Regularization for linear models

L1 (Lasso), L2 (Ridge)

MIGHT BE MOST POPULAR INTERVIEW QUESTION

Terence Parr
MSDS program
University of San Francisco

UNIVERSITY OF SAN FRANCISCO

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

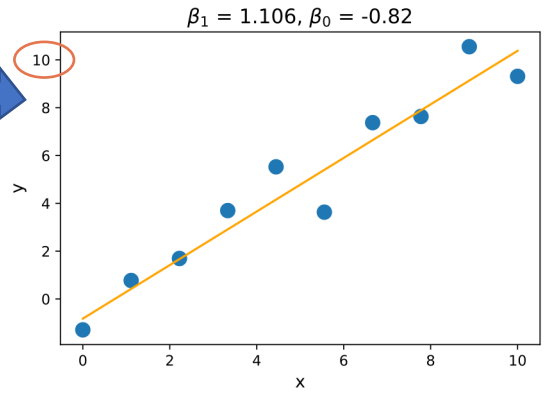
Regularization premise

- 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 on output of linear model anyway)
- Preview: add L1 or L2 norm of β coefficient vector to loss function

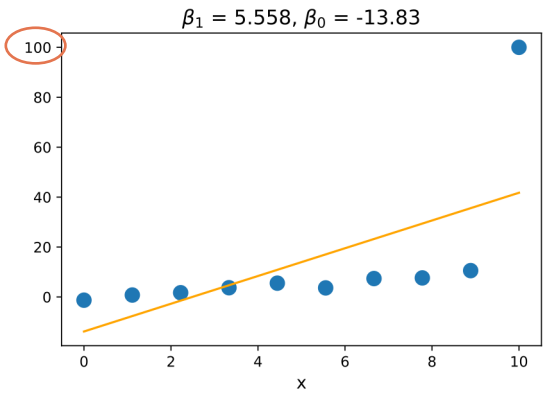
Let's make a deal!

- 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

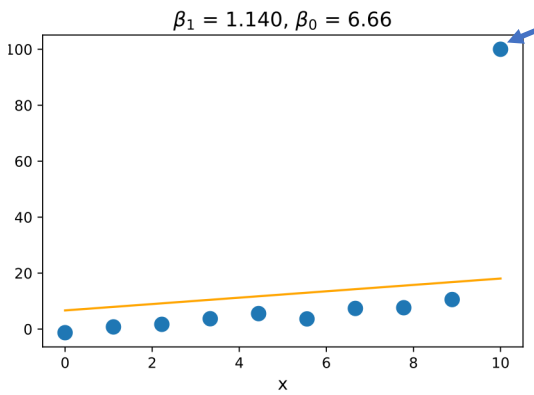
Ordinary least squares



OLS with outlier

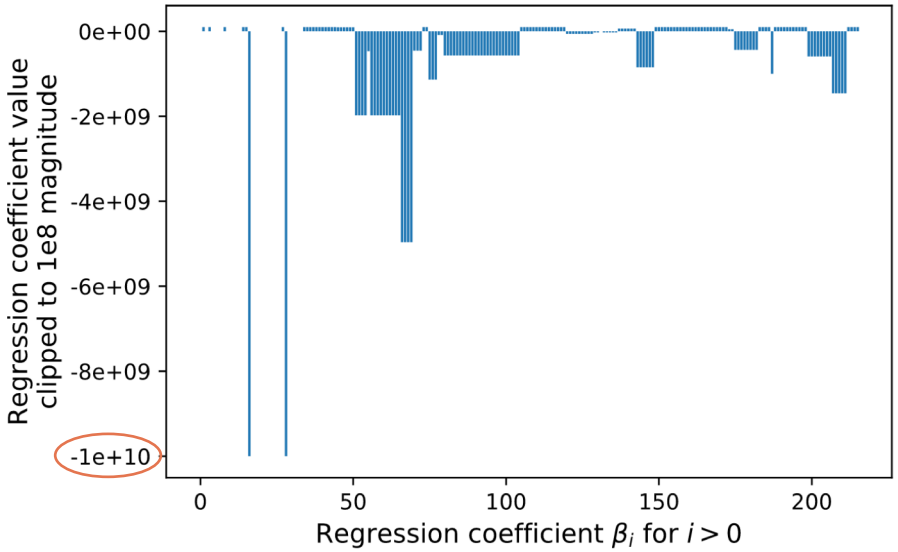


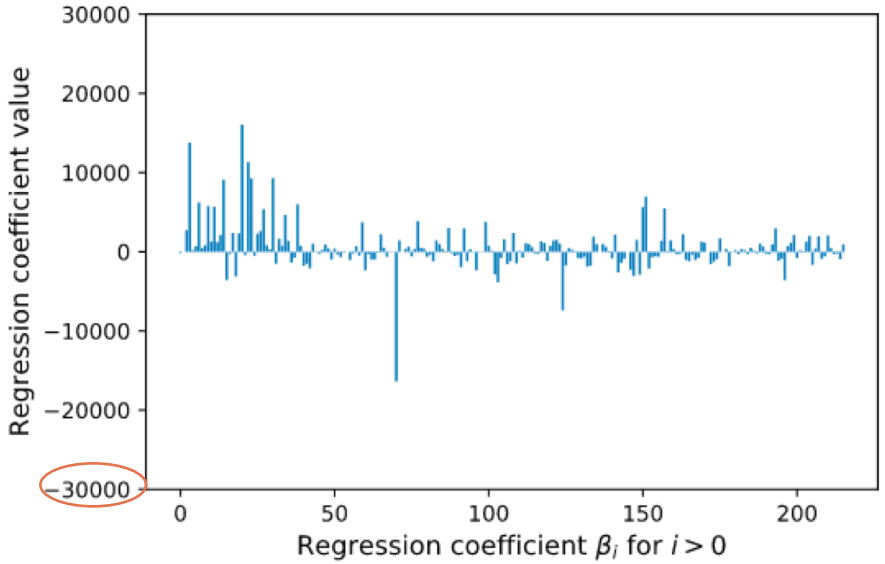
L1 regularization with outlier



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^2 is -1e6 w/o L2 regularization and ~0.82 with





