JOHNS HOPKINS UNIVERSITY

EN.600.475 Machine Learning

Regularization

Raman Arora
Lecture 8
February 20, 2017

- Model complexity and overfitting
- Shrinkage methods: ridge regression, Lasso

Slides credit: Greg Shakhnarovich

Review: noise model and log-likelihood

- Statistical model: noise as a Gaussian random variable

$$y = f(\mathbf{x}; \mathbf{w}) + \nu, \qquad \nu \sim \mathcal{N}\left(\nu; 0, \sigma^2\right)$$

  equivalent to $p(y|\mathbf{x}; \mathbf{w}, \sigma) = \mathcal{N}\left(y; f(\mathbf{x}; \mathbf{w}), \sigma^2\right)$
- Maximizing log-likelihood under this model

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}} \sum_i \log p(y_i|\mathbf{x}_i; \mathbf{w}, \sigma)$$

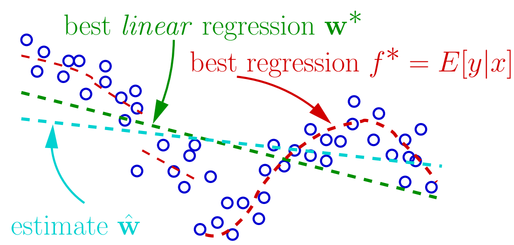
  is equivalent to least-squares regression

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_i \left(y_i - f(\mathbf{x}_i; \mathbf{w})\right)^2$$

Review: Decomposition of error

- Approximation error

$$E\left[\left(y - {\mathbf{w}^*}^T\mathbf{x}\right)^2\right]$$



- Estimation error

$$E\left[\left({\mathbf{w}^*}^T\mathbf{x} - \hat{\mathbf{w}}^T\mathbf{x}\right)^2\right]$$

- Approximation error: due to the failure to include optimal predictor in the model class, plus inherent uncertainty in $y|\mathbf{x}$
- Estimation error: due to failure to select the best predictor in the chosen model class; could be reduced with more data

Review: generalized linear regression

$$f(\mathbf{x};\mathbf{w}) \;=\; w_0 + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + \ldots + w_m\phi_m(\mathbf{x}),$$

- Still the same ML estimation technique applies:

$$\hat{\mathbf{w}} \;=\; \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

where $\mathbf{X}$ is the *design matrix*

$$\begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \ldots & \phi_m(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \ldots & \phi_m(\mathbf{x}_2) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \ldots & \phi_m(\mathbf{x}_N) \end{bmatrix}$$

Polynomial regression

- Consider 1D for simplicity:

$$f(x; \mathbf{w}) \;=\; w_0 + w_1 x + w_2 x^2 + \ldots + w_m x^m.$$

- No longer linear in x – but still linear in $\mathbf{w}$!
- Define $\boldsymbol{\phi}(x) = [1, x, x^2, \ldots, x^m]^T$
- Then, $f(x; \mathbf{w}) = \mathbf{w} \cdot \boldsymbol{\phi}(x)$ and we are back to the familiar simple linear regression. The least squares solution:

$$\hat{\mathbf{w}} \;=\; \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}, \text{ where } \mathbf{X} \;=\; \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^m \\ 1 & x_2 & x_2^2 & \ldots & x_2^m \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_N & x_N^2 & \ldots & x_N^m \end{bmatrix}$$

General additive regression models

- A general extension of the linear regression model:

$$f(\mathbf{x}; \mathbf{w}) \;=\; w_0 + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + \ldots + w_m\phi_m(\mathbf{x}),$$

  where $\phi_j(\mathbf{x}) : \mathcal{X} \to \mathbb{R}, j = 1, \ldots, m$ are the *basis functions*.
- This is still linear in $\mathbf{w}$,

$$f(\mathbf{x}; \mathbf{w}) \;=\; \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x})$$

  even when $\boldsymbol{\phi}$ is non-linear in the inputs $\mathbf{x}$.

