

Overview

Regularization

Introduction to Machine Learning

Regularization Introduction



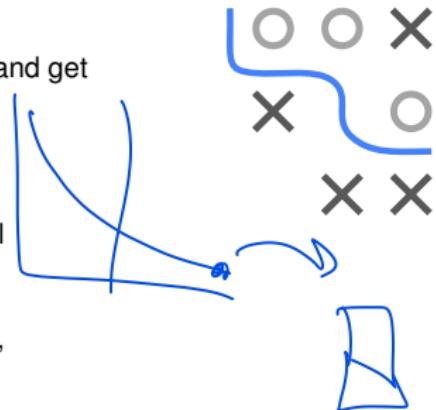
Learning goals

- Overfitting
- Motivation of regularization
- First overview of techniques
- Pattern of regularized ERM formula

WHAT IS REGULARIZATION?

Methods that add **inductive bias** to model, usually some “low complexity” priors (shrinkage and sparsity) to reduce overfitting and get better bias-variance tradeoff

- **Explicit regularization:** penalize explicit measure of model complexity in ERM (e.g., $L1/L2$)
- **Implicit regularization:** early stopping, data augmentation, parameter sharing, dropout or ensembling
- **Structured regularization:** structural prior knowledge over groups of parameters or subnetworks (e.g., group lasso → Yuan and Lin 2006)



$$f(x) =$$

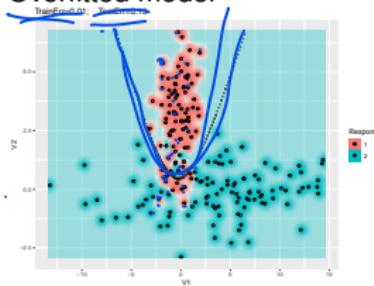
$$\theta$$

RECAP: OVERFITTING

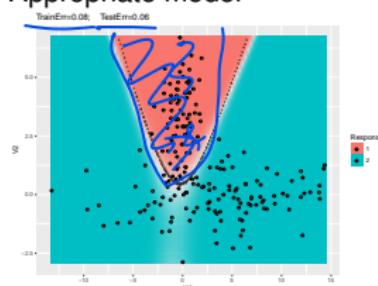
- Occurs when model reflects noise or artifacts in training data
- Model often then does not generalize well (small train error, high test error) – or at least works better on train than on test data



Overfitted model

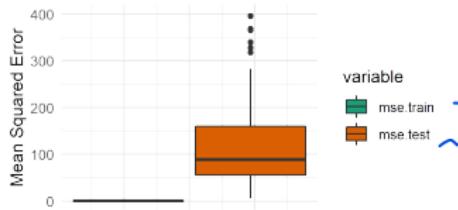
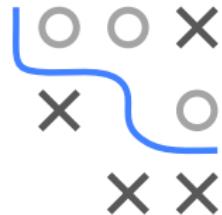


Appropriate model



EXAMPLE I: OVERFITTING

- Data set: daily maximum **ozone level** in LA; $n = 50$
- 12 features: time (weekday, month); weather (temperature at stations, humidity, wind speed); pressure gradient
- Orig. data was subsetted, so it feels “high-dim.” now (low n in relation to p)
- LM with all features (L2 loss)
- MSE evaluation under 10×10 REP-CV



Model fits train data well, but generalizes poorly.

EXAMPLE II: OVERFITTING

- We train an MLP and a CART on the mtcars data
- Both models are not regularized
- And configured to make overfitting more likely



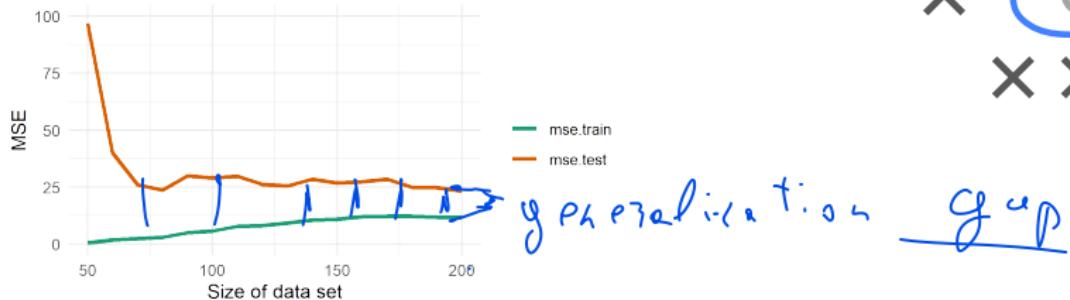
	Train MSE	Test MSE
Neural Network	3.68	19.98
CART	0.00	10.21

T_{Tree}

(And we now switch back to the Ozone example...)

