# 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

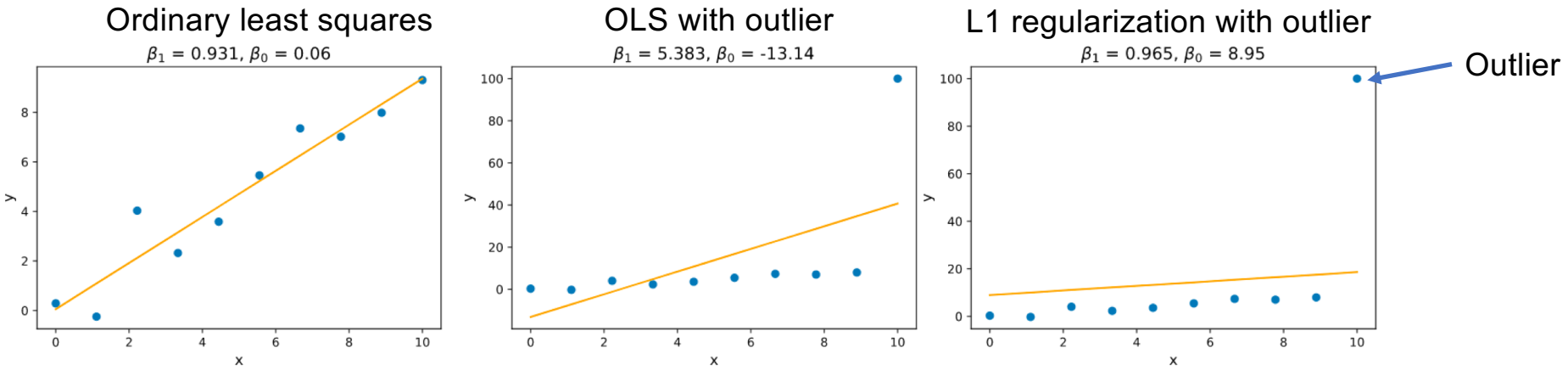
# Motivation for regularization

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

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

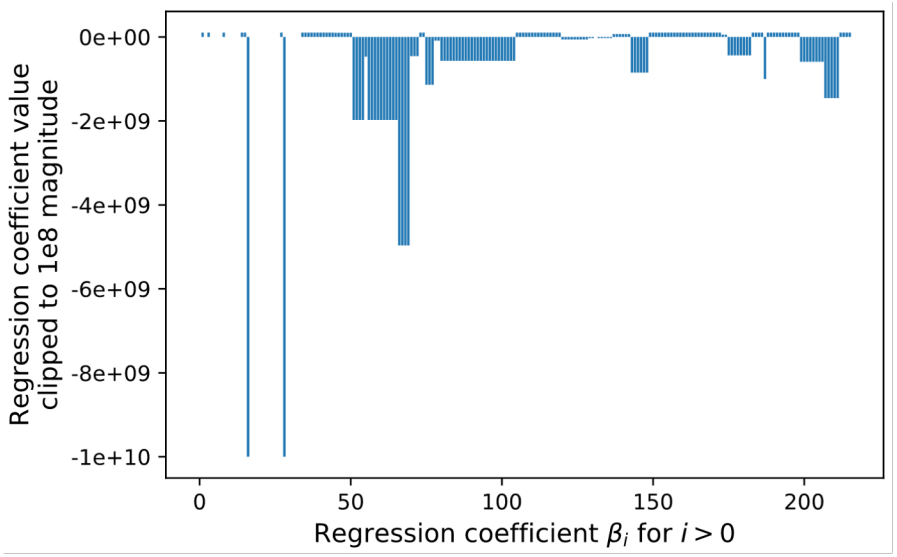
# Let's make a deal!

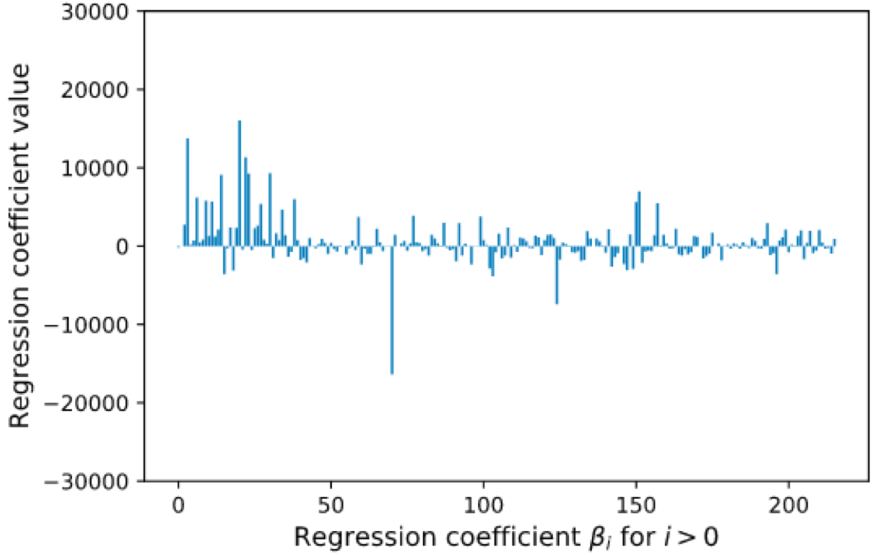
- Let's trade some model bias for improved generality
- Consider an example of a simple data set with OLS fit (on left)
- Now, send y[x==10] to 100; we get skewed line & bad R^2
- Regularization brings slope back down but with some bias