General additive regression models

$$f(\mathbf{x};\mathbf{w}) \;=\; w_0 + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + \ldots + w_m\phi_m(\mathbf{x}),$$

- Still the same ML estimation technique applies:

$$\hat{\mathbf{w}} \;=\; \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

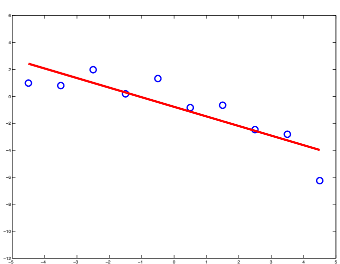
where $\mathbf{X}$ is the *design matrix*

$$\begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \ldots & \phi_m(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \ldots & \phi_m(\mathbf{x}_2) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \ldots & \phi_m(\mathbf{x}_N) \end{bmatrix}$$

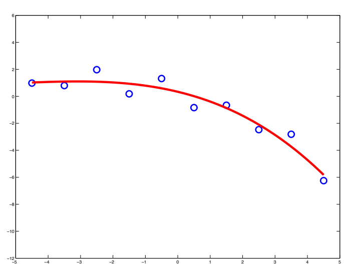
(for convenience we will denote $\phi_0(\mathbf{x}) \equiv 1$)

Model complexity and overfitting

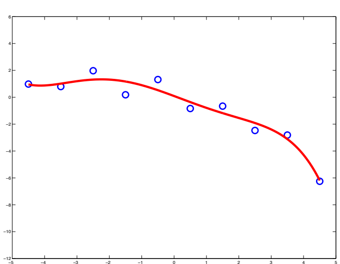
- Data drawn from 3rd order model:



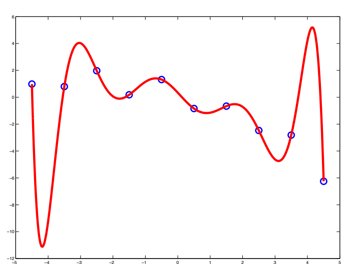
$m = 1$



$m = 3$



$m = 5$



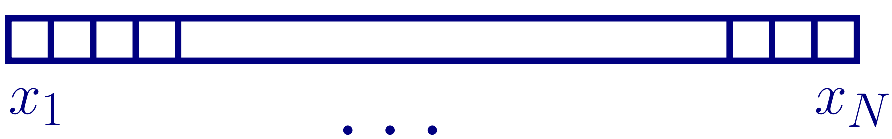
$m = 10$

How to avoid overfitting

- The basic idea: if a model overfits (is *too sensitive* to data) it will be unstable. I.e. removal part of the data will change the fit significantly.
- We can *hold out* part of the data; this is *validation (val)* set, or *development (dev)* set.
- Fit the model to the rest, and then test on the heldout data.
- What are the problems of this approach?
  - If the heldout set too small, we are susceptible to chance.
  - If it's too large, we get overly pessimistic (training on too little data compared to what we could do).

Cross-validation

- The improved holdout method: k-fold *cross-validation*
  - Partition data into k roughly equal parts;
  - Train on all but j-th part, test on j-th part

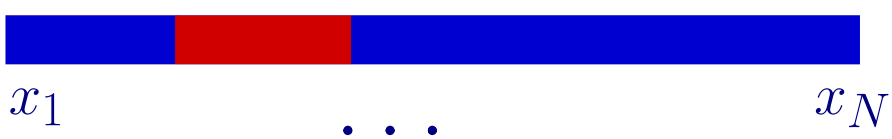


Cross-validation

- The improved holdout method: k-fold *cross-validation*
  - Partition data into k roughly equal parts;
  - Train on all but j-th part, test on j-th part

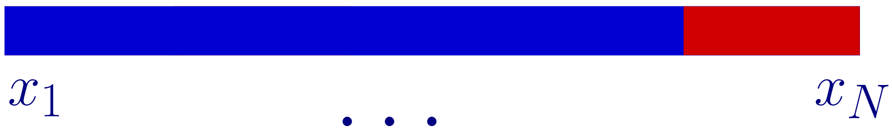
Cross-validation

- The improved holdout method: k-fold *cross-validation*
  - Partition data into k roughly equal parts;
  - Train on all but j-th part, test on j-th part



