CPSC 340:
Machine Learning and Data Mining

Regularization

Summer 2021

In This Lecture

- Regularization Intro (10 minutes)
- L2-regularization (10 minutes)
- L1-regularization (10 minutes)
- Standardization (10 minutes)

When you don't regularize your model

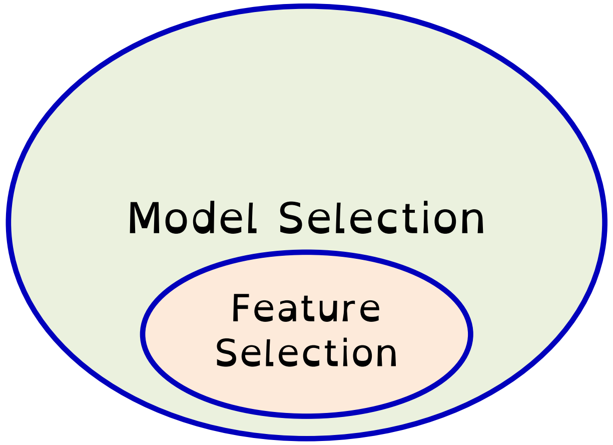
Coming Up Next

REGULARIZATION INTRO

(This is probably the MOST important topic in this course)

“Feature” Selection vs. “Model” Selection?

- Model selection: “which model should I use?”
  - KNN vs. decision tree, depth of decision tree, degree of polynomial basis.
- Feature selection: “which features should I use?”
  - Using feature 10 or not, using x_i^2 as part of basis.



- These two tasks are highly-related:
  - It’s a different “model” if we add x_i^2 to linear regression.
  - But the x_i^2 term is just a “feature” that could be “selected” or not.
  - Usually, “feature selection” means choosing from some “original” features.
    - You could say that “feature” selection is a special case of “model” selection.

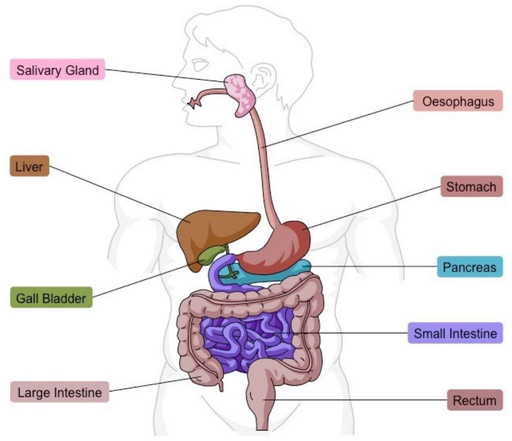
Is It Good to Throw Features Away?

- (Yes/No), because linear regression can overfit with large ‘d’.
  - Even though it’s “just” a hyper-plane.

- Consider using d=n, with completely random features.
  - With high probability, you will be able to get a training error of 0.
  - But the features were random, this is completely overfitting.

- You could view “number of features” as a hyper-parameter.
  - Model gets more complex as you add more features.

Controlling Complexity

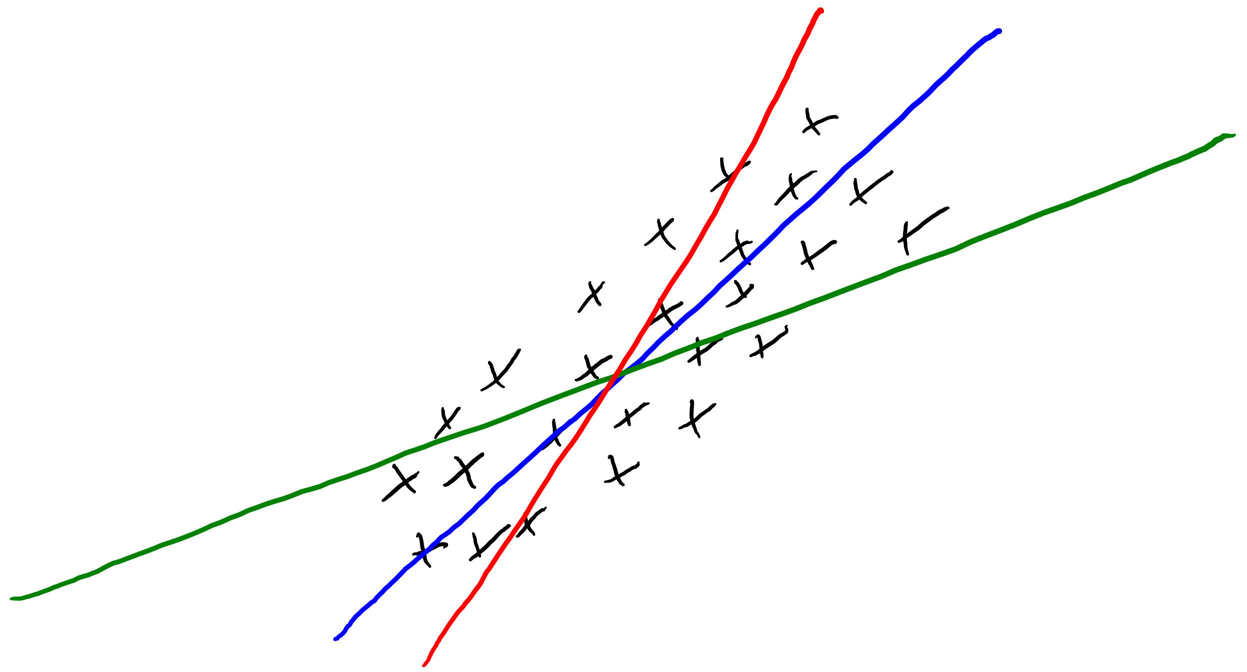


- Usually, "true" mapping from x_i to y_i is complex.
  - Might need high-degree polynomial.
  - Might need to combine many features, and don't know "relevant" ones.
- But complex models can overfit.
- So what do we do???

- Our main tools:
  - Model averaging: average over multiple models to decrease variance.
  - Regularization (today): add a penalty on the complexity of the model.

Would you rather?