# Ames housing data set (regressor)

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

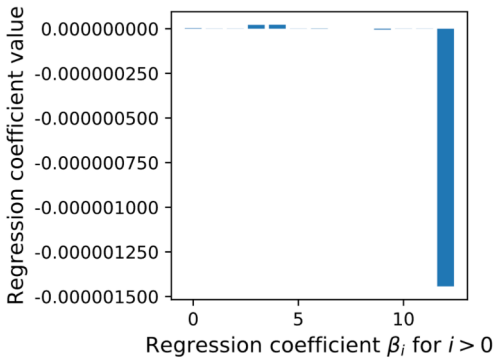


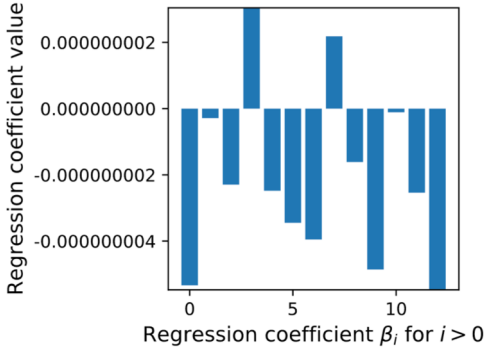
# Wine classifier fails w/o regularization

- Wine data set (130 records, 14 numeric vars)
- Test accur=0.59 (normalized data) w/OLS; accur=.98 w/regularization
- Normalizing also helps coefficient interpretation
- Regularization constrains coefficients to sane range

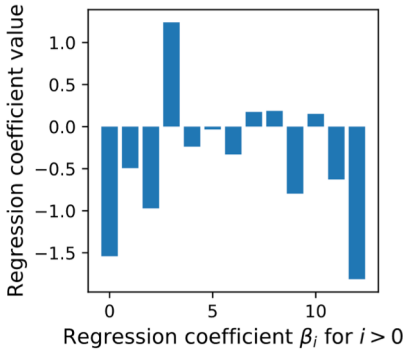
Ordinary least squares




Normalized




Normalized, regularized

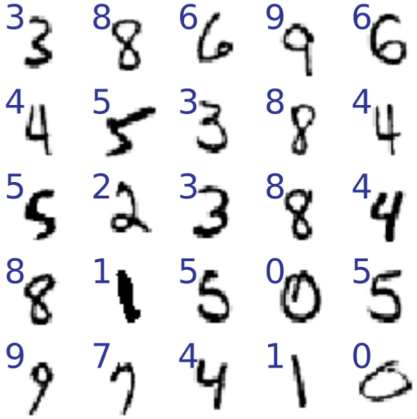



See https://github.com/parrt/msds621/blob/master/notebooks/linear-models/classifier-regularization.ipynb

# Example: classifier w/too many features

- Distinguish between ones and sevens (MNIST dataset)
- W/o regularization, accuracy score = .50 (same as guessing)

