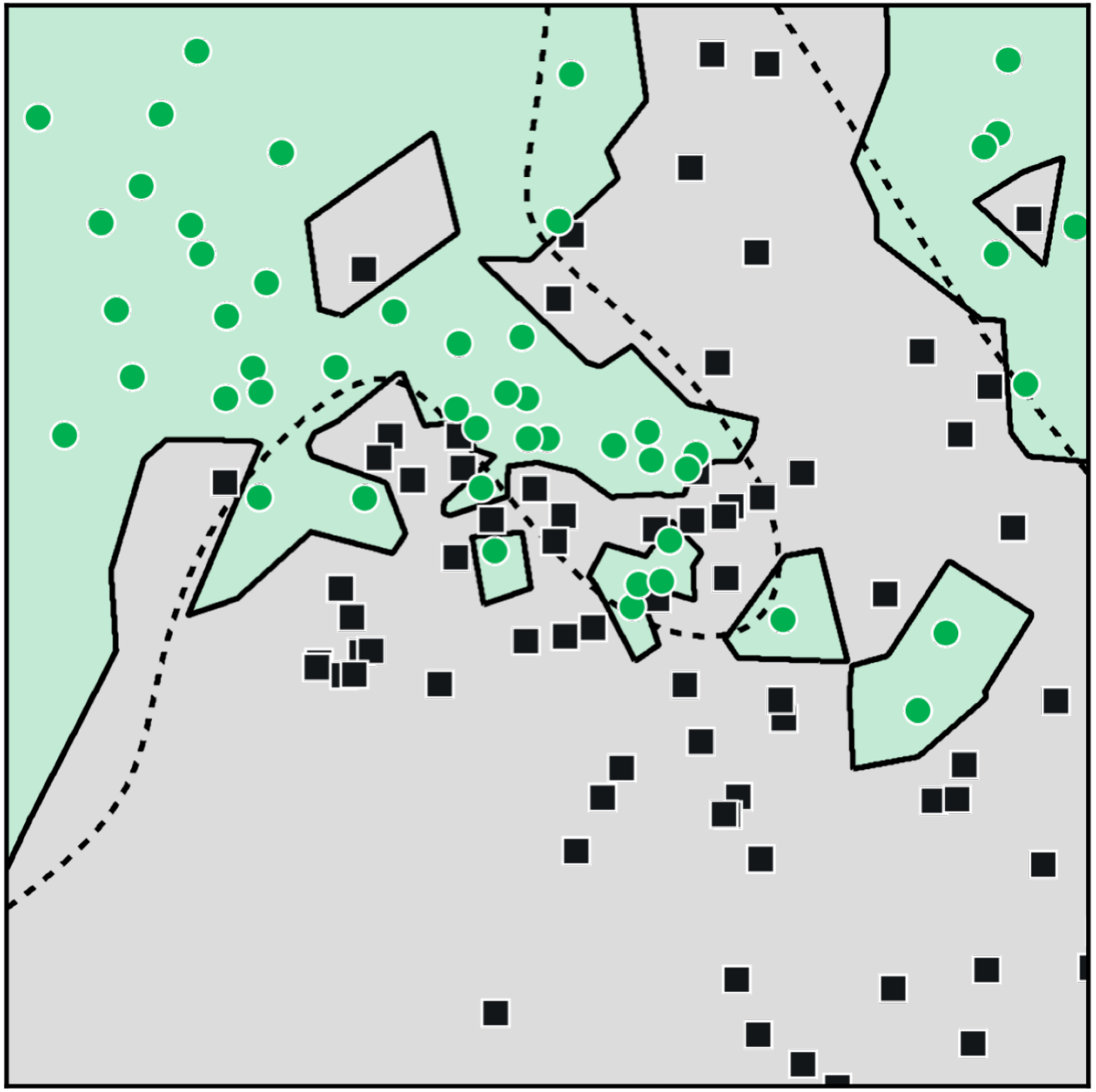
...is also our quest to **prevent overfit**