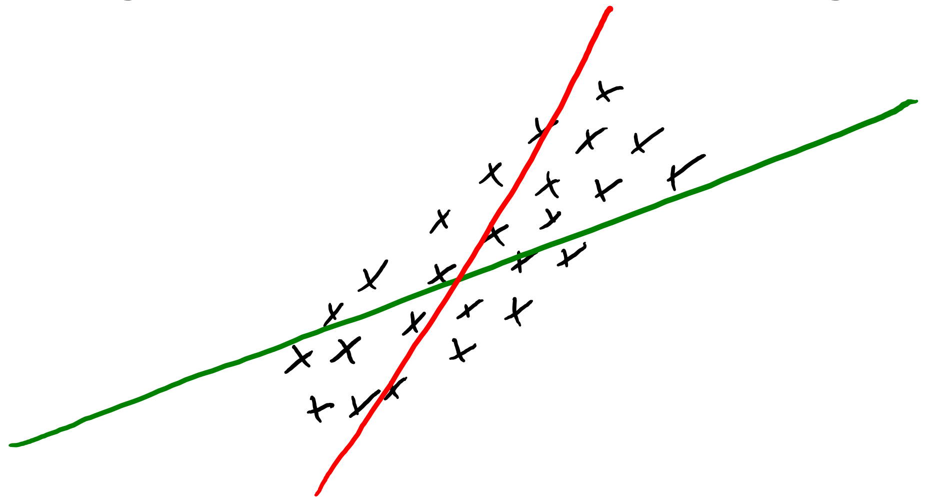
- Consider the following dataset and 3 linear regression models:



Q: Which one is the "best" model??

Would you rather?

- Consider the following dataset and 3 linear regression models:



- What if you are forced to choose between red and green?
  - And assume they have the same training error.
- You should pick green.
  - Since slope is smaller, small change in x_i has a _____________ in prediction y_i.
    - Green line's predictions are (more/less) sensitive to having 'w' exactly right.
  - Since green 'w' is less sensitive to data, test error might be lower.

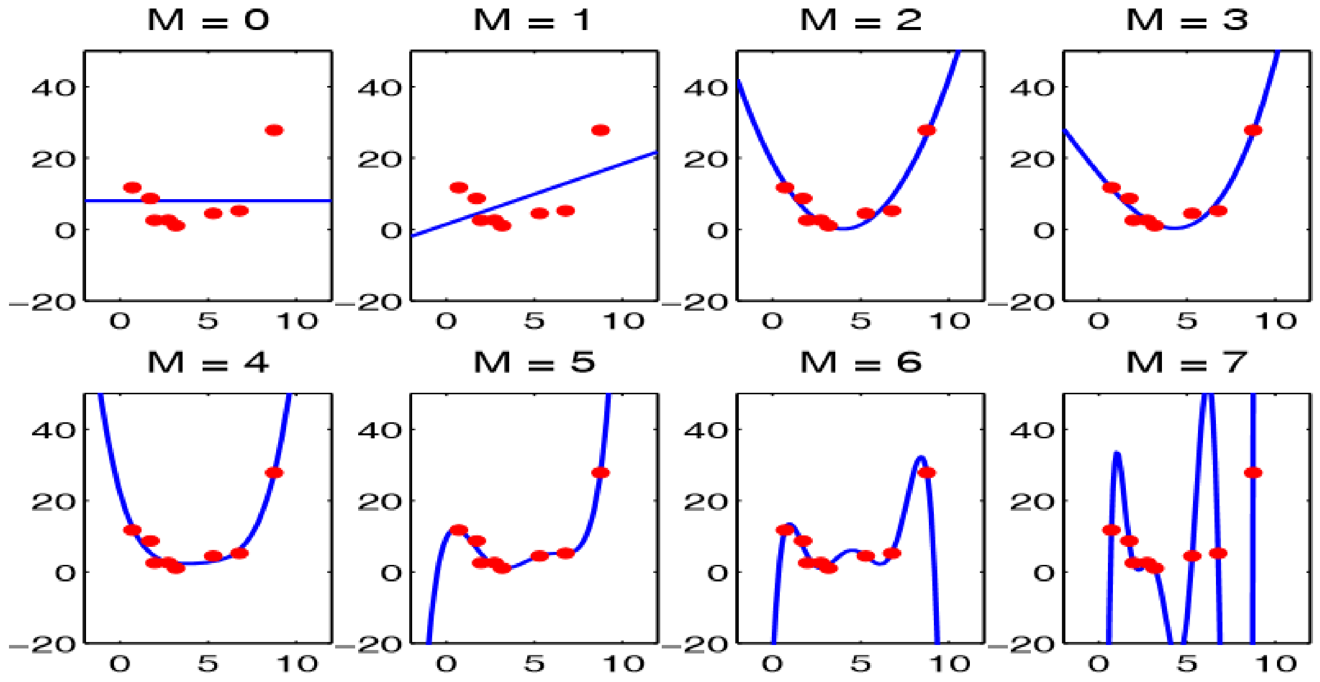
“Regularization”

- “Regularization”: reducing a property of parameters
  - e.g. L2-norm of w, L1-norm of w, number of non-zeros in w, etc.
  - Optimization must take this term into account when minimizing

- Assumption: we can express our goal as minimizing some quantity.
  - for linear models, small norm of w => low complexity

- Naive Bayes with Laplace smoothing:
  an instance of regularization
  - reduce heterogeneity of $p(x_{ij} \mid y_i)$ to control model complexity

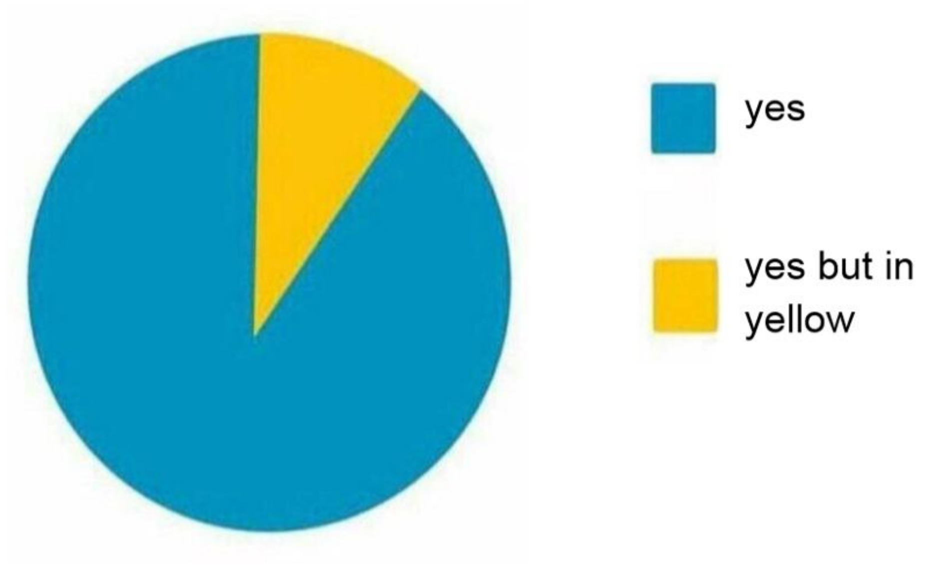
Size of Regression Weights are Overfitting



- The regression weights w_j with degree-7 are huge in this example.
- The degree-7 polynomial would be less sensitive to the data, if we "regularized" the w_j so that they are small.

$$\hat{y}_i = 0.0001(x_i)^7 + 0.03(x_i)^3 + 3 \qquad \text{vs.} \qquad \hat{y}_i = 1000(x_i)^7 - 500(x_i)^6 + 890x_i$$

Should you regularize your model?