MNIST sample



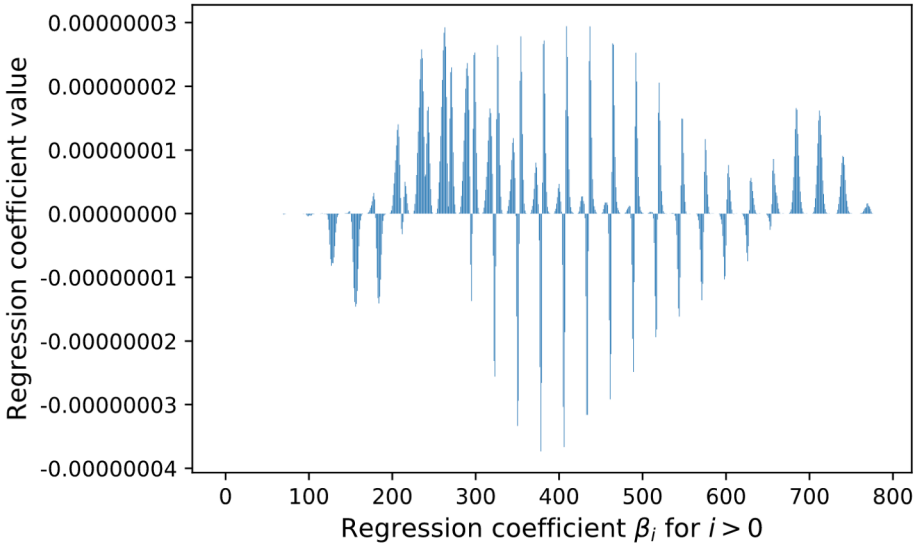
ones sample

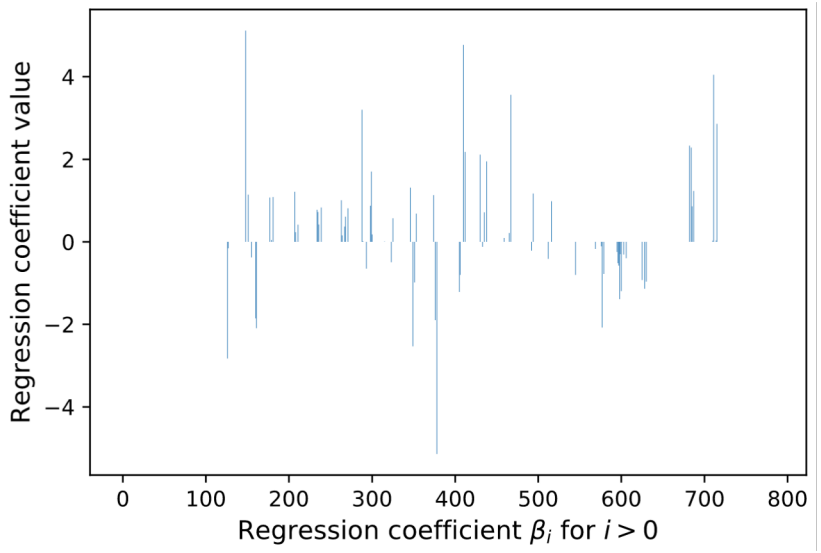
sevens sample

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

# Compare coefficients w/o & with L1 reg

Check out the scale difference and how reg (L1) kills off some variables
Test accur=0.49 (normalized data) w/OLS; accur=.98 w/regularization (L1)