Cross-validation

- The improved holdout method: k-fold *cross-validation*
  - Partition data into k roughly equal parts;
  - Train on all but j-th part, test on j-th part



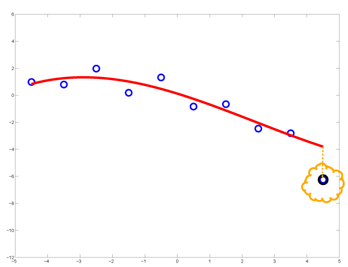
- An extreme case: *leave-one-out* cross-validation

$$\hat{L}_{\text{cv}} = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - f(\mathbf{x}_i; \hat{\mathbf{w}}_{-i})\right)^2$$

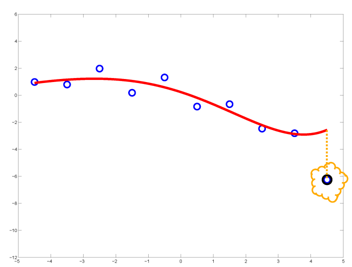
where $\hat{\mathbf{w}}_{-i}$ is fit to all the data but the i-th example.

Overfitting and complexity

Cross-validation: example



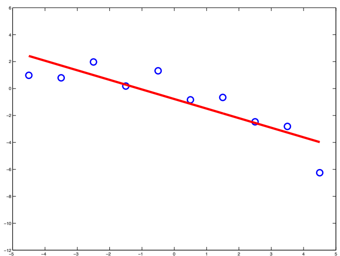
$m = 3$



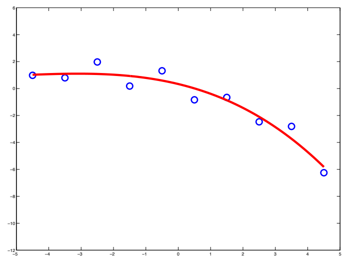
$m = 5$

- This is a very good estimate, although expensive to compute
  - Need to run N estimation problems each on $N - 1$ examples!
  - An important research area: devising tricks for efficiently computing cross-validation estimates (by taking advantage of overlap between folds).

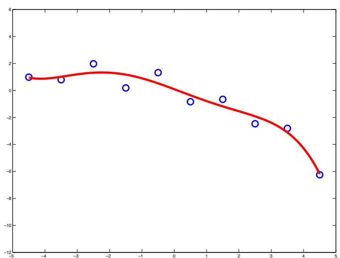
Cross-validation: example



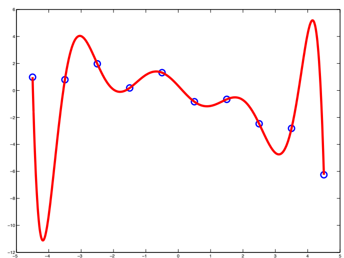
$m = 1 : L = 1.4, \hat{L}_{\mathsf{CV}} = 2.6$



$m = 3 : L = 0.4, \hat{L}_{\mathsf{CV}} = 1.3$



$m = 5 : L = 0.3, \hat{L}_{\mathsf{CV}} = 2.7$



$m = 10 : L = 0, \hat{L}_{\mathsf{CV}} = 4 \times 10^4$

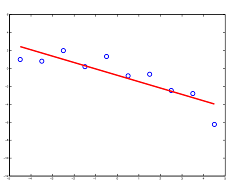
Understanding overfitting

- Cross validation provides some means of dealing with overfitting
- What is the source of overfitting? Why do some models overfit more than others?
- We can try to get some insight by thinking about the estimation process for model parameters

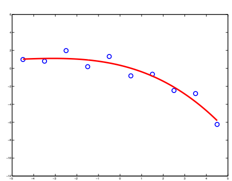
Roadmap

- So far: least squares regression (with arbitrary feature functions)
  - Closed form solution for maximum likelihood
  - Overfitting is a problem
- Today: regularization – main tool to combat overfitting
- Also: gradient descent as an alternative to closed form solution

