

# Regularization for linear models

L1 (Lasso), L2 (Ridge)

Terence Parr  
MSDS program  
**University of San Francisco**

MIGHT BE MOST POPULAR  
INTERVIEW QUESTION

# Motivation for regularization

- 3 main problems with least-squares (OLS) regression:
  - Model with “too many” parameters (e.g., neural nets) will overfit
  - Data sets w/outliers can skew line too much to fit outliers; bad generalization
  - Data sets w/many features can get extreme coefficients in linear models (see notebook “L1 regularization with normalization”)
- Often unregularized models work but we get extreme coefficients (extreme negative and positive coefficients must be canceling out)
- L1 (Lasso) regularization also has the advantage that it allows superfluous coefficients to shrink to zero
- Having zero coefficients helps reduce model complexity (fewer coefficients), improving interpretability and usually improving generality

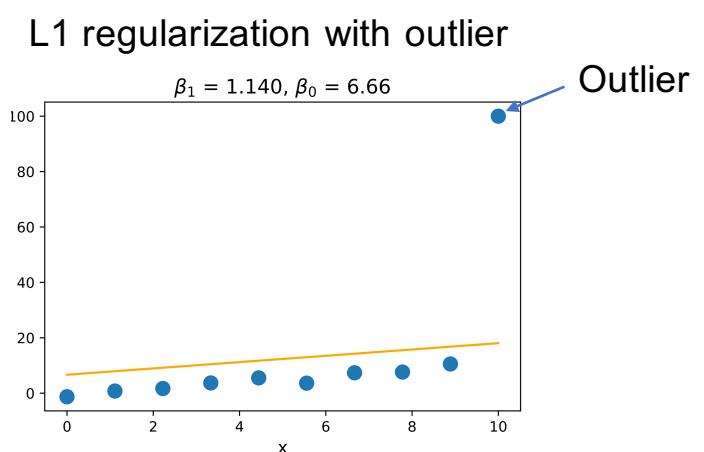
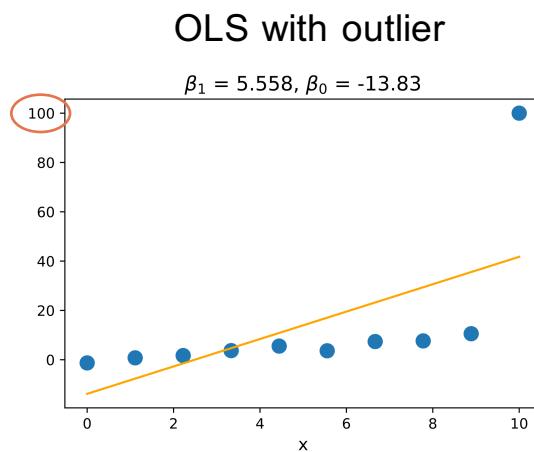
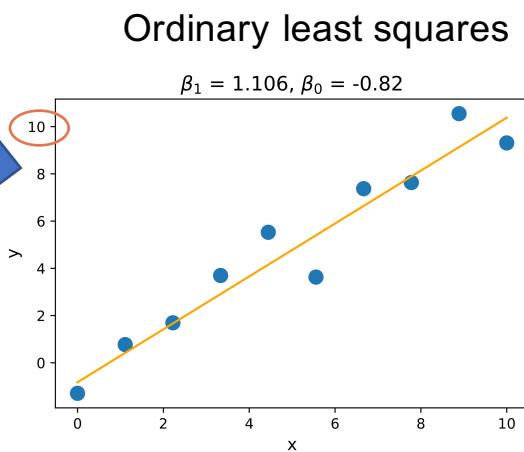
See <https://github.com/parrt/msds621/blob/master/notebooks/linear-models/regressor-regularization.ipynb>

# Regularization premise

- Extreme coefficients are unlikely to yield good generalization
- So, we're simply going to constrain model coefficient magnitudes
- Same technique works for linear and logistic regression  
(logistic is just a sigmoid of output of linear model anyway)
- Preview: add L1 or L2 norm of  $\beta$  coefficient vector to loss function

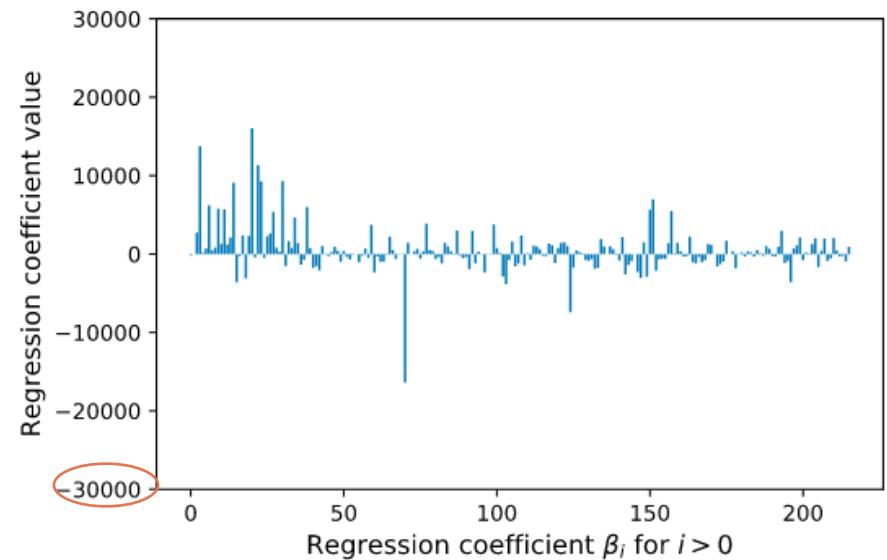
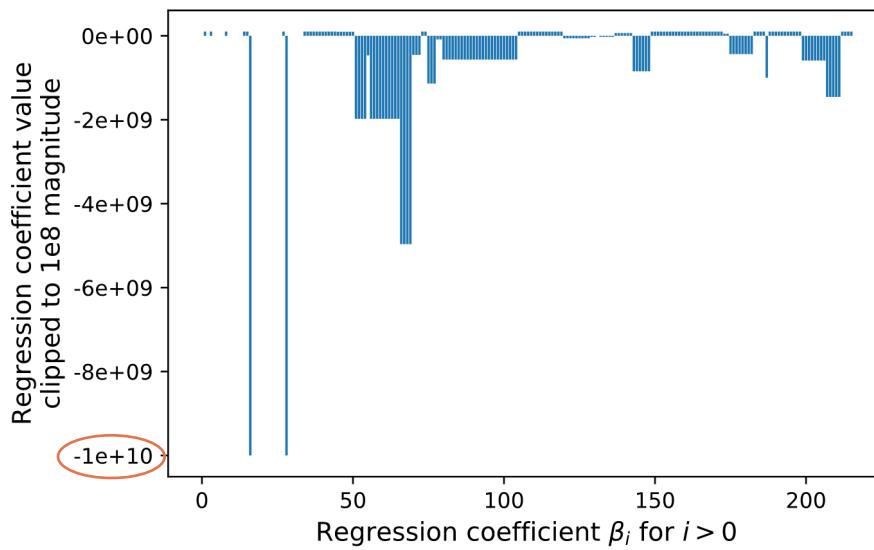
# Let's make a deal!

