STA 314: Statistical Methods for Machine Learning I

Lecture 5 - Regularized linear regression and gradient descent

Xin Bing

Department of Statistical Sciences
University of Toronto

Why consider alternatives to the OLS estimator?

Recall the linear model is

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \epsilon.$$

Alternative fitting procedures to OLS could yield **better prediction accuracy** and **model interpretability**.

- Prediction: OLS estimator has large variance when p is large. Especially, if $p > n$, then OLS estimator is not unique and its variance is infinite.
- Interpretability: By removing irrelevant features – that is, by setting some coefficient estimates to zero – we can obtain a model that is more parsimonious hence more interpretable.

Shrinkage Methods / Regularization

- We can fit a model containing all p predictors using a technique that constrains or regularizes the coefficient estimates, or equivalently, that shrinks the coefficient estimates towards zero.
- Shrinking the coefficient estimates can significantly reduce their variance.
- The two best-known techniques for shrinking the regression coefficients towards zero are the **ridge regression** and the **lasso**.

Ridge Regression

- Recall that the OLS fitting procedure estimates $\beta_0, ..., \beta_p$ using the values that minimize

$$RSS = \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2.$$

- The **ridge regression** estimates $\beta_0, ..., \beta_p$ using the values that minimize

$$\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

  where $\lambda \geq 0$ is a tuning (regularization) parameter, to be determined later.

Comments

$$\hat{\beta}_{\lambda}^{R} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \underbrace{\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2}_{RSS} + \lambda \sum_{j=1}^{p}\beta_j^2.$$

- We usually denote the ridge regression estimator by $\hat{\beta}_{\lambda}^{R}$, because different λ's produce distinct estimators.
- The term $\lambda \sum_{j=1}^{p} \beta_j^2$ is called a **shrinkage / regularization penalty**, which shrinks the estimates of β_i towards 0.
- We usually do not penalize the intercept β_0.
- Comparing to the OLS estimator, the ridge regression finds the coefficient estimate of $\boldsymbol{\beta}$ that has small entries (toward 0) by affording a slightly larger RSS. The balance is controlled by λ.

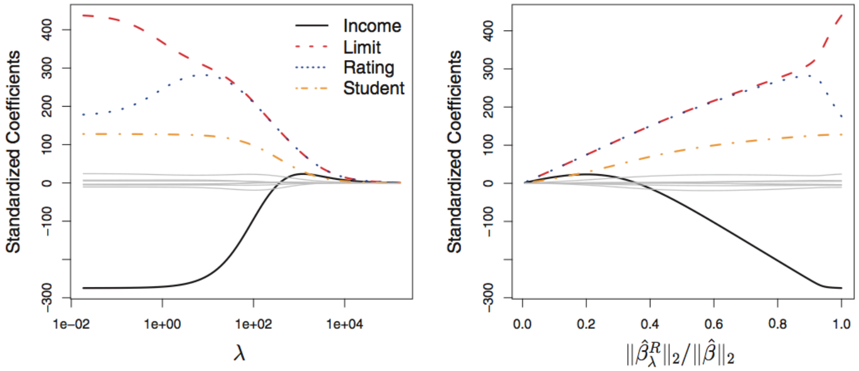
More Comments

- Selecting a good value for λ is critical. For $\lambda = 0$, the ridge estimator of β coincides with the OLS estimator. Later, we use cross-validation to select λ.
- The ridge regression coefficient estimates are not equivariant (to any linear transformation of $\mathbf{X}$), due to the sum of squared coefficients term in the penalty part of the ridge regression objective function.
- In practice, we recommend the standardized predictors for ridge regression, using the formula

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2}}.$$

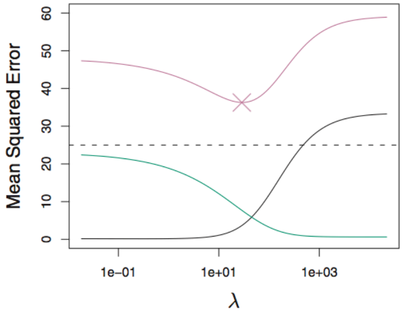
All standardized predictors have standard deviation equal to one.

