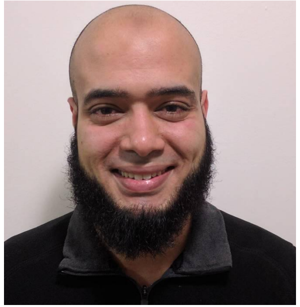
Machine Learning

Regularization - Lasso

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / MSc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



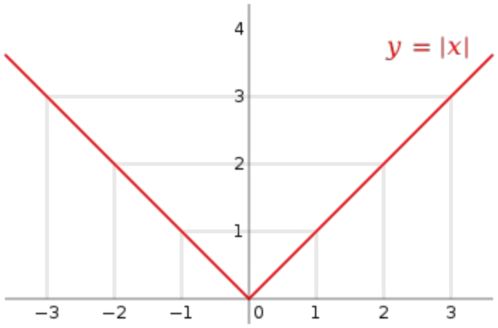
Question!

- Assume we have 1000 features for our N examples
- We trained a **regularized** linear regression and it works great on train/val/test
- However, on investigation, we found that 950 of the weights were all zeros
- What does this imply? What kind of actions we can take?

- The model doesn't depend on these features for its computations
- So we can consider **removing** them and trained with less features
- This is an example of how regularization can help with feature selection
- **Feature selection** is the process of reducing the number of input variables
- While Ridge regularization doesn't typically have this property, Lasso regularization is known for its ability to perform feature selection

Lasso Regression (L1 Regularization)

$$\underset{W}{\text{minimize}} \frac{1}{2N} \sum_{n=1}^{N} (y(X^n, W) - t^n)^2 + \frac{\lambda}{2} \sum_{i=1}^{M} |W_i|$$



- Another popular choice is L1 regularization
  - Instead of squaring each weight value, the absolute value is taken
  - This may seem similar to Ridge, but there are important implications to consider
- Is the absolute function a differentiable function?
- It is differentiable **everywhere** except for x = 0.

Lasso Regression vs Ridge

- Assume we have 2 weights w1 = -0.1 and w2 = 4
- Which one assigns a higher penalty: Lasso or Ridge?

- For w1: $(-0.1)^2 = 0.01$ while $|-0.1| = 0.1$ ⇒ Lasso assigns a higher penalty
  - For |weights| < 1, Lasso pushes them toward zero more strongly than Ridge
- For w2: $(4)^2 = 16$ while $|4| = 4$ ⇒ Ridge assigns a higher penalty
  - Side note: the squared L2 norm is sensitive to outliers in the error computation of the data (y-t)
    - If predict_price(home) is 1000 and the ground_truth(home) is 1001000, this would be considered an outlier

Lasso Regression vs Ridge

- Assume we have w1 = 10 and w2 = 20
- Does ridge **prioritize** the parameter equally? What about lasso?

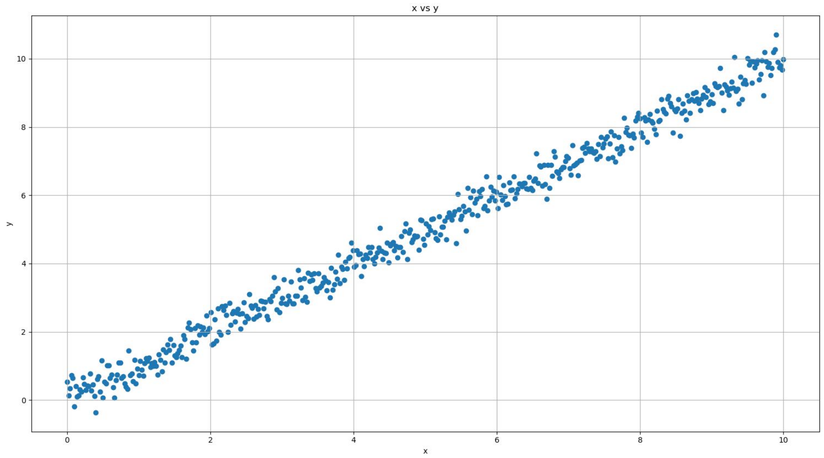
- For Ridge, the current cost is: $10^2 + 20^2 = 500$
  - If we reduce w1 by 1 the cost is: $9^2 + 20^2 = 481$ ⇒ a drop of 19
  - If we reduce w2 by 1 the cost is: $10^2 + 19^2 = 461$ ⇒ a drop of 39: a bigger effect
  - Ridge minimizes the weights **proportionally relative** to their **magnitude**
    - Larger weights lose **more**, smaller ones lose **less** ($dw^2 = w$)
    - For more mathematical treatment: *See Introduction to Statistical Learning: ch6*
- For Lasso, the current cost is: 10 + 20 = 30
  - If we reduce w1 by 1 the cost is: 9 + 20 = 29 ⇒ a drop of 1 ($dw = 1$)
  - If we reduce w2 by 1 the cost is: 10 + 19 = 29 ⇒ a drop of 1 also
  - Lasso does not favor a specific weight to reduce (treating 5 is the same as 500)