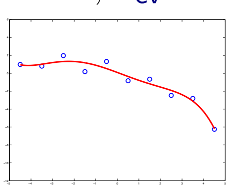
Controlling for overfitting



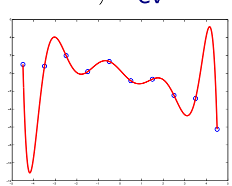
$L = 1.4, \hat{L}_{\text{CV}} = 2.6$



$L = 0.4, \hat{L}_{\text{CV}} = 1.3$



$L = 0.3, \hat{L}_{\text{CV}} = 2.7$

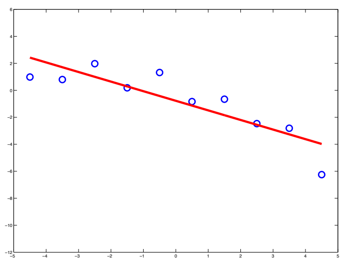


$L = 0, \hat{L}_{\text{CV}} = 4 \times 10^4$

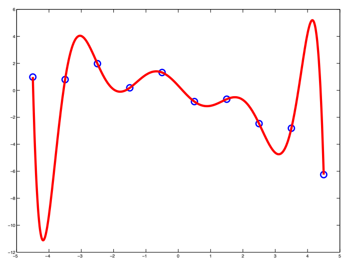
- More complex model (10th degree) overfits more than simple model (linear)
- Pure ERM would always prefer complex models
- Holdout/validation/cross-validation is a way to control for this in *model selection*

Model complexity - intuition

- Intuitively, the complexity of the model can be measured by the number of "degrees of freedom" (independent parameters).
  - The more complex the model, the more data needed to fit $\Rightarrow$ For a given number of points, a more complex model more likely to overfit.



$m = 1$, 2 parameters



$m = 10$, 11 parameters

- This is an issue only because of finite training data!

Penalizing model complexity

- Idea 1: restrict model complexity based on amount of data
  - Rule of thumb: $\approx$ 10 examples per parameter
- Idea 2: directly penalize by the number of parameters. Akaike information criterion (AIC): maximize

$$\log p\left(X \mid \widehat{\mathbf{w}}\right) - \#\text{params}$$

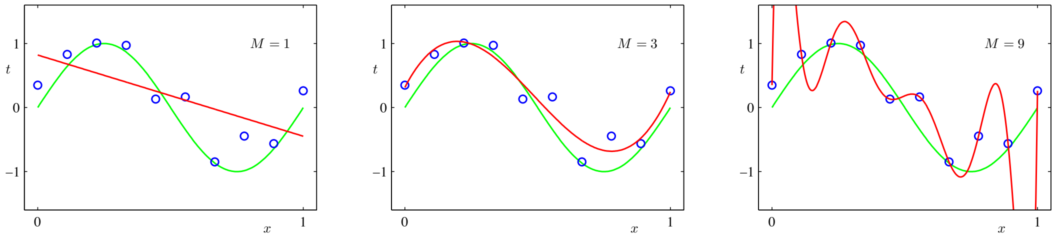
- But: Definition of model complexity as a number of parameters is a bit too simplistic. Consider feature vector

$$\phi x = \begin{bmatrix} 1 & x & -2x & 2x & x^2 & \frac{1}{2}x^2 \end{bmatrix}$$

  Does linear regression $\phi(x) \rightarrow y$ really have 6 parameters?
- Idea: look at the behavior of the values of $\mathbf{w}^*$

Linear regression complexity

- Example: polynomial regression, true [from Bishop, Ch. 1]



- Value of the optimal (ML) regression coefficients:

	$m=0$	$m=1$	$m=3$	$m=9$
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*			17.37	48568.31
w_4^*				-231639.30
w_5^*				640042.26
w_6^*				-1061800.52
w_7^*				1042400.18
w_8^*				-557682.99
w_9^*				125201.43

Description length

- Intuition: should penalize not the parameters, but the number of bits required to encode the parameters
- With finite set of parameter values, these are equivalent
- With "infinite" set, we can limit the effective number of degrees of freedom by restricting the value of the parameters.
- Then we have penalized log-likelihood:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}} \left\{ \frac{1}{2} \sum_{i=1}^{N} \log p(\text{data}_i;\, \mathbf{w}) - \text{penalty}(\mathbf{w}) \right\}$$