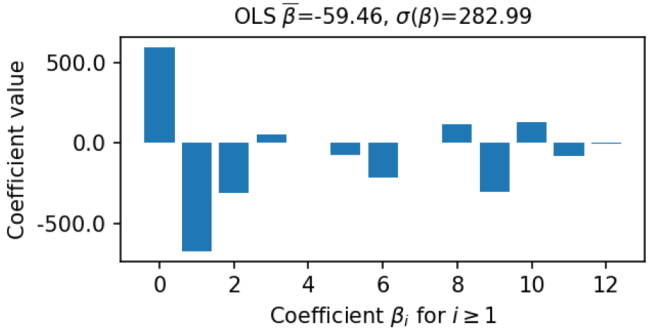
UNIVERSITY OF SAN FRANCISCO

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

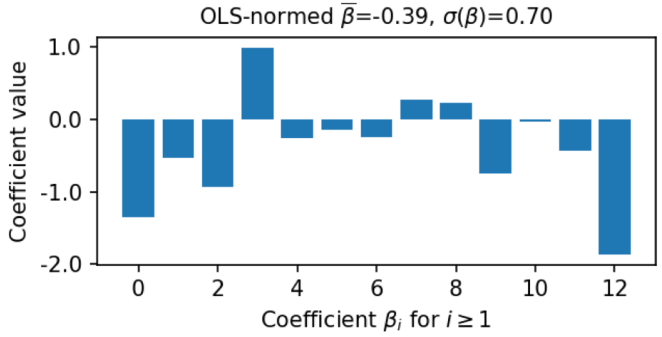
Wine classifier less accurate w/o reg.

- Wine data set (130 records, 14 numeric vars)
- Test accur=.96 (normalized data) w/OLS; accur=1.0 w/regularization
- Normalizing also helps with coefficient interpretation
- L2 regularization 1.0 test score, L1 still .96 but can drop 4 coeff

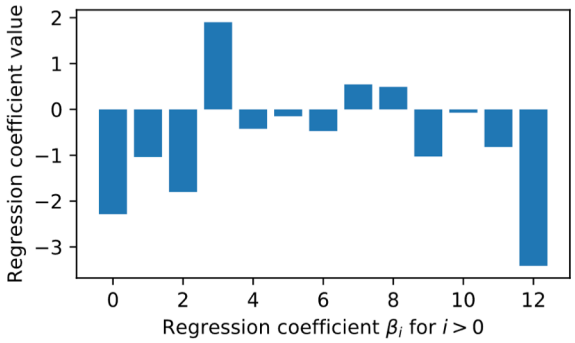
Ordinary least squares



Normalized



Normalized, L2 regularized

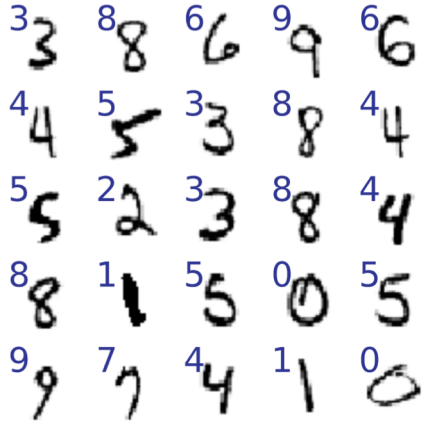


See https://github.com/parrt/msds621/blob/master/notebooks/linear-models/classifier-regularization.ipynb

Example: classifier w/too many features

- Distinguish between ones and sevens (MNIST dataset)

