Regularization for Sparsity

Kazi Aminul Islam

Department of Computer Science

Kennesaw State University

What is Regularization?

- A technique to minimize the complexity of the model by penalizing the loss function to solve overfitting.
- The above definition involves the following characteristics:
  - Make the parameters (weights) as small as possible.
  - Minimize the complexity: remove terms by in objective functions, i.e., reduce dimensions of data
  - Penalize parameters in the objective function: by setting a regularizer (λ)
  - Solve the overfitting problem: add a bias intentionally. So machine would learn too much from the training data (e.g., noise), which results in poor performance in testing data.

Sparsity

- In Linear algebra, a sparse matrix is one that contains lot of zeros.

Sparse Matrix

$$\begin{bmatrix} 1.1 & 0 & 0 & 0 & 0 & 0 & 0.5 \\ 0 & 1.9 & 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 2.6 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 7.8 & 0.6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.5 & 2.7 & 0 & 0 \\ 1.6 & 0 & 0 & 0 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.9 & 1.7 \end{bmatrix}$$

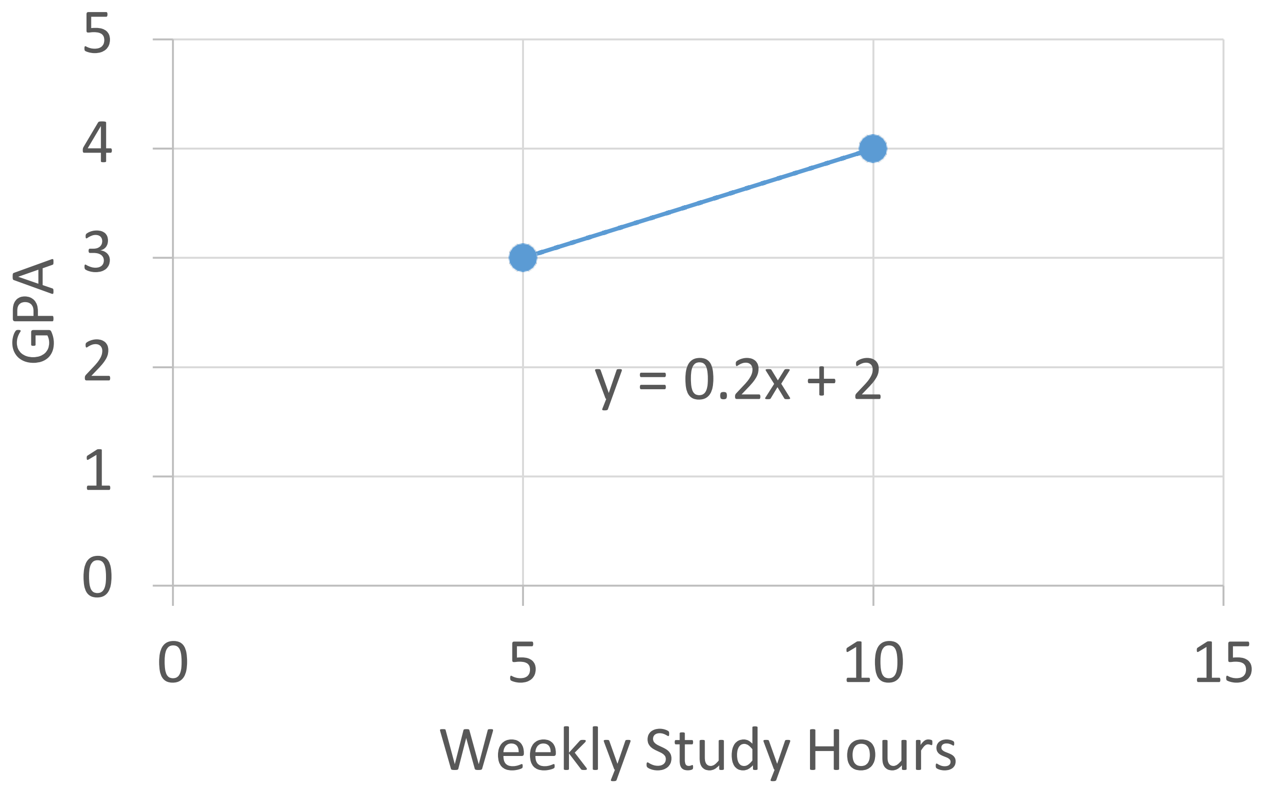
ComputerHope.com

- In linear regression, we want to turn $\hat{\beta}$ into sparse a matrix (vector) for a number of reasons:
  - Not all features are equally corrected, or unrelated at all.
  - Zero out some $\hat{\beta}_i$ means remove them from the model. So simplify the model.
  - Solve overfitting problem