Shrinkage methods

- Shrinkage methods impose penalty on the size of $\mathbf{w}$
- We can measure "size" in a few different ways. Let us start with L_2 norm:

$$\mathbf{w}^*_{\text{ridge}} = \operatorname*{argmax}_{\mathbf{w}} \left\{ \sum_{i=1}^{N} \log p(\text{data}_i;\, \mathbf{w}) \,-\, \lambda \|\mathbf{w}\|^2 \right\}$$

in regression "data$_i$" $= y_i|\mathbf{x}_i$

- This is *ridge regression*; λ is the *regularization* parameter
- Does it matter that log-likelihood is not averaged? Consider relative effect of the value of λ

Ridge regression

$$\mathbf{w}^*_{\text{ridge}} = \operatorname*{argmin}_{\mathbf{w}} \left\{ \sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 \,+\, \lambda \sum_{j=1}^{m} w_j^2 \right\}$$

- Recall: $\mathbf{w} = [w_0, w_1, \ldots, w_m]$
- Usually do not include w_0 in regularization (why?)
- Closed form solution:

$$\widehat{\mathbf{w}}^*_{\text{ridge}} = \left(\lambda \mathbf{I} + \mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y}.$$

- Careful: solution *not* invariant to scaling! Should normalize input before solving.

Lasso regression

- The L_1-penalized maximum likelihood under Gaussian noise model:

$$\mathbf{w}^*_{\text{lasso}} = \operatorname*{argmax}_{\mathbf{w}} \left\{ -\sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 - \lambda \sum_{j=1}^{m} |w_j| \right\}$$

- This is still concave (i.e. unique maximum), but not "smooth" (differentiable).
- Can solve it efficiently using convex programming methods or first-order numerical optimization (gradient descent)
- Why is it called "lasso"?
  least absolute shrinkage and selection operator

Optimization of ridge regression

- Can rewrite the optimization problem

$$\min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 + \lambda \sum_{j=1}^{m} w_j^2$$

  in the proper objective/constraint form:

$$\min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2$$

$$\text{subject to } \sum_{j=1}^{m} w_j^2 \leq t$$

- Correspondence $\lambda \Rightarrow t$ can be shown using Lagrange multipliers.

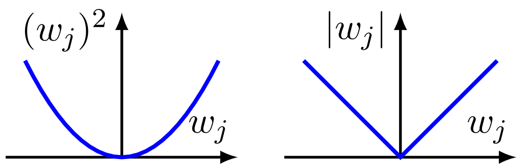
Optimization for Lasso

- Similarly, for Lasso:

$$\min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2$$

$$\text{subject to} \sum_{j=1}^{m} |w_j| \leq t$$

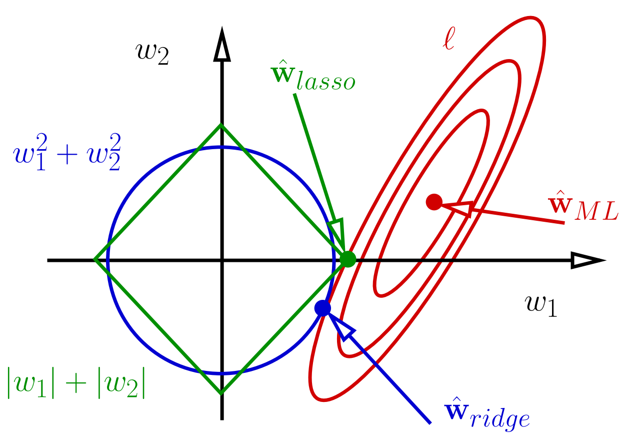
- Compare shape of the penalty as a function of w_j:



Lasso vs. ridge: geometry of error surfaces

- An equivalent formulation for L_p regularization: constrained maximization

$$\hat{\mathbf{w}} = \underset{\mathbf{w}:\, \sum_{j=1}^{m} |w_j|^p \leq \beta}{\operatorname{argmax}} - \sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2.$$