MNIST sample



ones sample

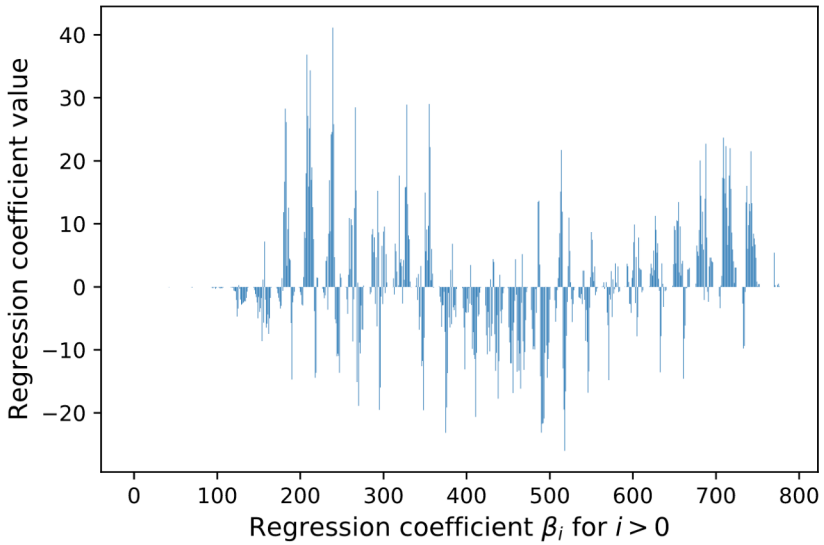
sevens sample

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

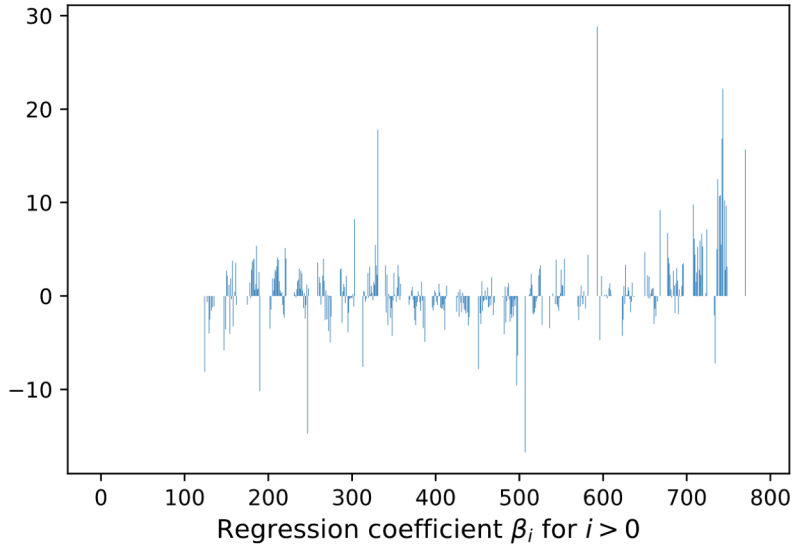
Compare coefficients w/o & with L1 reg

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

Normalized, OLS



Normalized, L1 regularized



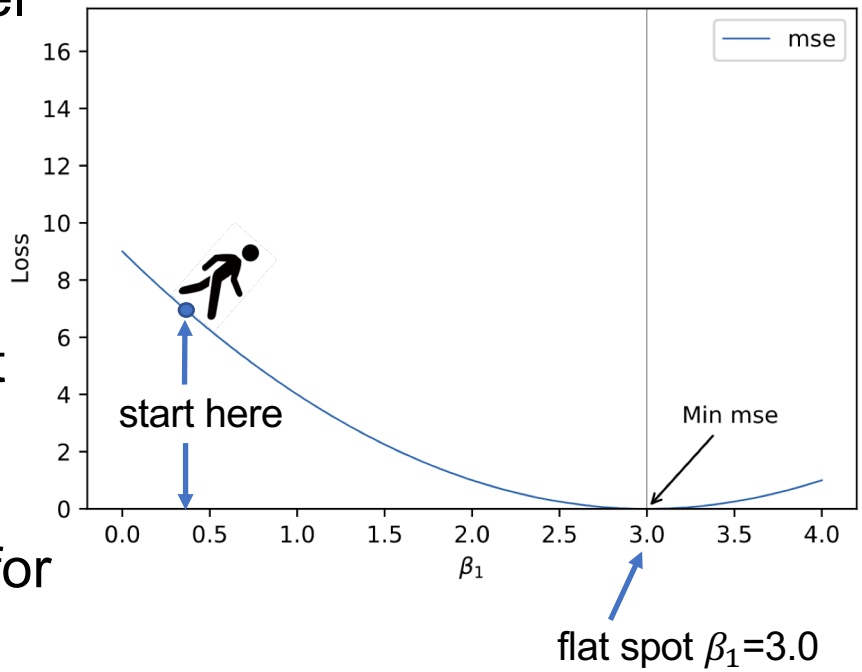
The regularization mechanism

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

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

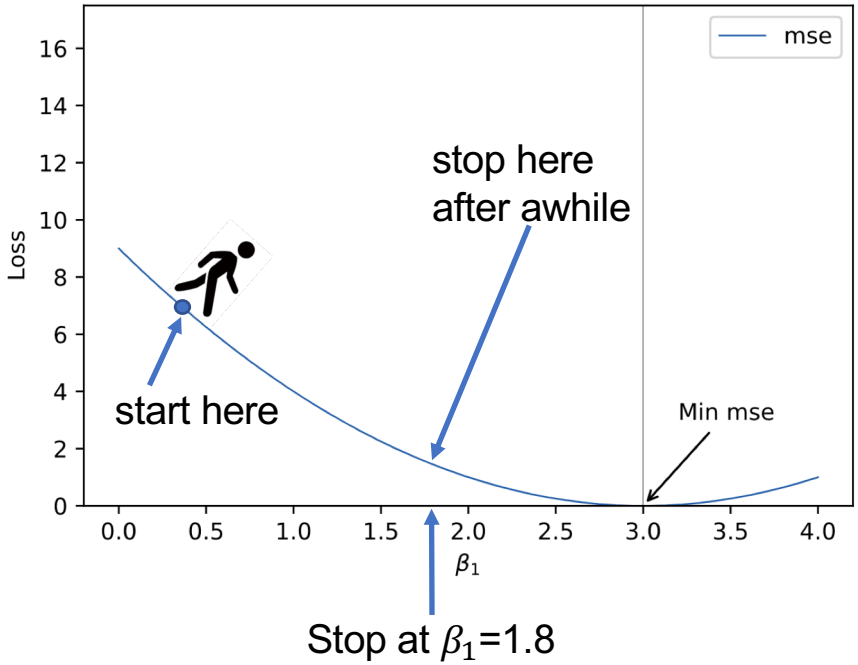
Detour: How training works in 1 dimension

- 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 β_1 and then just walk β_1 downhill until you hit flat spot
- The derivative/slope of loss function is 0 at the minimum loss
- The β_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