Linear Regression using Least Squares

- Suppose we have a training data set,

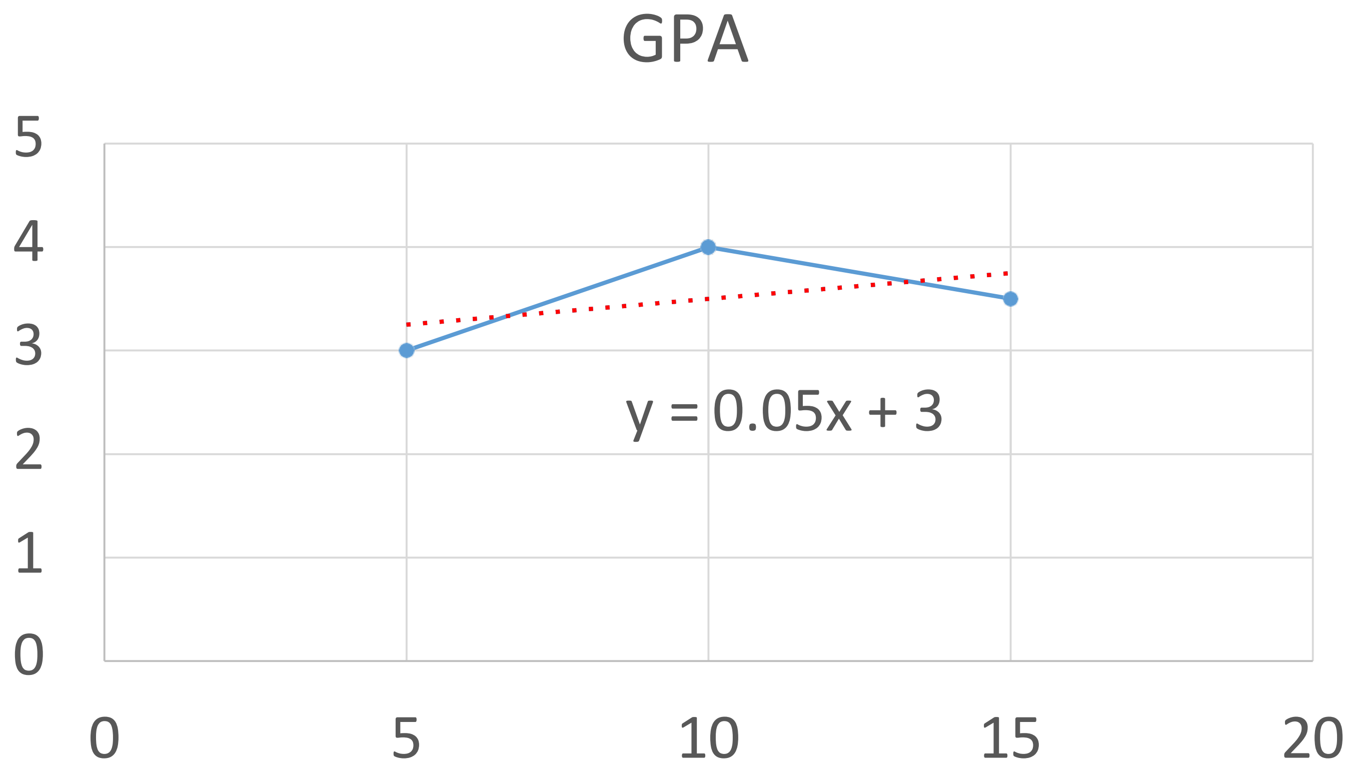
Weekly Study Hours	GPA
5	3
10	4



Prediction Performance

- What if we have a test data (15, 3.5), e.g.,

Weekly Study Hours	GPA
5	3
10	4
15	3.5



Overfitting

- The first regression line trained from 2 data points is overfitting to the training data.
- So the performance is obvious very bad when predicting the test data.
- To overcome this problem, we add a penalty to the objective function.
- This mechanism is called regularization.

Ridge Regression (L2)

- We add a regularization term to the objective function in linear regression model as follows:
- $f(\beta_0, \beta_1) = \sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2 + \lambda\beta_1^2$
- Since we are minimizing the objective function, the added term $\lambda\beta_1^2$ tends to get β_1 smaller, i.e., the slope of the regression line.
- What's next is estimate the parameters β_0, β_1 again using differentiation.

Find the Parameters β_0, β_1

$f(\beta_0, \beta_1) = \sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2 + \lambda\beta_1^2$

$\frac{\partial f(\beta_0,\beta_1)}{\partial \beta_0} = -2\sum(y_i - \beta_0 - \beta_1 x_i) = 0$

$n\bar{y} - n\beta_0 - n\beta_1\bar{x} = 0 \Rightarrow \beta_0 = \bar{y} - \beta_1\bar{x}$

Now partial differentiate on β_1

$\frac{\partial f(\beta_0,\beta_1)}{\partial \beta_1} = -2\sum(x_i(y_i - \beta_0 - \beta_1 x_i)) + 2\lambda\beta_1 = 0$