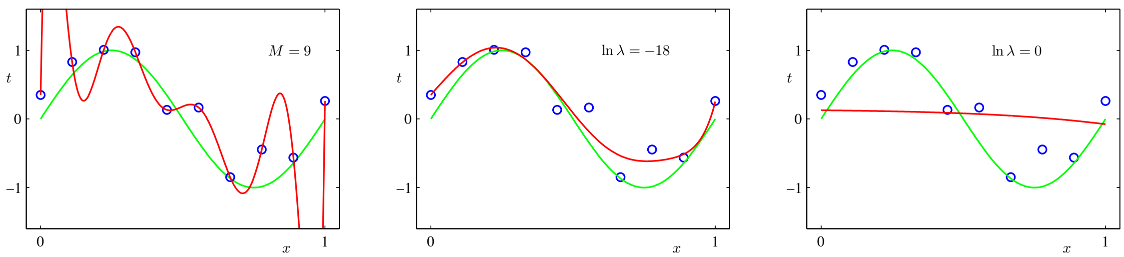


- With sufficiently large λ (=sufficiently small β) lasso leads to *sparsity*.
- Must explicitly solve the above optimization problem – e.g., using Lagrange multipliers.

Choice of λ

- Example [from Bishop, Ch. 1]: 9th deg polynomial with varying λ:

$\lambda = 0$ $\quad$ $\lambda = e^{-18}$ $\quad$ $\lambda = 1$

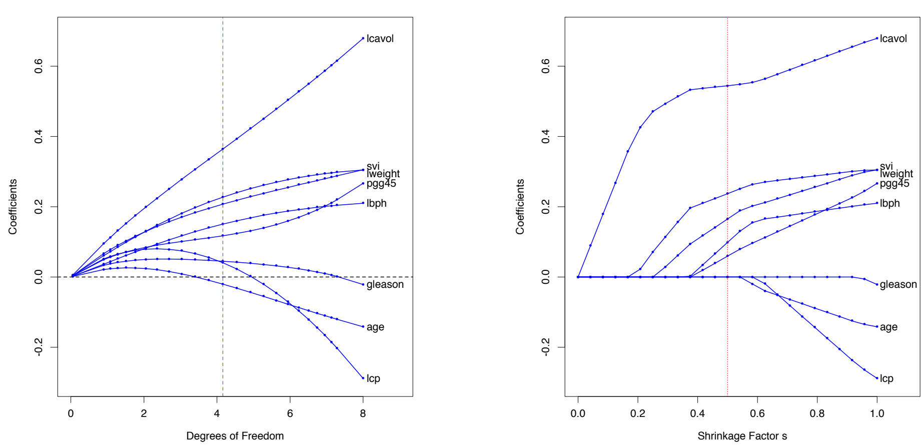


$\|\mathbf{w}^*\|^2 > 10^{12}$ $\quad$ $\|\mathbf{w}^*\|^2 \approx 21595$ $\quad$ $\|\mathbf{w}^*\|^2 \approx 0.027$

- Most often: choose λ by (cross) validation

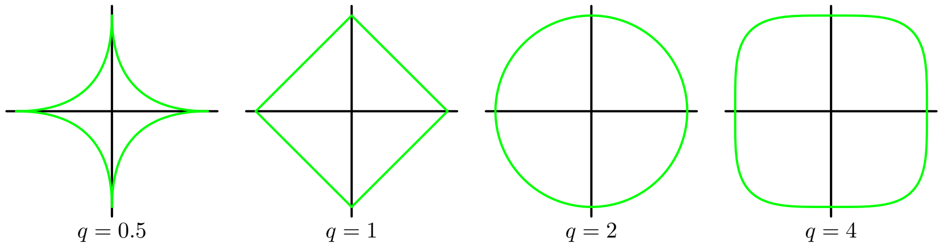
Example: lasso vs. ridge regularization paths

- Example: prostate data [Hastie, Tibshirani and Friedman]
  Red lines: choice of λ by 10-fold CV.