- Let's trade some model bias for improved generality
- Consider an example of a simple data set with OLS fit
- Now, send  $y[x==10]$  to 100; we get skewed line & bad  $R^2$
- Regularization brings slope back down but with some bias



# Ames housing data set (regressor)

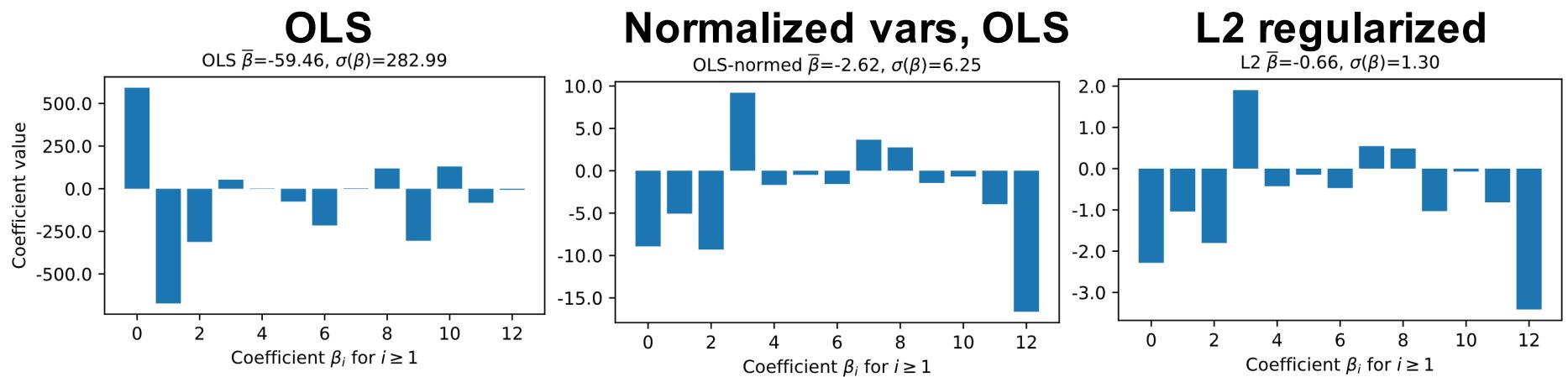
- With dummy vars, number of columns explodes from 81 to 216
- Compare scale of coeff (huge coefficients don't generalize)
- Regressor test R<sup>2</sup> is -1e6 w/o L2 regularization and ~0.82 with



See <https://github.com/parrt/msds621/blob/master/notebooks/linear-models/regressor-regularization.ipynb>

# Wine classifier less accurate w/o reg.

- Wine data set (130 records, 14 numeric vars)
- Test accur=.96 (normalized data) w/OLS; accur=1.0 w/regularization
- L2 regularization 1.0 test score
- L1 regularization still .96 but can drop 4 coefficients from model



See <https://github.com/parr/msds621/blob/master/notebooks/linear-models/classifier-regularization.ipynb>

# Example: classifier w/ too many features

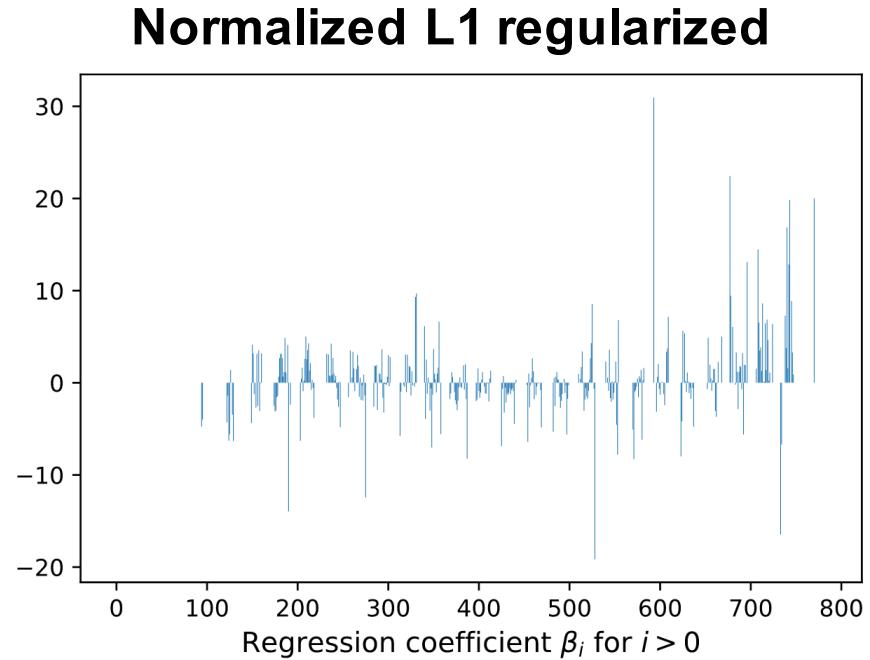
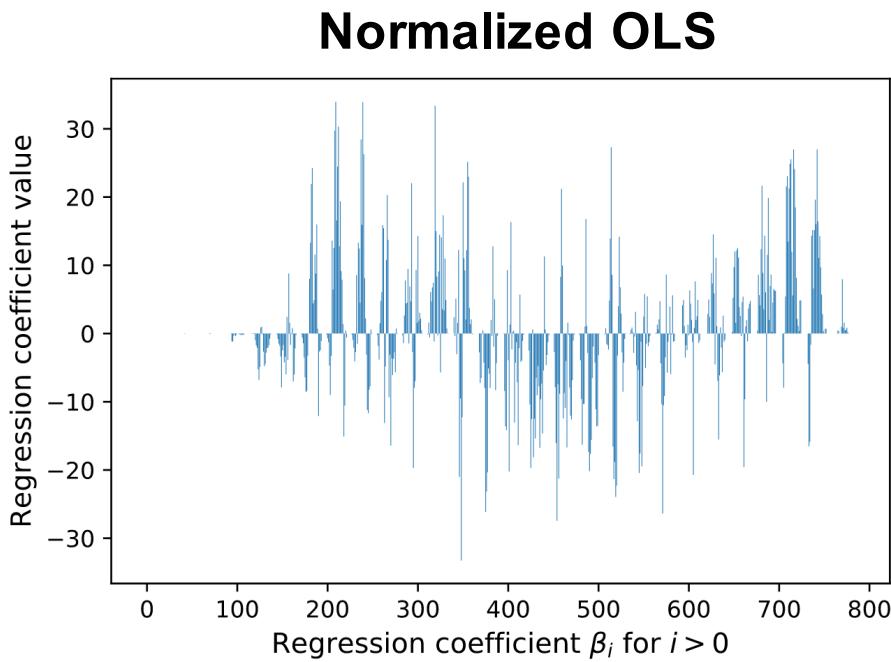
- Distinguish between ones and sevens (MNIST dataset)

MNIST sample	ones sample	sevens sample
3 8 6 9 6	1 1 1 1 1	7 7 7 7 7
4 5 3 8 4	1 1 1 1 1	7 7 7 7 7
5 2 3 8 4	1 1 1 1 1	7 7 7 7 7
8 1 5 0 5	1 1 1 1 1	7 7 7 7 7
9 7 4 1 0	1 1 1 1 1	7 7 7 7 7

See <https://github.com/parrt/msds621/blob/master/notebooks/linear-models/classifier-regularization.ipynb>

# Compare coefficients w/o & with L1 reg

- Test accur=0.96 (normalized data) w/OLS and w/L1 but log loss drops a lot
- L1 regularization has zeroed out lots of coefficients and still accurate!