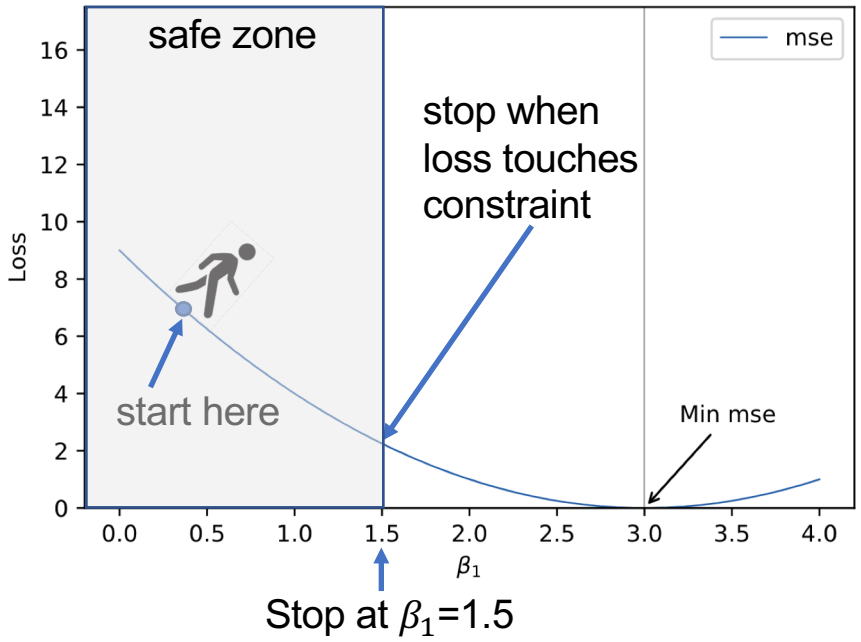


Improving early termination of training

- 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 β_1 in [0,t) and go downhill and stop at minimum or when $\beta_1 \geq t$

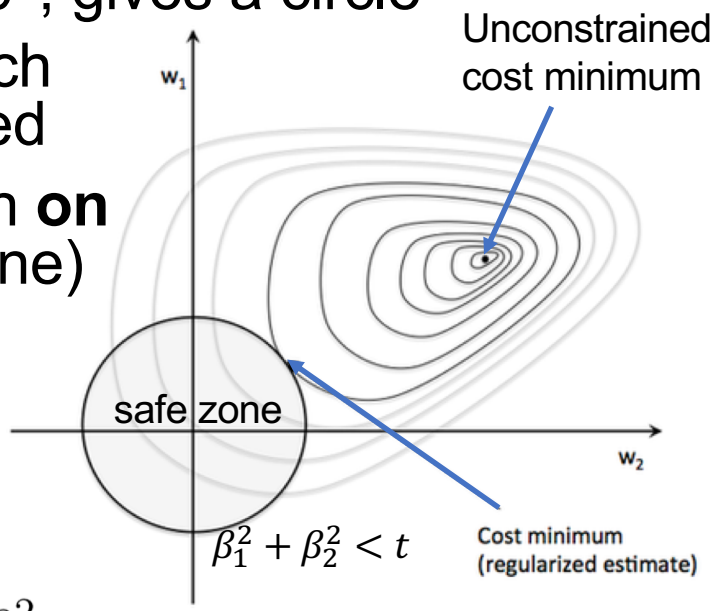
Minimizing subject to a *hard constraint*

- 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°, gives a circle
- Recall formula for circle; constrain β_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**



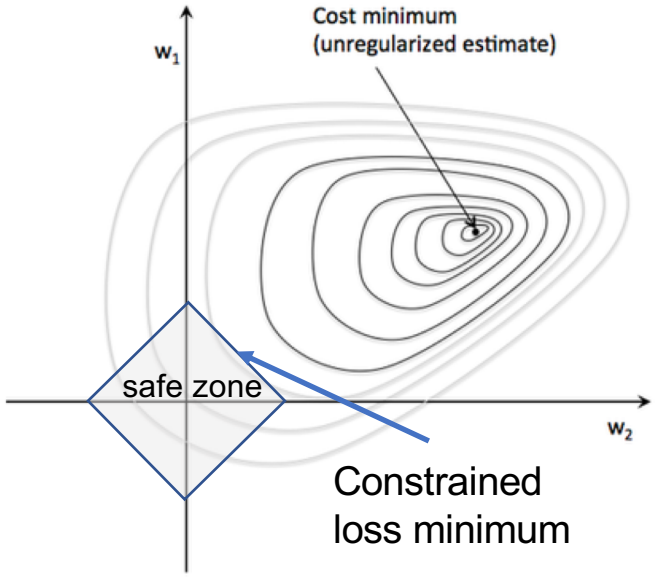
$$\sum_{j=1}^{p} \beta_j^2 < t$$

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 β_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:

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

- Subject to either (L2 or L1):

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

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

Detour: Lagrange Multipliers

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

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

$$\mathscr{L}(\beta, \lambda) = \mathscr{L}(\beta) + \lambda(\sum_{j=1}^{p} \beta_j^2 - t)$$ for some λ

(λ 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

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

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

- Drop *t* since t is constant & doesn't affect minimizing
- This is a **soft constraint** and penalty increases as β_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 β_i's and hyperparameter λ

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

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

- β_0 is just $\overline{y}$, assuming zero-centered data set (next slide)
- **Note**: β_0 is NOT included in penalty, but is used in $\mathscr{L}(\beta)$

Normalizing/standardizing data

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

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

- Computing β_0 as simply mean(y) for L1/L2 linear regression requires normalization