1. Use cross validated performance evaluation to accurately measure generalization performance

2. Reduce the complexity of flexible models

Our problem…

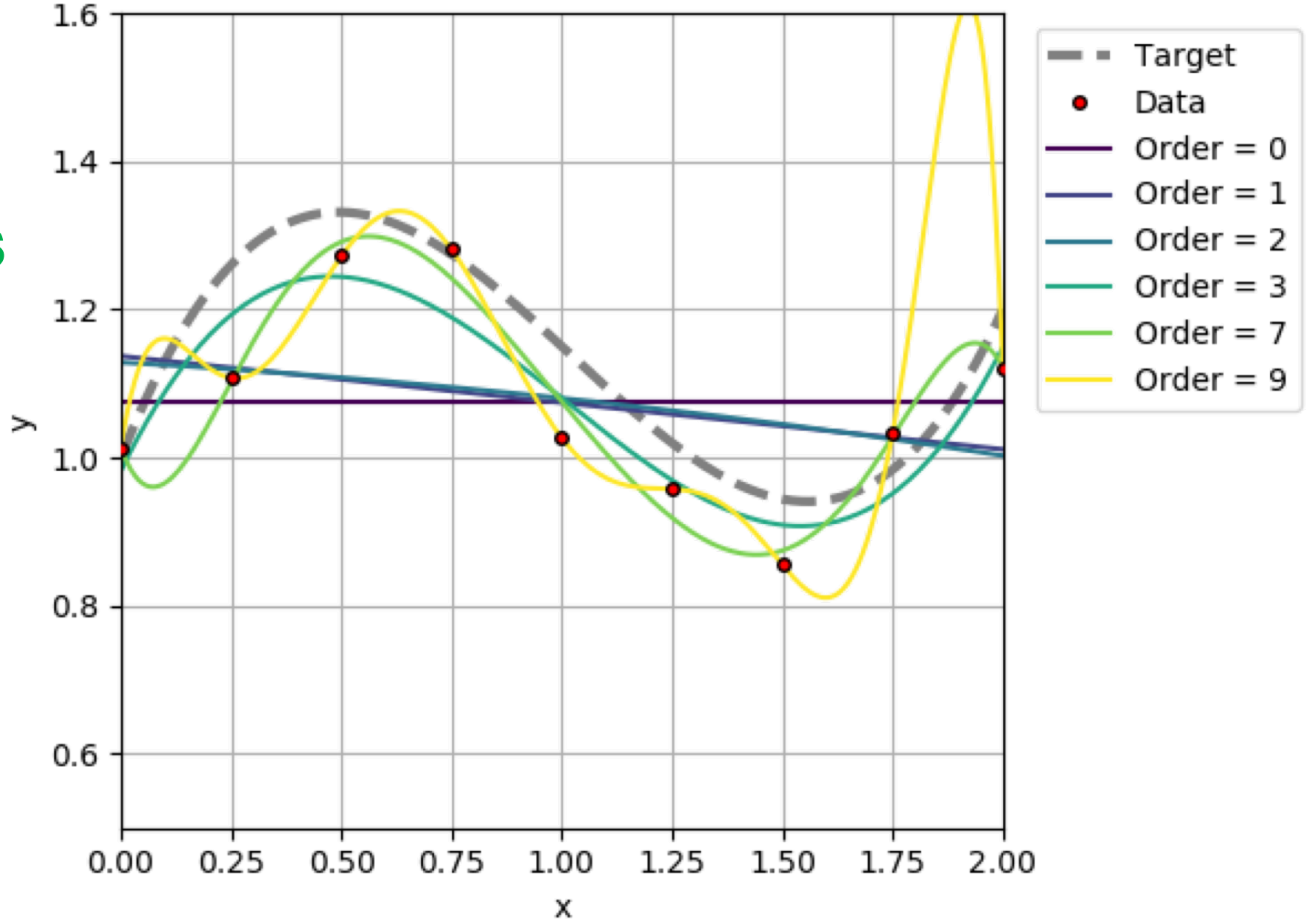
Overfitting to the training data



Linear regression model with 9 features **overfits** the training data

$$\hat{y}_i = \sum_{j=0}^{N} a_j x_i^j$$

N is the model order



Overfitting and high model variance are related… we want to reduce both!

Our tool…

Image from Speckyboy.com

Occam's Razor / Law of Parsimony

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

Option 1: reduce the number of features

- Some algorithms scale poorly with increased dimensions (computationally)
- Irrelevant and redundant features can confuse algorithms
- Removal of features can increase generalization performance
- Often reduces training data needs

Subset selection: wrapper methods

Search for subsets of features that perform well

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

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)

Option 2: Regularization

Reduce the variance by simplifying the model

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

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

Square error

L_2 regularization penalty

This term causes the estimated parameter values to "shrink"

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

Norms

Norms

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

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

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

You'll often see this in its squared form:

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

Norms

There's also the $\boldsymbol{L_1}$ **norm** (a.k.a taxicab or Manhattan distance)

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

The general $\boldsymbol{L_p}$ **norm**:

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

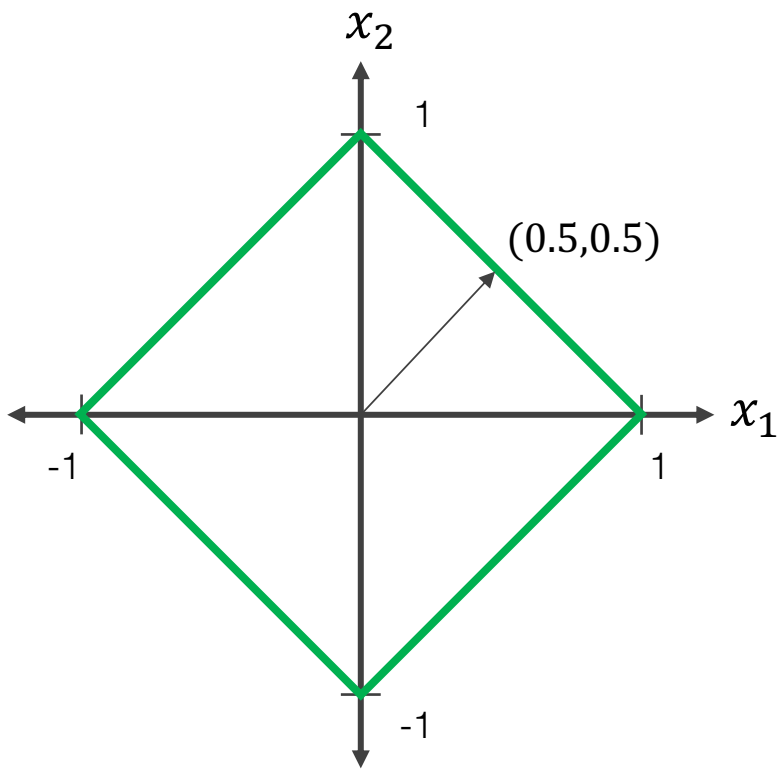
In the limit, the **infinity norm** is the maximum entry of the vector $\boldsymbol{x}$:

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

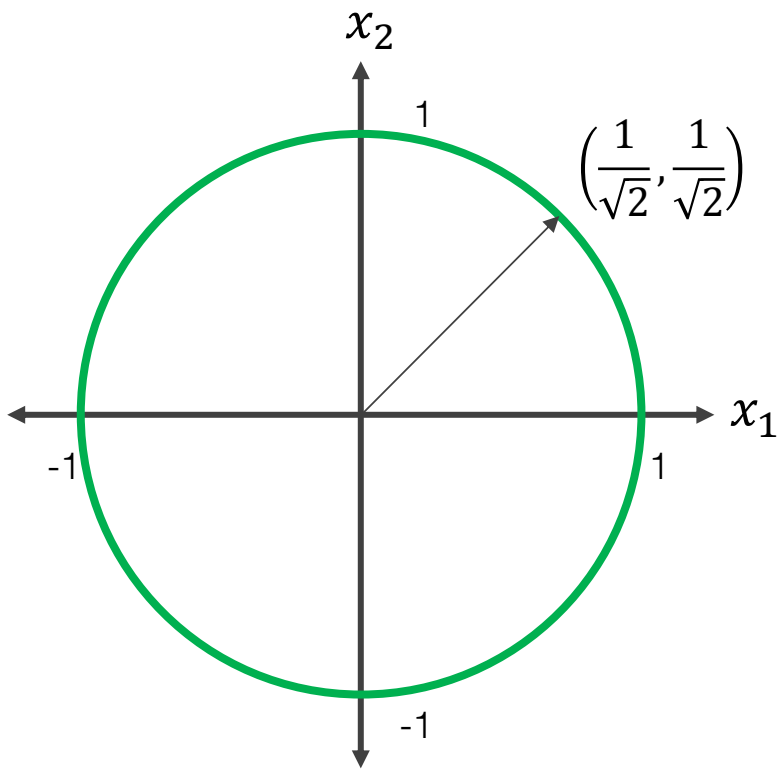
Norms of length 1

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

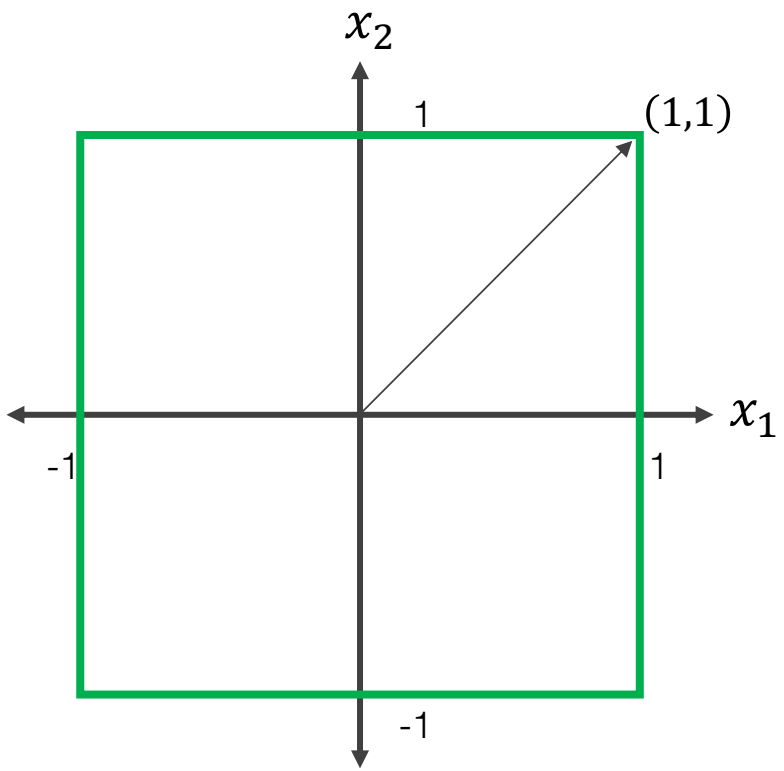
$\|\boldsymbol{w}\|_1 = 1$



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



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



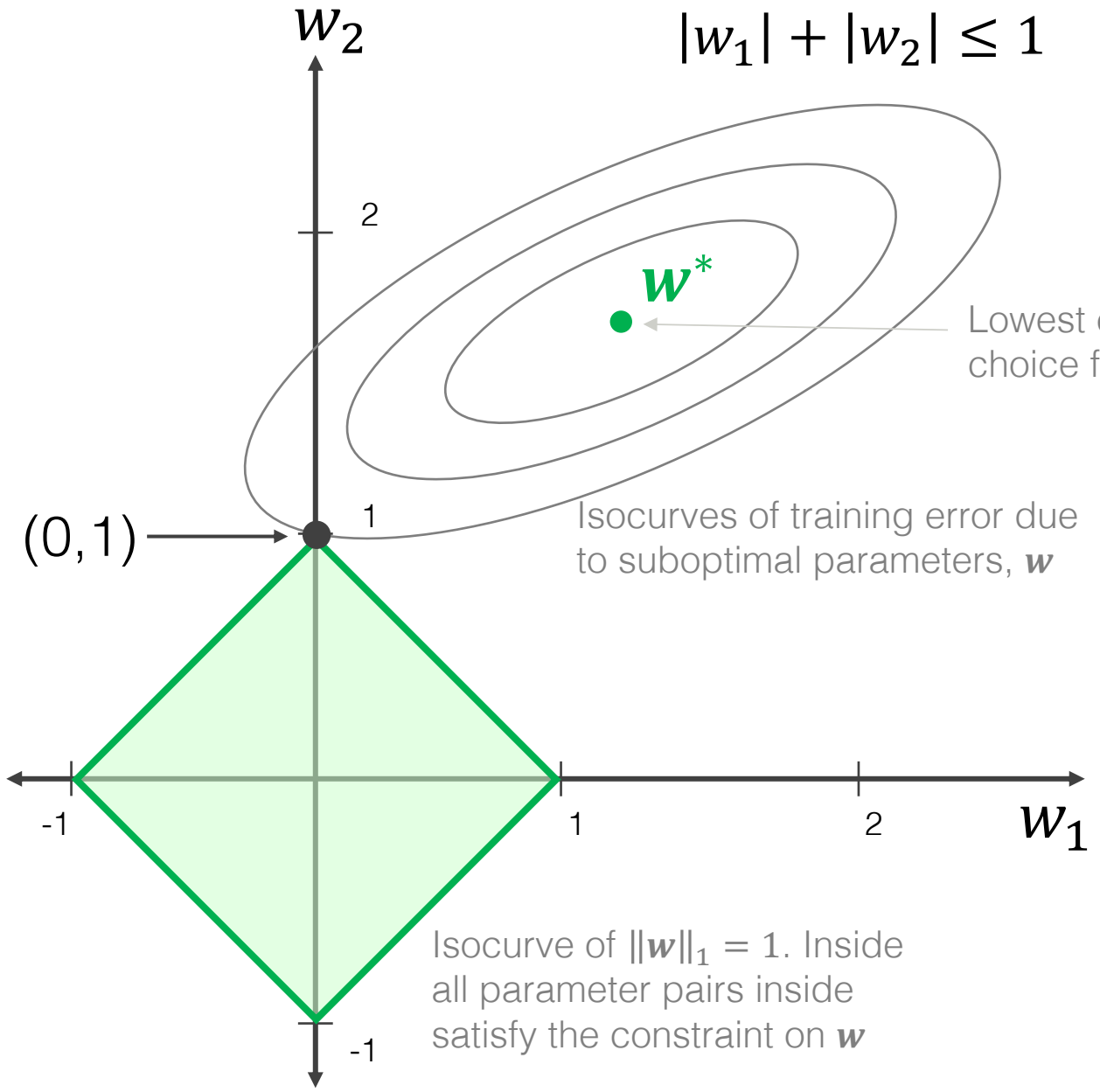
Regularization

a.k.a. shrinkage

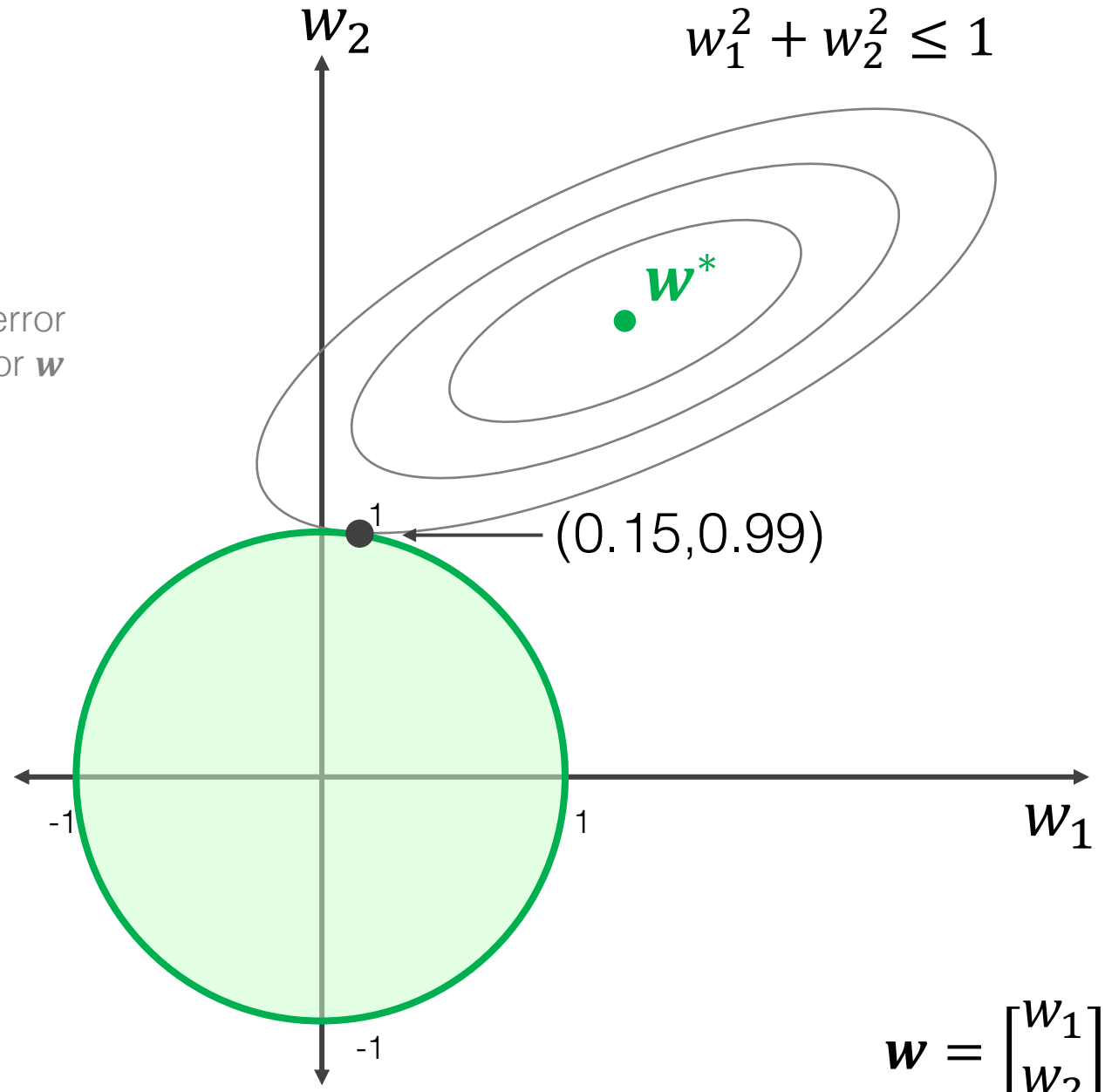
Adjust the **cost/loss function** to penalize larger parameter values

		a.k.a.…
L_2 loss/cost	$L(\boldsymbol{w}) = \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_n)^2 + \lambda \sum_{i=1}^{n} w_i^2$	**ridge regression** or weight decay
L_1 loss/cost	$L(\boldsymbol{w}) = \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_n)^2 + \lambda \sum_{i=1}^{n} \lvert w_i \rvert$	least absolute shrinkage and selection operator (**LASSO**)
L_2 and L_1 loss/cost	$L(\boldsymbol{w}) = \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_n)^2 + \lambda_1 \sum_{i=1}^{n} \lvert w_i \rvert + \lambda_2 \sum_{i=1}^{n} w_i^2$	**elastic net** regularization