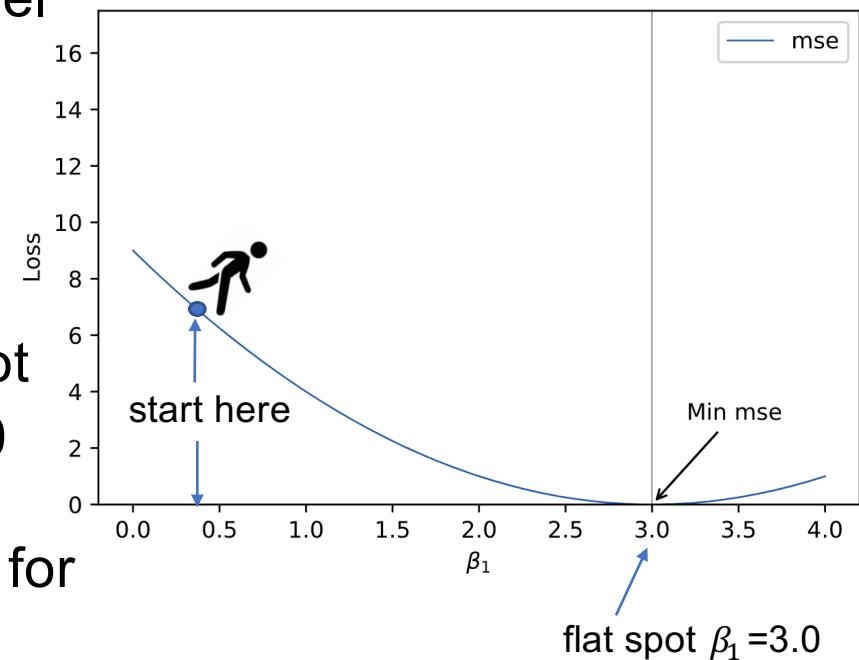
# The regularization mechanism

The goal: Don't let optimization get too specific to training data; inhibit best fit

The cost: Sacrifice some model bias (underfit) for generality

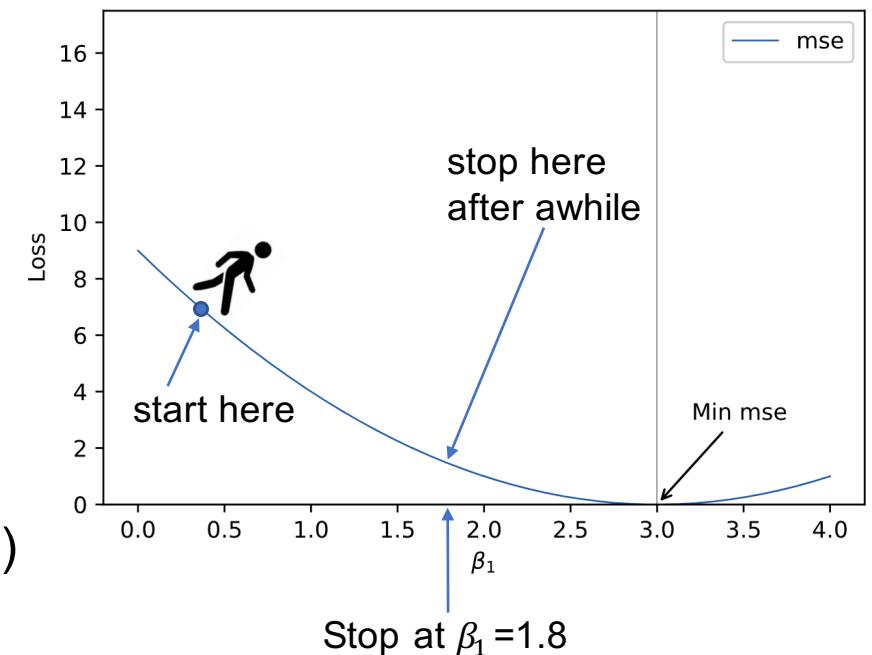
# Detour: How training works in 1 dimension

- Loss function (cost) is a function of model parameters (betas) and data set
- Minimize MSE loss computed on the **training set**
- Loss is a quadratic  $(y - \hat{y})^2$  so pick a random starting point for  $\beta_1$  and then just walk  $\beta_1$  downhill until you hit flat spot
- The derivative/slope of loss function is 0 at the minimum loss
- The  $\beta_1$  at loss minimum is best fit coeff. for training set
- (We'll have full lecture on gradient descent, don't worry!)



# Simplest regularization

- Just terminate minimization process early; don't let the coefficients fully reach the minimal loss location
- (Called *early stopping* in neural nets)
- Walk “downhill” on training set loss curve until either:
  - You run out of AWS CPU credits
  - After fixed amount of time (you're bored)
  - Loss on **validation set** starts going up, not down

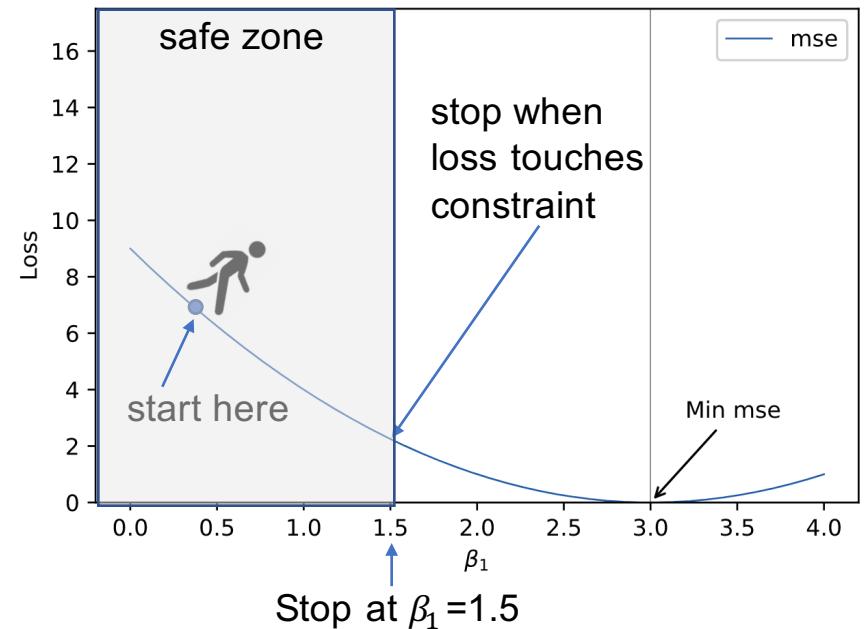


# Improving early termination of training

- Stopping after fixed amount of time depends on how fast our “hiker” moves along loss curve and it’s hard to pick duration
- Instead, let’s just restrict magnitude of  $\beta_1 < t$  for some  $t$
- Pick initial  $\beta_1$  in  $[0,t)$  and go downhill and stop at minimum or when  $\beta_1 \geq t$