Coming Up Next

L2-REGULARIZATION

L2-Regularization

- Standard regularization strategy is L2-regularization:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^Tx_i - y_i)^2 + \frac{\lambda}{2}\sum_{j=1}^{d} w_j^2 \quad \text{or} \quad f(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2$$

- Intuition: large slopes w_j tend to lead to overfitting.

- Objective balances getting low error vs. having small slopes 'w_j'.
  - "You can increase the training error if it makes 'w' much smaller."
  - Nearly-always reduces overfitting.

  - Regularization parameter $\lambda > 0$ controls ____________ of regularization.
    - Large λ puts large penalty on slopes.

L2-Regularization

- Standard regularization strategy is L2-regularization:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^Tx_i - y_i)^2 + \frac{\lambda}{2}\sum_{j=1}^{d} w_j^2 \quad \text{or} \quad f(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2$$

- In terms of fundamental trade-off:
  - Regularization (increases/decreases) training error.
  - Regularization (increases/decreases) approximation error.

- How should you choose λ?
  - Theory: as 'n' grows λ should be in the range O(1) to (√n).
  - Practice: optimize validation error or cross-validation error.
    - This almost always decreases the test error.

Q: Does this mean optimization bias is not a problem anymore?

L2-Regularization “Shrinking” Example

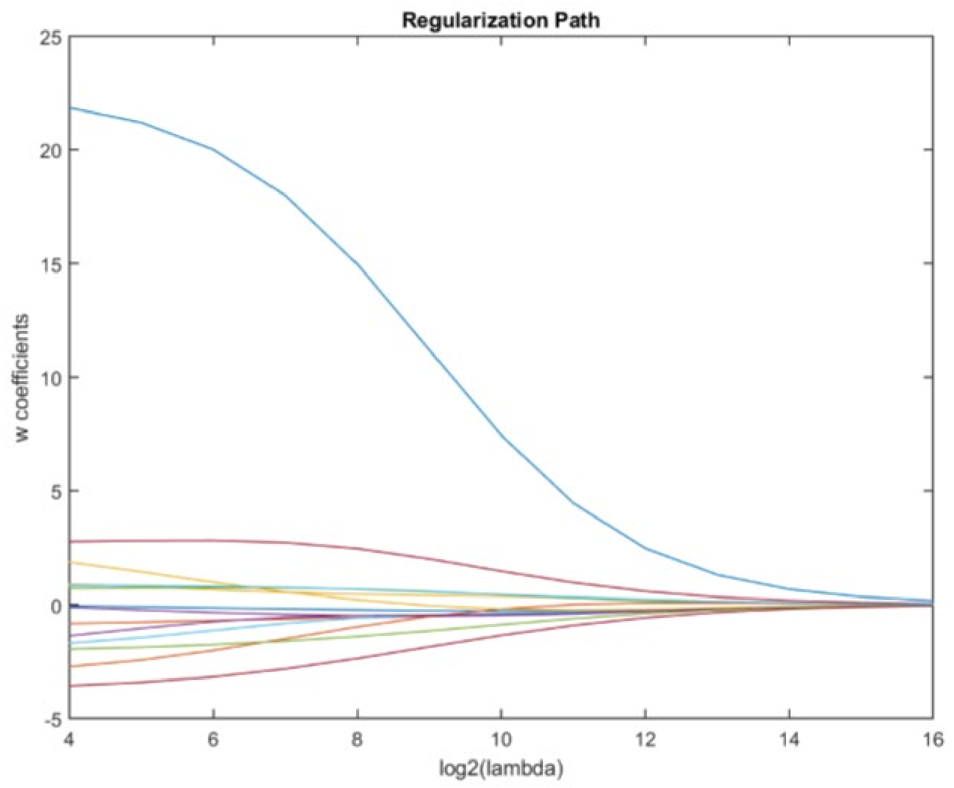
- Solution to a “least squares with L2-regularization” for different λ:

λ	w_1	w_2	w_3	w_4	w_5	$\|\|Xw - y\|\|^2$	$\|\|w\|\|^2$
0	-1.88	1.29	-2.63	1.78	-0.63	285.64	15.68
1	-1.88	1.28	-2.62	1.78	-0.64	285.64	15.62
4	-1.87	1.28	-2.59	1.77	-0.66	285.64	15.43
16	-1.84	1.27	-2.50	1.73	-0.73	285.71	14.76
64	-1.74	1.23	-2.22	1.59	-0.90	286.47	12.77
256	-1.43	1.08	-1.70	1.18	-1.05	292.60	8.60
1024	-0.87	0.73	-1.03	0.57	-0.81	321.29	3.33
4096	-0.35	0.31	-0.42	0.18	-0.36	374.27	0.56

- We get least squares with $\lambda = 0$.
  - But we can achieve similar training error with smaller $||w||$.
- $||Xw - y||$ increases with λ, and $||w||$ decreases with λ.
  - Though individual w_j can increase or decrease with lambda.
  - Because we use the L2-norm, the large ones decrease the most.

Regularization Path

- Regularization path is a plot of the optimal weights 'w_j' as 'λ' varies:



- Starts with least squares with $\lambda = 0$, and w_j converge to 0 as λ grows.

L2-regularized Least Squares Normal Equations

- When using L2-regularized squared error, we can solve for ∇ f(w) = 0.
- Loss before: $f(w) = \frac{1}{2}\|Xw - y\|^2$
- Loss after: $f(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2$
- Gradient before: $\nabla f(w) = X^T X w - X^T y$
- Gradient after: $\nabla f(w) = X^T X w - X^T y + \lambda w$
- Linear system before: $X^T X w = X^T y$
- Linear system after: $(X^T X + \lambda I)w = X^T y$
- But unlike X^TX, the matrix $(X^TX + \lambda I)$ is always invertible:
  - Multiply by its inverse for unique solution: $w = (X^T X + \lambda I)^{-1}(X^T y)$

Gradient Descent for L2-Regularized Least Squares

- The L2-regularized least squares objective and gradient:

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2 \qquad \nabla f(w) = X^T(Xw - y) + \lambda w$$

- Gradient descent iterations for L2-regularized least squares:

$$w^{t+1} = w^t - \alpha^t \left[\underbrace{X^T(Xw^t - y) + \lambda w^t}_{\nabla f(w^t)}\right]$$

- Cost of gradient descent iteration is still O(nd).
  - Can show number of iterations decrease as λ increases (not obvious).