AVOIDING OVERFITTING – COLLECT MORE DATA

We explore our results for increased dataset size.



Fit slightly worsens, but test error decreases.

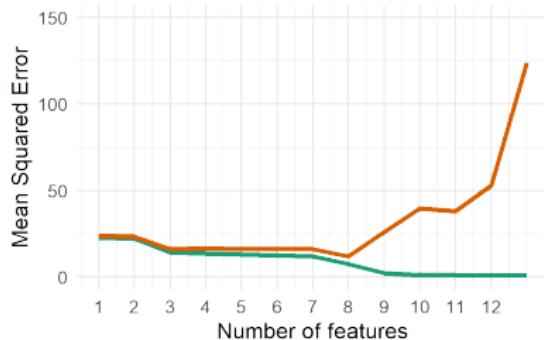
But: Often not feasible in practice.



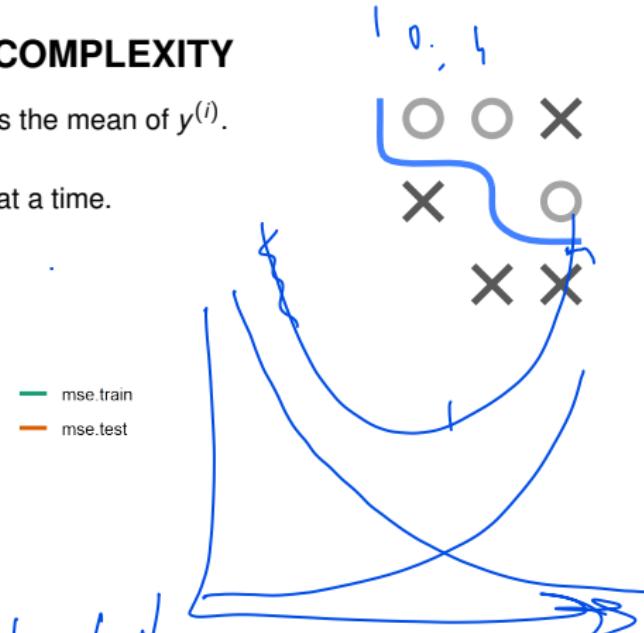
AVOIDING OVERFITTING – REDUCE COMPLEXITY

We try the simplest model: a constant. So for $L2$ loss the mean of $y^{(i)}$.

We then increase complexity by adding one feature at a time.



NB: We added features in a specific (clever) order, so we cheated a bit.

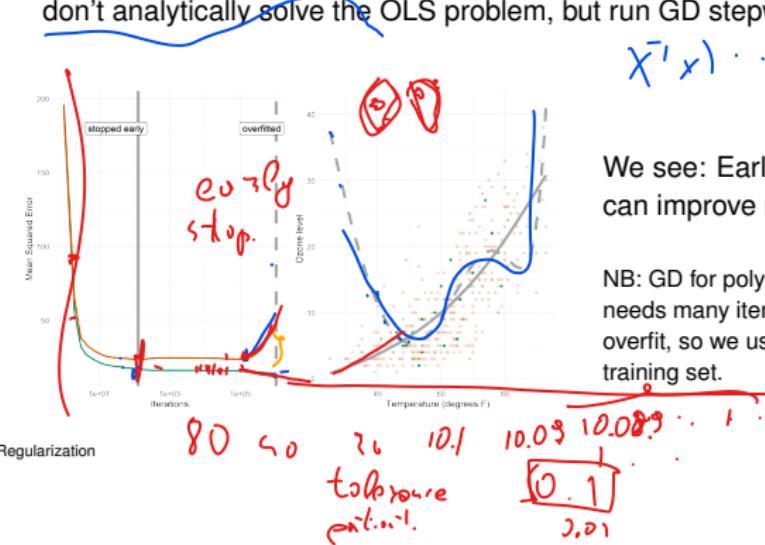


AVOIDING OVERFITTING – OPTIMIZE LESS

Now: polynomial regression with temperature as single feature

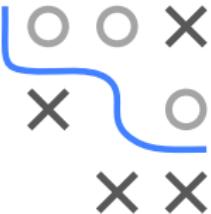
$$f(\mathbf{x} | \boldsymbol{\theta}) = \sum_{k=0}^d \theta_k \cdot (x_T)^k$$

We set $d = 15$ to overfit to small data. To investigate early stopping, we don't analytically solve the OLS problem, but run GD stepwise.



We see: Early stopping GD can improve results.

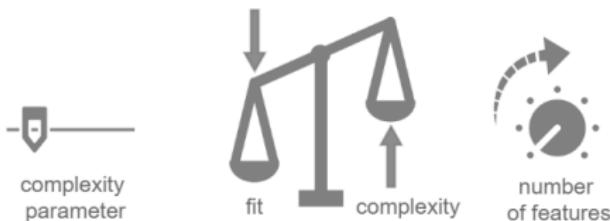
NB: GD for poly-regr usually needs many iters before it starts to overfit, so we used a very small training set.