# Minimizing subject to a *hard constraint*

- Let  $t=1.5$ , which defines “safe zone”
- Stop at min loss point or  $\beta_1 = t$
- The “best” coefficient is where constraint  $t$  and loss function meet, if min loss is outside safe zone
- If min loss is in safe zone, then regularization constraint wasn’t needed (same as OLS)



# Minimizing in 2D subject to hard constraint

- Spin line segment  $[0,t]$  around origin  $360^\circ$ , gives a circle
- Recall formula for circle; constrain  $\beta_i$  such that  $\beta_1^2 + \beta_2^2 < t$  where  $t$  is radius squared
- The “best” coefficient is min loss function **on constraint curve** (if loss min outside zone)
- If loss min inside radius, same as OLS
- Pick initial  $[\beta_1, \beta_2]$  inside safe zone, then start walking downhill, stop at min loss or edge of the safe zone
- **This is L2 (Ridge) regularization** 
$$\sum_{j=1}^p \beta_j^2 < t$$

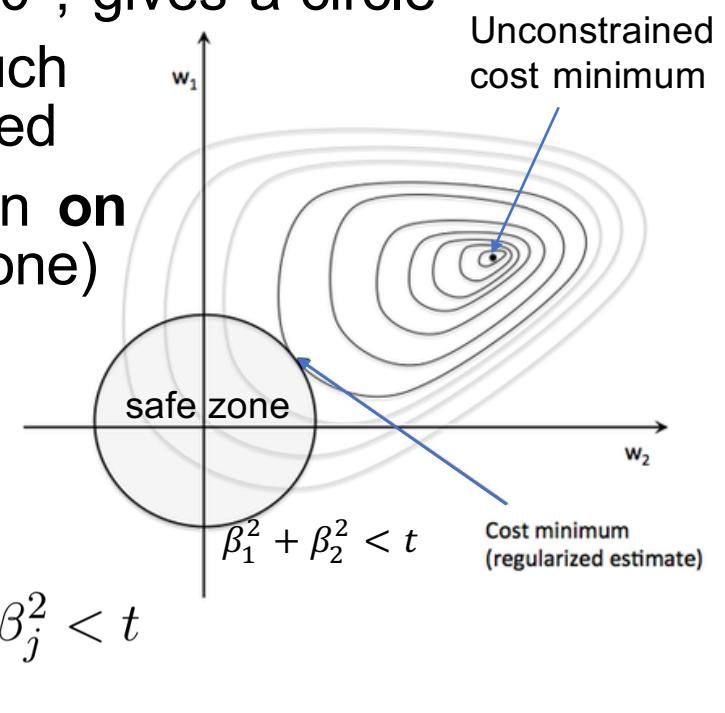


Image credit: <https://www.kdnuggets.com/2016/06/regularization-logistic-regression.html>

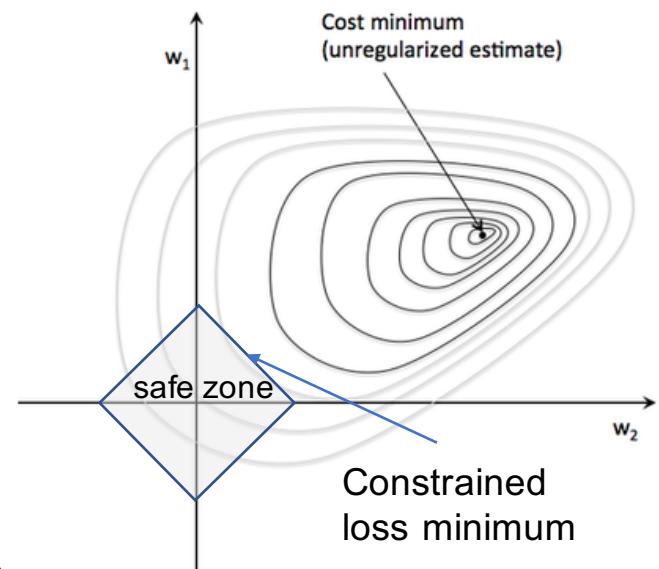
# Minimizing L1 in 2D (hard constraint)

- Instead, we can sum the  $|\beta_i|$ 's not  $\beta_i^2$ 's, giving diamond-shaped safe zone
- **This is L1 (Lasso) regularization**

$$\sum_{j=1}^p |\beta_j| < t$$



Notice how statisticians have this backwards; L1 constraint zone looks like a ridge not lasso, and L2 safe zone (circle) looks like a lasso



# Fitting regularized linear model (Conceptually)

- Minimize the usual MSE loss function:

$$\mathcal{L}(\beta) = \sum_{i=1}^n (y^{(i)} - (\mathbf{x}'^{(i)} \cdot \beta))^2$$

- Subject to either (L2 or L1):

$$\sum_{j=1}^p \beta_j^2 < t \quad \text{or} \quad \sum_{j=1}^p |\beta_j| < t$$

- Problem is “subject to” constraint is hard to implement

# Detour: Lagrange Multipliers

- Magic of Lagrange multipliers lets us incorporate constraint into loss function:

$$\mathcal{L}(\beta) \text{ s.t. } \sum_{j=1}^p \beta_j^2 \leq t \text{ same as}$$

$$\mathcal{L}(\beta, \lambda) = \mathcal{L}(\beta) + \lambda \left( \sum_{j=1}^p \beta_j^2 - t \right) \text{ for some } \lambda$$

( $\lambda$  and  $t$  are related one-to-one per ESLII book  
but by no relationship I can find)

# How we *actually* fit regularized models

- Minimizing loss function subject to a constraint is harder to implement than just function minimization
- Invoke magic of Lagrange

$$\text{For L2: } \mathcal{L}(\beta) + \lambda \sum_{j=1}^p \beta_j^2$$

$$\text{For L1: } \mathcal{L}(\beta) + \lambda \sum_{j=1}^p |\beta_j|$$

- Drop  $t$  since  $t$  is constant & doesn't affect minimizing
- This is a **soft constraint** and penalty increases as  $\beta_i$ 's move away from origin; *there's no hard cutoff!*
- Net effect is that regularization pulls min loss location closer to origin!

# Fitting both $\beta_i$ 's and hyperparameter $\lambda$

- $\lambda$  is unknown like  $t$  but at least we have a single function now
- Find  $\lambda$  by finding minimum loss for different  $\lambda$  values, pick  $\lambda$  that gets min loss on validation set

$$\mathcal{L}(\beta) + \lambda \sum_{j=1}^p \beta_j^2$$

$$\mathcal{L}(\beta) + \lambda \sum_{j=1}^p |\beta_j|$$

- $\beta_0$  is just  $\bar{y}$ , assuming zero-centered data set (next slide)
- **Note:**  $\beta_0$  is NOT included in penalty, but is used in  $\mathcal{L}(\beta)$