Why use L2-Regularization?

- It's a weird thing to do, but Mark says "always use regularization".
  - "Almost always decreases test error" should already convince you.

- But here are 6 more reasons:
  1. Solution 'w' is unique.
  2. X^TX does not need to be invertible (no collinearity issues).
  3. Less sensitive to changes in X or y.
  4. Gradient descent converges faster (bigger λ means fewer iterations).
  5. Stein's paradox: if $d \geq 3$, 'shrinking' moves us closer to 'true' w.
  6. Worst case: just set λ small and get the same performance.

Regularizing the y-Intercept?

- Should we regularize the y-intercept?
- No! Why encourage it to be closer to zero? (It could be anywhere.)
  - You should be allowed to shift function up/down globally.
- Yes! It makes the solution unique and it easier to compute ‘w’.
- Compromise: regularize by a smaller amount than other features.

$$f(w, w_0) = \frac{1}{2}\|Xw + w_0 - y\|^2 + \frac{\lambda}{2}\|w\|^2 + \frac{\lambda_0}{2}w_0^2$$

Coming Up Next

L1-REGULARIZATION

Lasso, pronounced "la sue"

Previously: Search and Score

- We talked about search and score for feature selection:
  - Define a "score" and "search" for features with the best score.
- Usual scores count the number of non-zeroes ("L0-norm"):

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda \underbrace{\|w\|_0}_{\text{number of non-zeroes in 'w'}}$$

- But it's hard to find the 'w' minimizing this objective.
- We discussed forward selection, but requires fitting O(__) models.

Previously: Search and Score

- What if we want to pick among millions or billions of features?

- If 'd' is large, forward selection is too slow (A4):
  - For least squares, need to fit $O(d^2)$ models at cost of $O(nd^2 + d^3)$.
  - Total cost $O(nd^4 + d^5)$.

- The situation is worse if we aren't using basic least squares:
  - For robust regression, need to run gradient descent $O(d^2)$ times.
  - With regularization, need to search for lambda $O(d^2)$ times.

L1-Regularization

- Instead of L0- or L2-norm, consider regularizing by the L1-norm:

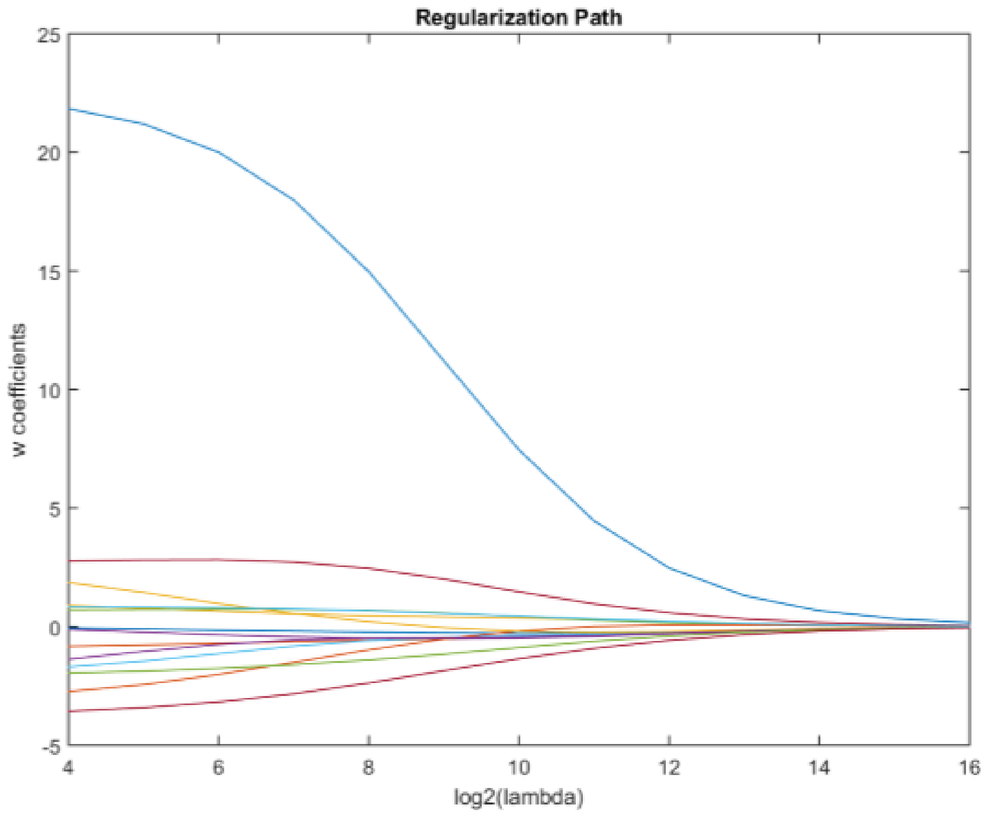
$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_1$$

- Like L2-norm, it's convex and improves our test error.
- Like L0-norm, it encourages elements of 'w' to be exactly zero.

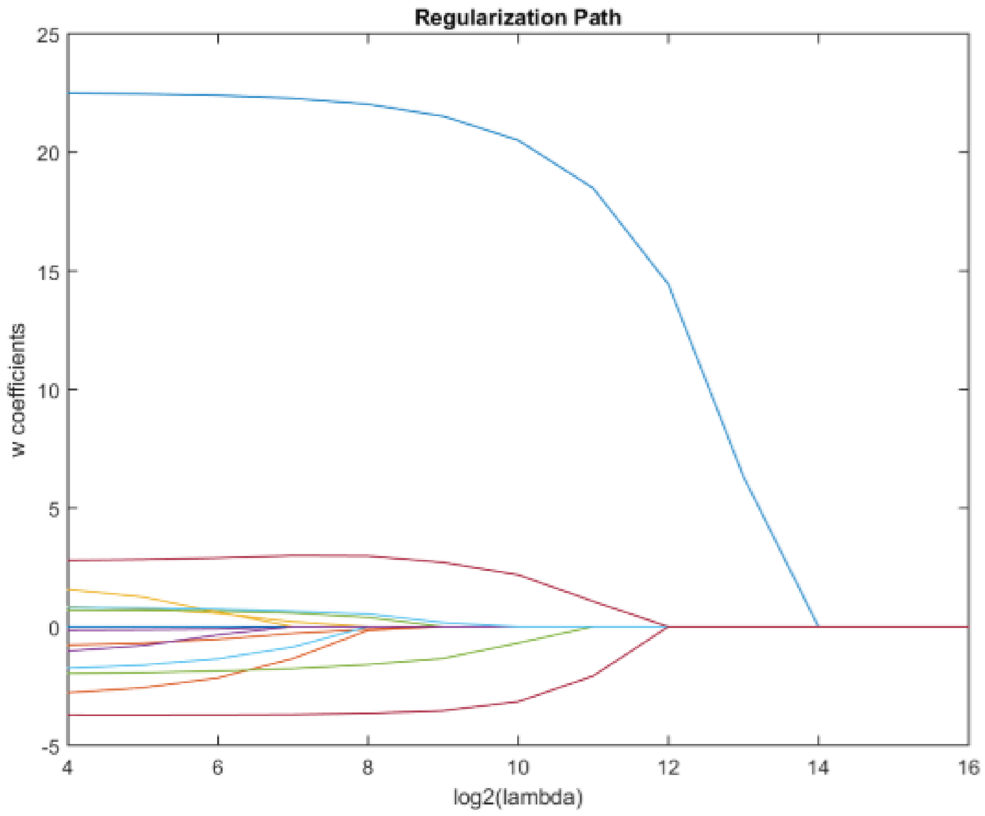
- L1-regularization simultaneously regularizes and selects features.
  - Very fast alternative to search and score.
  - Sometimes called "LASSO" regularization.
    - least absolute shrinkage and selection operator