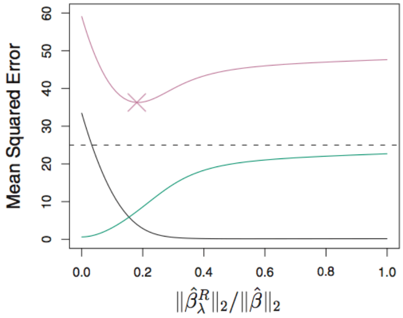
Credit Card Data Example



- In the left-hand panel, each curve corresponds to the ridge regression coefficient estimate for one of the 10 variables, plotted as a function of λ.
- The right-hand panel displays the same ridge coefficient estimates as the left-hand panel, but we now display $||\hat{\beta}^R_\lambda||_2/||\hat{\beta}||_2$, where $\hat{\beta}$ denotes OLS estimator.

Ridge Regression Improves Over OLS





Squared bias (black), variance (green), and test mean squared error (purple) for the ridge regression. The dashed lines indicate the smallest possible MSE.

- Ridge does a better job for prediction than the OLS approach by reducing the coefficient estimates.
- Ridge regression is computationally efficient, comparable to the OLS approach. In particular, it has substantial computational advantages over the best subset selection.
- Can we use ridge regression for variable selection (excluding features that are not important)?
  - No, it tends to include all p features in the model! The resulting model is difficult to interpret.
- Lasso does a better job than the OLS for both prediction and variable selection!

The Lasso

- Different from ridge, lasso shrinks the coefficients by penalizing their absolute values.
- Specifically, the lasso coefficients, $\hat{\beta}_\lambda^L$, minimize the quantity

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}|\beta_j| = RSS + \lambda \sum_{j=1}^{p}|\beta_j|,$$

  where $\lambda \geq 0$ is a tuning parameter, to be determined later.

- Different from the ridge regression that uses the ℓ_2 penalty

$$||\beta||_2^2 = \sum_{j=1}^{p}\beta_j^2,$$

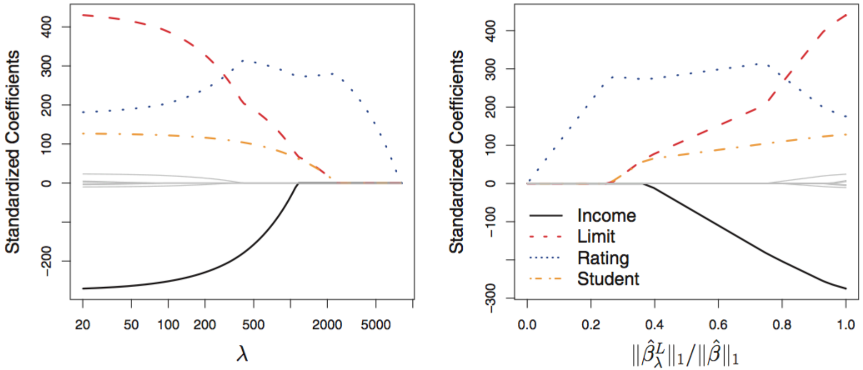
  lasso uses the ℓ_1 penalty

$$||\beta||_1 = \sum_{j=1}^{p}|\beta_j|.$$

More Comments

- Similar to ridge regression, the lasso shrinks the coefficient estimates towards zero.
- However, in the case of the lasso, the ℓ_1 penalty has the effect of forcing some of the coefficient estimates to be **exact zero** when the tuning parameter λ is sufficiently large.
- Therefore, the lasso performs variable selection.
- We say that the lasso yields a **sparse model** if the fitted model involves only a subset of the variables.
- Similar to ridge regression, selecting a good value of the regularization parameter λ for the lasso is critical; cross-validation is again the method of choice.