Lasso Regression: Sparsity

- With a higher λ, Lasso may push several weights toward being exactly zero
  - Why zero? Why with a higher λ? Later
- This implies the following:
- Lasso can help us do feature selection (cancel features of **zero weights**)
  - Why matter?
- Lasso is performing 2 things simultaneously: regularization and selection
  - The acronym Lasso stands for Least Absolute **Shrinkage and Selection** Operator

Lasso Regression vs Ridge Regression

- Let's generate simple simple data with 500 values for y = x + noise
  - Optimal values: c = 0 and m = 1
- Clearly we can fit the data with a single feature
- However, we will add 3 random (entirely useless) features for each value
  - Input data: [500 x 4] and output is 1 feature
- Let's explore Ridge vs Lasso
  - We can see if Lasso recognizes the useless features



Lasso Regression vs Ridge Regression

```
    lasso 7    : MSE 5.896 - intercept 4.20- Weights 0.159 0.000 0.000 0.000
    lasso 5    : MSE 3.028 - intercept 3.00- Weights 0.398 0.000 0.000 0.000
    lasso 3    : MSE 1.115 - intercept 1.81- Weights 0.637 0.000 0.000 0.000
    lasso 1    : MSE 0.159 - intercept 0.61- Weights 0.876 0.000 0.000 0.000
    lasso 0.1  : MSE 0.041 - intercept 0.08- Weights 0.984 0.000 0.000 0.000
Q2  lasso 0.01 : MSE 0.040 - intercept 0.02- Weights 0.994 0.000 0.001 0.000
Q1  lasso 0.001: MSE 0.039 - intercept 0.02- Weights 0.995 0.001 0.011 0.007

    Ridge 7    : MSE 0.040 - intercept 0.03- Weights 0.994 0.002 0.013 0.008
    Ridge 1    : MSE 0.039 - intercept 0.02- Weights 0.995 0.002 0.013 0.008
    Ridge 0.1  : MSE 0.039 - intercept 0.02- Weights 0.995 0.002 0.013 0.008
    Ridge 0.01 : MSE 0.039 - intercept 0.02- Weights 0.995 0.002 0.013 0.008
    Ridge 0.001: MSE 0.039 - intercept 0.02- Weights 0.995 0.002 0.013 0.008
```

- Q1) Which λ will cross validation choose for the Lasso method?
- Q2) Which λ will inform you about the useless features while having fair error?

Lasso Regression: Careful with Cross-Validation

- There is a blocking problem when using cross-validation for lasso λ
- The last 2 questions reveal 2 different goals:
  - Minimum prediction error (max accuracy) vs most-removed features
- The goal of cross-validation is to find the minimum prediction error
- The goal of feature selection is to find the λ that eliminates the most unnecessary features
- No guarantee their lambda matches!
  - In practice, typically λ_cross_prediction < λ_most_useless_features
    - Recall, higher λ cancels more features
  - This inconsistency is mathematically [proven](#) in the paper:
    - High-dimensional graphs and variable selection with the Lasso By Nicolai Meinshausen and Peter Buhlmann, 2006
- So, overall we don't use cross validation to determine λ if we want to select important features

Lasso Regression: Improving the Performance

- Sometimes people try to improve their models further
- Here are 2 possible approaches:
- Two-stage procedure
  - 1) Using Lasso with several lambda values to perform feature selection (without cross validation)
  - 2) Applying another model (such as Lasso, Ridge, or Linear Regression)
- Use Elastic net regularization
  - It linearly combines the L1 and L2 penalties (with two hyperparameters)
    - Deep learning solutions typically have multiple linearly combined error functions

$$\underset{W}{\text{minimize}} \frac{1}{2N} \sum_{n=1}^{N} (y(X^n, W) - t^n)^2 + \frac{\lambda 1}{2} \sum_{i=1}^{M} |W_i| + \frac{\lambda 2}{2} \sum_{i=1}^{M} W_i^2$$