Note out L1 regularization has zeroed out lots of coefficients

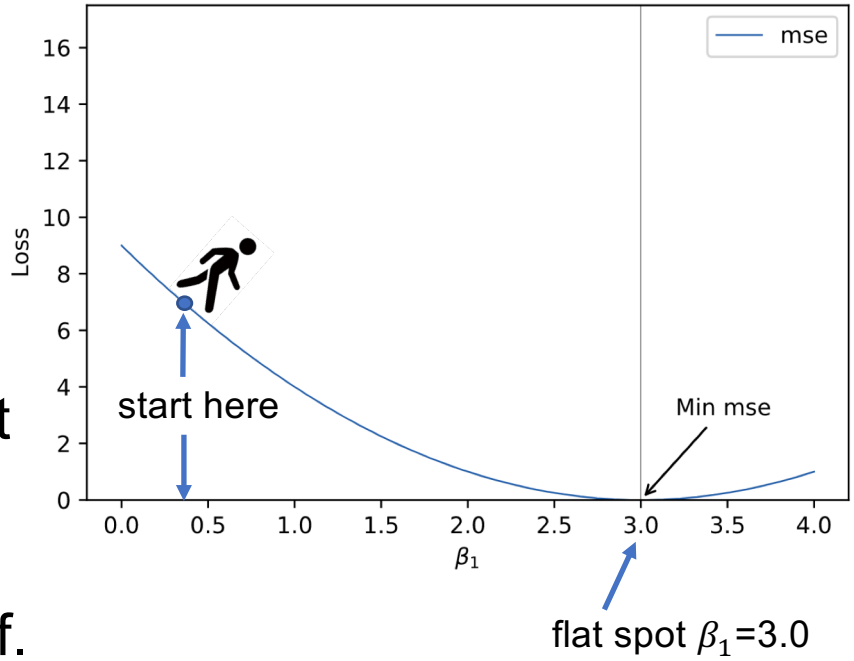
# The regularization mechanism

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

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

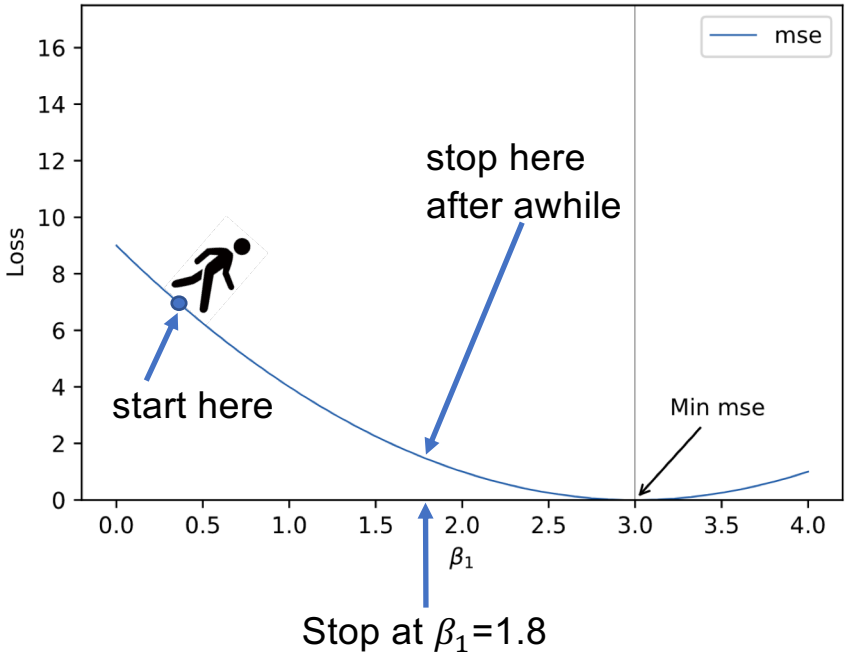
# Detour: How training works in 1 dimension

- Loss function (cost) is a function of model parameters (betas), data set
- Minimize 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 downhill until you hit flat spot
- The derivative of loss function is 0 at the minimum loss
- The $\beta_1$ at loss minimum is best fit coeff. for training set



# 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

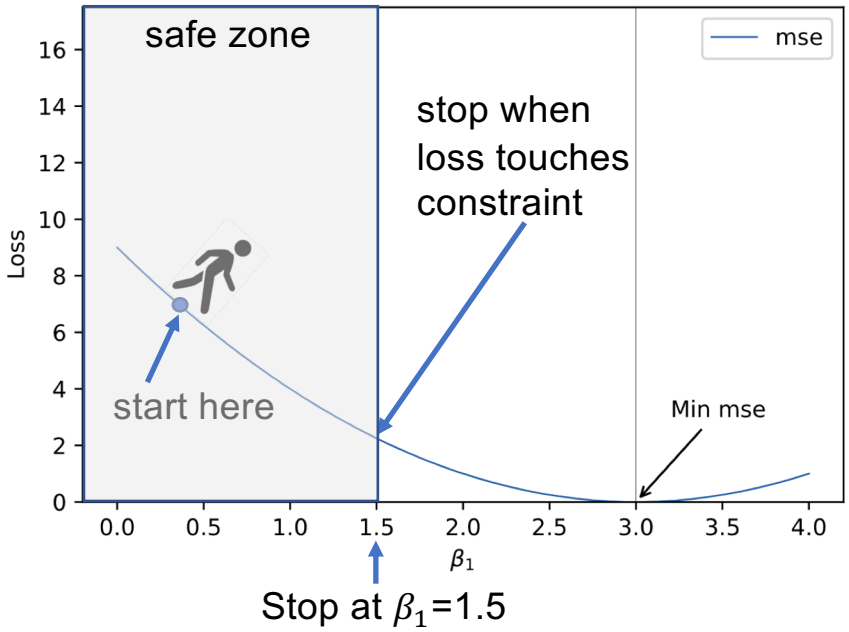


# Improving early termination of training

- Stopping after fixed amount of time depends on how fast our ”hiker” moves along loss curve and hard to pick duration
- Checking if the loss on **validation set** starts going up is more expensive; each iteration must compute loss on another set
- 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 $\beta_1 \geq t$

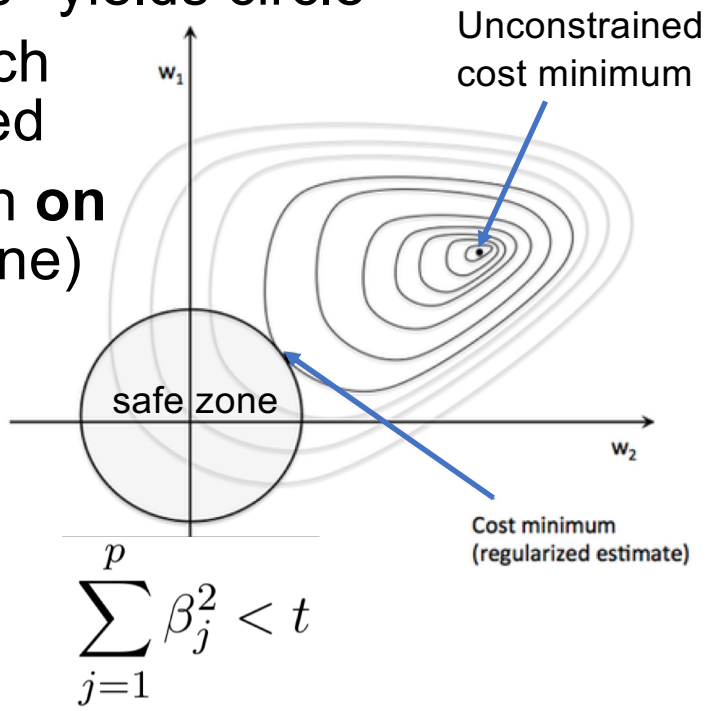
# Minimizing subject to a *hard constraint*