Credit Card Data Example



Why does the lasso, unlike ridge regression, yield coefficient estimates that have exact zero?

Another Formulation for Ridge Regression and Lasso

The lasso and ridge regression coefficient estimates solve the problems

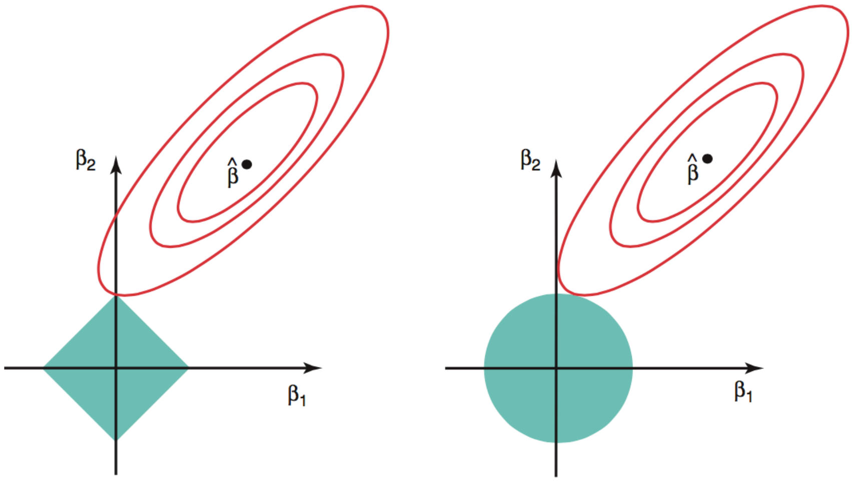
$$\underset{\beta}{\text{minimize}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s$$

and

$$\underset{\beta}{\text{minimize}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \leq s,$$

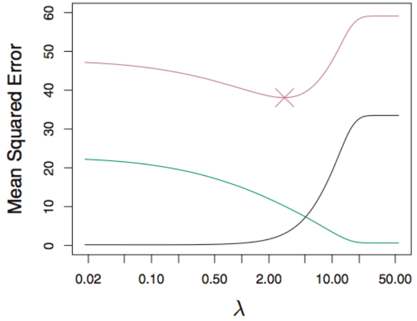
Here $s \geq 0$ is some regularization parameter.

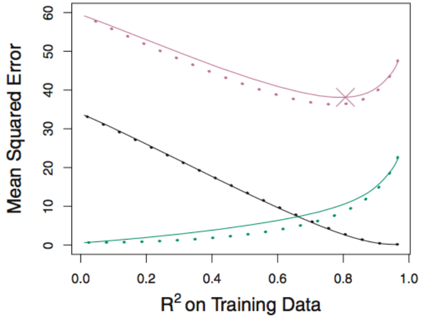
Understand why the Lasso yields zero estimates



The solid areas are the constraint regions, $|\beta_1| + |\beta_2| \le s$ and $\beta_1^2 + \beta_2^2 \le s$, while the red ellipses are the contours of the RSS.

Comparing the MSE of Lasso and Ridge

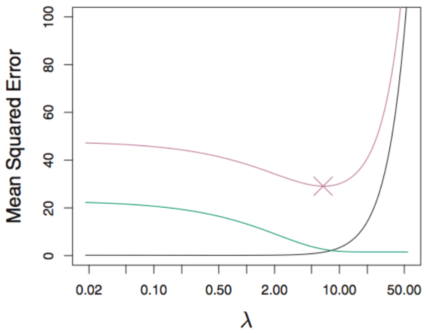


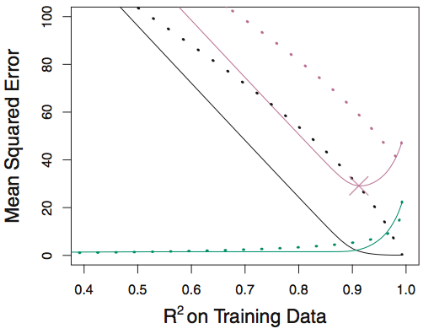


Left: Plots of squared bias (black), variance (green), and test MSE (purple) for the lasso on a simulated data set.
Right: Comparison of squared bias, variance and test MSE between lasso (solid) and ridge (dotted). Both are plotted against their R^2 on the training data, as a common form of indexing. The crosses in both plots indicate the lasso model for which the MSE is smallest.