Lasso Regression: Why Sparsity

- For **mathematical** details:
  - ‘Machine Learning A Probabilistic Perspective’ book ch 13.3
  - ‘An Introduction to Statistical Learning’ book ch 6.2.2 (inventor of Lasso)
- First, we re-formulate the penalty as a constraint

$$\underset{W}{\text{minimize}} \frac{1}{2N} \sum_{n=1}^{N} (y(X^n, W) - t^n)^2 \quad \text{subject to} \quad \frac{\lambda}{2} \sum_{i=1}^{M} |W_i| \leq s$$

- Now we have 2 separate contours: cost function and constraint
- **Theory of constrained optimization**: solution to the constrained optimization lies at the **intersection** between the contours of the two functions

Lasso Regression: Why Sparsity

- The cost function is represented by the red ellipses in the illustration
- The shadowed region represents the contours of the penalty constraint
- Lasso constraint has **corners** at each of the **axes**, and so the ellipse often intersects the constraint region at an axis
- Ridge regression has a circular constraint without any sharp points

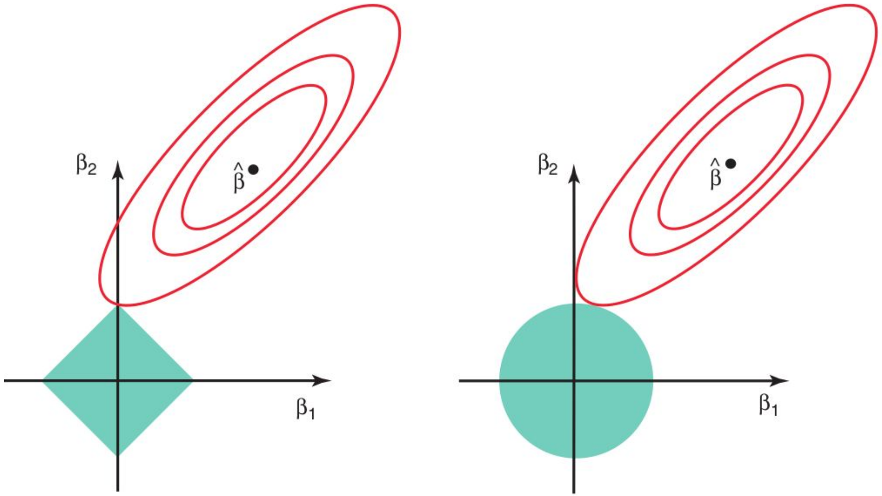
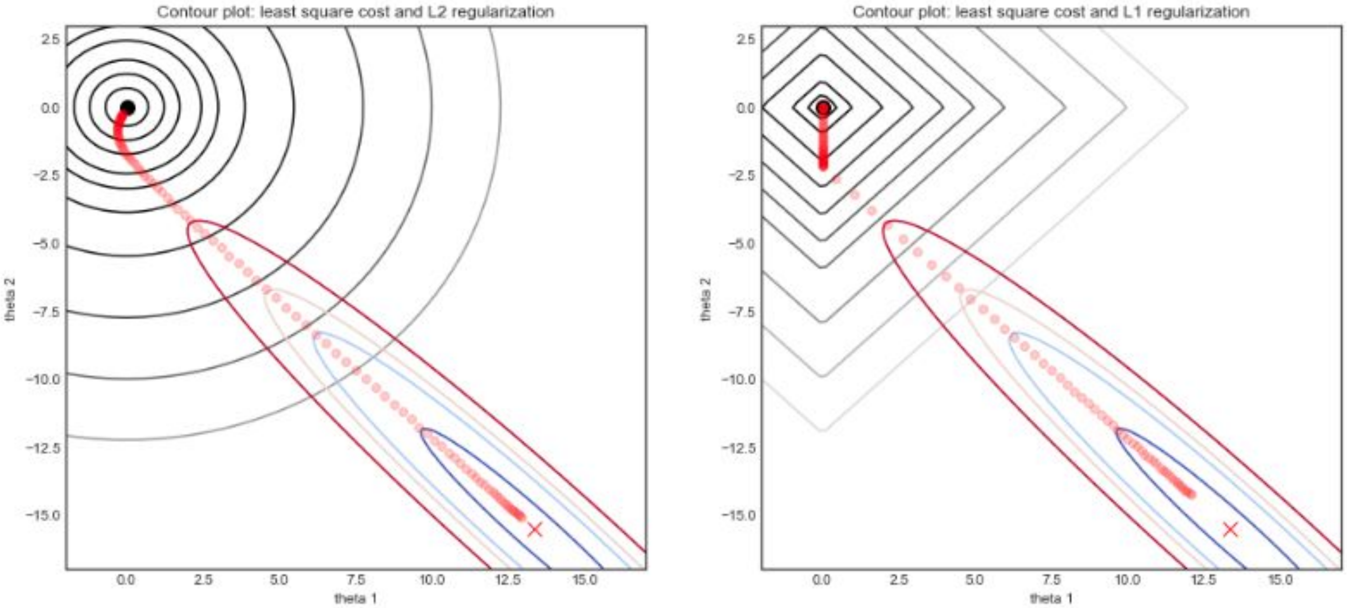