- 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
- It could be min loss is in safe zone, meaning regularization wasn't req'd



# Minimizing in 2D subject to hard constraint

- Spinning segment [0,$t$] around origin 360° yields 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**

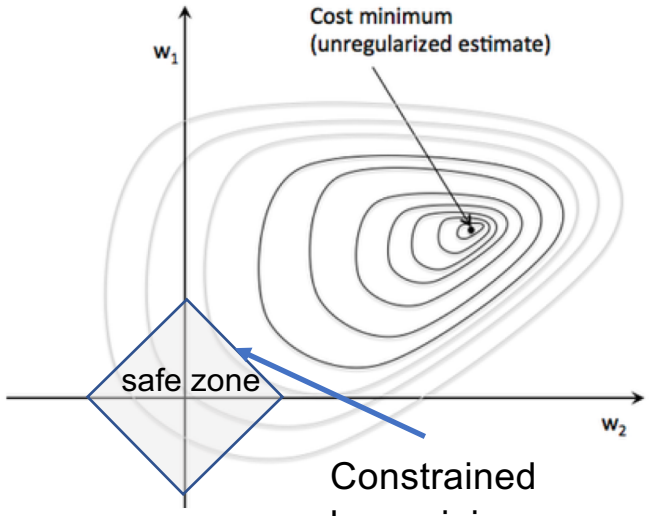



$$\sum_{j=1}^{p} \beta_j^2 < t$$

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

# Minimizing in 2D subject to hard constraint

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




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

Notice 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 loss function:

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

- Subject to either (L2 or L1):

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

# Detour: Lagrange Multipliers

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

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

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

($\lambda$ and *t* are related one-to-one 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: $\mathscr{L}(\beta) + \lambda \sum_{j=1}^{p} \beta_j^2$

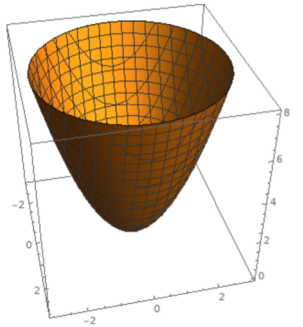
For L1: $\mathscr{L}(\beta) + \lambda \sum_{j=1}^{p} |\beta_j|$

- Drop *t* since $t$ is constant & doesn't affect minimizing
- This is a **soft constraint** and penalty increases as $\beta_i$'s move away from origin; there's no hard cutoff

# 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 increases with sum of coefficient magnitudes
- Regularization penalty term **increases loss** but skews it towards origin as penalty curve is anchored at origin; that **shifts $\beta_i$'s to towards 0**
- Soft constraint makes larger $\beta_i$ very unlikely due to increased penalty away from origin