L2-Regularization vs. L1-Regularization

- Regularization path of w_j values as 'λ' varies:



L2-regularization



L1-regularization

- L1-Regularization sets values to exactly 0 (WHY?)

Regularizers and Sparsity

- L1-regularization gives sparsity but L2-regularization doesn't.
  - But don't they both shrink features towards zero?

- What is the penalty for setting $w_j = 0.00001$?

- L0-regularization: penalty of λ.
  - A constant penalty for any non-zero value.
  - Encourages you to set w_j exactly to zero, but otherwise doesn't care if w_j is small or not.

- L2-regularization: penalty of $(\lambda/2)(0.00001)^2 = 0.0000000005\lambda$.
  - The penalty gets smaller as you get closer to zero.
  - The penalty asymptotically vanishes as w_j approaches 0 (no incentive for "exact" zeroes).

- L1-regularization: penalty of $\lambda|0.00001| = 0.00001\lambda$.
  - The penalty stays is proportional to how far away w_j is from zero.
  - There is still something to be gained from making a tiny value exactly equal to 0.

L2-Regularization vs. L1-Regularization

- L2-Regularization:
  - Insensitive to changes in data.
  - Decreased variance:
    - Lower test error.
  - Closed-form solution.
  - Solution is unique.
  - All 'w_j' tend to be non-zero.
  - Can learn with *linear* number of irrelevant features.
    - E.g., only O(d) relevant features.

- L1-Regularization:
  - Insensitive to changes in data.
  - Decreased variance:
    - Lower test error.
  - Requires iterative solver.
  - Solution is not unique.
  - Many 'w_j' tend to be zero.
  - Can learn with **exponential** number of irrelevant features.
    - E.g., only O(log(d)) relevant features.

Paper on this result by Andrew Ng

L1-Regularization Applications

- Used to give super-resolution in imaging black holes.
  - Sparsity arises in a particular basis.

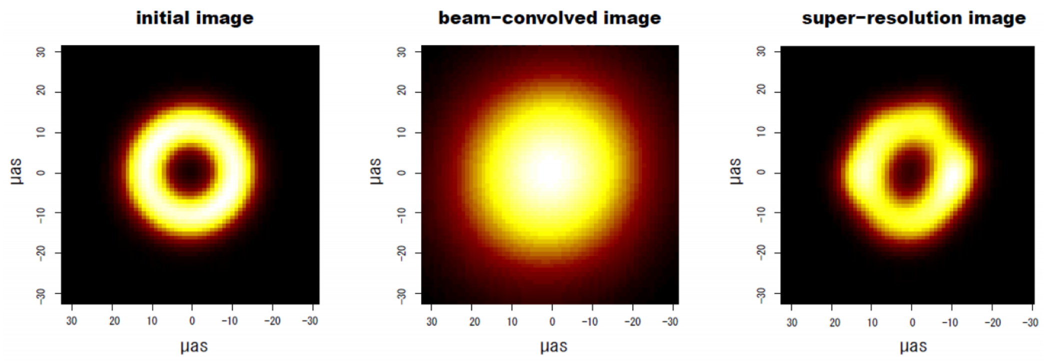


Figure 2. Simulated images of M87. From left to right, the initial model, the image with 0-filling, and the image with LASSO. Improvement of resolution in the LASSO image is significant.

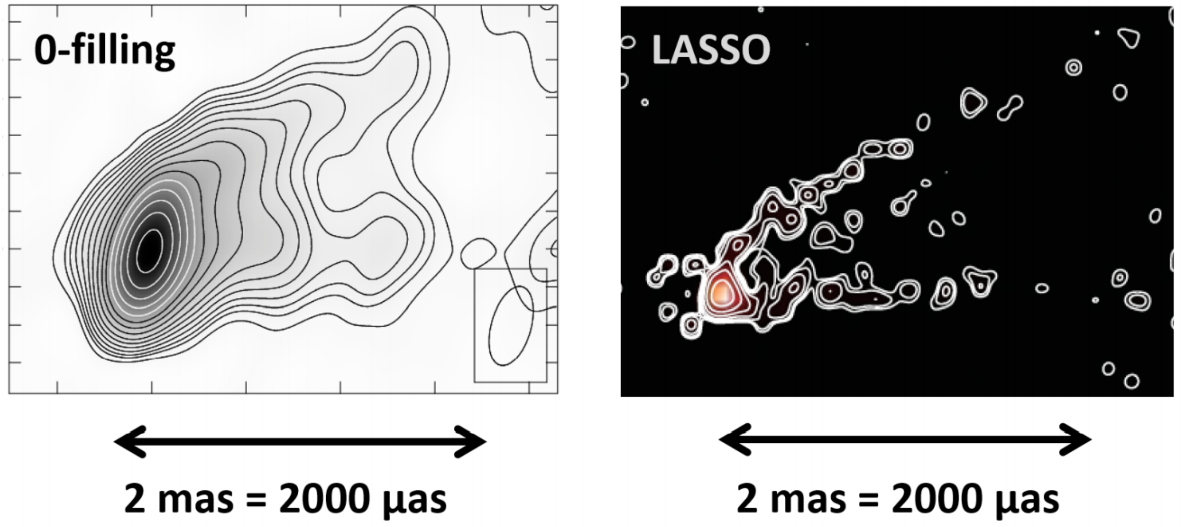


Figure 3. Standard and LASSO images of M87 observed with VLBA at a wavelength of 7 mm. In the two plots, exactly the same data are used. The angular resolution is better in the LASSO image, and the detailed structure of the M87 jet can be traced in more detail.