REGULARIZED EMPIRICAL RISK MINIMIZATION

We have contradictory goals:

- **maximizing fit** (minimizing the train loss)
- **minimizing complexity** of the model



We saw how we can include features in a binary fashion.
But we would rather control complexity **on a continuum**.

REGULARIZED EMPIRICAL RISK MINIMIZATION

Common pattern:

$$\mathcal{R}_{\text{reg}}(f) = \mathcal{R}_{\text{emp}}(f) + \lambda \cdot J(f) = \sum_{i=1}^n L\left(y^{(i)}, f\left(\mathbf{x}^{(i)}\right)\right) + \lambda \cdot J(f)$$

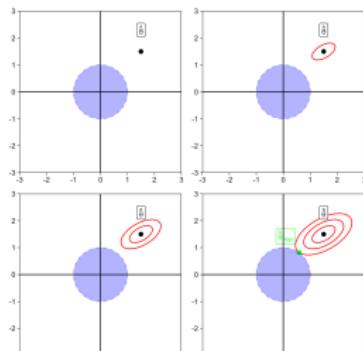


- $J(f)$: **complexity penalty, roughness penalty or regularizer**
- $\lambda \geq 0$: **complexity control** parameter
- The higher λ , the more we penalize complexity
- $\lambda = 0$: We just do simple ERM; $\lambda \rightarrow \infty$: we don't care about loss, models become as "simple" as possible
- λ is hard to set manually and is usually selected via CV
- As for \mathcal{R}_{emp} , \mathcal{R}_{reg} and J are often defined in terms of θ :

$$\mathcal{R}_{\text{reg}}(\theta) = \mathcal{R}_{\text{emp}}(\theta) + \lambda \cdot J(\theta)$$

Introduction to Machine Learning

Regularization Ridge Regression



Learning goals

- Regularized linear model
- Ridge regression / L2 penalty
- Understand parameter shrinkage
- Understand correspondence to constrained optimization

REGULARIZATION IN LM

- Can also overfit if p large and n small(er)
- OLS estimator requires full-rank design matrix
- For highly correlated features, OLS becomes sensitive to random errors in response, results in large variance in fit
- We now add a complexity penalty to the loss:

$$\mathcal{R}_{\text{reg}}(\theta) = \sum_{i=1}^n \left(y^{(i)} - \theta^\top \mathbf{x}^{(i)} \right)^2 + \lambda \cdot J(\theta).$$



RIDGE REGRESSION / L2 PENALTY

Intuitive measure of model complexity is deviation from 0-origin; coeffs then have no or a weak effect. So we measure $J(\theta)$ through a vector norm, shrinking coeffs closer to 0.



$$\begin{aligned}\hat{\theta}_{\text{ridge}} &= \arg \min_{\theta} \sum_{i=1}^n \left(y^{(i)} - \theta^T \mathbf{x}^{(i)} \right)^2 + \lambda \sum_{j=1}^p \theta_j^2 \\ &= \arg \min_{\theta} \|\mathbf{y} - \mathbf{X}\theta\|_2^2 + \lambda \|\theta\|_2^2\end{aligned}$$

Can still analytically solve this:

$$\hat{\theta}_{\text{ridge}} = (\mathbf{X}^T \mathbf{X} + \lambda I)^{-1} \mathbf{X}^T \mathbf{y}$$

Name: We add pos. entries along the diagonal "ridge" of $\mathbf{X}^T \mathbf{X}$

$$\frac{1}{100} (2x^2 + 4x + 7) \rightarrow \frac{1}{100} [100 + (2^2 + 4^2 + 7^2)] = 150$$

$$\frac{1}{100} (10x^2 + 4x^2 + 2x^2 + 3x + 6) \rightarrow \frac{1}{100} [0.1(10 + 10 + 4 + 2) + 6] \approx 150$$