penalty term

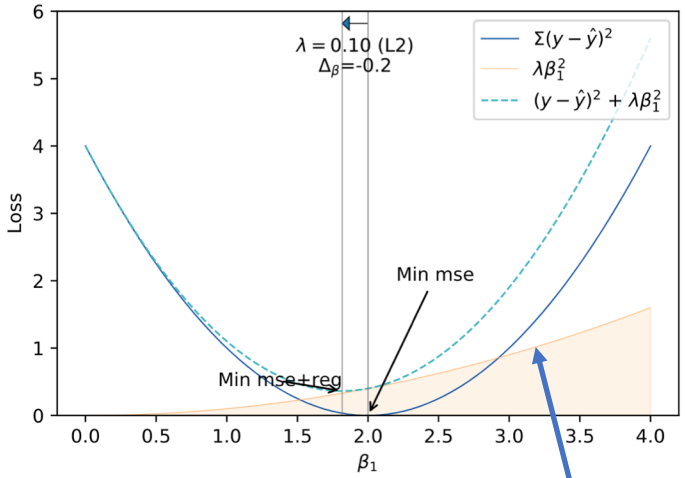


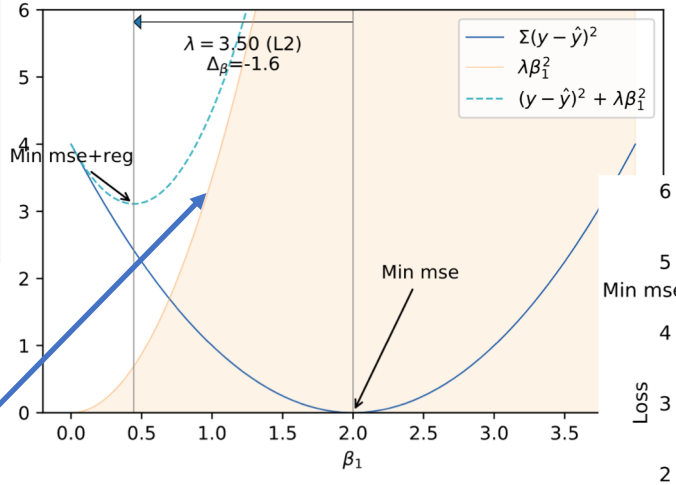
L2

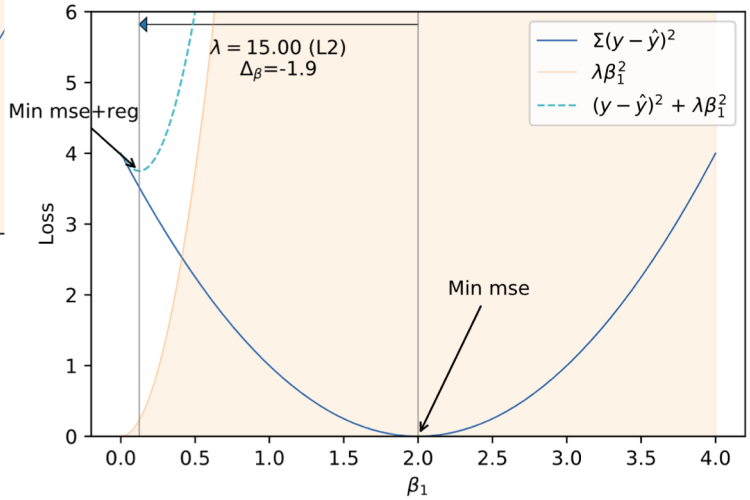
# The effect of regularization

- $\lambda$ is independent of training data, so can reduce model's dependence on data during fit, which regularizes model
- 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 to 0
- L1 does not discourage $\beta_i$ from fading to 0, but L2 discourages
- Squaring coefficients for L2 in constraint discourages any $\beta_i$ from getting much bigger than the others; squeezes $\beta_i$ together

