Table of Contents I

notes

- this is the table of contents of this document; each section corresponds to a specific part of the course

Regularization

notes

Contents map

developed content units	**taxonomy levels**
regularization	u1, e1
regularization path	u1, e1
ridge regression	u1, e1
Lasso	u1, e1

prerequisite content units	**taxonomy levels**
bias variance tradeoff	u1, e1

notes

Main ILO of sub-module "Regularization"

Compare different regularization techniques (ridge, lasso, and elastic net) by evaluating their mathematical formulations, graphical interpretations, and practical implications

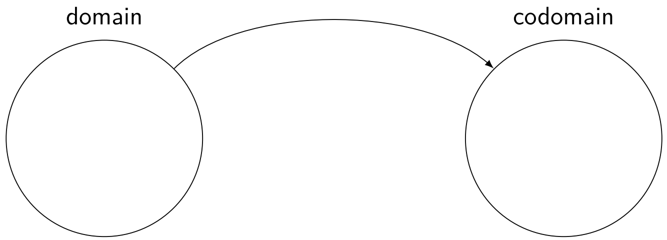
Interpret regularization paths from Lasso regression plots to identify the relative importance of features in predictive modeling

notes

- by the end of this module you shall be able to do this

Regularization = trading off variance with some bias

main intuition: if $\widehat{\theta}$ has a variance V, then $0.9\widehat{\theta}$ has a variance $0.81V$



(i.e., the potential datasets) *(i.e., the potential estimators)*

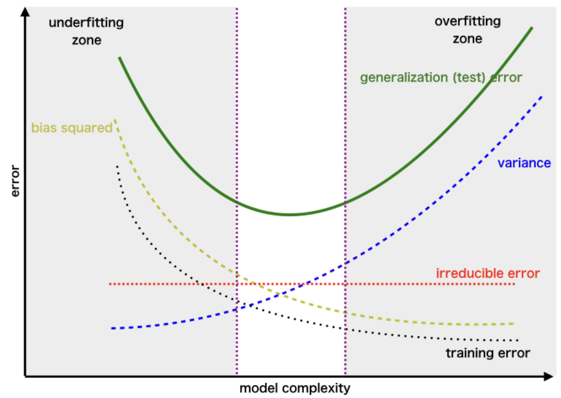
at the same time this will **likely** increase the bias:

notes

- Let me explain this important concept carefully. When we shrink our estimator by multiplying it by 0.9, we're reducing its variance - that's good! But look what happens to bias...
- The blue line represents our original estimator - it has some variance (the spread) but is unbiased (centered at the true value). The green line shows what happens when we shrink it - the variance decreases, but now we've introduced some bias.
- This is the fundamental trade-off in regularization: we accept some bias to reduce variance. The key question is: how much bias should we accept to get how much variance reduction?

Regularization = trading off variance with some bias

if $\widehat{\theta}$ has a variance V and bias B, then there will be a specific $\gamma\widehat{\theta}$ that minimizes $V + B^2$ (but we can't know a priori which γ is best):



notes

- This graph shows the sweet spot we're trying to find. On the left, we have high bias (underfitting) - our model is too simple. On the right, high variance (overfitting) - our model is too complex.
- The magic happens in the middle! Regularization helps us find that optimal point where we balance these two competing objectives.
- Notice how the test error (what we really care about) is minimized at a point where neither bias nor variance is zero. This is counterintuitive but crucial!

the Stein's effect

notes

notes

An interesting example: the Stein's effect (in words)

(caveat: the next 3 slides are just motivational, not for the exam)

when estimating several parameters simultaneously, it's possible to improve overall estimation accuracy by borrowing strength across parameters, even if individual estimators may appear less accurate when considered in isolation

- This is one of my favorite statistical phenomena - it's quite surprising when you first see it!
- Imagine you're estimating multiple parameters. Normally, you'd estimate each one separately. But Stein showed that's not optimal - you can do better by "shrinking" estimates towards each other.
- Think about it like this: if you have limited data for each parameter, you can get better overall estimates by letting them share information.
- This is the foundation for many regularization techniques we use today.