- *When the true coefficients are non-sparse, ridge and lasso have the same bias but ridge has a smaller variance hence a smaller MSE.*

Another Case





- *When the true coefficients are sparse, Lasso outperforms ridge regression of having both a smaller bias and a smaller variance.*

Conclusions

- These two examples illustrate that neither ridge regression nor the lasso will universally dominate the other.
- In general, one might expect the lasso to perform better when the response is only related with a relatively small number of predictors.
- As the ridge regression, when the OLS estimates have excessively high variance, the lasso solution can yield a reduction in variance at the expense of a small increase in bias, and consequently can lead to more accurate predictions.
- Unlike ridge regression, the lasso performs variable selection, and hence yields models that are easier to interpret.

A simple example of the shrinkage effects of ridge and lasso

- Assume that $n = p$ and $\mathbf{X} = \mathbf{I}_n$. We force the intercept term $\beta_0 = 0$.
- The OLS approach is to find $\beta_1, \ldots, \beta_p$ that minimize

$$\sum_{j=1}^{p} (y_j - \beta_j)^2.$$

This gives the OLS estimator

$$\hat{\beta}_j = y_j, \qquad \forall j \in \{1, \ldots, p\}.$$

The ridge estimator

- The ridge regression is to find $\beta_1, \dots, \beta_p$ that minimize

$$\sum_{j=1}^{p} (y_j - \beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2.$$

  This leads to the ridge estimator

$$\hat{\beta}_j^R = \frac{y_j}{1 + \lambda}, \qquad \forall j \in \{1, \dots, p\}.$$

  Since $\lambda \geq 0$, the magnitude of each estimated coefficient is shrinked toward 0.

The lasso estimator

- The lasso is to find $\beta_1, \dots, \beta_p$ that minimize

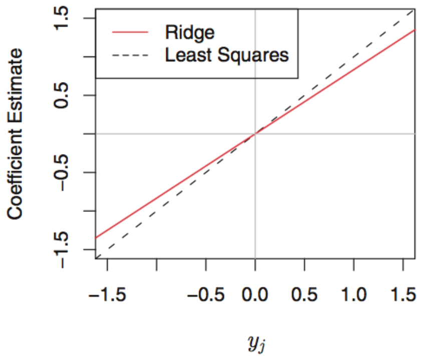
$$\sum_{j=1}^{p}(y_j - \beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|.$$

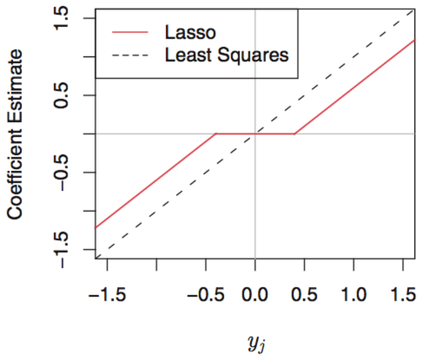
This gives estimator

$$\hat{\beta}_j^L = \begin{cases} y_j - \lambda/2 & \text{if } y_j > \lambda/2; \\ y_j + \lambda/2 & \text{if } y_j < -\lambda/2; \\ 0 & \text{if } |y_j| \le \lambda/2. \end{cases}$$

The estimated coefficients from Lasso are also shrinked. The above shrinkage is known as the **soft-thresholding**.