# L2 Visually



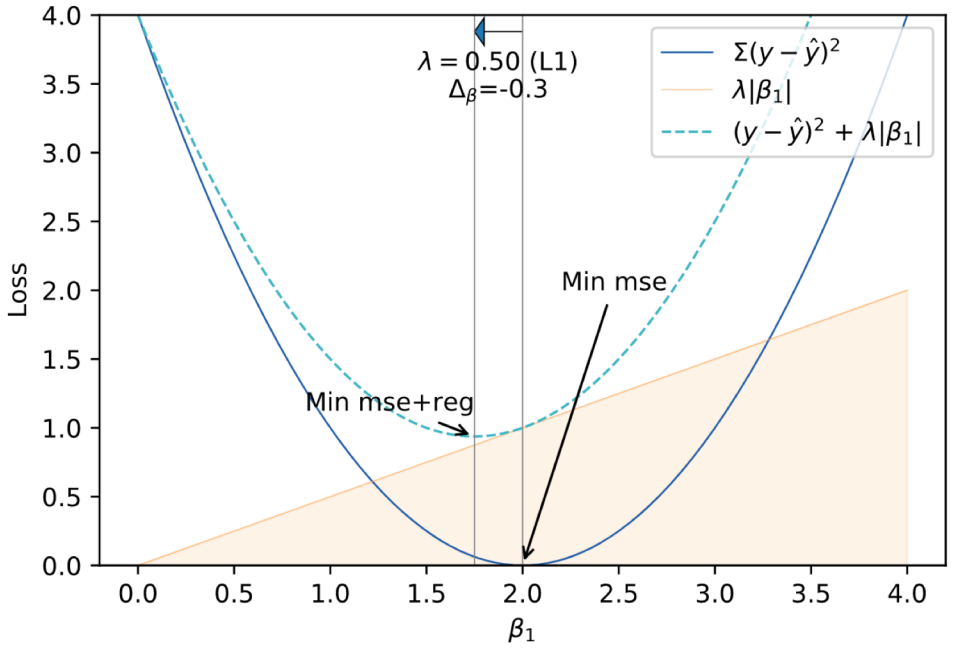



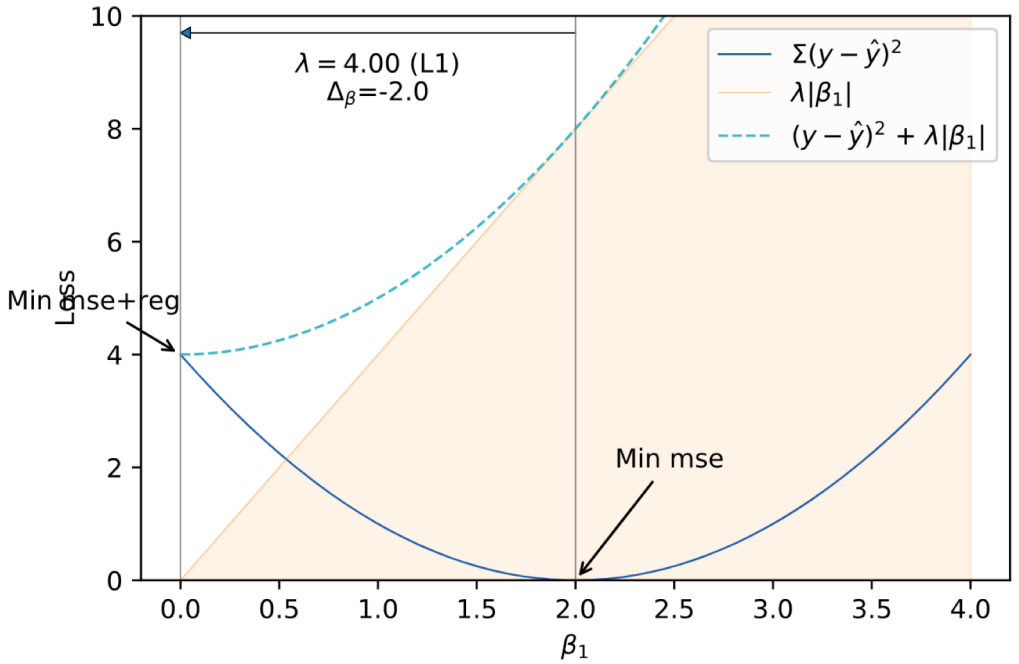




Uneven increase in loss, shifting coeff towards 0

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

# L1 Visually

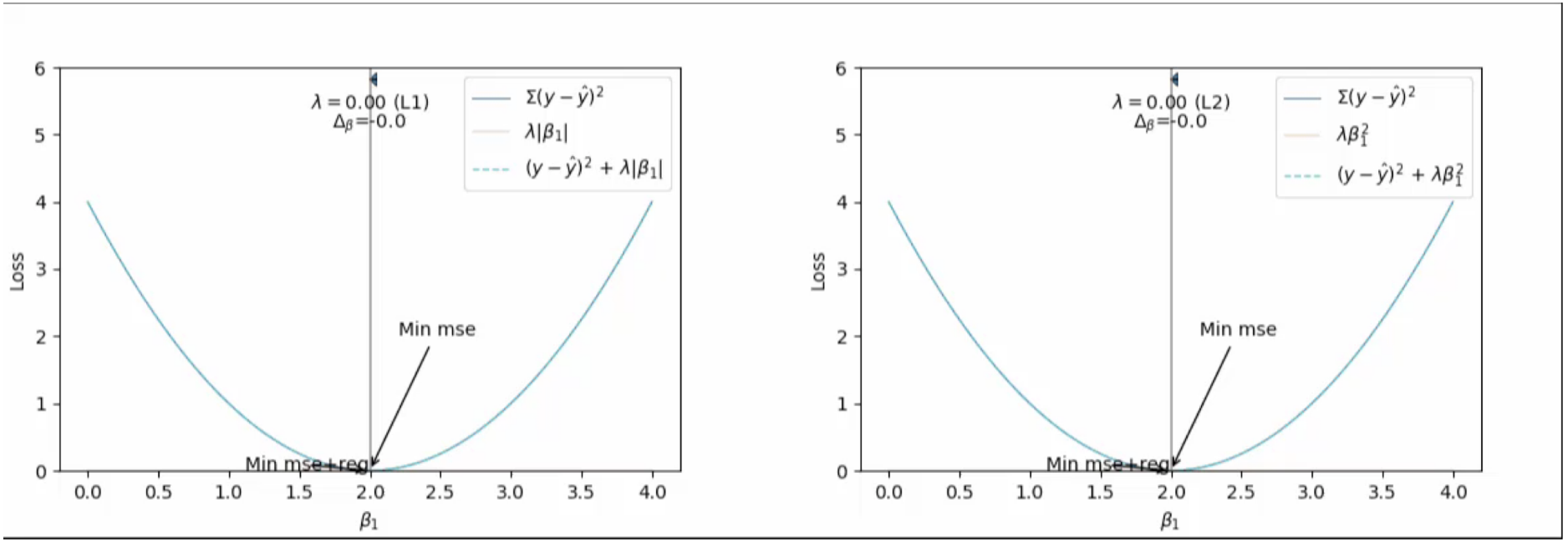



See https://github.com/parrt/msds621/blob/master/notebooks/linear-models/viz-regularization.ipynb