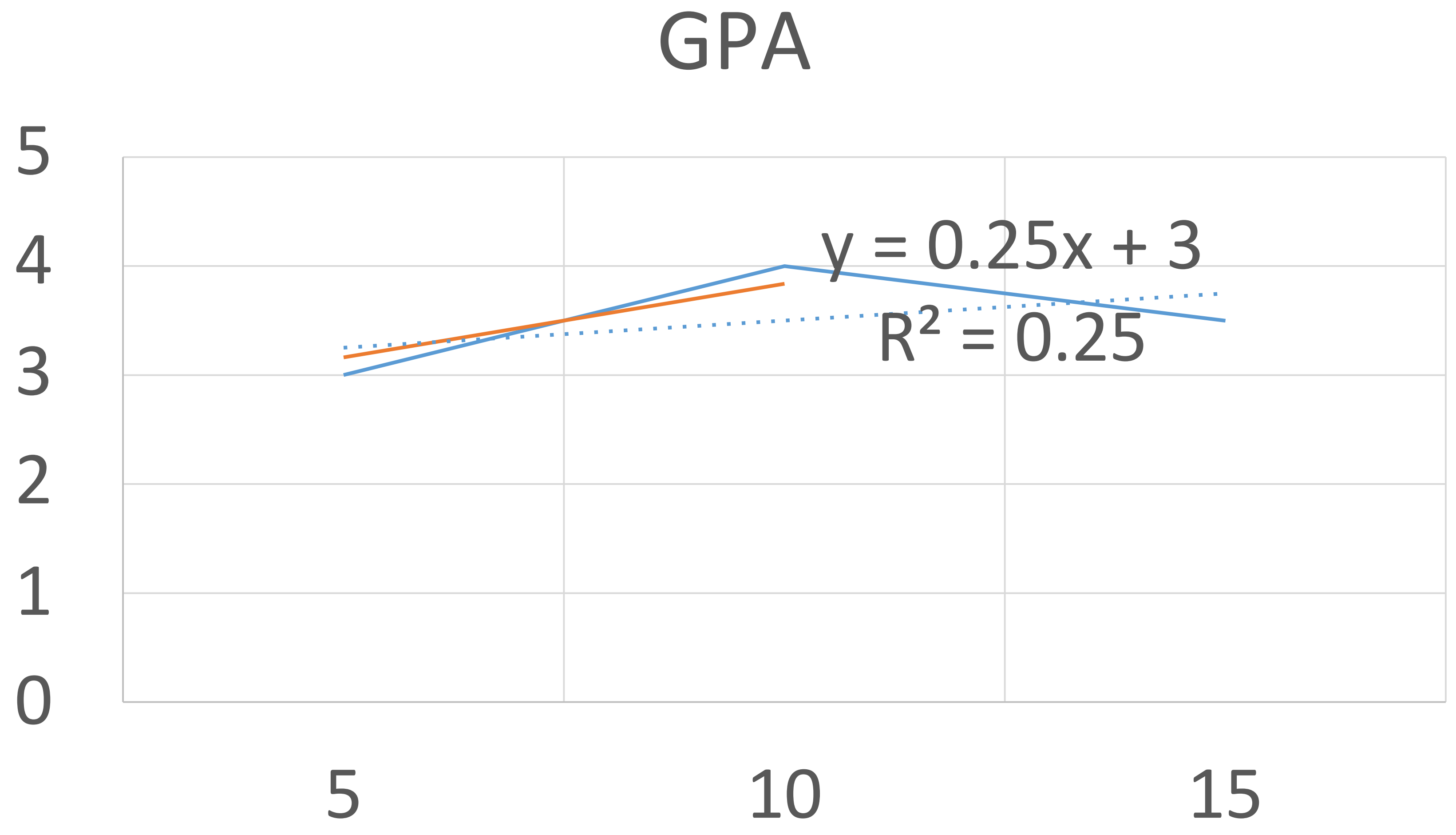
$-2\sum(x_i(y_i - \bar{y} + \beta_1\bar{x} - \beta_1 x_i)) + 2\lambda\beta_1 = -\sum(x_i(y_i - \bar{y}) + \beta_1\sum x_i(x_i - \bar{x}) + \lambda\beta_1 = 0$

$\beta_1(\sum x_i(x_i - \bar{x}) + \lambda) = \sum x_i(y_i - \bar{y})$

$$\beta_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})(x_i - \bar{x}) + \lambda} = \frac{SSxy}{SSxx + \lambda/(n-1)}$$

L2 Regularization

Let $\lambda = 12$. The red line shows that slope is 0.25 and leans toward the regression line with all three data points.



Regularization

- Many standard machine learning methods can be formulated as a convex optimization problem, i.e., minimize f(w), where w is weights in d dimension.
- $f(w) = \lambda R(w) + \frac{1}{n} \sigma_{i=1}^{n} L(w; x_i, y_i)$ where $x_i \in R^d$ are the training data samples and $y_i \in R$ are their corresponding labels.
- $R(w)$ is a regularizer that controls the complexity of the model.
- The purpose of the regularizer is to encourage simple models and avoid overfitting.

L0 Regularization

- An idea (L0) would be to try and create a regularization term that penalizes the count of non-zero coefficient values in a model.
- Increasing this count would only be justified if there was a sufficient gain in the model’s ability to fit the data.
- Unfortunately, it would turn our convex optimization problem into a non-convex optimization problem (NP-hard)
- L0 regularization is not something we can use effectively in practice.

L2 Regularization

- Adding an appropriately chosen regularization term to encode the previous idea into optimization problem done at training time.
- It penalizes the weight by a percentage.
- $\lambda R(w) = \frac{\lambda}{2}\left||w|\right|_2^2, \lambda \geq 0, \left||w|\right|_2^2 = \sigma w_i^2$ where $\lambda \geq 0$ and $w_i = \beta_i, \forall\, i > 0$.
- L2 regularization encourages weights to be small but does not force them to exactly 0.0.
- L2 regularization aka Ridge regression.