An example of the Stein's effect in formulas

Given

$$y_t = \theta_t + e_t \qquad e_t \sim \mathcal{N}\left(0, \sigma^2\right) \text{ i.i.d.} \qquad \theta_t \in \mathbb{R} \qquad \boldsymbol{y} := \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \qquad \boldsymbol{\theta} := \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_N \end{bmatrix}$$

Then

$$\boldsymbol{\theta}_{LS} = \boldsymbol{y} \quad \text{does not minimize} \quad \mathbb{E}\left[\left\|\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta}\right\|^2\right]$$

and

$$\boldsymbol{\theta}_{JS} := \left(1 - \frac{N-2}{\|\boldsymbol{y}\|_2^2}\sigma^2\right)\boldsymbol{y}$$

has lower MSE than the LS solution

notes

- Here's the math behind what I just described. The maximum likelihood estimate (LS) seems natural, but it's not optimal in terms of mean squared error!
- The James-Stein estimator (JS) looks strange - it's shrinking our estimates towards zero. But remarkably, it gives better overall performance.
- The key is in the shrinkage factor $(1 - \frac{N-2}{\|\boldsymbol{y}\|^2}\sigma^2)$. When our estimates are large, we shrink less. When they're small, we shrink more.
- This was so surprising when discovered that it's called the "Stein paradox". The LS estimator is actually inadmissible when N 3!

What is happening?

In this case

$$\boldsymbol{\theta}_{JS} = \left(1 - \frac{N-2}{\|\boldsymbol{y}\|_2^2}\sigma^2\right)\boldsymbol{y}$$

is a "regularized" version of

$$\boldsymbol{\theta}_{LS} = \boldsymbol{y}$$

notes

- What we're seeing here is regularization in action! The JS estimator is regularizing the LS estimate by shrinking it towards zero.
- This shrinkage introduces some bias (our estimates are no longer unbiased) but reduces variance enough that the overall MSE improves.
- This is exactly the bias-variance tradeoff we saw earlier, now in a concrete mathematical form.
- Modern regularization techniques like ridge and lasso are direct descendants of this insight.

ridge regularization

notes

One of the most used regularization techniques: L_2 (a.k.a. "ridge")

$$J(\theta) = J_{\text{original}}(\theta) + \gamma \|\theta\|_2^2$$

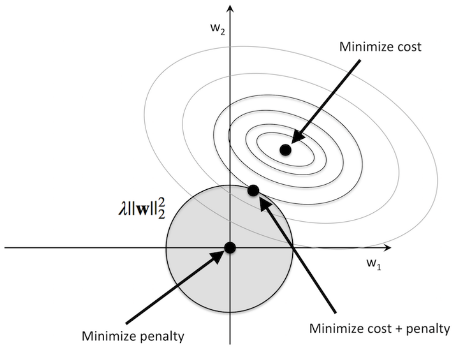
Animation: `https://www.geogebra.org/m/myfghjzg`

notes

- Now let's talk about ridge regression, one of the most practical regularization techniques.
- The key idea is simple: we add a penalty term to our cost function that discourages large parameter values.
- The γ parameter controls how much we penalize large values - larger γ means more regularization.
- I highly recommend checking out the linked animation - it shows visually how ridge regression pulls our estimates towards zero.
- Remember: we're not saying the true parameters are zero! We're just accepting some bias to reduce variance.

Ridge regression = the most common approach to regularization

$$J(\theta) = \sum_{i=1}^{m}\left(y_i - \theta_0 - \sum_{j=1}^{p}\theta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2$$



notes

- Here's ridge regression in its standard form. The first term is our usual least squares, and we've added an L2 penalty on the parameters.
- Notice we typically don't penalize the intercept term (θ_0) - we want to let the baseline prediction be whatever it needs to be.
- The figure shows how ridge regression shrinks parameters towards zero (but not exactly to zero). The stronger our λ, the more shrinkage.
- A practical tip: always standardize your features before applying ridge! The penalty treats all parameters equally, so they should be on similar scales.
- How to choose λ? We'll talk about cross-validation soon - that's the standard approach.

Lasso regularization

notes

The second most used regularization technique: L_1 (a.k.a. "lasso")

(actually this one typically works better than ridge!)

$$J(\theta) = J_{\text{original}}(\theta) + \gamma\|\theta\|_1$$

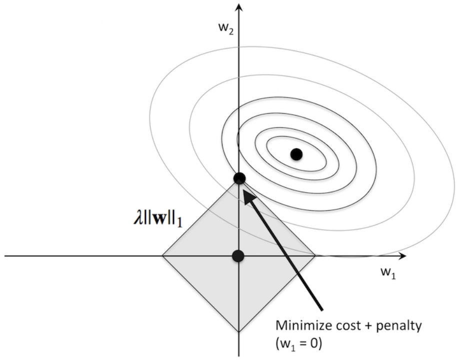
Animation: `https://www.geogebra.org/m/gaujemka`

notes

- Now let's meet lasso, ridge's more aggressive cousin. Instead of L2 penalty, we use L1.
- This small change has huge consequences! Lasso doesn't just shrink parameters - it can set them exactly to zero, performing feature selection.
- The animation shows this beautifully - notice how some parameters hit zero while others remain relatively large.
- In practice, lasso often outperforms ridge when we have many features but only a few are truly important.
- Like with ridge, standardization is crucial before applying lasso.