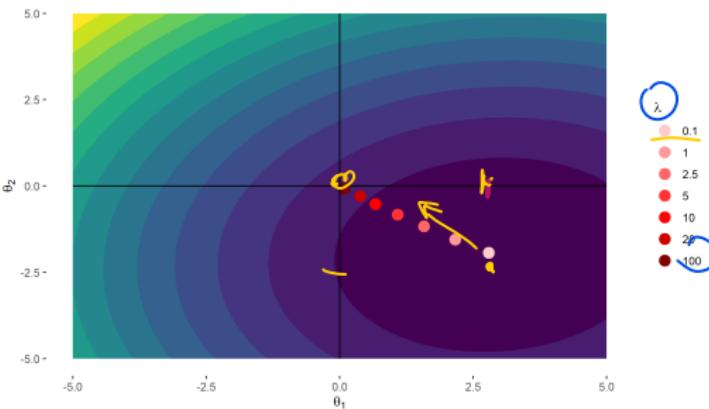
$$2.10.0x + 7 + 600$$

RIDGE REGRESSION / L2 PENALTY

Let $y = 3x_1 - 2x_2 + \epsilon$, $\epsilon \sim N(0, 1)$. The true minimizer is $\theta^* = (3, -2)^T$, with $\hat{\theta}_{\text{ridge}} = \arg \min_{\theta} \|y - X\theta\|^2 + \lambda \|\theta\|^2$.



Effect of L2 Regularization on Linear Model Solutions

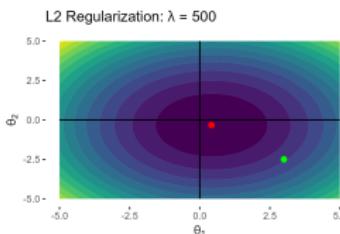
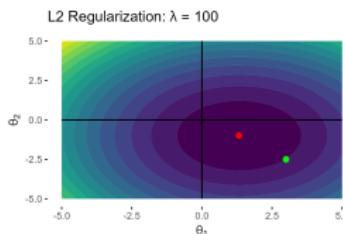
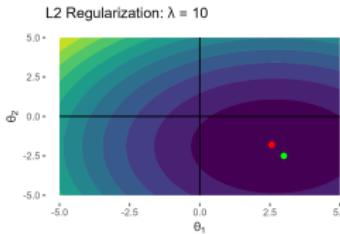
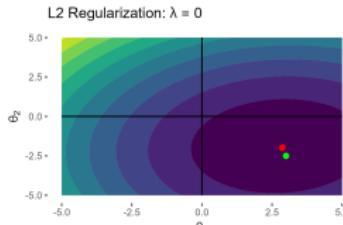


With increasing regularization, $\hat{\theta}_{\text{ridge}}$ is pulled back to the origin
(contour lines show unregularized objective).

RIDGE REGRESSION / L2 PENALTY

Contours of regularized objective for different λ values.

$$\hat{\theta}_{\text{ridge}} = \arg \min_{\theta} \|\mathbf{y} - \mathbf{X}\theta\|^2 + \lambda \|\theta\|^2.$$



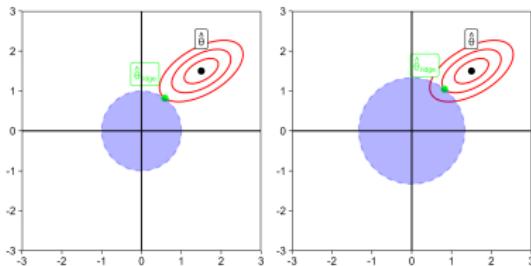
Green = true coeffs of the DGP and red = ridge solution.



RIDGE REGRESSION / L2 PENALTY

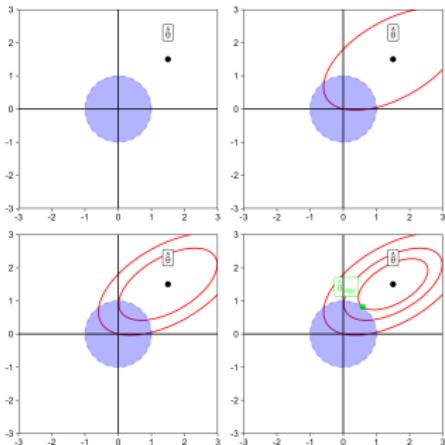
We understand the geometry of these 2 mixed components in our regularized risk objective much better, if we formulate the optimization as a constrained problem (see this as Lagrange multipliers in reverse).

$$\begin{aligned} \min_{\theta} \quad & \sum_{i=1}^n \left(y^{(i)} - f(\mathbf{x}^{(i)} | \theta) \right)^2 \\ \text{s.t.} \quad & \|\theta\|_2^2 \leq t \end{aligned}$$

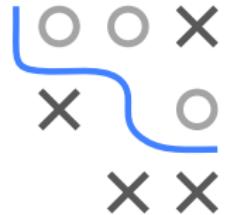


NB: There is a bijective relationship between λ and t : $\lambda \uparrow \Rightarrow t \downarrow$ and vice versa.