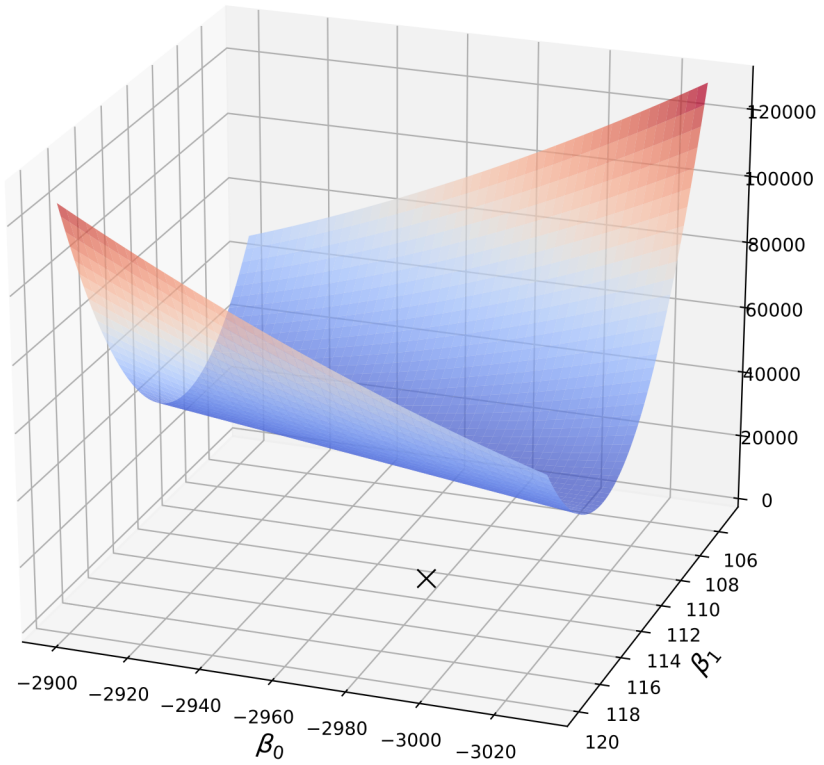
More reasons to standardize variables

- We have just one λ for all coefficients so vars must be in same range or big β's prevent regularization of small coefficients (big var ranges cause big β's)

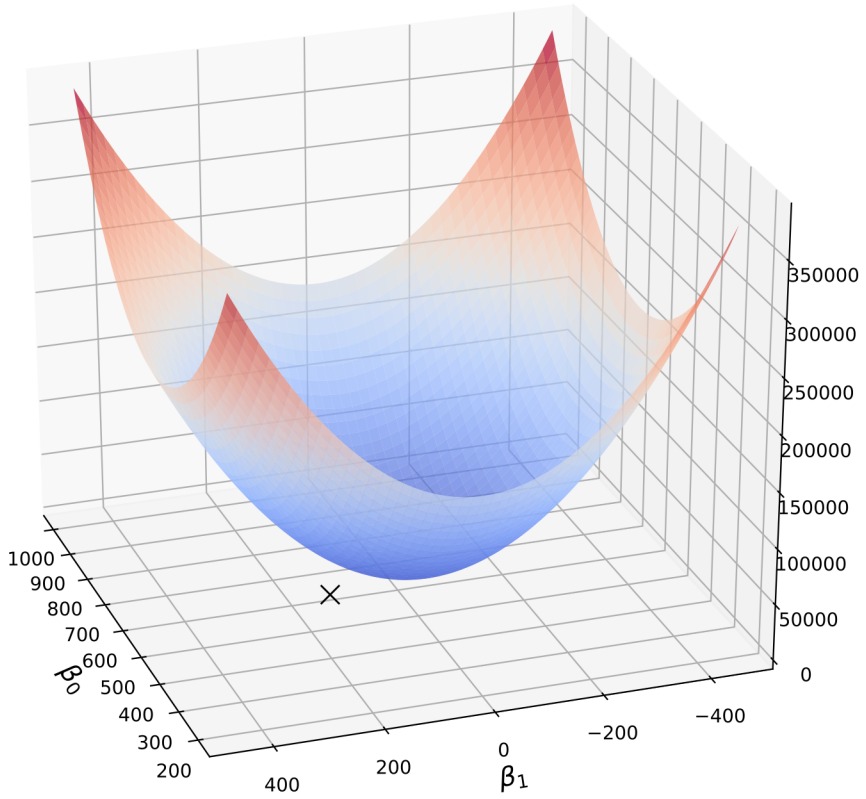
$$\lambda \sum_{j=1}^{p} \beta_j^2$$

- 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, η, that is small enough for x_2 (so we converge) is too slow for x_1

Loss function shape for normalized vars



Untouched variables



Normalized variables

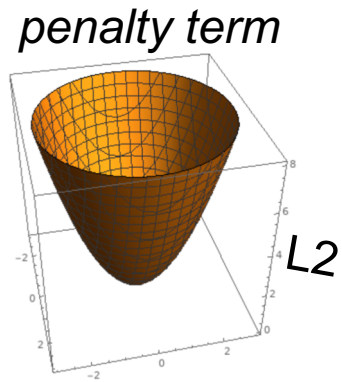
Cheese consumption vs bedsheet strangulation death data set

UNIVERSITY OF SAN FRANCISCO

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

How penalty term restricts β_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 β_i location towards origin as penalty curve is anchored at origin
- L1&L2 **shift β_i's towards 0** because penalty approaches 0 only at β=0
- Soft constraint makes larger β_i very unlikely due to increased penalty away from origin (hard constraint makes large coefficients > t impossible)



The effect of regularization

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

Regularized logistic regression

Optimizing likelihood with penalty terms

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

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

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

Note: β_0 is NOT included in penalty, but is used in $\mathscr{L}(\beta)$ term

Fitting logistic regression β_i's and λ

- ESLII book p125 says to find L1 β_0 and $\beta_{1..p}$ that maximize:

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

- We minimize the negative of that to find β of max likelihood
- Means we must find β_0 differently than for $\beta_{1..p}$
- As before, find λ by computing minimum loss for different λ values, pick λ that gets min loss on validation set