# Normalizing/standardizing data

- Regularization requires that you normalize your data set
- Zero center each variable and divide by the standard deviation

$$x^{(i)} = \frac{(x^{(i)} - \bar{x})}{\sigma_x}$$

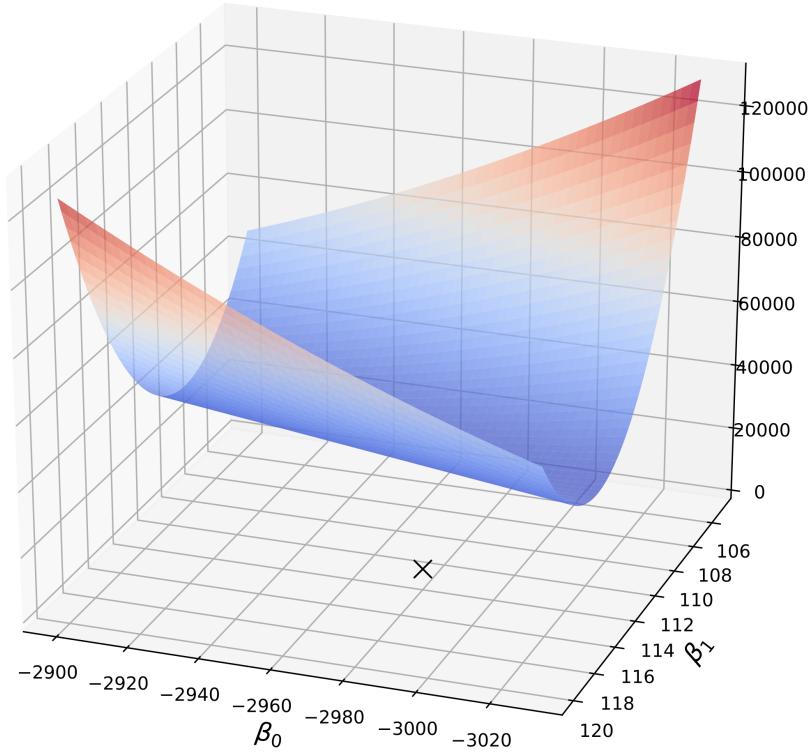
- Computing  $\beta_0$  as simply mean(y) for L1/L2 linear regression requires normalization

# More reasons to standardize variables

- We have just one  $\lambda$  for all coefficients so vars must be in same range or big  $\beta$ 's prevent regularization of small coefficients (big var ranges cause big  $\beta$ 's)
- Finding coefficients is often more efficient for normalized data
  - Loss contours for normalized vars are less erratic due to nature of finite precision floating point; operations (like subtraction) on data in very different ranges are ill-conditioned
  - Loss contours for normalized vars are more spherical and yield gradients that point more directly at min loss
  - If  $x_1$  is in [0, 100] and  $x_2$  in [0, 0.001], then a **single** learning rate,  $\eta$ , that is small enough for  $x_2$  (so we converge) is too slow for  $x_1$

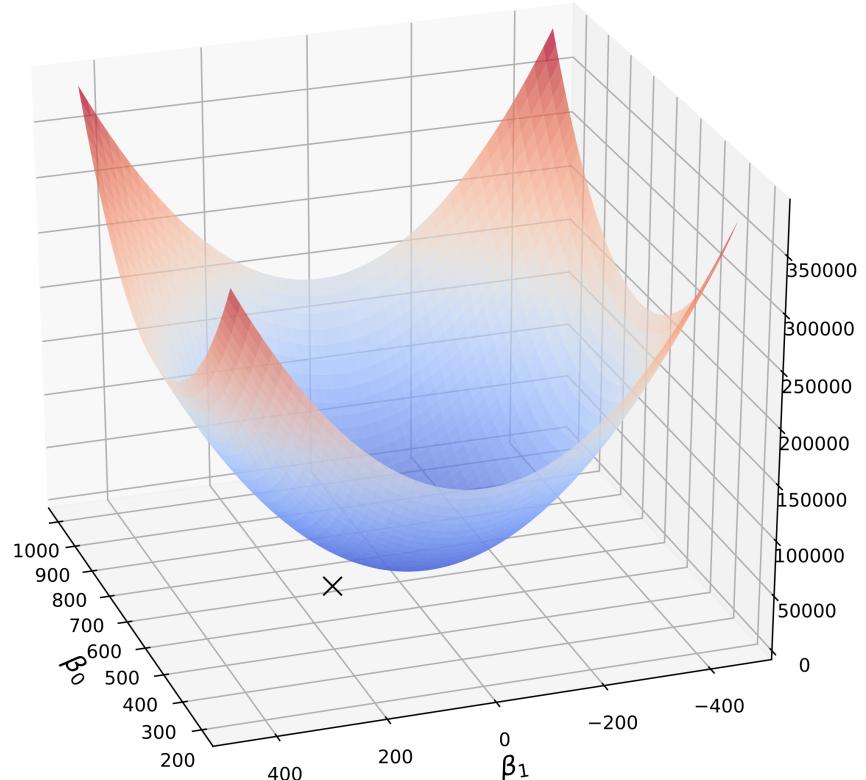
$$\lambda \sum_{j=1}^p \beta_j^2$$

# Loss function shape for normalized vars



Untouched variables

*Cheese consumption vs bedsheets strangulation death data set*

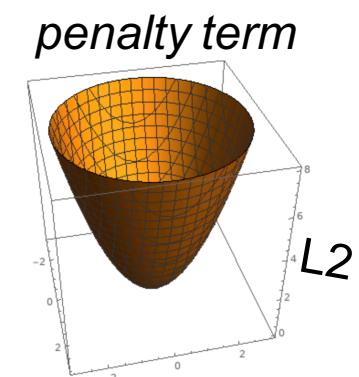


Normalized variables

See <https://github.com/parrt/msds621/blob/master/notebooks/linear-models/viz-gradient-descent.ipynb>

# How penalty term restricts $\beta_i$ 's

- Think of regularization as two different cost functions, MSE and regularization, added together
- L2 loss increases with the sum of squared coefficients
- L1 loss increases with sum of coefficient magnitudes
- Regularization penalty term **increases loss** but skews min loss  $\beta_i$  location towards origin as penalty curve is anchored at origin
- L1&L2 **shift**  $\beta_i$ 's **towards 0** because penalty approaches 0 only at  $\beta=0$
- Soft constraint makes larger  $\beta_i$  very unlikely due to increased penalty away from origin (hard constraint makes large coefficients  $> t$  impossible)



# The effect of regularization

- What happens when  $\lambda$  is 0? Regularization is turned off
- What happens when we crank up  $\lambda$ ? Loss function strives for small  $\beta$
- Pushing  $\lambda$  higher with L1 pushes more coefficients to 0
- L1 does not discourage  $\beta_i$  from fading to 0, but L2 discourages
- Squaring coefficients for L2 in constraint discourages any single  $\beta_i$  from getting much bigger than the others; squeezes  $\beta_i$ 's together
- $\lambda$  is independent of training data, so can reduce model's dependence on data during fit, which regularizes model

# Lab time

- Exploring regularization for linear regression  
<https://nbviewer.jupyter.org/github/parrt/msds621/blob/master/labs/linear-models/regularization-regr.ipynb>
- Note: that links is for viewing since github screws that up; you will need to go to the course repo to get interactive labs  
<https://github.com/parrt/msds621/tree/master/labs/linear-models>

# Regularized logistic regression

# Optimizing likelihood with penalty terms

- Same mechanism, minimizing (**negation** of maximum likelihood) via Lagrangian interpretation:

$$\mathcal{L}(\beta, \lambda) = - \sum_{i=1}^n \left\{ y^{(i)} \mathbf{x}'^{(i)} \beta - \log(1 + e^{\mathbf{x}' \beta}) \right\} + \lambda \sum_{j=1}^p \beta_j^2$$

$$\mathcal{L}(\beta, \lambda) = - \sum_{i=1}^n \left\{ y^{(i)} \mathbf{x}'^{(i)} \beta - \log(1 + e^{\mathbf{x}' \beta}) \right\} + \lambda \sum_{j=1}^p |\beta_j|$$