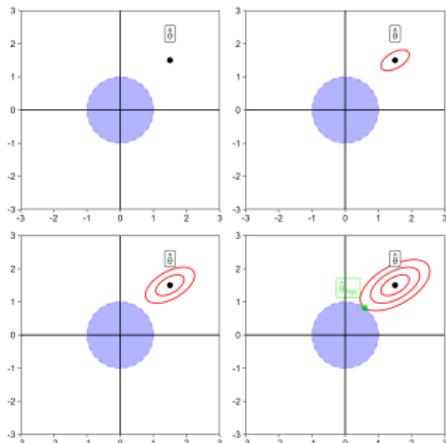
RIDGE REGRESSION / L2 PENALTY



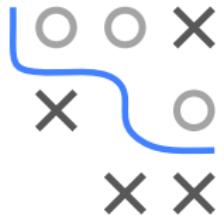
- Inside constraints perspective: From origin, jump from contour line to contour line (better) until you become infeasible, stop before.
- We still optimize the $\mathcal{R}_{\text{emp}}(\theta)$, but cannot leave a ball around the origin.
- $\mathcal{R}_{\text{emp}}(\theta)$ grows monotonically if we move away from $\hat{\theta}$ (elliptic contours).
- Solution path moves from origin to border of feasible region with minimal L_2 distance.



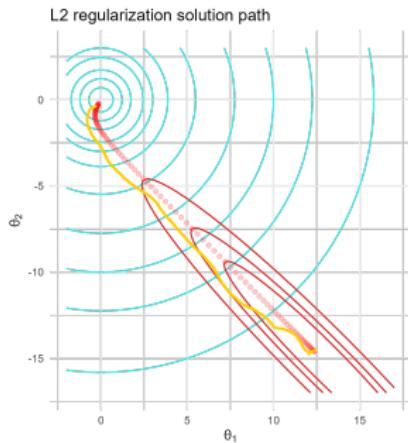
RIDGE REGRESSION / L2 PENALTY



- Outside constraints perspective:
From $\hat{\theta}$, jump from contour line to contour line (worse) until you become feasible, stop then.
- So our new optimum will lie on the boundary of that ball.
- Solution path moves from unregularized estimate to feasible region of regularized objective with minimal L_2 distance.



RIDGE REGRESSION / L2 PENALTY



- Here we can see entire solution path for ridge regression
- Cyan contours indicate feasible regions induced by different λ s
- Red contour lines indicate different levels of the unreg. objective
- Ridge solution (red points) gets pulled toward origin for increasing λ



EXAMPLE: POLYNOMIAL RIDGE REGRESSION

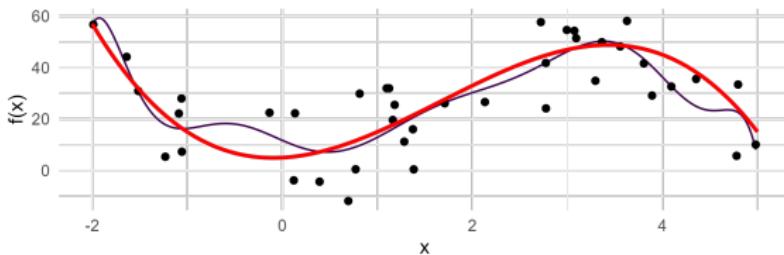
Consider $y = f(x) + \epsilon$ where the true (unknown) function is
 $f(x) = 5 + 2x + 10x^2 - 2x^3$ (in red).

Let's use a d th-order polynomial

$$f(x) = \theta_0 + \theta_1 x + \cdots + \theta_d x^d = \sum_{j=0}^d \theta_j x^j.$$

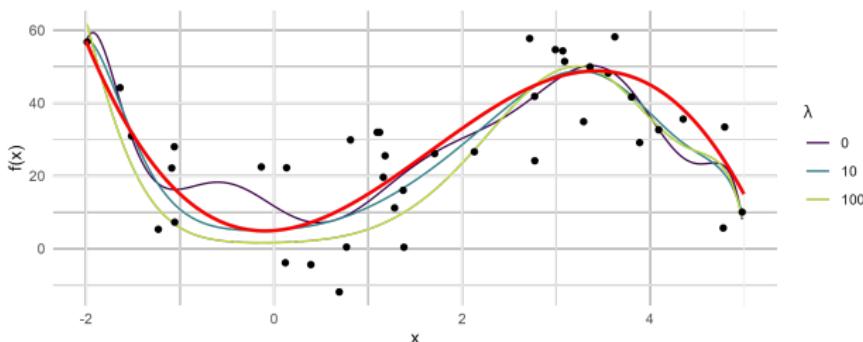


Using model complexity $d = 10$ overfits:



EXAMPLE: POLYNOMIAL RIDGE REGRESSION

With an L^2 penalty we can now select d "too large" but regularize our model by shrinking its coefficients. Otherwise we have to optimize over the discrete d .