https://iopscience.iop.org/article/10.1088/1742-6596/699/1/012006/pdf

L1-loss vs. L1-regularization

- Don't confuse the L1 loss with L1-regularization!
  - L1-loss is robust to outlier data points.
    - You can use this instead of removing outliers.
  - L1-regularization is robust to irrelevant features.
    - You can use this instead of removing features.
- And note that you can be robust to outliers and irrelevant features:

$$f(w) = \underbrace{\|Xw - y\|_1}_{L_1-\text{loss}} + \underbrace{\lambda\|w\|_1}_{L_1-\text{regularizer}}$$

- Can we smooth and use "Huber regularization"?
  - Huber regularizer is still robust to irrelevant features.
  - But it's the non-smoothness that sets weights to exactly 0.

L*-Regularization

- L0-regularization (AIC, BIC, Mallow's Cp, Adjusted R^2, ANOVA):
  - Adds penalty on the number of non-zeros to select features.

$$f(w) = \|Xw - y\|^2 + \lambda \|w\|_0$$

- L2-regularization (ridge regression):
  - Adding penalty on the L2-norm of 'w' to decrease overfitting:

$$f(w) = \|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2$$

- L1-regularization (LASSO):
  - Adding penalty on the L1-norm decreases overfitting and selects features:

$$f(w) = \|Xw - y\|^2 + \lambda \|w\|_1$$

L0- vs. L1- vs. L2-Regularization

	Sparse 'w' (Selects Features)	Speed	Unique 'w'	Coding Effort	Irrelevant Features
L0-Regularization	Yes	Slow	No	Few lines	Not Sensitive
L1-Regularization	Yes*	Fast*	No	1 line*	Not Sensitive
L2-Regularization	No	Fast	Yes	1 line	A bit sensitive

- L1-Regularization isn't as sparse as L0-regularization.
  - L1-regularization tends to give more false positives (selects too many).
  - And it's only "fast" and "1 line" with specialized solvers (optimizers.py).
- Cost of L2-regularized least squares is $O(nd^2 + d^3)$.
  - Changes to O(ndt) for 't' iterations of gradient descent (same for L1).
- "Elastic net" (L1- and L2-regularization) is sparse, fast, and unique.
- Using L0+L2 does not give a unique solution.

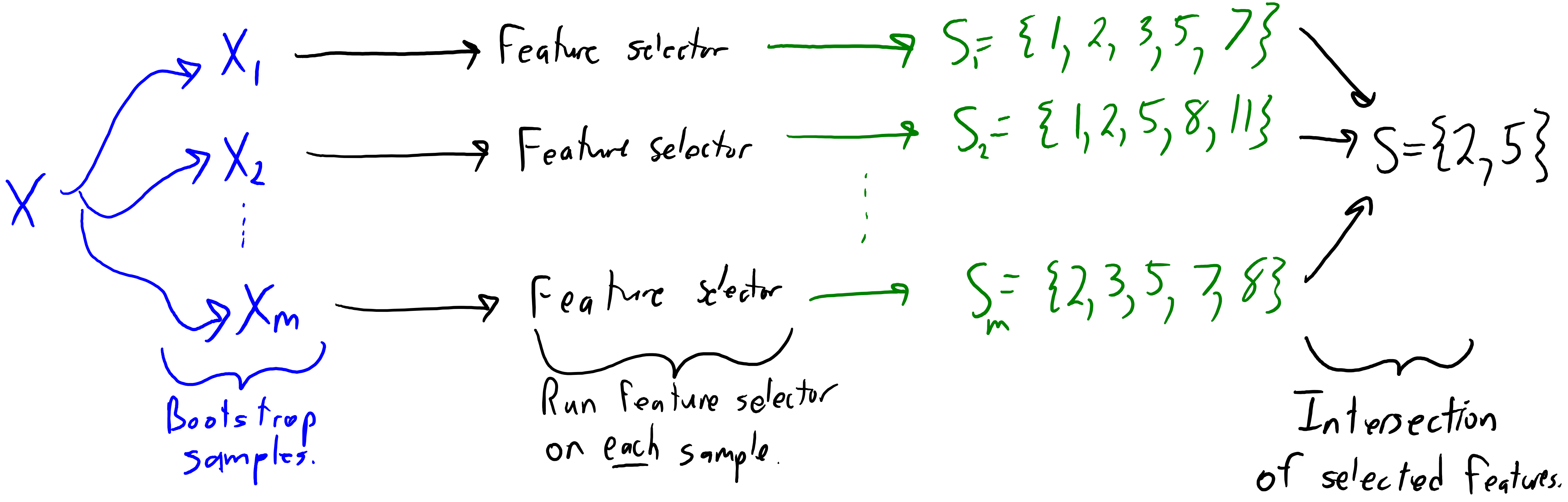
Ensemble Feature Selection

- We can also use ensemble methods for feature selection.
  - Usually designed to reduce false positives or reduce false negatives.
    - False positive: irrelevant feature is selected
    - False negative: relevant feature is excluded

- In this case of L1-regularization, we want to reduce false positives.
  - Unlike L0-regularization, continuous tension between performance/selection
    - "Irrelevant" features can be included before "relevant" w_j reach best value.

- A bootstrap approach to reducing false positives:
  - Apply the method to bootstrap samples of the training data.
  - Only take the features selected in all bootstrap samples.

Ensemble Feature Selection


X
X_1
X_2
X_m
Bootstrap samples.
Feature selector
Feature selector
Feature selector
Run feature selector on each sample.
$S_1 = \{1, 2, 3, 5, 7\}$
$S_2 = \{1, 2, 5, 8, 11\}$
$S_m = \{2, 3, 5, 7, 8\}$
$S = \{2, 5\}$
Intersection of selected features.

- Example: bootstrapping plus L1-regularization ("BoLASSO").
  - Reduces false positives.
  - It's possible to show it recovers "correct" features with weaker conditions.
    - Can replace "intersection" with "selected frequency" if has false negatives too.

Coming Up Next

STANDARDIZATION

Features with Different Scales

- Consider continuous features with different scales:

Egg (#)	Milk (mL)	Fish (g)	Pasta (cups)
0	250	0	1
1	250	200	1
0	0	0	0.5
2	250	150	0

- Should we convert to some standard 'unit'?
  – It doesn't matter for decision trees or naïve Bayes.
    - They only look at one feature at a time.
  – It doesn't matter for least squares:
    - w_j*(100 mL) gives the same model as w_j*(0.1 L) with a different w_j.