**Note:**  $\beta_0$  is NOT included in penalty, but is used in  $\mathcal{L}(\beta)$  term

# Fitting logistic regression $\beta_i$ 's and $\lambda$

- ESLII book p125 says to find L1  $\beta_0$  and  $\beta_{1..p}$  that maximize:

$$\max_{\beta_0, \beta} \left\{ \sum_{i=1}^N \left[ y_i(\beta_0 + \beta^T x_i) - \log(1 + e^{\beta_0 + \beta^T x_i}) \right] - \lambda \sum_{j=1}^p |\beta_j| \right\}. \quad (4.31)$$

- We minimize the negative of that to find  $\beta$  of max likelihood
- Means we must find  $\beta_0$  differently than for  $\beta_{1..p}$
- As before, find  $\lambda$  by computing minimum loss for different  $\lambda$  values, pick  $\lambda$  that gets min loss on validation set

# L1 Logistic gradient is tricky to get right

**Algorithm:**  $L1NegLogLikelihood(\mathbf{X}', \mathbf{y}, \beta')$

$$err = \mathbf{y} - \sigma(\mathbf{X}' \cdot \beta')$$

(error vector is  $n \times 1$  column vector)

$$\frac{\partial}{\partial \beta_0} = mean(err)$$

(usual log-likelihood gradient; use current  $\beta'$ )

$$r = \lambda \text{sign}(\beta')$$

(regularization term  $p + 1 \times 1$  column vector)

$$r[0] = 0$$

(kill  $\beta_0$  position but keep as  $p + 1 \times 1$  vector)

$$\nabla = \frac{1}{n} \left\{ \mathbf{X}'^T err - r \right\}$$

$$\text{return} - \begin{bmatrix} \frac{\partial}{\partial \beta_0} \\ \nabla_1 \\ \vdots \\ \nabla_p \end{bmatrix}$$

# Resources with code

- <https://aimotion.blogspot.com/2011/11/machine-learning-with-python-logistic.html>
- <https://stackoverflow.com/questions/38853370/matlab-regularized-logistic-regression-how-to-compute-gradient>
- You may look at but not cut-paste from these examples for your project (and I think there are some bugs too)

# Key takeaways

- Regularization increases generality at cost of some bias
- Does so by restricting size of coefficients with hard constraint (conceptually) or soft constraint using penalty term (impl.)
- For hard constraint, min loss inside safe zone or on zone border
- Soft constraint penalty just makes bigger parameters less likely
- L2 discourages any  $\beta_i$  from getting much bigger than the others
- L1 does not discourage  $\beta_i$  from fading to 0, but L2 discourages
- For  $\beta < 1$ ,  $\beta^2$  gets smaller, not bigger; L2 can't push  $\beta$  to left very much near 0 as it's shifting by fraction of  $\beta$
- In practice, we see L1 zero out unpredictive vars

# Key implementation details

- Minimize  $\mathcal{L}(\beta, \lambda)$  by trying multiple  $\lambda$  and choosing parameters from fitted model getting lowest **validation** error
- Always normalize X before fitting models
- If 0-centered  $x_i$  then  $\beta_0 = \text{mean}(y)$  for L1 and L2 regression
- Logistic regression has no closed form for  $\beta_0$
- OLS & L2 regularized linear regression have symbolic solutions
- L1 linear regression and L1/L2 logistic regression require iterative solution: usually some gradient descent variation

# Interview questions (L1 vs L2)

- Both L1 and L2 increase bias (reduce "accuracy" of fit, predictions)
- L1 useful for variable / feature selection as some  $\beta_i$  go to 0
- L2 useful when you have collinear features (e.g., both tumor radius and tumor circumference features)
  - Collinearity increases coefficient variance, making them unreliable
  - And L2 reduces variance of  $\beta_i$  estimate which counteracts effect of collinearity
- With regularization, we don't get unbiased  $\beta_i$  estimates
  - Expected value of  $\beta_i$  estimate is no longer  $\beta_i$  since we've constrained  $\beta_i$ 's
  - Consequence: we no longer know effect of  $x_i$  on target  $y$  via  $\beta_i$
- In my experiments, regularization improves regression much more than classification in terms of R^2 and simple accuracy; (due to sigmoid constraining the solutions to the loss function?)

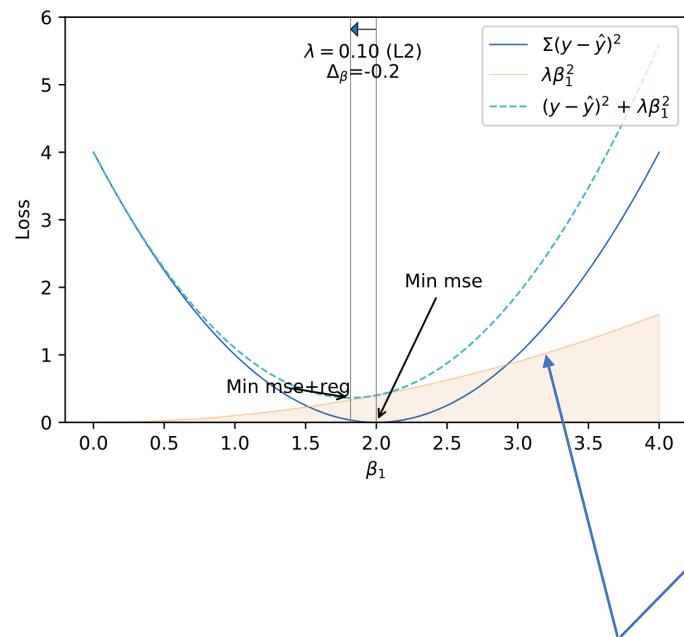
# Lab time

- Regularization for logistic regression  
<https://nbviewer.jupyter.org/github/parrt/msds621/blob/master/labs/linear-models/regularization-logi.ipynb>
- Note: that links is for viewing since github screws that up; you will need to go to the course repo to get interactive labs  
<https://github.com/parrt/msds621/tree/master/labs/linear-models>

# EPILOGUE:

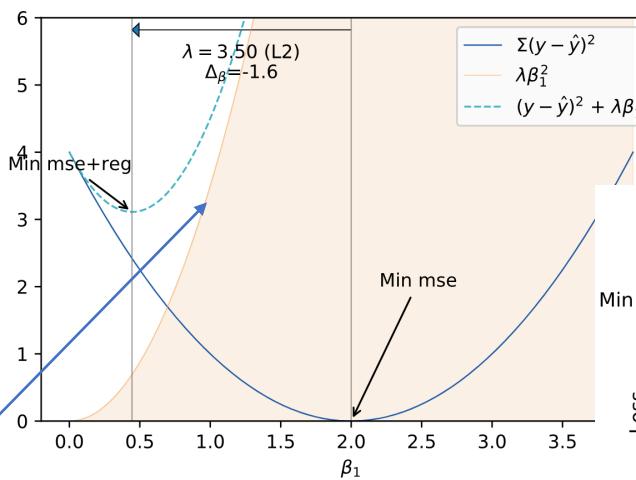
## More on the effect of increasing hyper parameter $\lambda$