An illustrative figure

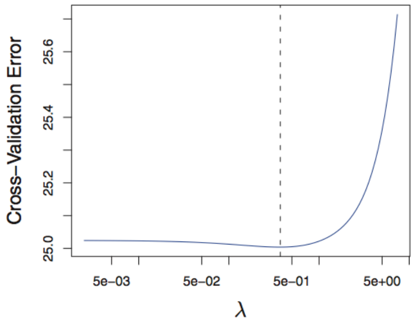


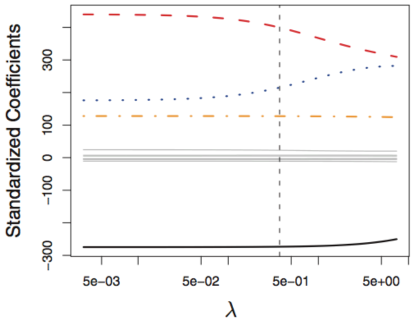


Selecting the Tuning Parameter

- Similar as the subset selection, for ridge and lasso, we require a systematic way of choosing the best model under a sequence of fitted models (from different choices of λ)
- Equivalently, we require a method to select the optimal value of the tuning parameter λ or equivalently, the value of the constraint s.
- Cross-validation provides a simple way to tackle this problem. We choose a grid of λ, and compute the cross-validation error rate for each value of λ.
- We then select the tuning parameter value for which the cross-validation error is smallest.
- Finally, the model is re-fitted by using all of the available observations and the selected value of the tuning parameter.

Credit Card Data Example





Cross-validation errors that result from applying ridge regression to the Credit data set with various value of λ.

More choices of penalties

- There are many other penalties in addition to the ℓ_2 and ℓ_1 norms used by ridge and lasso.
  - the elastic net:

$$\underset{\boldsymbol{\beta}}{\operatorname{argmin}} \quad ||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda\left[(1-\alpha)||\boldsymbol{\beta}||_1 + \alpha||\boldsymbol{\beta}||_2\right]$$

    for some tuning parameters $\lambda \geq 0$ and $\alpha \in [0,1]$. Ridge corresponds to $\alpha = 1$ while lasso corresponds to $\alpha = 0$.

The group lasso

- If we suspect the model is nonlinear in X_1 or X_2, we can add quadratic terms, say

  $$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \beta_3 X_2 + \beta_4 X_2^2 + \epsilon.$$

  The **group lasso** estimator minimizes

  $$RSS + \lambda \left(\sqrt{\beta_1^2 + \beta_2^2} + \sqrt{\beta_3^2 + \beta_4^2} \right).$$

  In this penalty, we view β_1 and β_2 (coefficient of X_1 and X_1^2) as if they belong to the same group. The group Lasso can shrink the parameters in the same group (both β_1 and β_2) exactly to 0 simultaneously.

- There are a lot more penalties out there ......

Regularization in more general settings

- The ridge and lasso regressions are not restricted to the linear models.
- The idea of penalization is generally applicable to almost all parametric models.

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \underbrace{L(\boldsymbol{\beta}, \mathcal{D}^{train}) + Pen(\boldsymbol{\beta})}_{g(\boldsymbol{\beta}; \mathcal{D}^{train})}.$$

  - OLS: $L(\boldsymbol{\beta}, \mathcal{D}^{train}) = ||\mathbf{y} - \mathbf{X}\beta||_2^2$, $Pen(\boldsymbol{\beta}) = 0$.
  - Ridge: $L(\boldsymbol{\beta}, \mathcal{D}^{train}) = ||\mathbf{y} - \mathbf{X}\beta||_2^2$, $Pen(\boldsymbol{\beta}) = ||\boldsymbol{\beta}||_2^2$.
  - Lasso: $L(\boldsymbol{\beta}, \mathcal{D}^{train}) = ||\mathbf{y} - \mathbf{X}\beta||_2^2$, $Pen(\boldsymbol{\beta}) = ||\boldsymbol{\beta}||_1$.
  - In general,
    - L can be any loss function, i.e. negative likelihood, 0-1 loss.
    - Pen could be any penalty function.