Lasso regression = the most common approach to regularization when one wants to promote sparsity (i.e., parsimonious models)

$$J(\theta) = \sum_{i=1}^{m}\left(y_i - \theta_0 - \sum_{j=1}^{p}\theta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{n}|\theta_j|$$

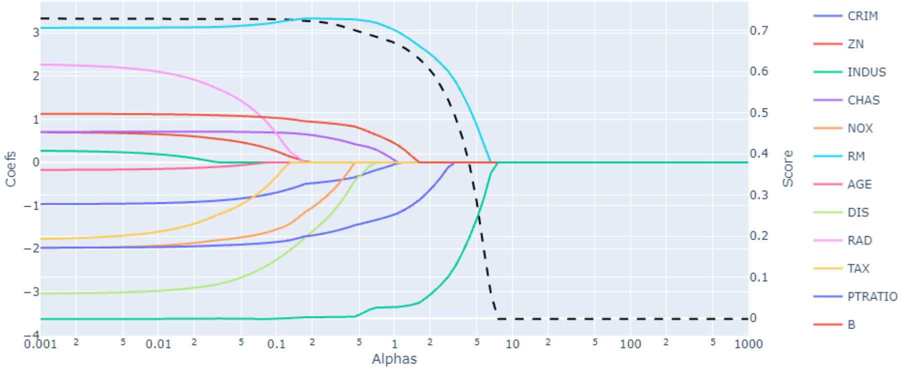


notes

- Here's the lasso formulation. It looks almost identical to ridge, but that absolute value makes all the difference!
- The figure shows why - the "pointy" constraint region means solutions often lie on the axes, setting some parameters to zero.
- This is incredibly useful when you want interpretable models or when you suspect many features are irrelevant.
- Practical note: lasso tends to select one feature from a group of correlated features, while ridge spreads the weight among them.
- If you have domain knowledge suggesting only a few features matter, lasso is often your best bet.

A plot you should always include in your reports: the L_1 regularization path

(this is an implicit way to understand the relative importance of the features)



notes

- This is one of the most informative plots you can make when using lasso. Let me walk you through it.
- On the x-axis we have the regularization strength (log scale). As we move right, λ increases and regularization gets stronger.
- Each line represents a feature's coefficient. Watch how as λ increases, coefficients get shrunk towards zero.
- The order in which coefficients hit zero tells us about feature importance - the later a feature disappears, the more important it is.
- In your projects, always include this plot! It gives immediate visual intuition about which features matter most.
- Notice how some features are only important for certain λ ranges - this is why choosing λ carefully matters.

extensions

notes

Elastic net = ridge + lasso

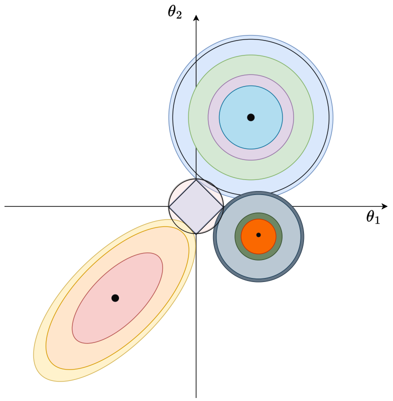
Last most-common regularization approach: combine the two into $\lambda_1 \|\boldsymbol{\theta}\|_1^2 + \lambda_2 \|\boldsymbol{\theta}\|_2^2$

notes

- Sometimes you want the best of both worlds - that's where elastic net comes in.
- It combines L1 and L2 penalties, getting some feature selection from lasso and some stability from ridge.
- This is particularly useful when you have many correlated features - lasso might pick one arbitrarily, while elastic net keeps similar weights for correlated features.
- The downside? Now you have two parameters to tune (λ_1 and λ_2) instead of one.
- In practice, I recommend starting with simple lasso or ridge, and only using elastic net if you have specific reasons to.

A common way to represent regularization graphically

(Damiano's opinion: not as good as the 3D ones in geogebra)



notes

- This figure shows the conceptual difference between L1 and L2 regularization.
- The blue regions represent the constraints imposed by each penalty type. The red ellipses are contours of the original cost function.
- L2 (ridge) gives smooth, circular constraints - solutions tend to have many small non-zero coefficients.
- L1 (lasso) gives diamond-shaped constraints with sharp corners - solutions often lie on the axes, giving exact zeros.
- While this is a classic visualization, I agree with Damiano - the 3D interactive visualizations give better intuition.
- Still, this helps understand why lasso leads to sparsity while ridge doesn't.