# Visual effect of increasing $\lambda$ for L2 (single variable)

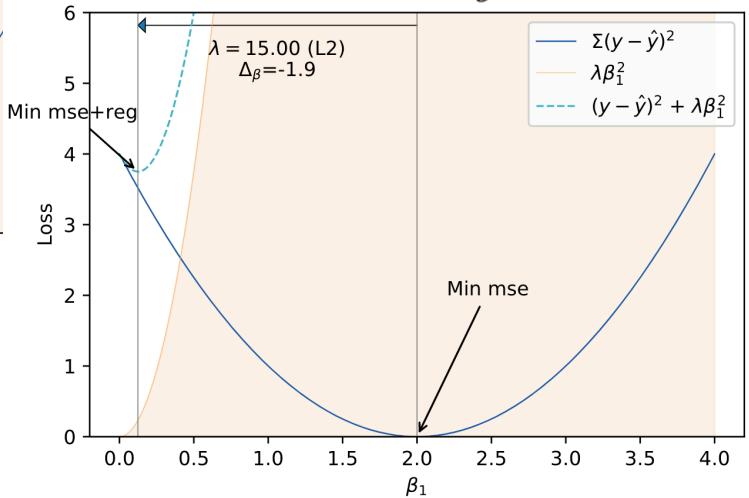


Takes roughly  $\lambda = 300$  to get  $\beta_1$  to 0!

$$\mathcal{L}(\beta) + \lambda \sum_{j=1}^p \beta_j^2$$



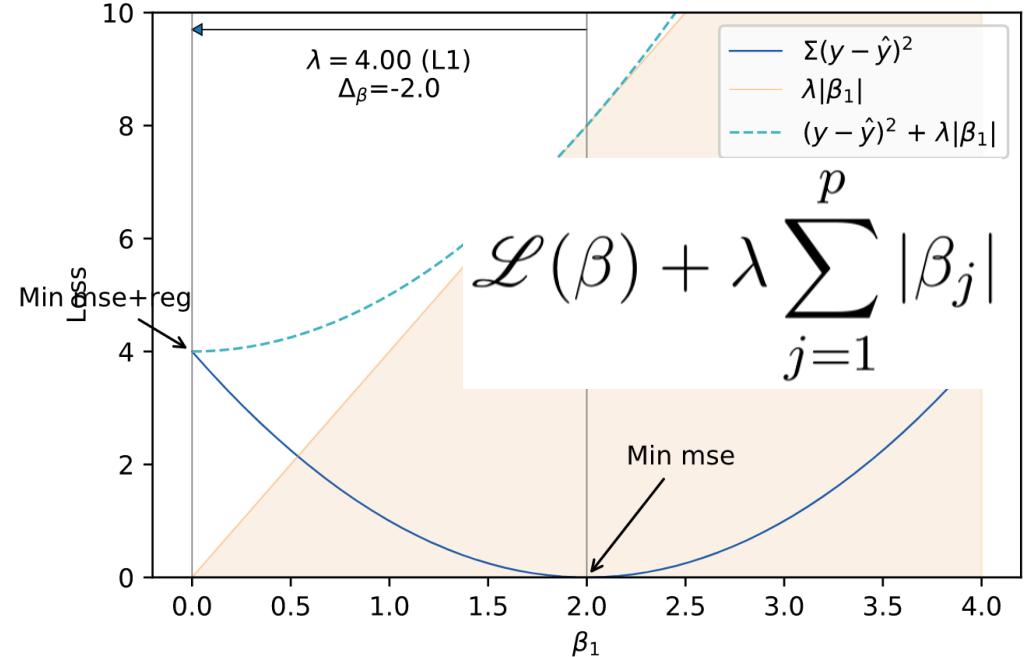
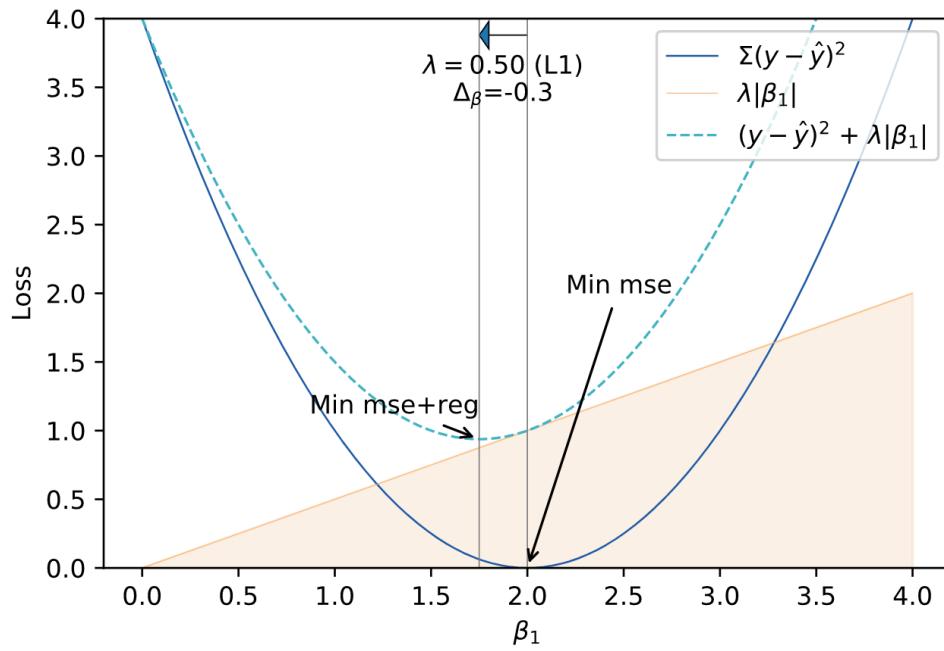
Decelerating increase in loss,  
decelerating shift of  $\beta_1$  towards 0



See <https://github.com/parrt/msds621/blob/master/notebooks/linear-models/viz-regularization.ipynb>