L2 Multiple Regression

From multiple regression, we have $\mathrm{Y} = X\beta + \epsilon$ and $\widehat{\mathrm{Y}} = X\hat{\beta}$.

$SSE = \left(Y - X\hat{\beta}\right)^T\left(Y - X\hat{\beta}\right), \hat{\beta} = \left(X^TX\right)^{-1}X^TY$

With L2, we are minimizing $SSE + \lambda\sum\hat{\beta}_i^2 = \left(Y - X\hat{\beta}\right)^T\left(Y - X\hat{\beta}\right) + \lambda\hat{\beta}^{\mathrm{T}}\hat{\beta}$

$$\frac{\partial\left(SSE+\lambda\left|\left|\widehat{\beta}\right|\right|_2^2\right.}{\partial\widehat{\beta}} = -2Y^TX + 2\hat{\beta}^TX^TX + 2\lambda\hat{\beta}^T = 0$$

$$\hat{\beta}^T\left(X^TX + \lambda I\right) = Y^TX \Rightarrow \hat{\beta}^T = Y^TX\left(X^TX + \lambda I\right)^{-1}$$

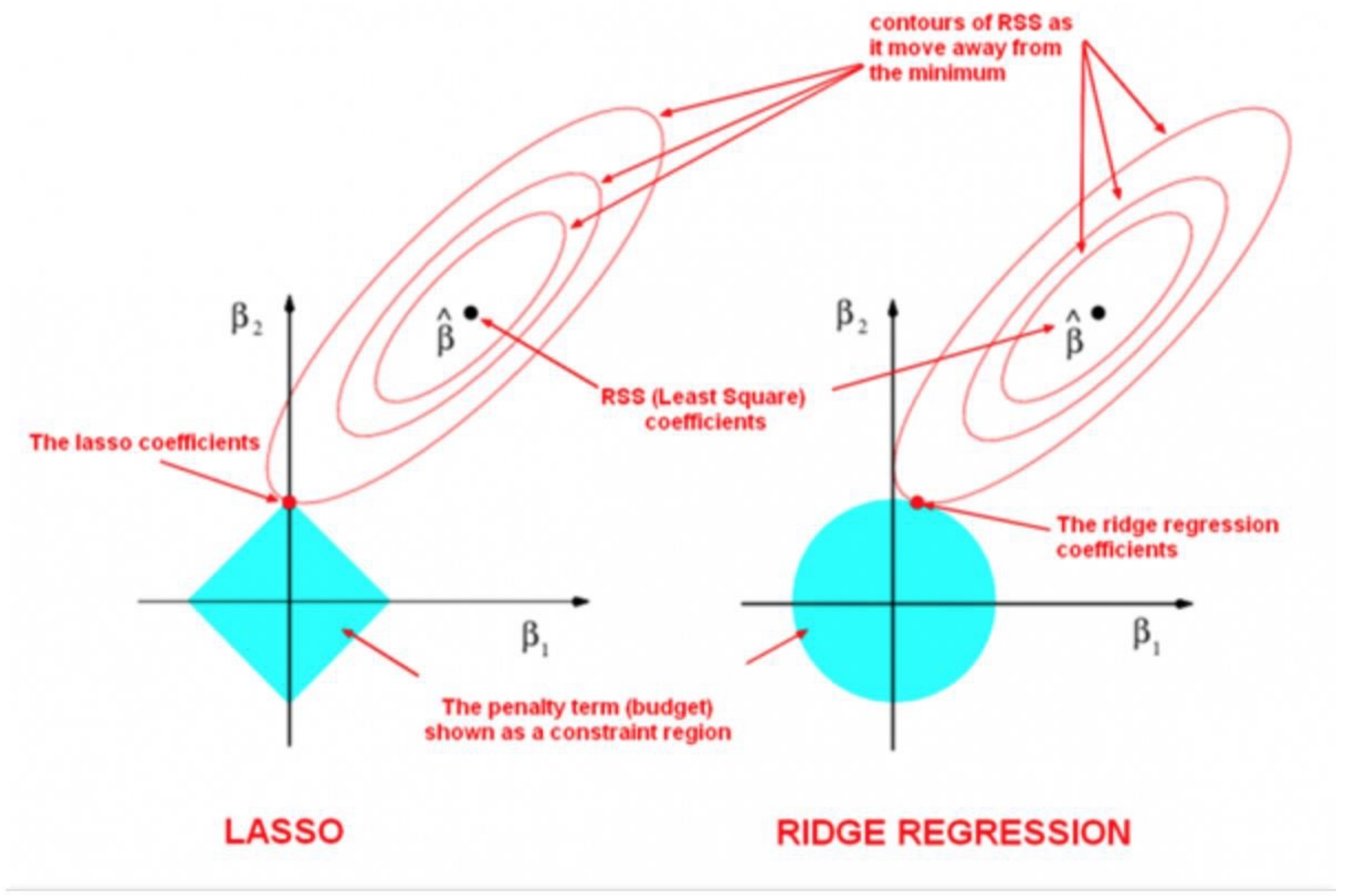
$$\hat{\beta} = \left(X^TX + \lambda I\right)^{-1}X^TY$$