L1 Logistic gradient is tricky to get right

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

$err = \mathbf{y} - \sigma(\mathbf{X}' \cdot \beta')$ $(error\ vector \text{ is } n \times 1\ column\ vector)$

$\frac{\partial}{\partial \beta_0} = mean(err)$ $(usual\ log\text{-}likelihood\ gradient;\ use\ current\ \beta')$

$r = \lambda \operatorname{sign}(\beta')$ $(regularization\ term\ p+1 \times 1\ column\ vector)$

$r[0] = 0$ $(kill\ \beta_0\ position\ but\ keep\ as\ p+1 \times 1\ vector)$

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

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

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 β_i from getting much bigger than the others
- L1 does not discourage β_i from fading to 0, but L2 discourages
- For $\beta < 1$, β^2 gets smaller, not bigger; L2 can't push β to left very much near 0 as it's shifting by fraction of β
- In practice, we see L1 zero out unpredictive vars

Key implementation details

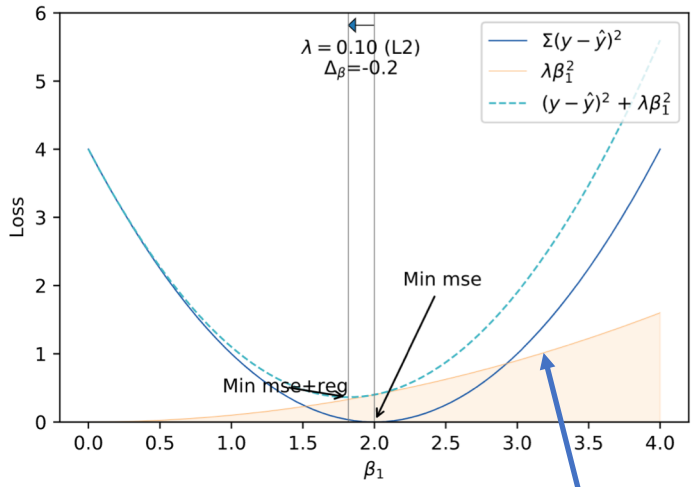
- Minimize $\mathscr{L}(\beta, \lambda)$ by trying multiple λ and choosing parameters from fitted model getting lowest **validation** error
- Always normalize X before fitting models
- If 0-centered x_i then $\beta_0 = mean(y)$ for L1 and L2 regression
- Logistic regression has no closed form for β_0
- 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)

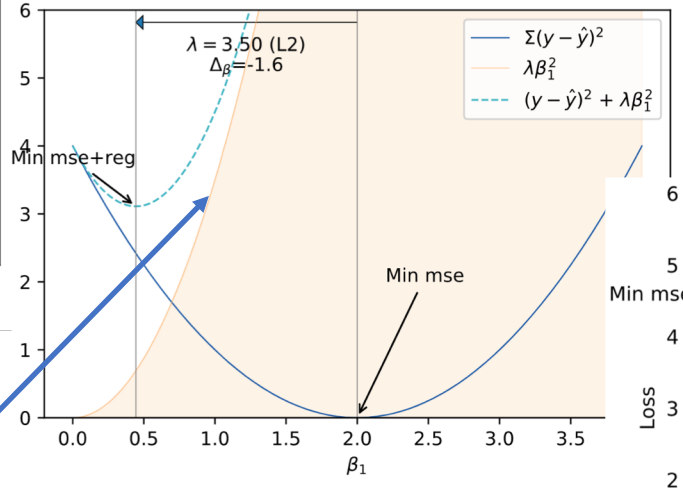
- Both L1 and L2 increase bias (reduce ”accuracy” of fit, predictions)
- L1 useful for variable / feature selection as some β_i go to 0
- L2 useful when you have collinear features (e.g., both tumor radius and tumor circumference features)
  - Collinearity increases coefficient variance, making them unreliable
  - And L2 reduces variance of β_i estimate which counteracts effect of collinearity
- With regularization, we don’t get unbiased β_i estimates
  - Expected value of β_i estimate is no longer β_i since we’ve constrained β_i’s
  - Consequence: we no longer know effect of x_i on target *y* via β_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?)

EPILOGUE:
More on the effect of increasing hyper parameter λ

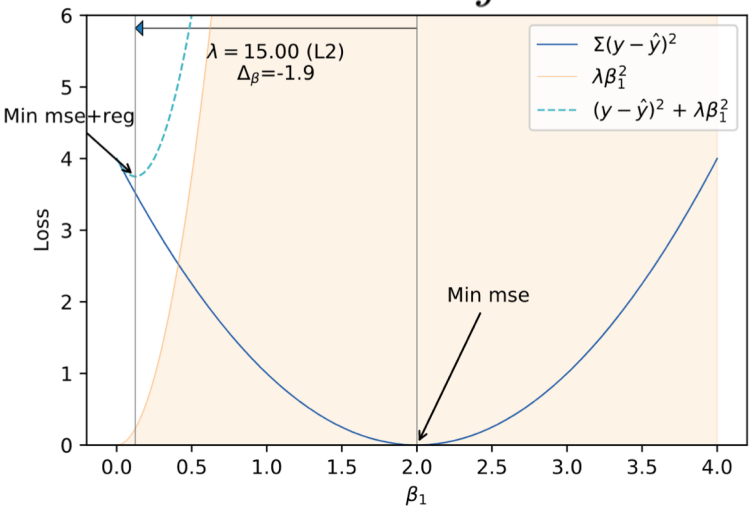
Visual effect of increasing λ for L2
(single variable)



Takes roughly $\lambda = 300$ to get β_1 to 0!