Bayesian interpretation

notes

Regularization may be seen with the Bayesian googles

(also this part is not for the exam)

Interesting mathematical objects:

- regularized optimization: $\min_{\theta} J(\theta) + \gamma R(\theta)$
- bayesian MAP estimation: $\max_{\theta} p(\theta|y) \propto p(y|\theta)p(\theta)$

Actually sometimes they coincide!

- L_2 regularization $\Leftrightarrow$ Gaussian prior
- L_1 regularization $\Leftrightarrow$ Laplace prior

notes

- Let me reveal a deep connection that often surprises students - regularization is secretly Bayesian inference in disguise!
- When we add a penalty term to our loss function, it's mathematically equivalent to putting a prior distribution on our parameters.
- That L_2 penalty you've been using in ridge regression? It corresponds to assuming each parameter has a Gaussian prior centered at zero.
- The L_1 penalty in lasso? That comes from a Laplace (double exponential) prior - a distribution with sharp peaks at zero.
- This means every time you've used regularization, you've actually been doing Bayesian statistics without knowing it!

From regularization to MAP estimation

Example for linear regression with L_2 regularization

$$\text{Ridge: } \min_\theta \|y - X\theta\|^2 + \lambda\|\theta\|^2$$

$$\text{MAP: } \max_\theta \mathcal{N}(y|X\theta, \sigma^2 I) \cdot \mathcal{N}(\theta|0, \tau^2 I) \quad \text{with } \lambda = \sigma^2/\tau^2$$

notes

- Let's make this connection precise. Take ridge regression - the standard formulation minimizes SSE plus L2 penalty.
- The Bayesian approach says: maximize the posterior probability, which is likelihood times prior.
- When you work through the math, these become identical if you set $\lambda = \sigma^2/\tau^2$!

From regularization to MAP estimation

Example for linear regression with L_1 regularization

$$\text{Lasso: } \min_\theta \|y - X\theta\|^2 + \lambda\|\theta\|_1$$

$$\text{MAP: } \max_\theta \mathcal{N}(y|X\theta, \sigma^2 I) \cdot \text{Laplace}(\theta|0, b)$$

notes

- The same as above holds for lasso - its penalty corresponds to a Laplace prior on the parameters.
- This gives us a powerful interpretation: the regularization parameter λ encodes our prior belief about how large the parameters should be.
- Smaller λ means weaker prior (parameters can be larger), larger λ means stronger prior (shrink parameters more).

Important implication

> if you have your own prior for your own problem, you should design your regularization term in an ad-hoc way

Very interesting example of this: "stable splines kernels" for system identification

notes

- there is a lot of new literature showing how classical system identification is generally outperformed by new algorithms that are essentially nonparametric estimators of impulse responses on the time domain with a ad-hoc kernel that mimics how impulse responses of BIBO stable system are

Summarizing

> Compare different regularization techniques (ridge, lasso, and elastic net) by evaluating their mathematical formulations, graphical interpretations, and practical implications

> Interpret regularization paths from Lasso regression plots to identify the relative importance of features in predictive modeling

- L2 and L1 regularization techniques place different additional weights to the cost functions
- graphically seeing how they work is essential to fix the understanding behind them
- L1 is especially useful to perform features selection

notes

- you should now be able to do this, following the pseudo-algorithm in the itemized list

Most important python code for this sub-module

notes

Core Scientific Computing

- **NumPy**: Fundamental package for numerical computations
- **SciPy**: Advanced scientific computing (optimization, linear algebra)
- **Matplotlib**: Publication-quality visualization
- **Pandas**: Data manipulation and analysis

notes

- These are the absolute basics we'll use in every session
- NumPy handles all our matrix operations - crucial for regularization math
- SciPy contains special functions we'll need for advanced optimization
- Pro tip: Have students install these via `pip install numpy scipy matplotlib pandas`

Machine Learning Focus

- **scikit-learn**: Main library for LS implementations
  - Ridge/Lasso/ElasticNet implementations
  - Cross-validation tools
  - Regularization path visualization
- **statsmodels**: Formal statistical modeling
- **autograd/JAX**: For advanced gradient computations

notes

- scikit-learn will be our workhorse - it has production-grade implementations
- Show students `Ridge` and `Lasso` classes - we'll use these extensively
- statsmodels is great for showing formal statistical outputs (p-values, etc.)
- autograd/JAX for when we want to implement custom regularized objectives