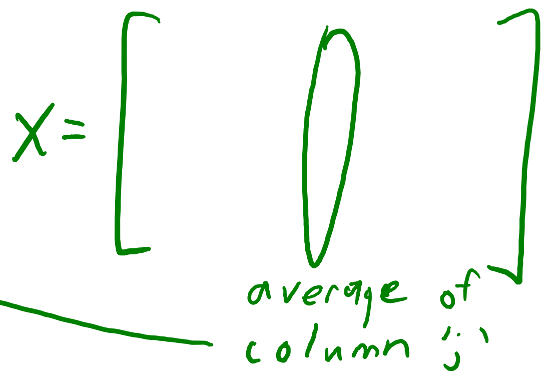
Features with Different Scales

- Consider continuous features with different scales:

Egg (#)	Milk (mL)	Fish (g)	Pasta (cups)
0	250	0	1
1	250	200	1
0	0	0	0.5
2	250	150	0

- Should we convert to some standard 'unit'?
  - It matters for k-nearest neighbours:
    - "Distance" will be affected more by large features than small features.
  - It matters for regularized least squares:
    - Penalizing $(w_j)^2$ means different things if features 'j' are on different scales.

Standardizing Features



- It is common to standardize continuous features:
  - For each feature:
    1. Compute mean and standard deviation:

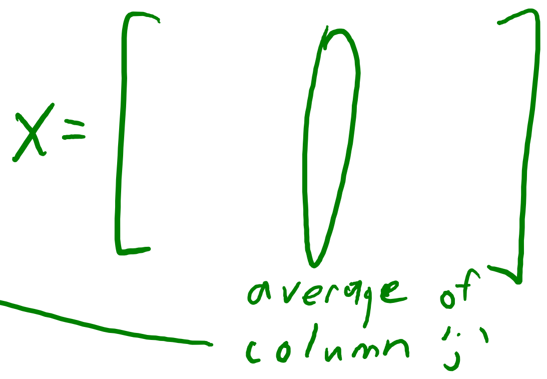
$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij} \qquad \sigma_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\mu_j)^2}$$

    2. Subtract mean and divide by standard deviation ("z-score")

Replace x_{ij} with $\frac{x_{ij}-\mu_j}{\sigma_j}$

  - Now changes in 'w_j' have similar effect for any feature 'j'.
- How should we standardize test data?
  - Wrong approach: use mean and standard deviation of test data.
  - Training and test mean and standard deviation might be very different.
  - Right approach: use mean and standard deviation of training data.

Standardizing Features



- It is common to standardize continuous features:
  - For each feature:
    1. Compute mean and standard deviation:

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij} \qquad \sigma_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\mu_j)^2}$$

    2. Subtract mean and divide by standard deviation ("z-score")

Replace x_{ij} with $\frac{x_{ij}-\mu_j}{\sigma_j}$

  - Now changes in 'w_j' have similar effect for any feature 'j'.
- If we're doing 10-fold cross-validation:
  - Compute μ_j and σ_j based on the 9 training folds (e.g., average over 9/10s of data).
  - Standardize the remaining ("validation") fold with this "training" μ_j and σ_j.
  - Re-standardize for different folds.

Standardizing Target

- In regression, we sometimes standardize the targets y_i.
  - Puts targets on the same standard scale as standardized features:

Replace y_i with $\frac{y_i - \mu_y}{\sigma_y}$

- With standardized target, setting w = 0 predicts average y_i:
  - High regularization makes us predict closer to the average value.
- Again, make sure you standardize test data with the training stats.
- Other common transformations of y_i are logarithm/exponent:

Use $\log(y_i)$ or $\exp(\gamma y_i)$

  - Makes sense for geometric/exponential processes.

Summary

- Regularization:
  - Adding a penalty on model complexity.
- L2-regularization: penalty on L2-norm of regression weights ‘w’.
  - Almost always improves test error.
- L1-regularization: penalty on L1-norm of regression weights ‘w’.
  - Simultaneous regularization and feature selection.
  - Robust to having lots of irrelevant features.
- Feature standardization:
  - Change the unit of every feature into “z-score”

- Next time: non-parametric feature transform and linear classifiers

Review Questions

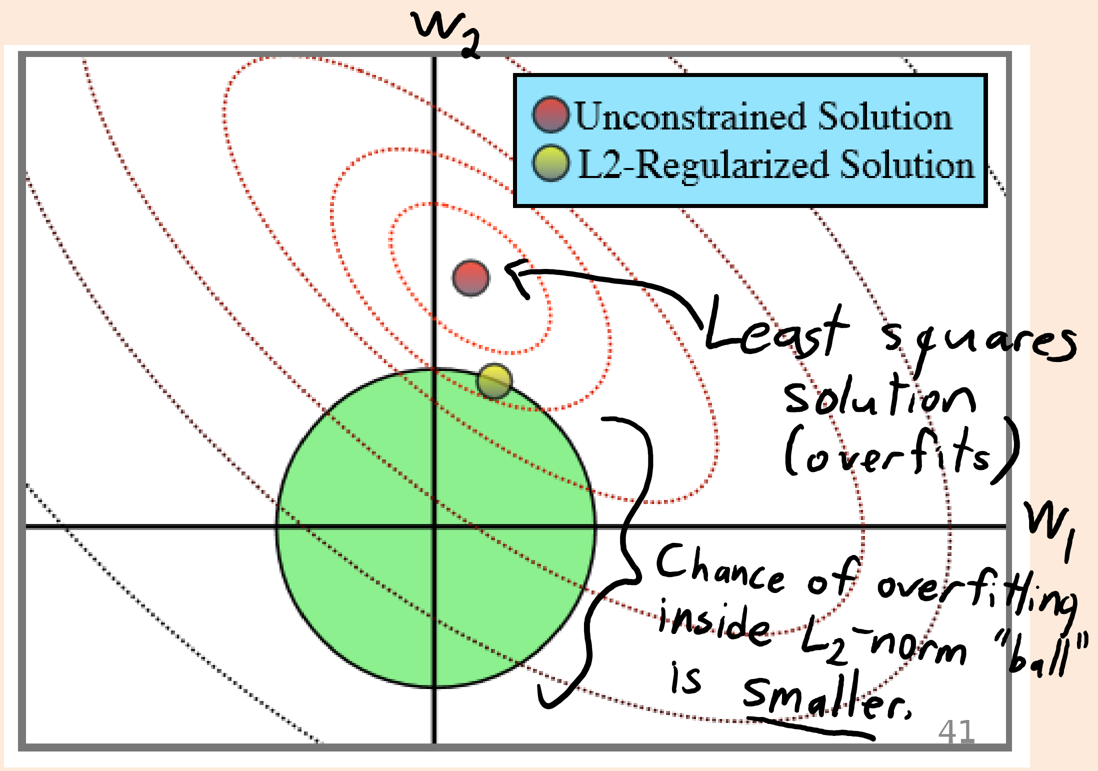
- Q1: In what ways can standardizing the features help reduce a linear model's complexity?
- Q2: Why is L1-regularization able to perform feature selection while L2-regularization cannot?
- Q3: Why are we allowed to use $(X^{\top}X + \lambda I)^{-1}$ in the solution to L2-regularized least squares?
- Q4: What happens to colinear features when L1-regularization is used?
- Q5: What parameters are we "learning" for standardization?