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



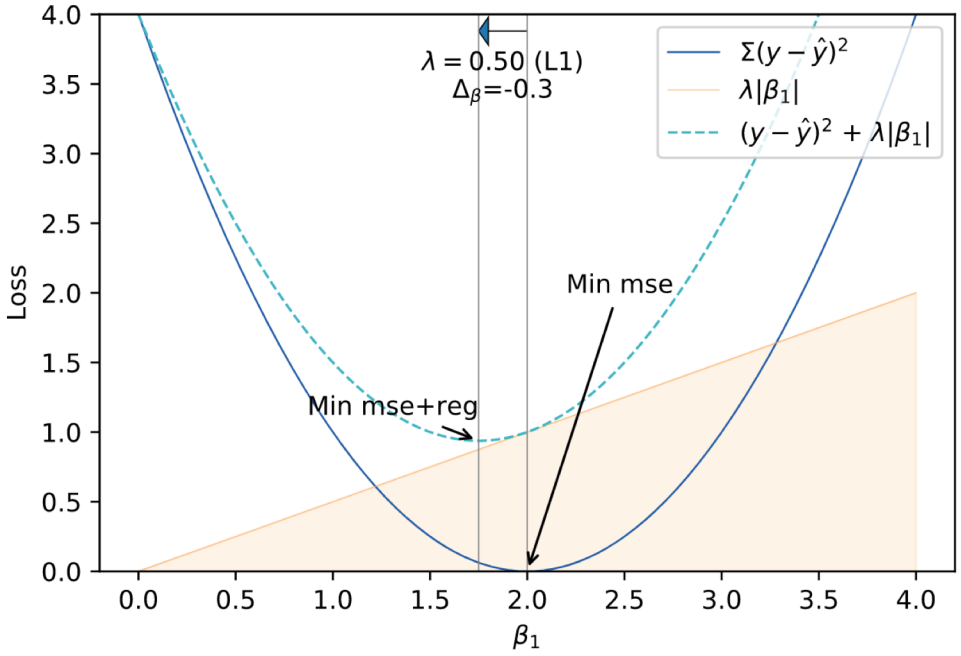
Decelerating increase in loss,
decelerating shift of β_1 towards 0

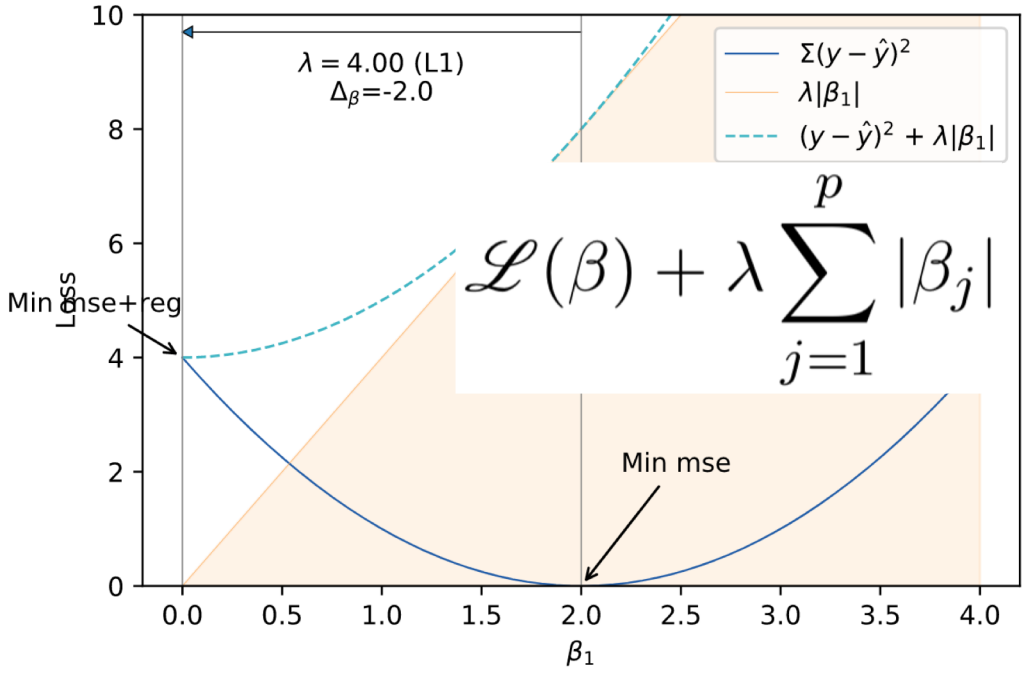
See https://github.com/parrt/msds621/blob/master/notebooks/linear-models/viz-regularization.ipynb