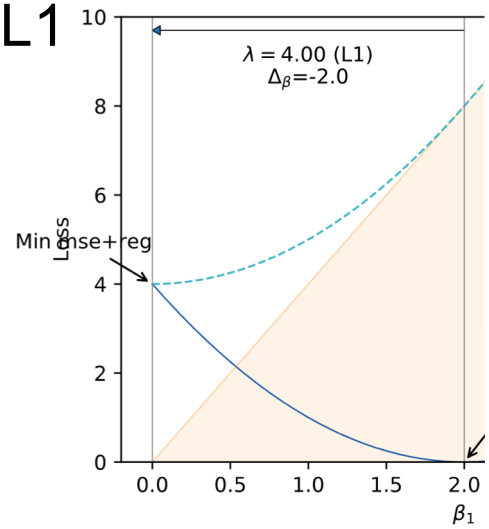
# L1 vs L2 regularization visually

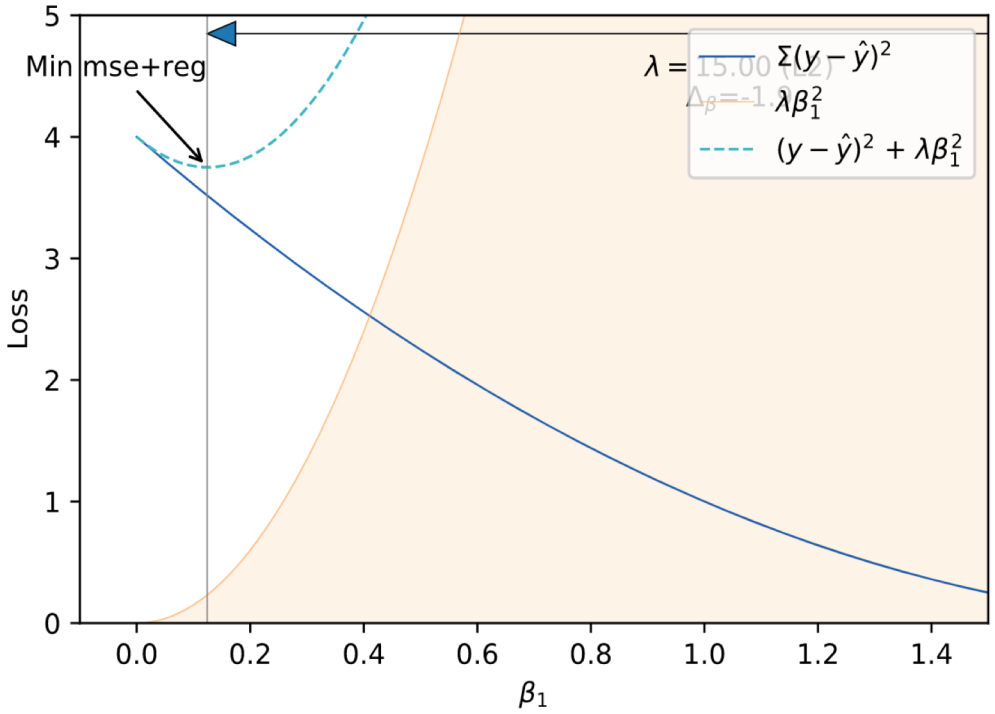
- Notice how L1 has no problem pushing $\beta$ to 0 whereas L2 struggles to get there (this is a video page)



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

- For $\beta < 1$, $\beta^2$ gets smaller, not bigger
- L2 can't push loss up, $\beta$ to left very much near 0
- Compare to L1



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

- (p=1) Derivative of penalty term w.r.t $\beta$ is $2\lambda\beta$ for L2, 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 $\beta$
- For L1, derivative is just $\lambda$sign($\beta$), independent of $\beta$ magnitude, so increases loss at same rate ($\lambda$) even near 0

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

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

# 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 $\lambda$ 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 $\beta_i$ sits on an edge of the diamond; mostly likely sits at a discontinuity

# 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 computing minimum loss for different $\lambda$ values, pick $\lambda$ that gets min loss on validation set

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

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

# Normalizing data

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

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

- Reasons:
  - One $\lambda$ for all coefficients so must be in same range or big coefficients prevent regularization of small coefficients

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

  - Finding coefficients is often more efficient for normalized data sets
  - Allows us to compute $\beta_0$ as simply mean(y) for L1/L2 linear regression

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

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

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

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

- Means we must find $\beta_0$ differently than for $\beta_{1..p}$
- Find $\lambda$ by computing minimum loss for different $\lambda$ values, pick $\lambda$ that gets min loss on validation set
- We'll examine the implementation in gradient descent lecture

# Key takeaways

- Regularization increases generality at cost of some bias
- Does so by restricting size of parameters with hard constraint (conceptually) or soft constraint using penalty term
- 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 $\mathscr{L}(\beta, \lambda)$ 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 $\beta_0 = mean(y)$ 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 $\beta_i$