Apply a constraint: $\|\boldsymbol{w}\|_1 \leq 1$



Apply a constraint: $\|\boldsymbol{w}\|_2^2 \leq 1$



$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

Regularization reduces variance

L_1 regularization also performs variable selection

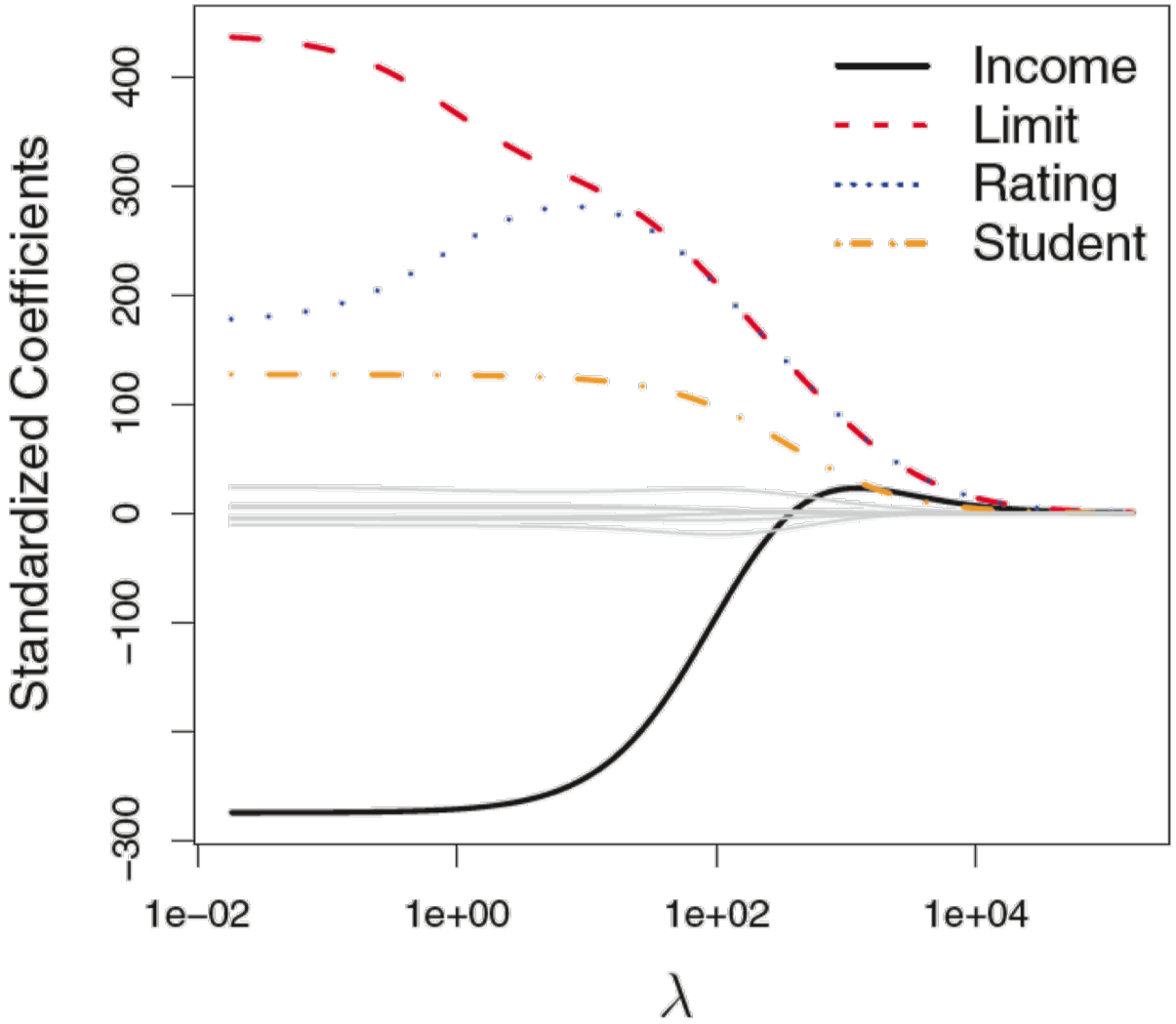
Predicting credit default

11 features to use to predict default:

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

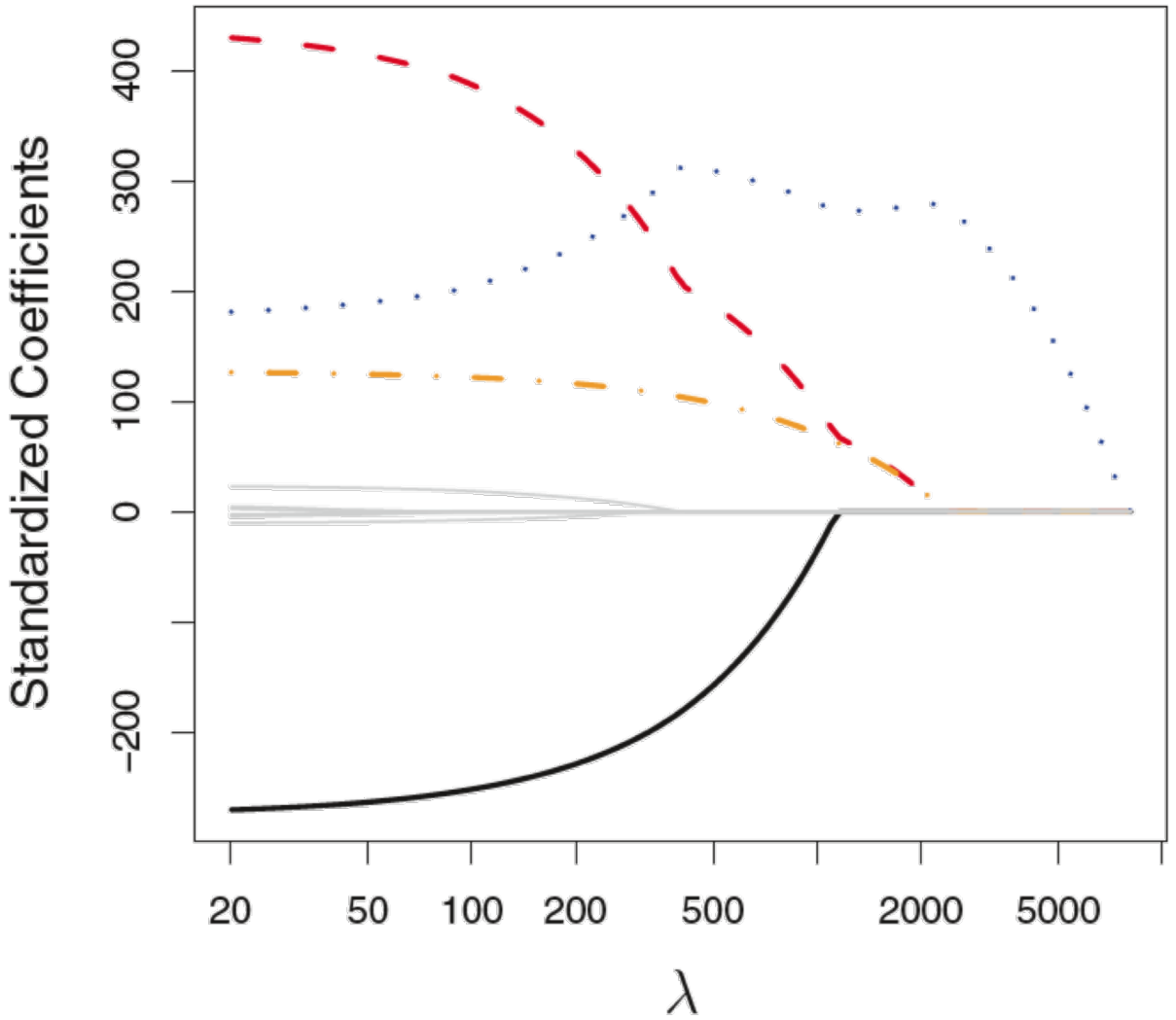
L_2 regularization

Ridge regression



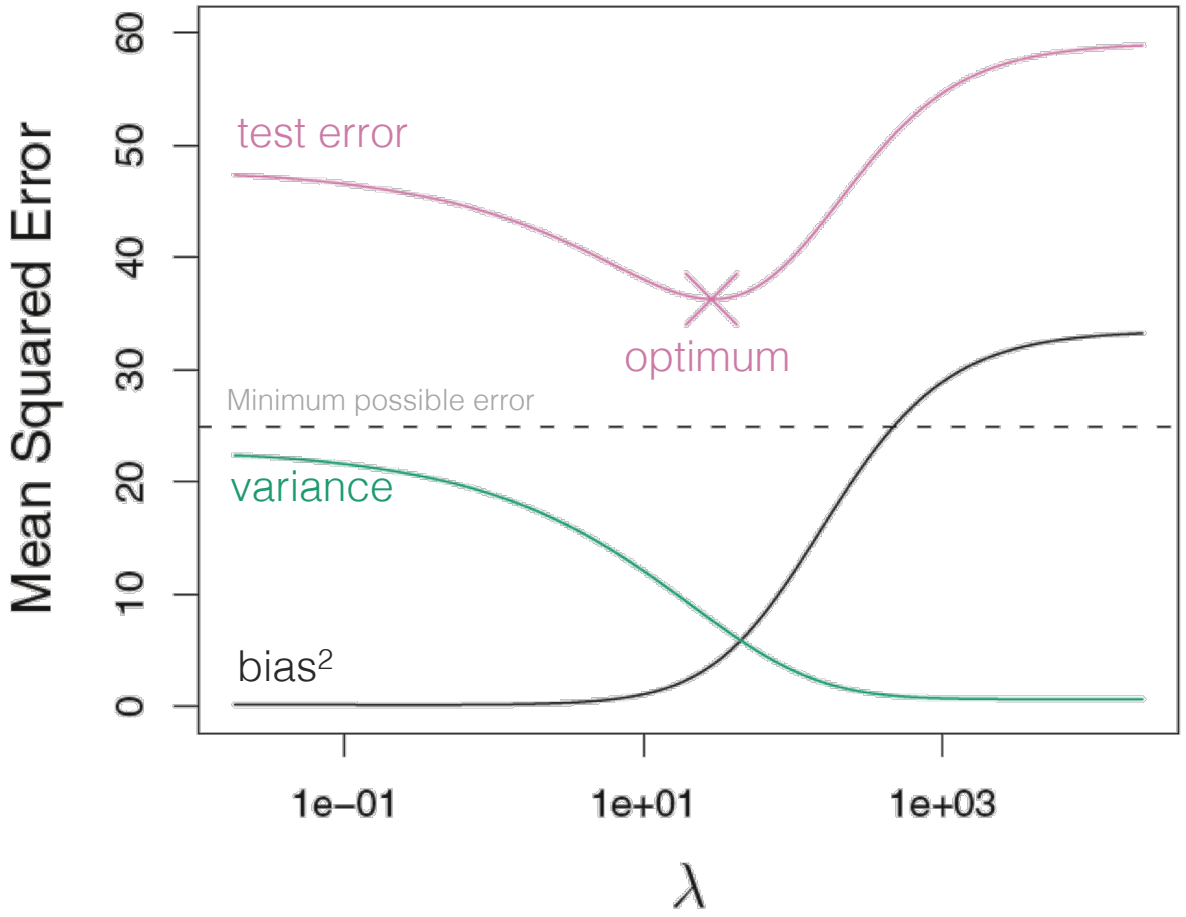
L_1 regularization

LASSO regularization