FIGURE 6.7. *Contours of the error and constraint functions for the lasso (left) and ridge regression (right). The solid blue areas are the constraint regions, $|\beta_1| + |\beta_2| \leq s$ and $\beta_1^2 + \beta_2^2 \leq s$, while the red ellipses are the contours of the RSS.*

Lasso Regression: Why Sparsity

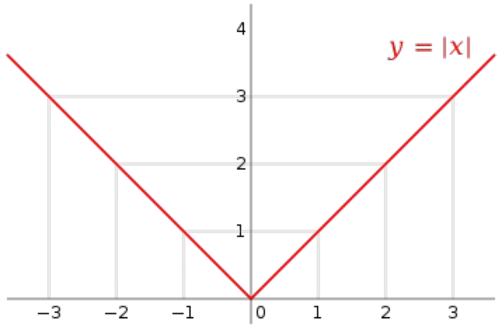
- Why does a large λ likely result in more sparsity than a small λ?
    - With large λ, the weights shrink which makes use closer to the narrower contours and this increase the chances of solutions at the corners of the diamond



Img src

Lasso Regression: Differentiability

- The absolute function is a convex function
  - However, it is NOT differentiable at 0 (a sharp corner)
  - Hence:
    - Our cost function does not have a closed form
    - We can't apply a gradient descent
- About differentiability
  - The absolute value function is a piecewise function
  - When x > 0, we know derivative of **f(x) = x is 1**
  - When x < 0, we know derivative of **f(x) = -x is -1**
  - However, there is no derivative for f(x - 0)
    - The neighbour gradients jumps from -1 to 1
  - If there is no derivative, then we **can't update** the weights in the **multivariate** case



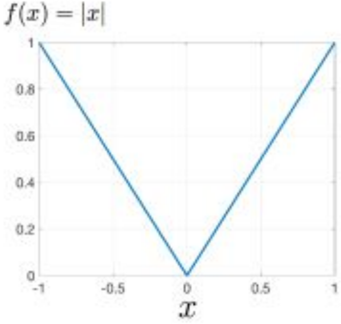
$$f(x)=\begin{cases} x; & x\geq 0 \\ -x; & x<0 \end{cases}$$

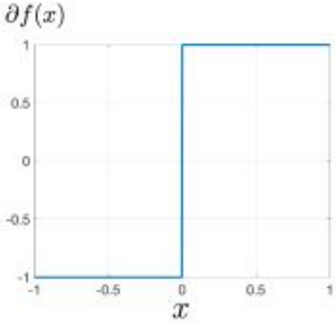
Lasso Regression: Implementation

- There are several techniques
  - Subgradient method
  - Coordinate descent
    - Update only a single weight at each step
      - Recall how Gradient descent updates all of the weights at once
    - SKLearn uses coordinate descent method to implement Lasso optimization
    - Refer to resources / internet

Implementing Lasso: Subgradient method

- We can use it for **non-differentiable** objective functions
- It is not a descent method; the function value can increase
- Optionally, refer to the math details in the resources section
- We compute some kind of gradient for the piecewise function





$$f(x) = |x| \qquad \partial f(x) = \begin{cases} \{-1\}, & \text{if } x < 0 \\ [-1, 1], & \text{if } x = 0 \\ \{1\}, & \text{if } x > 0 \end{cases}$$