Specialized Visualization

- **Seaborn**: Enhanced statistical visuals
- **Plotly**: Interactive regularization path plots
- **mpld3**: D3.js integration for matplotlib

notes

- Regularization concepts need good visuals - these libraries help
- Seaborn's `regplot()` is great for showing regularization effects
- Plotly makes those regularization path plots interactive
- mpld3 can help create web-friendly visualizations for your course materials

Teaching-Specific Tools

- **ipywidgets**: Interactive demonstrations
- `sklearn-evaluation`: Enhanced model evaluation
- **alive-progress**: For long computations during demos

notes

- ipywidgets lets us create sliders to show λ effects in real-time
- `sklearn-evaluation` extends scikit-learn's plotting capabilities
- alive-progress shows pretty progress bars during lengthy cross-validation demos
- Consider creating a `requirements-teaching.txt` with these extras

Self-assessment material

notes

Question 1

What is the primary purpose of regularization in statistical learning?

Potential answers:

I: **(wrong)** To increase model complexity and fit training data perfectly
II: **(correct)** To reduce overfitting by trading some bias for lower variance
III: **(wrong)** To eliminate all bias from the model estimates
IV: **(wrong)** To make computations faster by reducing matrix dimensions
V: **(wrong)** I do not know

Solution 1:

The correct answer is **To reduce overfitting by trading some bias for lower variance**. Regularization intentionally introduces some bias to constrain model complexity, which typically reduces variance and improves generalization performance. This is the fundamental bias-variance tradeoff we discussed in the slides.

notes

- see the associated solution(s), if compiled with that ones :)

Question 2

In ridge regression, what Bayesian prior does the L2 penalty term correspond to?

Potential answers:

I: **(wrong)** Uniform prior over all parameters
II: **(wrong)** Laplace (double exponential) prior
III: **(correct)** Gaussian prior centered at zero
IV: **(wrong)** Poisson prior with =1
V: **(wrong)** I do not know

Solution 1:

The correct answer is **Gaussian prior centered at zero**. The L2 penalty in ridge regression is mathematically equivalent to placing independent Gaussian priors on each parameter, with mean zero and variance determined by the regularization strength . This connection was shown in the Bayesian interpretation slides.

notes

- see the associated solution(s), if compiled with that ones :)

Question 3

Why does L1 regularization (lasso) tend to produce sparse solutions with exactly zero coefficients?

Potential answers:

I: **(wrong)** Because it uses a logarithmic penalty term
II: **(correct)** Due to the sharp corners of the L1 constraint region
III: **(wrong)** Because it maximizes the likelihood more aggressively
IV: **(wrong)** It doesn't - this is a common misconception
V: **(wrong)** I do not know

Solution 1:

The correct answer is **Due to the sharp corners of the L1 constraint region**. The geometry of the L1 penalty's diamond-shaped constraint region causes solutions to frequently land exactly on the axes, setting some coefficients to zero. This was visualized in both the lasso slides and the regularization path plot.

notes

- see the associated solution(s), if compiled with that ones :)

Question 4

What surprising result does the James-Stein estimator demonstrate about maximum likelihood estimation?

Potential answers:

I: **(wrong)** LS estimators always have minimum variance
II: **(correct)** LS can be dominated by shrinkage estimators when estimating multiple parameters
III: **(wrong)** LS becomes biased when sample size exceeds 30
IV: **(wrong)** LS requires normally distributed errors
V: **(wrong)** I do not know

Solution 1:

The correct answer is **LS can be dominated by shrinkage estimators when estimating multiple parameters**. The James-Stein estimator shows that when estimating three or more parameters simultaneously, shrinking the LS estimates toward zero can achieve lower overall mean squared error, despite introducing

notes

- see the associated solution(s), if compiled with that ones :)

Question 5

When examining a lasso regularization path plot, how should you interpret features whose coefficients become non-zero earliest as decreases?

Potential answers:

- I: **(wrong)** They are likely measurement errors
- II: **(wrong)** They should be removed from the model
- III: **(correct)** They are the most important predictors
- IV: **(wrong)** They have the smallest scale
- V: **(wrong)** I do not know

Solution 1:

The correct answer is **They are the most important predictors**. In regularization path plots, features that "enter" the model (become non-zero) at larger values are more strongly associated with the response. This makes them effectively the most important predictors, as discussed in the regularization path slide.

notes

- see the associated solution(s), if compiled with that ones :)

Recap of sub-module "Regularization"

- adding regularization and non-L2 costs noticeably extends capabilities of estimators, at the cost though of introducing some hyperparameters that need to be tuned too from the data

notes

- the most important remarks from this sub-module are these ones