L2-Regularization

- Standard regularization strategy is L2-regularization:

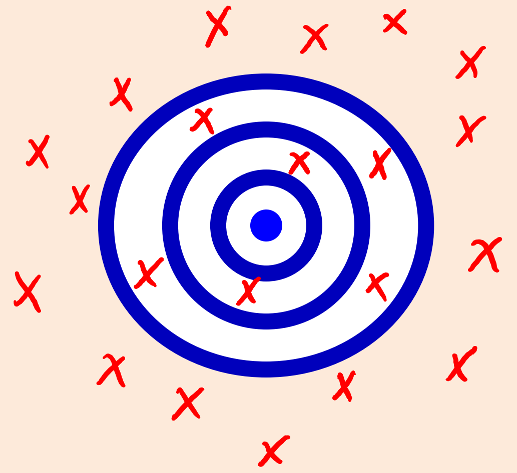
$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^Tx_i - y_i)^2 + \frac{\lambda}{2}\sum_{j=1}^{d} w_j^2 \quad \text{or} \quad f(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2$$

- Equivalent to minimizing squared error but keeping L2-norm small.


w_2
w_1
Unconstrained Solution
L2-Regularized Solution
Least squares solution (overfits)
Chance of overfitting inside L_2-norm "ball" is smaller.

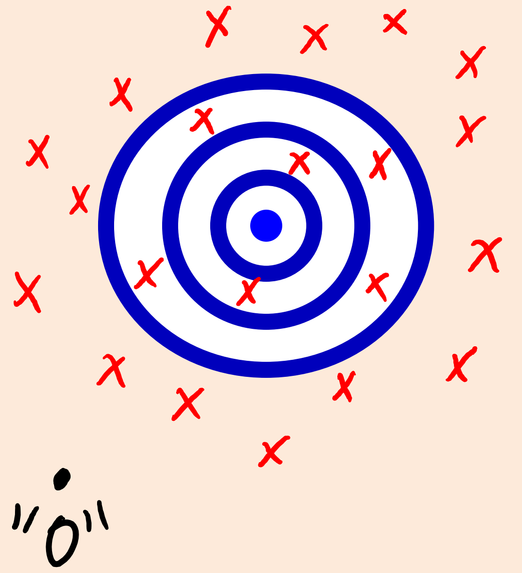
Regularization/Shrinking Paradox

- We throw darts at a target:
  - Assume we don't always hit the exact center.
  - Assume the darts follow a symmetric pattern around center.



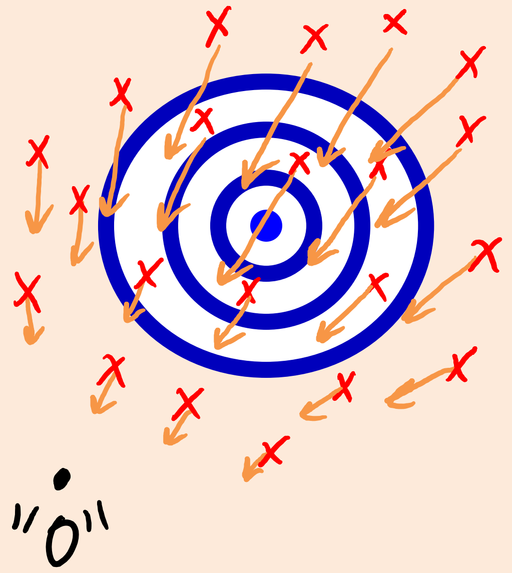
Regularization/Shrinking Paradox

- We throw darts at a target:
  - Assume we don't always hit the exact center.
  - Assume the darts follow a symmetric pattern around center.
- Shrinkage of the darts :
  1. Choose some arbitrary location '0'.
  2. Measure distances from darts to '0'.



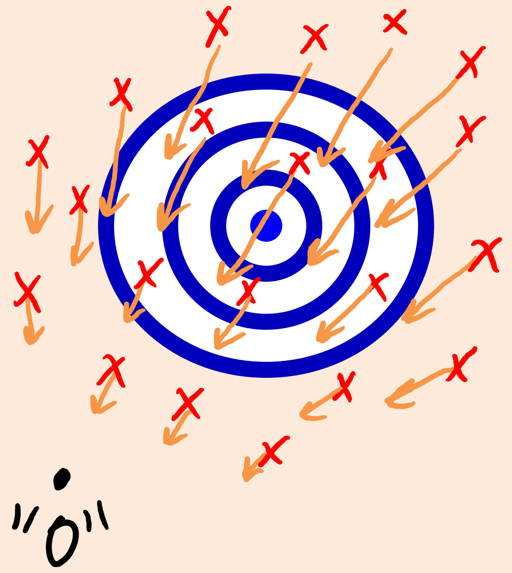
Regularization/Shrinking Paradox

- We throw darts at a target:
  - Assume we don't always hit the exact center.
  - Assume the darts follow a symmetric pattern around center.
- Shrinkage of the darts :
  1. Choose some arbitrary location '0'.
  2. Measure distances from darts to '0'.
  3. Move misses towards '0', by *small* amount proportional to distance from 0.
- If small enough, darts will be closer to center on average.



Regularization/Shrinking Paradox

- We throw darts at a target:
  - Assume we don't always hit the exact center.
  - Assume the darts follow a symmetric pattern around center.
- Shrinkage of the darts :
  1. Choose some arbitrary location '0'.
  2. Measure distances from darts to '0'.
  3. Move misses towards '0', by *small* amount proportional to distance from 0.
- If small enough, darts will be closer to center on average.



Visualization of the related higher-dimensional paradox that the mean of data coming from a Gaussian is not the best estimate of the mean of the Gaussian in 3-dimensions or higher: https://www.naftaliharris.com/blog/steinviz