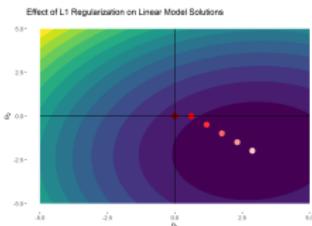


λ	θ_0	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}
0.00	12.00	-16.00	4.80	23.00	-5.40	-9.30	4.20	0.53	-0.63	0.13	-0.01
10.00	5.20	1.30	3.70	0.69	1.90	-2.00	0.47	0.20	-0.14	0.03	-0.00
100.00	1.70	0.46	1.80	0.25	1.80	-0.94	0.34	-0.01	-0.06	0.02	-0.00



Introduction to Machine Learning

Regularization Lasso Regression



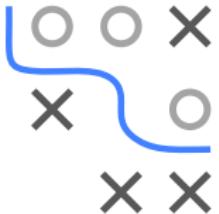
Learning goals

- Lasso regression / $L1$ penalty
- Know that lasso selects features
- Support recovery

LASSO REGRESSION

Another shrinkage method is the so-called **lasso regression** (least absolute shrinkage and selection operator), which uses an $L1$ penalty on θ :

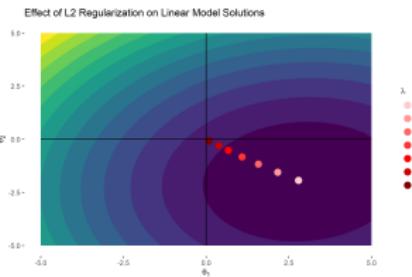
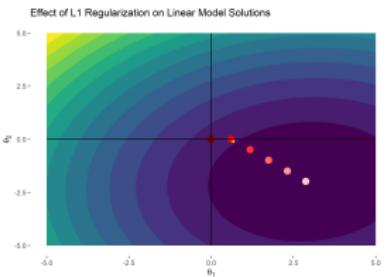
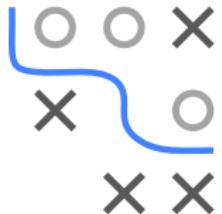
$$\begin{aligned}\hat{\theta}_{\text{lasso}} &= \arg \min_{\theta} \sum_{i=1}^n \left(y^{(i)} - \theta^T \mathbf{x}^{(i)} \right)^2 + \lambda \sum_{j=1}^p |\theta_j| \\ &= \arg \min_{\theta} (\mathbf{y} - \mathbf{X}\theta)^T (\mathbf{y} - \mathbf{X}\theta) + \lambda \|\theta\|_1\end{aligned}$$



Optimization is much harder now. $\mathcal{R}_{\text{reg}}(\theta)$ is still convex, but in general there is no analytical solution and it is non-differentiable.

LASSO REGRESSION

Let $y = 3x_1 - 2x_2 + \epsilon$, $\epsilon \sim N(0, 1)$. The true minimizer is $\theta^* = (3, -2)^T$. LHS = L1 regularization; RHS = L2

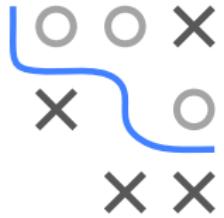
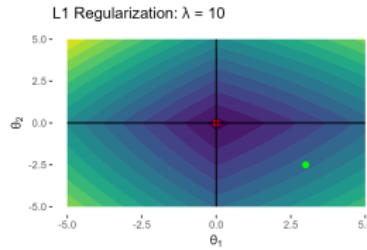
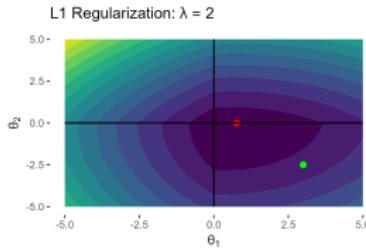
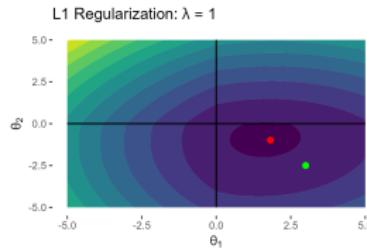
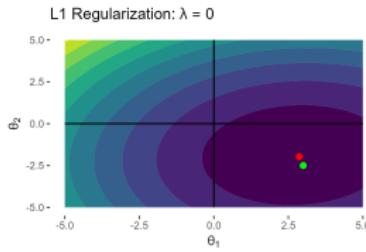


With increasing regularization, $\hat{\theta}_{lasso}$ is pulled back to the origin, but takes a different “route”. θ_2 eventually becomes 0!

$$0x^{10} + 1x^8 + 0.x^8$$

LASSO REGRESSION

Contours of regularized objective for different λ values.



Green = true minimizer of the unreg.objective and red = lasso solution.