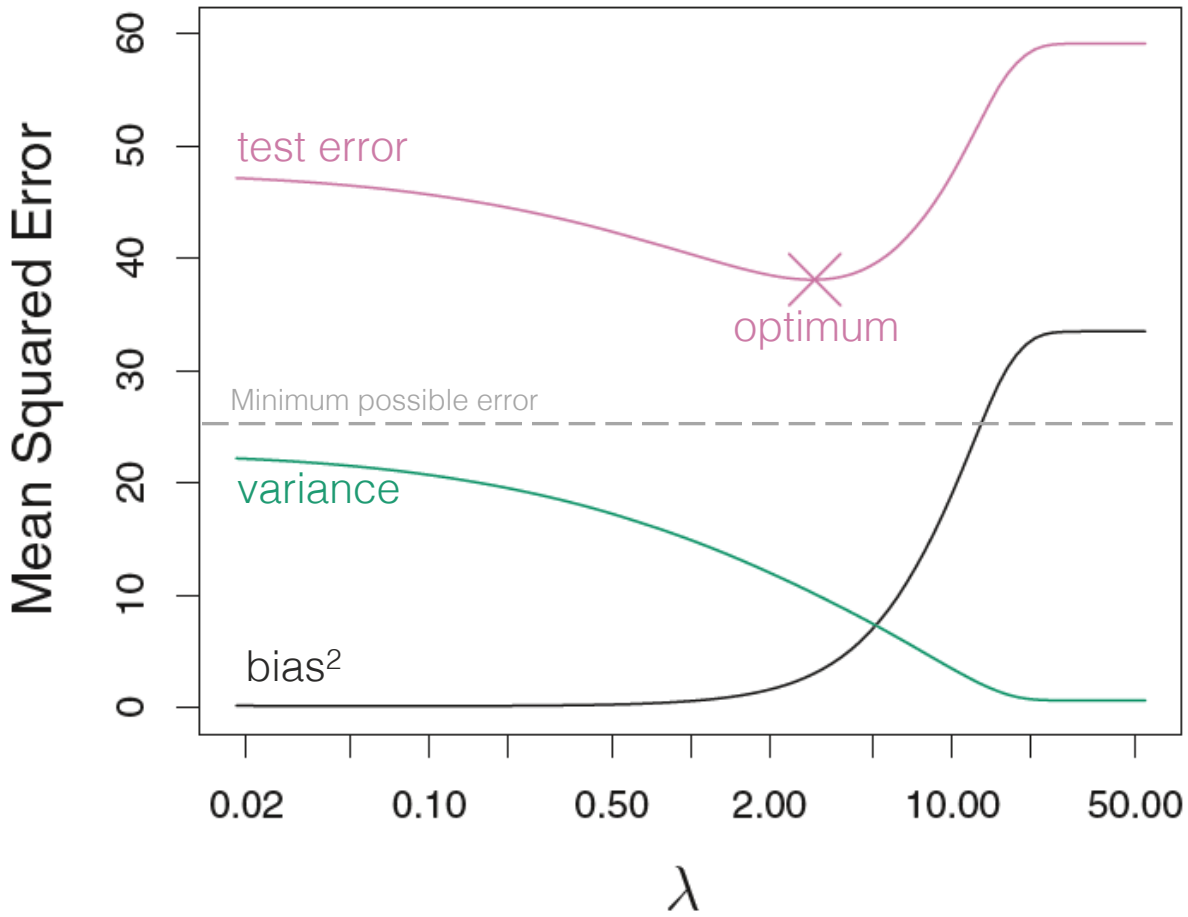


Images from James et al., An Introduction to Statistical Learning

L_2 regularization



L_1 regularization



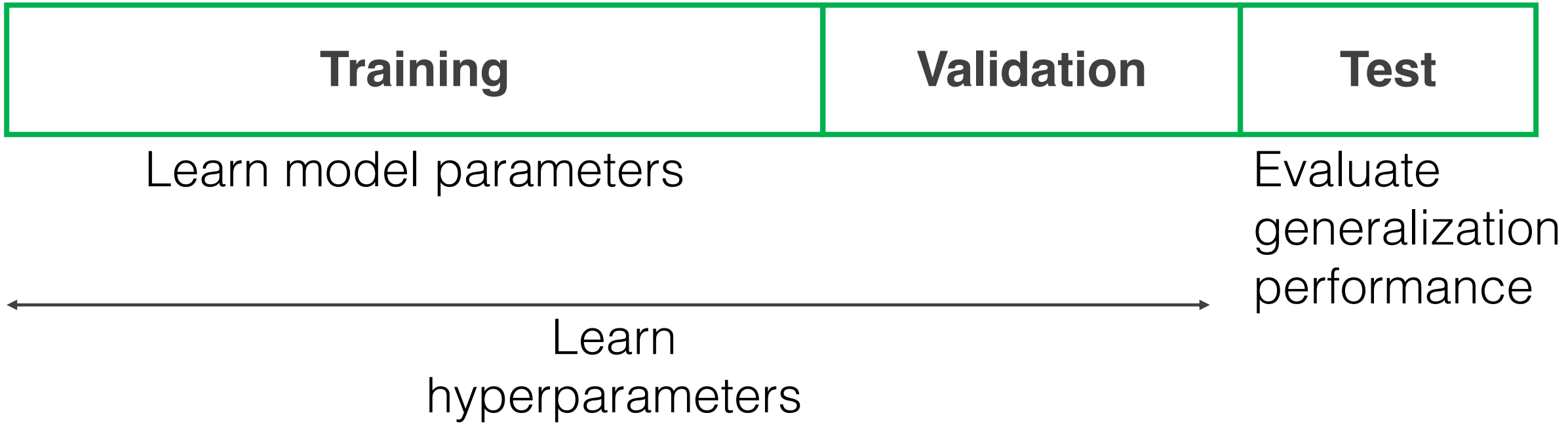
Images from James et al., An Introduction to Statistical Learning

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

Choosing the parameter λ

- λ is a hyperparameter
- Include a training, validation, and test set
- Can apply cross validation



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