Solving the Minimization Problem

Suppose we want to solve the following problem

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, g(\boldsymbol{\beta}; \mathcal{D}^{train})$$

where $g(\boldsymbol{\beta}; \mathcal{D}^{train})$ is a smooth function depending on $\boldsymbol{\beta}$ and $\mathcal{D}^{train}$.

- The optimal solution (if exists) must be a critical point, i.e. point to which the derivative is zero (partial derivatives to zero for multi-dimensional parameter).

- Finding the optimal solution needs to solve the equations. Solutions may be direct or iterative
  - Sometimes we obtain a direct solution, closed-form expression.
  - We may also use optimization techniques that iteratively get us closer to the solution.

Direct solution

- Partial derivatives: derivatives of a multivariate function with respect to one of its arguments.

$$\frac{\partial}{\partial x_1} f(x_1, x_2) = \lim_{h \to 0} \frac{f(x_1 + h, x_2) - f(x_1, x_2)}{h}$$

- The minimum must occur at a point where the partial derivatives are zero.

$$\begin{bmatrix} \frac{\partial g}{\partial \beta_1} \\ \vdots \\ \frac{\partial g}{\partial \beta_p} \end{bmatrix} = 0$$

- This turns out to give a system of linear equations, which we can solve analytically in some scenarios.

Direct solution

- OLS:

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, g(\boldsymbol{\beta}; \mathcal{D}^{train}) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2.$$

  The partial derivatives w.r.t. $\boldsymbol{\beta}$ are

$$\frac{\partial g}{\partial \boldsymbol{\beta}} = -2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}).$$

  (If not familiar with multi-dimensional derivatives, calculate $\frac{\partial g}{\partial \beta_j}$ and stack them together).
  Setting the above equal to zero results

$$\mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}^\top\mathbf{y}, \qquad \Rightarrow \qquad \hat{\boldsymbol{\beta}} = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{y}.$$

Direct solution

- Ridge:

$$\hat{\beta}_\lambda^R = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, g(\boldsymbol{\beta}; \mathcal{D}^{train}) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2.$$

The partial derivatives w.r.t. $\boldsymbol{\beta}$ are

$$\frac{\partial g}{\partial \boldsymbol{\beta}} = -2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + 2\lambda\boldsymbol{\beta}.$$

Setting the above equal to zero results

$$(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})\hat{\beta}_\lambda^R = \mathbf{X}^\top\mathbf{y}, \qquad \Rightarrow \qquad \hat{\beta}_\lambda^R = \left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^\top\mathbf{y}.$$

Gradient Descent

- Now let's see a second way to solve

$$\hat{\mathbf{w}} = \min_{\mathbf{w}} \mathcal{J}(\mathbf{w})$$

  which is more broadly applicable: gradient descent.

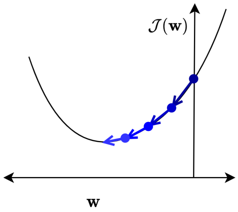
- Many times, we do not have a direct solution to

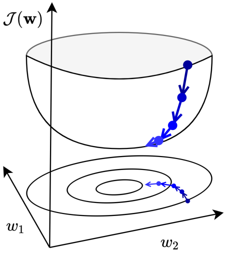
$$\frac{\partial g}{\partial \mathbf{w}} = 0.$$

- Gradient descent is an iterative algorithm, which means we apply an update repeatedly until some criterion is met.

Gradient Descent

We initialize **w** to something reasonable (e.g. all zeros) and repeatedly adjust them in the direction of steepest descent.





What is the direction of the steepest descent of $\mathcal{J}(\mathbf{w})$ at $\mathbf{w}$?