General view of L_q penalty

- Can be creative in design of penalty function $\|\mathbf{w}\|_q$



- For $q > 1$, no sparsity is achieved.
- For $q < 1$, non-convex
- What about L_0?

$$\min_{\mathbf{w}} \sum (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 \quad \text{s.t. } |\{w_j : w_j > 0\}| \leq M$$

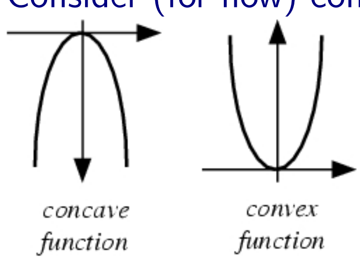
is NP-hard

Beyond closed form solution

- So far: solve (least squares) regression with a closed form solution

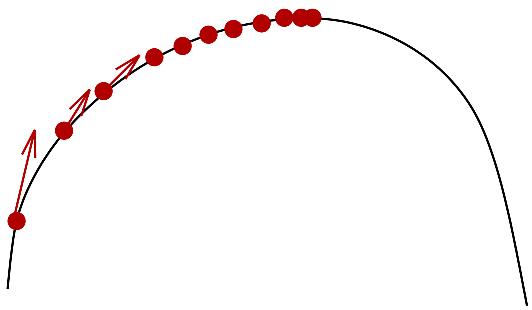
$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

- Sometimes we can not do this. E.g., the data matrix is too large to compute the pseudoinverse for
- If we move away from simple squared loss (e.g., in PS1: asymmetric loss) also lose the closed form solution
- Alternative: numerical optimization – gradient descent
- Consider (for now) convex or concave functions



Gradient ascent/descent

- The idea behind gradient ascent: "hill climbing" on the function surface.



- Start at a (random) location
- Make steps in the direction of maximal altitude increase.

- An equivalent: gradient *descent* on the *convex* loss $-\log p\,(y \mid \mathbf{x}; \mathbf{w})$

Gradient descent algorithm on $f(\mathbf{X}, \mathbf{y}; \mathbf{w})$

- Iteration counter $t = 0$
- Initialize $\mathbf{w}^{(t)}$ (to zero or a small random vector)
- for $t = 1, \ldots$:
  compute gradient

$$\mathbf{g}^{(t)} = \nabla f\left(\mathbf{X}, \mathbf{y}; \mathbf{w}^{(t-1)}\right)$$

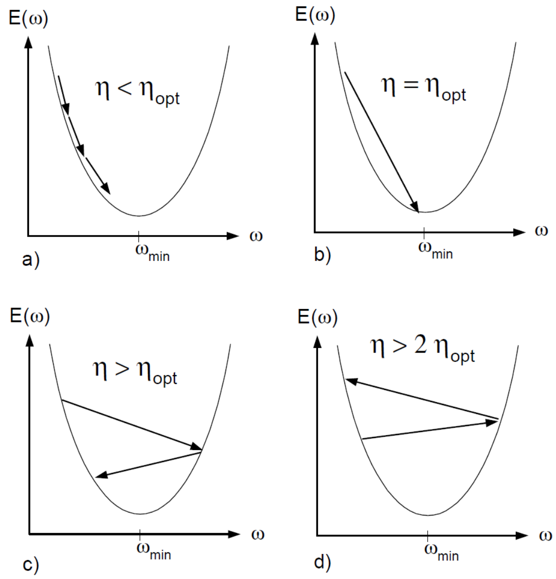
  update model

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta \mathbf{g}^{(t)}$$

  check for convergence
- The *learning rate* η controls the step size

Gradient descent convergence

- Generally, for convex functions, gradient descent will converge
- Setting the learning rate η may be very important to ensure rapid convergence



From Lecun et al, 1996

Gradient descent convergence

- A lot of theory on convergence of gradient descent
- Usually relies on various properties of the objective function: strong convexity, smoothness, etc.
- In practice, need to monitor the objective, tweak learning rate, and consider stopping ("convergence") criteria
- Common criteria (often use a combination):
  - Maximum number of iterations (time budget)
  - Minimum required change in objective value (loss)
  - Minimum required change in model parameters ($\mathbf{w}$)
- If stopped because of max iterations: may not have converged
- Problematic criteria: monitor absolute (not relative) value of something like objective or parameters. Often hard to know what the "right" value for these is.