Feature Scaling

Feature Scaling

REGULARIZATION AND FEATURE SCALING

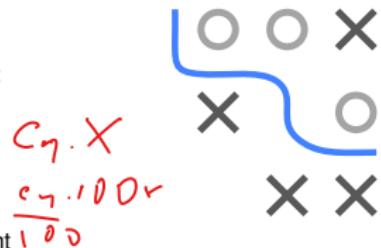
- Typically we omit θ_0 in penalty $J(\theta)$ so that the “infinitely” regularized model is the constant model (but can be implementation-dependent).
- Unregularized LM has **rescaling equivariance**, if you scale some features, can simply “anti-scale” coeffs and risk does not change.
- Not true for Reg-LM: if you down-scale features, coeffs become larger to counteract. They are then penalized stronger in $J(\theta)$, making them less attractive without any relevant reason.
- **So: usually standardize features in regularized models, whether linear or non-linear!**



REGULARIZATION AND FEATURE SCALING

- Let the DGP be $y = \sum_{j=1}^5 \theta_j x_j + \varepsilon$ for $\theta = (1, 2, 3, 4, 5)^\top$, $\varepsilon \sim \mathcal{N}(0, 1)$
- Suppose x_5 was measured in m but we change the unit to cm ($\tilde{x}_5 = 100 \cdot x_5$):

Method	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\theta}_4$	$\hat{\theta}_5$	MSE
OLS	0.984	2.147	3.006	3.918	5.205	0.812
OLS Rescaled	0.984	2.147	3.006	3.918	0.052	0.812



- Estimate $\hat{\theta}_5$ gets scaled by $1/100$ while other estimates and MSE are invariant
- Running ridge regression with $\lambda = 10$ on same data shows that rescaling of x_5 does not result in inverse rescaling of $\hat{\theta}_5$ (everything changes!)
- This is because $\hat{\theta}_5$ now lives on small scale while L2 constraint stays the same. Hence remaining estimates can "afford" larger magnitudes.

$$\begin{array}{l} \cancel{\hat{\theta}_5} \rightarrow \underline{0.01} \\ \cancel{\gamma} \leftarrow \underline{100} \quad \underline{0.1} \end{array}$$

Method	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\theta}_4$	$\hat{\theta}_5$	MSE
Ridge	0.709	1.874	2.661	3.558	4.636	1.366
Ridge Rescaled	0.802	1.943	2.675	3.569	0.052	1.08

- For lasso, especially for very correlated features, we could arbitrarily force a feature out of the model through a unit change.

Weight Decay

Weight Decay

WEIGHT DECAY VS. L2 REGULARIZATION

Let's optimize $L2$ -regularized risk of a model $f(\mathbf{x} | \theta)$

$$\min_{\theta} \mathcal{R}_{\text{reg}}(\theta) = \min_{\theta} \mathcal{R}_{\text{emp}}(\theta) + \frac{\lambda}{2} \|\theta\|_2^2$$

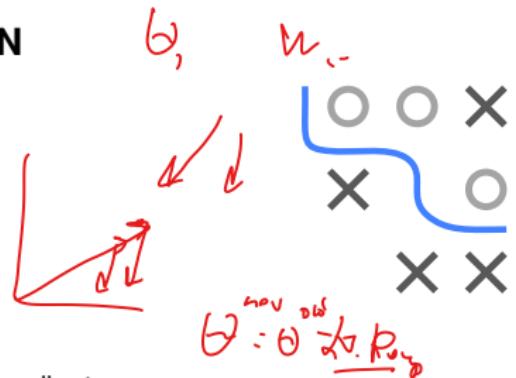
by GD. The gradient is

$$\nabla_{\theta} \mathcal{R}_{\text{reg}}(\theta) = \nabla_{\theta} \mathcal{R}_{\text{emp}}(\theta) + \lambda \theta$$

We iteratively update θ by step size α times the negative gradient

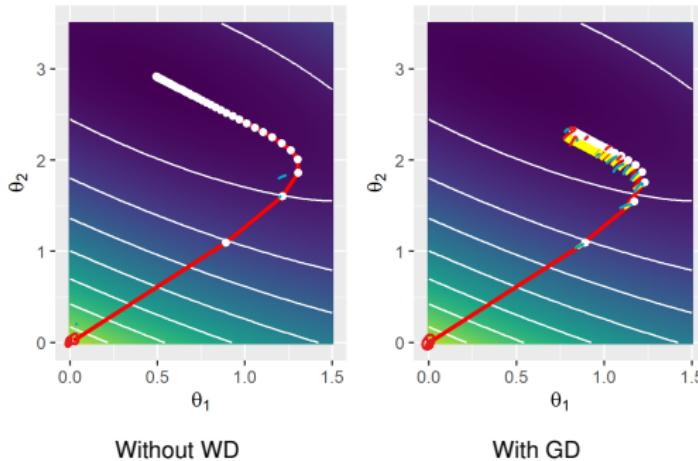
$$\begin{aligned}\theta^{[\text{new}]} &= \underline{\theta^{[\text{old}]}} - \alpha \left(\nabla_{\theta} \mathcal{R}_{\text{emp}}(\theta^{[\text{old}]}) + \lambda \theta^{[\text{old}]} \right) = \alpha \nabla - \alpha \cdot \lambda \theta^{[\text{old}]} \\ &= \underline{\theta^{[\text{old}]}} \left(1 - \alpha \lambda \right) - \underline{\alpha \nabla_{\theta} \mathcal{R}_{\text{emp}}(\theta^{[\text{old}]})}\end{aligned}$$