Gradient Descent

- By definition, the direction of the greatest increase in $\mathcal{J}(\mathbf{w})$ at $\mathbf{w}$ is its gradient $\partial\mathcal{J}/\partial\mathbf{w}$. So, we should update $\mathbf{w}$ in the opposite direction of the gradient descent.
- The following update always decreases the cost function for small enough α (unless $\partial\mathcal{J}/\partial w_j = 0$): at the $(k+1)$th iteration,

$$w_j^{(k+1)} \leftarrow w_j^{(k)} - \alpha \frac{\partial \mathcal{J}}{\partial w_j}\Big|_{\mathbf{w}=\mathbf{w}^{(k)}}$$

- $\alpha > 0$ is a learning rate (or step size). The larger it is, the faster $\mathbf{w}^{(k+1)}$ changes relative to $\mathbf{w}^{(k)}$
  - We'll see later how to tune the learning rate, but values are typically small, e.g. 0.01 or 0.0001.

Gradient descent for OLS

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}} \mathcal{J}(\mathbf{w}), \qquad \mathcal{J}(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2.$$

- Update rule in vector form at the $k+1$th iteration:

$$\begin{aligned}\mathbf{w}^{(k+1)} &\leftarrow \mathbf{w}^{(k)} - \alpha \frac{\partial \mathcal{J}}{\partial \mathbf{w}}\Big|_{\mathbf{w}=\mathbf{w}^{(k)}} \\ &= \mathbf{w}^{(k)} + 2\alpha \mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}^{(k)}).\end{aligned}$$

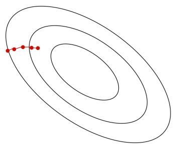
- Stopping criteria: when do we stop?
  - The objective value stops changing: $|\mathcal{J}(\mathbf{w}^{(k+1)}) - \mathcal{J}(\mathbf{w}^{(k)})|$ is small, i.e. $\leq 1e-6$.
  - The parameter stops changing: $\|\mathbf{w}^{(k+1)} - \mathbf{w}^{(k)}\|_2$ is small or $\|\mathbf{w}^{(k+1)} - \mathbf{w}^{(k)}\|_2 / \|\mathbf{w}^{(k)}\|_2$ is small.

Gradient Descent for Linear Regression

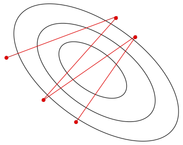
- The squared error loss of linear regression is a convex function. So there is a unique and direct solution. Even in this case, we sometimes need to use GD.
- Why gradient descent, if we can find the optimum directly?
  - When p is large, GD is more efficient than direct solution
    - Linear regression solution: $(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$
    - Matrix inversion is an $\mathcal{O}(p^3)$ algorithm
    - Each GD update costs $\mathcal{O}(np)$
    - Or less with stochastic GD (Stochastic GD, later)
    - Huge difference if $p \gg \sqrt{n}$
- In general, GD can be applied to much broader settings where there is no direct solution.

Learning Rate (Step Size)

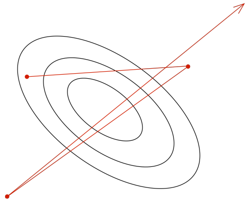
- In gradient descent, the learning rate α is a hyperparameter we need to tune. Here are some things that can go wrong:



α too small: slow progress



α too large: oscillations

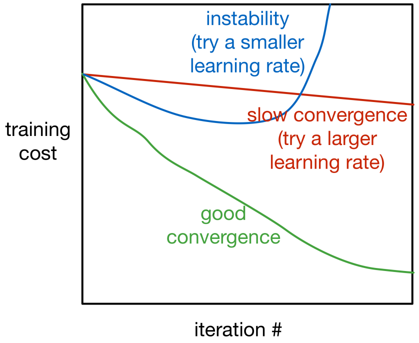


α much too large: instability

- Good values are typically between 0.001 and 0.1. You should do a grid search if you want good performance (i.e. try $0.1, 0.03, 0.01, \ldots$).

Training Curves

- To diagnose optimization problems, it's useful to look at the training cost: plot the training cost as a function of iteration.



- **Warning**: the training cost could be used to check whether the optimization problem reaches certain convergence. But
  - It does not tell whether we reach the global minimum or not
  - It does not tell anything on the performance of the fitted model

Gradient descent

Visualization:

`http://www.cs.toronto.edu/~guerzhoy/321/lec/W01/linear_regression.pdf#page=21`