- The final derivatives for absolute value function are simple
- For f(0), any value in the range [-1, 1] is accepted
  - We typically use 1
- Then it is all about the sign of f(x): positive or negative
  - **df(x) = sign(x)**
  - Multiplied by λ for lasso
    - **df(x) = sign(x)** * λ

Img src

SKlearn library

- Linear regression is a **simple** and old model. Many variates were created with deep mathematical analysis around them
  - In practice, we might try them and see the best

```
def evalaute(x, t, model, name):
    model.fit(x, t)
    pred_t = model.predict(x)
    err = mean_squared_error(t, pred_t)
    w = ' '.join([f'{w:.3f}' for w in model.coef_])
    print(f'{name}: MSE {err:.3f} - intercept {model.intercept_

if __name__ == '__main__':
    x, t = get_linear_data()

    evalaute(x, t, Lasso(alpha=0.01), 'lasso 0.01 ')
    evalaute(x, t, LassoLars(alpha=0.01, normalize=False), 'Las
    evalaute(x, t, ElasticNet(alpha=0.01), 'ElasticNet 0.01 ')
    evalaute(x, t, Ridge(alpha=1), 'Ridge 1     ')
    evalaute(x, t, BayesianRidge(), 'BayesianRidge')
    evalaute(x, t, LinearRegression(), 'LinearRegression')
```

Importance of scaling

- Very Large input features require tiny weights. Very small input features require large weights. Scale/standardize your data!

```
x, t = get_linear_data()

evalaute(x, t, Lasso(alpha=0.01), 'lasso 0.01 ')
evalaute(x, t, Ridge(alpha=1), 'Ridge 1    ')
evalaute(x, t, LinearRegression(), 'LinearRegression')
print()
x /= 10 ** 5
# Tiny features require big weights.
# But weights are penalized, and regualrizer will set to 0
evalaute(x, t, Lasso(alpha=0.01), 'lasso 0.01 ')
evalaute(x, t, Ridge(alpha=1), 'Ridge 1    ')
evalaute(x, t, LinearRegression(), 'LinearRegression')
```

Importance of scaling

```
lasso 0.01 : MSE 0.040 - intercept 0.02- Weights 0.994 0.000 0.001 0.000
Ridge 1    : MSE 0.039 - intercept 0.02- Weights 0.995 0.002 0.013 0.008
LinearRegression: MSE 0.039 - intercept 0.02- Weights 0.995 0.002 0.013 0.008

lasso 0.01 : MSE 8.334 - intercept 4.99- Weights 0.000 0.000 0.000 0.000
Ridge 1    : MSE 8.334 - intercept 4.99- Weights 0.042 0.000 0.001 0.000
LinearRegression: MSE 0.039 - intercept 0.02- Weights 99540.085 233.673 1257.960 813.894
```

Lasso Regression: In Practice

- We typically just try Ridge, Lasso and Elastic Net and see which one works better (**empirical** approach). Start with Ridge
- Lasso *might* works well if there are small number of significant features, while Ridge is the opposite
- Don't count on lasso features selection. Do your own investigations
  - If features D > examples N, lasso selects at most N features
  - Lasso might drop significant features that generate illogical models
  - Lasso might select only one feature from a group of **relevant** features
  - If the data is perturbed (changed) slightly, we might get different solutions
  - Then which features are the important ones?!
- Lasso is slower than ridge which is differentiable and has a closed form

Feature Selection

- Feature selection is selecting a subset of relevant features that are informative and discriminative
- In theory it can come with many advantages:
  - Dimensionality Reduction ⇒ less complex model ⇒ less overfitting
    - More features requires more weights which make the model more complex
  - Interpretability: easier to understand the factors that influence the model's predictions

Feature Selection Techniques

- Univariate Feature Selection: Investigate each feature independently (e.g. chi-square test)
- **Correlation**-based Methods: correlation between features and the target variable, as well as the intercorrelation between features (very common)
- **Lasso** (discard zero weights)
- Tree-based Methods (e.g. random forests) provides features importance
- Dimensionality reduction technique (e.g. PCA and **Autoencoders**)
- Tip: in practice, try using all of your data and run simple models first

Relevant Materials

- Lasso: Article, StatQuest, Proximal gradient, Article
- Subgradients: video, article, article, Probabilistic Perspective 13.3.2
- Coordinate descent: slides
- Deriving lasso coordinate descent: article / video coursera / code
- **Regularization path**: **plot** weights vs lambdas: article - code
- Soft thresholding: link, Probabilistic Perspective:ch:13.3.2

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."