We see how $\theta^{[\text{old}]}$ decays in magnitude – for small α and λ – before we do the gradient step. Performing the decay directly, under this name, is a very well-known technique in DL – and simply $L2$ regularization in disguise (for GD).



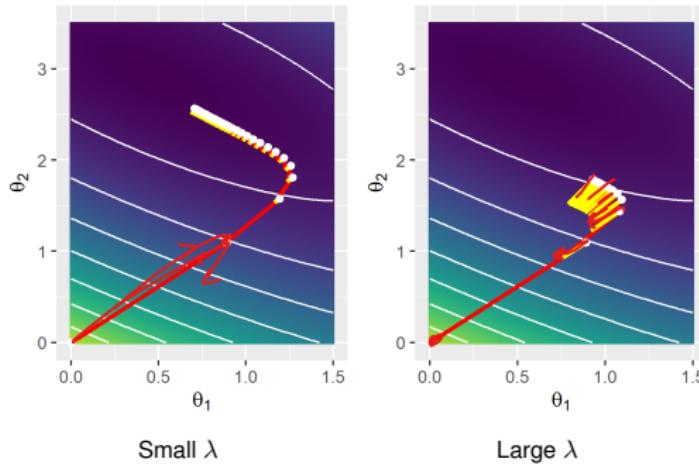
WEIGHT DECAY VS. L2 REGULARIZATION / 2

In GD With WD, we slide down neg. gradients of \mathcal{R}_{emp} ,
but in every step, we are pulled back to origin.



WEIGHT DECAY VS. L2 REGULARIZATION / 3

How strongly we are pulled back (for fixed α) depends on λ :

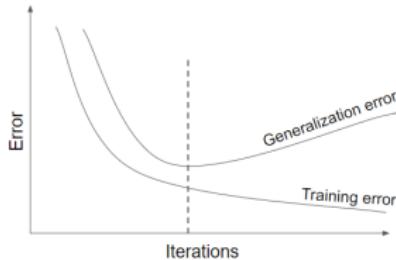
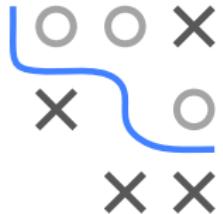


Early Stopping

Early Stopping

EARLY STOPPING

- Especially for complex nonlinear models we can easily overfit
- In optimization: Often, after a certain number of iterations, generalization error begins to increase even though training error continues to decrease



EARLY STOPPING / 2

For iterative optimizers like SGD,
we can monitor this step-by-step over small iterations:

- ➊ Split train data $\mathcal{D}_{\text{train}}$ into $\mathcal{D}_{\text{subtrain}}$ and \mathcal{D}_{val} (e.g. with ratio of 2:1)
- ➋ Train on $\mathcal{D}_{\text{subtrain}}$ and eval model on \mathcal{D}_{val}
- ➌ Stop when validation error stops decreasing
(after a range of “patience” steps)
- ➍ Use parameters of the previous step for the actual model



More sophisticated forms also apply cross-validation.

EARLY STOPPING AND L_2

Goodfellow, Bengio, and Courville 2016

Strengths	Weaknesses
Effective and simple	Periodical evaluation of validation error
Applicable to almost any model without adjustment	Temporary copy of θ (we have to save the whole model each time validation error improves)
Combinable with other regularization methods	Less data for training → include \mathcal{D}_{val} afterwards



- For simple case of LM with squared loss and GD optim initialized at $\theta = 0$: Early stopping has exact correspondence with L_2 regularization/WD: optimal early-stopping iter T_{stop} inversely proportional to λ scaled by step-size α

$$T_{\text{stop}} \approx \frac{1}{\alpha \lambda} \Leftrightarrow \lambda \approx \frac{1}{T_{\text{stop}} \alpha}$$

- Small λ (regu. \downarrow) \Rightarrow large T_{stop} (complexity \uparrow) and vice versa