# Visual effect of increasing $\lambda$ for L1

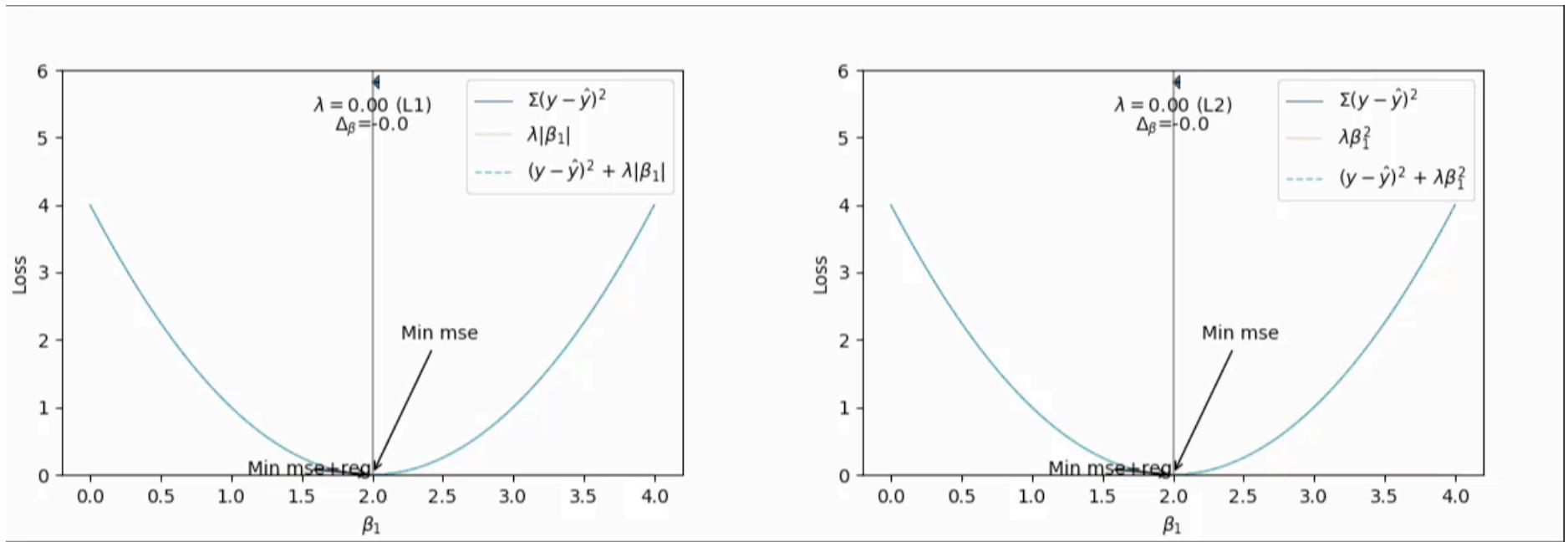
Smooth/linear increase in loss and shift of  $\beta_1$  towards 0  
Doesn't take huge  $\lambda$  to get  $\beta_1$  to 0



See <https://github.com/parrt/msds621/blob/master/notebooks/linear-models/viz-regularization.ipynb>

# L1 vs L2 regularization visually

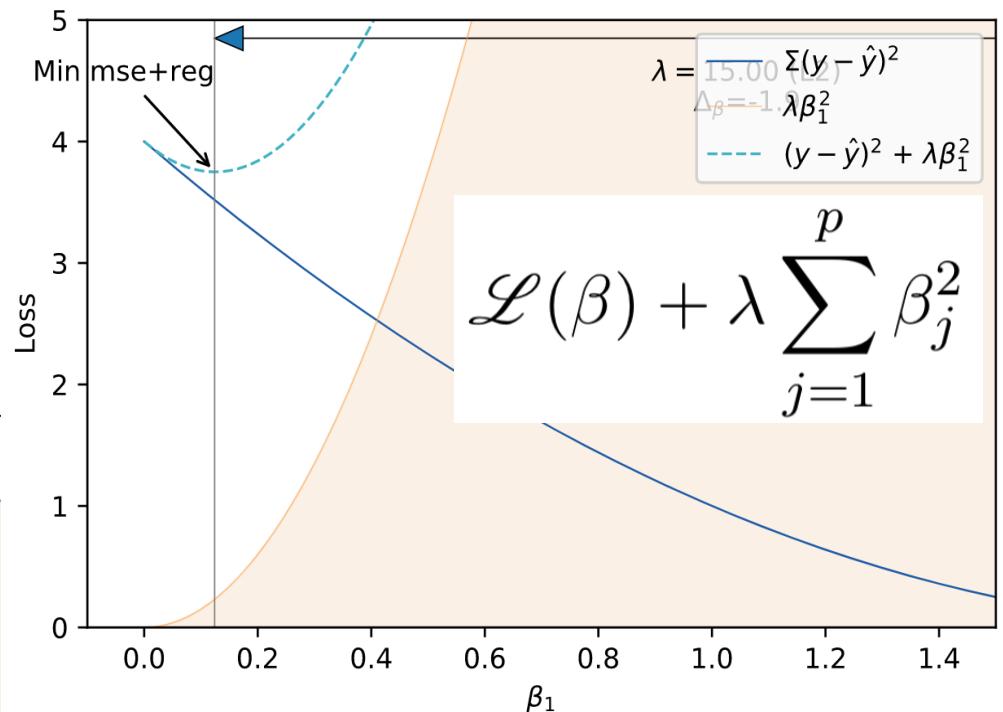
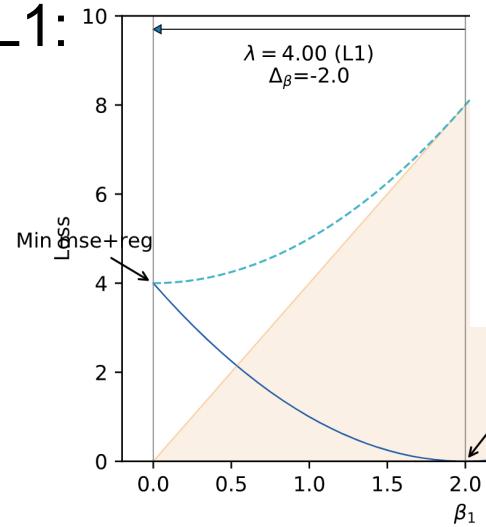
- Notice how L1 has no problem pushing  $\beta$  to 0 whereas L2 struggles to get there (this is a video page)



Animations from: <https://github.com/parrt/msds621/blob/master/notebooks/linear-models/viz-regularization.ipynb>

# Zoom in on L2 trying to push $\beta$ to 0

- For  $\beta < 1$ ,  $\beta^2$  gets smaller, not bigger
- Since  $\beta$  decelerates, L2 can't push  $\beta$  to left quickly near 0
- Compare to L1:



# Why L1 can yield $\beta_i=0$ (and why not L2?)

- (p=1) Derivative of L2 penalty term w.r.t  $\beta$  is  $2\lambda\beta$ , which is the increase in loss due to penalty term;  $2\lambda\beta$  vanishes as  $\beta \rightarrow 0$
- When  $\beta < 1$ , L2 adds a fraction of  $\lambda$  to loss each time; asymptotic
- The closer it comes to 0, the slower the increase in loss & shift in  $\beta$  associated with min loss
- For L1, derivative is just  $\lambda\text{sign}(\beta)$ , independent of  $\beta$  magnitude, so loss increases at same rate ( $\lambda$ ) even near  $\beta = 0$

$$\mathcal{L}(\beta) + \lambda \sum_{j=1}^p \beta_j^2$$

$$\mathcal{L}(\beta) + \lambda \sum_{j=1}^p |\beta_j|$$

# Why/when L1 encourages 0 coefficients

- In practice, we see L1 zero out unpredictive variables for large  $p$  ( $p$  is number of features)
- Or when  $\lambda$  gets big enough
- There's a theorem that says  $P(\beta_i = 0) \rightarrow 1$  as  $p \rightarrow \infty$
- As  $p \rightarrow \infty$ , diamond shape collapses in on itself to a point
- Distance between discontinuities (feasibility points / axis crossings) approaches zero, making it less likely  $\beta_i$  sits on an edge of the diamond; mostly likely sits at a discontinuity
- Uminsky drew a picture that convinced me 0's are more likely and I came up with a drawing showing 0's more likely for nonpredictive variables... (but we're way off our scope here  $\square$ )