Visual effect of increasing λ for L1

Smooth/linear increase in loss and shift of β_1 towards 0

Doesn't take huge λ to get β_1 to 0

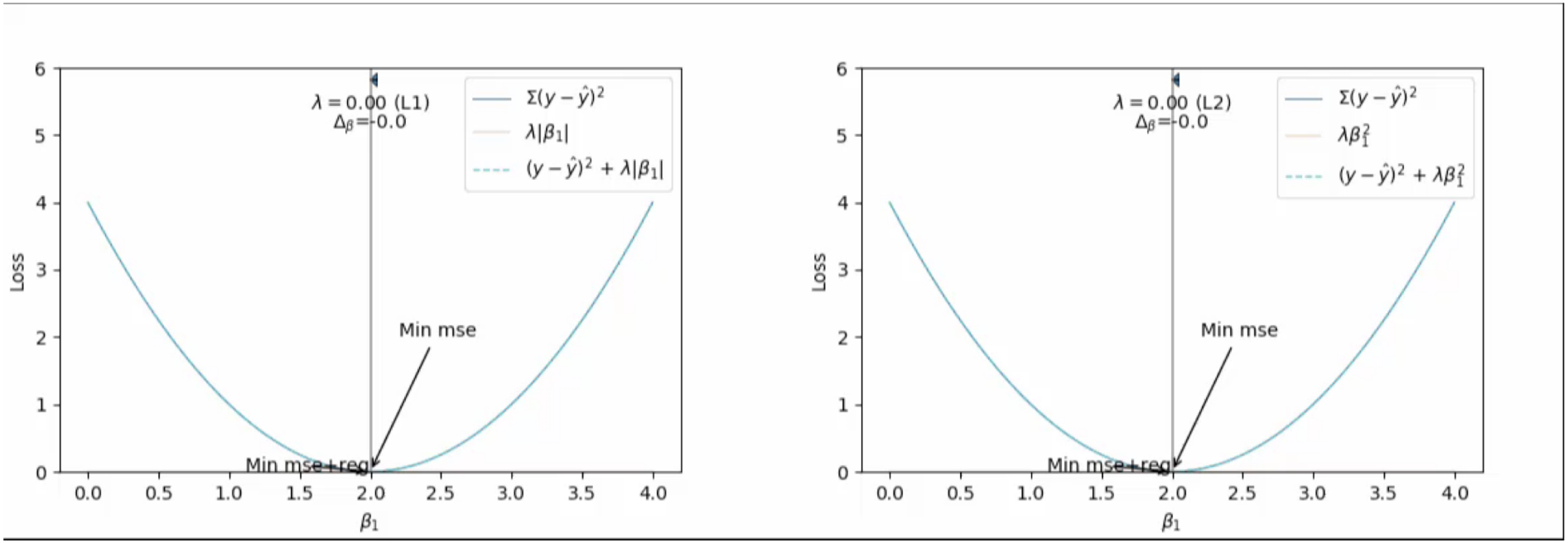




See https://github.com/parrt/msds621/blob/master/notebooks/linear-models/viz-regularization.ipynb

L1 vs L2 regularization visually

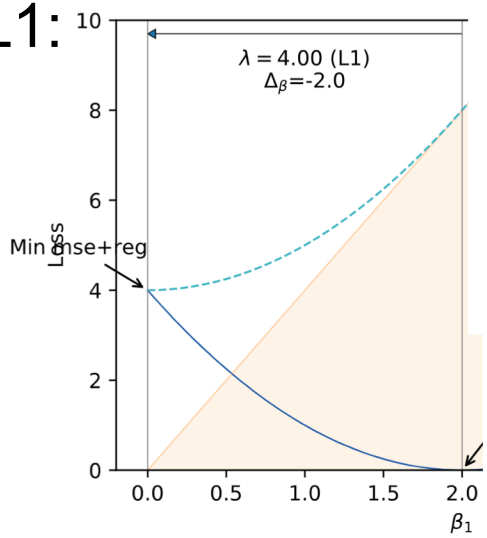
- Notice how L1 has no problem pushing β to 0 whereas L2 struggles to get there (this is a video page)

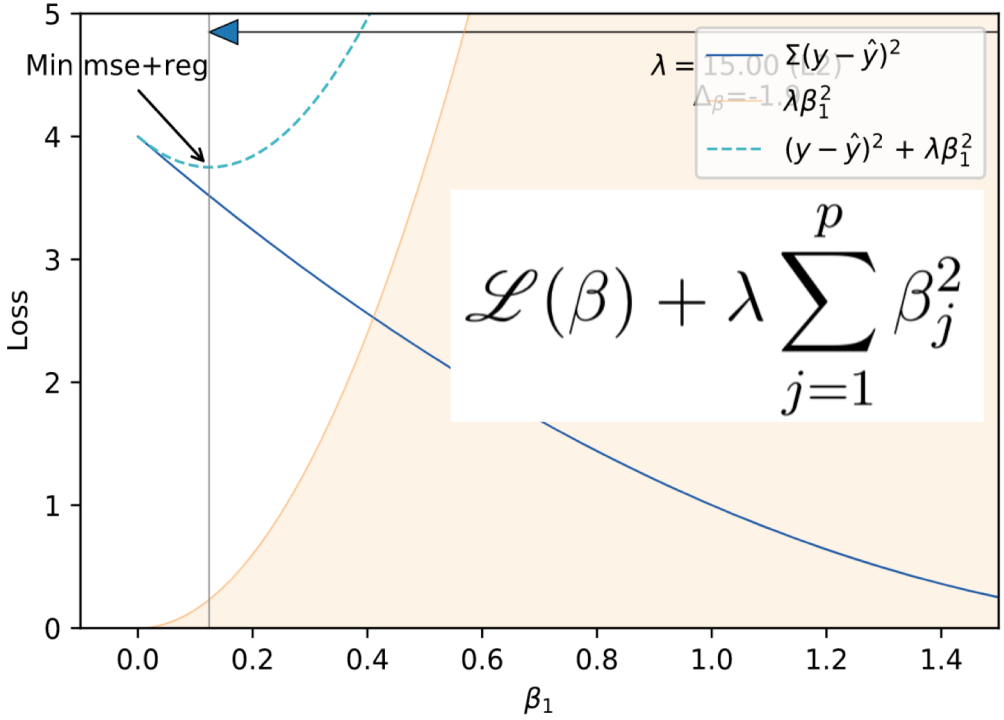


Animations from: https://github.com/parrt/msds621/blob/master/notebooks/linear-models/viz-regularization.ipynb

Zoom in on L2 trying to push β to 0

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





Why L1 can yield β_i=0 (and why not L2?)

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

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

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 λ gets big enough
- There's a theorem that says P($\beta_i = 0$) → 1 as $p \to \infty$
- As $p \to \infty$, diamond shape collapses in on itself to a point
- Distance between discontinuities (feasibility points / axis crossings) approaches zero, making it less likely β_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😀)