L1 Regularization

- L1 penalizes sum of abs(weights)
- $\lambda R(w) = \lambda||w||_1$ where $||w||_1 = \sigma\,|w_i|, \forall i > 0$ and $w_i = \beta_i, \forall\ i > 0$
- Convex problem
- L1 regularization serves as an approximation to L0, but has the advantage of being convex and thus efficient to compute.
- Therefore, we can use L1 regularization to encourage many of the uninformative coefficients in our model to be exactly 0, and thus reap RAM savings at inference time.
- Encourage sparsity unlike L2
- L1 aka Least Absolute Shrinkage and Selection Operator (LASSO) regression.

Graphical Representation for L1 and L2

- For L1, the penalty can be written $||\beta||_1 \leq c \Rightarrow |\beta_1| + |\beta_2| \leq c$
- For L2, the penalty can be written $||\beta||_2^2 \leq c \Rightarrow \beta_1^2 + \beta_2^2 \leq c$


contours of RSS as it move away from the minimum
β₂
β̂
RSS (Least Square) coefficients
The lasso coefficients
β₁
The penalty term (budget) shown as a constraint region
The ridge regression coefficients
LASSO
RIDGE REGRESSION

L1 vs. L2 Regularization

- L2 and L1 penalize weights differently:
  - L2 penalizes weight
  - L1 penalizes |weight|
- L2 and L1 have different derivatives
  - The derivative of L2 is 2*weight
  - The derivative of L1 is k (a constant whose value is independent of weight)
- L2 removes x% of the weigh every time. It does not drive weights to zero.
- L1 subtracts some constant from the weight every time. L1 will set the weight to zero.

Elastic Net Regularization

- We may combine L1 and L2 in the penalty term.
- $\lambda R(w) = \alpha\lambda_1||w||_1 + (1-\alpha)\frac{\lambda_2}{2}||w||_2^2, \alpha \in [0,1], \lambda_1, \lambda_2 \geq 0$
- By setting α properly, elastic net contains both L1 and L2 regularization as special cases.
- For example, if a linear regression model is trained with the elastic net parameter α set to 1, it is equivalent to a Lasso model.
- On the other hand, if α is set to 0, the trained model reduces to a ridge regression model.

Check for Understanding

Which of the following statements are true?

1. L1 regularization will encourage many of the non-informative weights to be nearly (but not exactly) 0.0.
2. L1 regularization may cause informative features to get a weigh of exactly 0.0.
3. L1 regularization will encourage most of the non-informative weights to be exactly 0.0.

Check for Understanding

Which type of regularization will produce the smaller model?

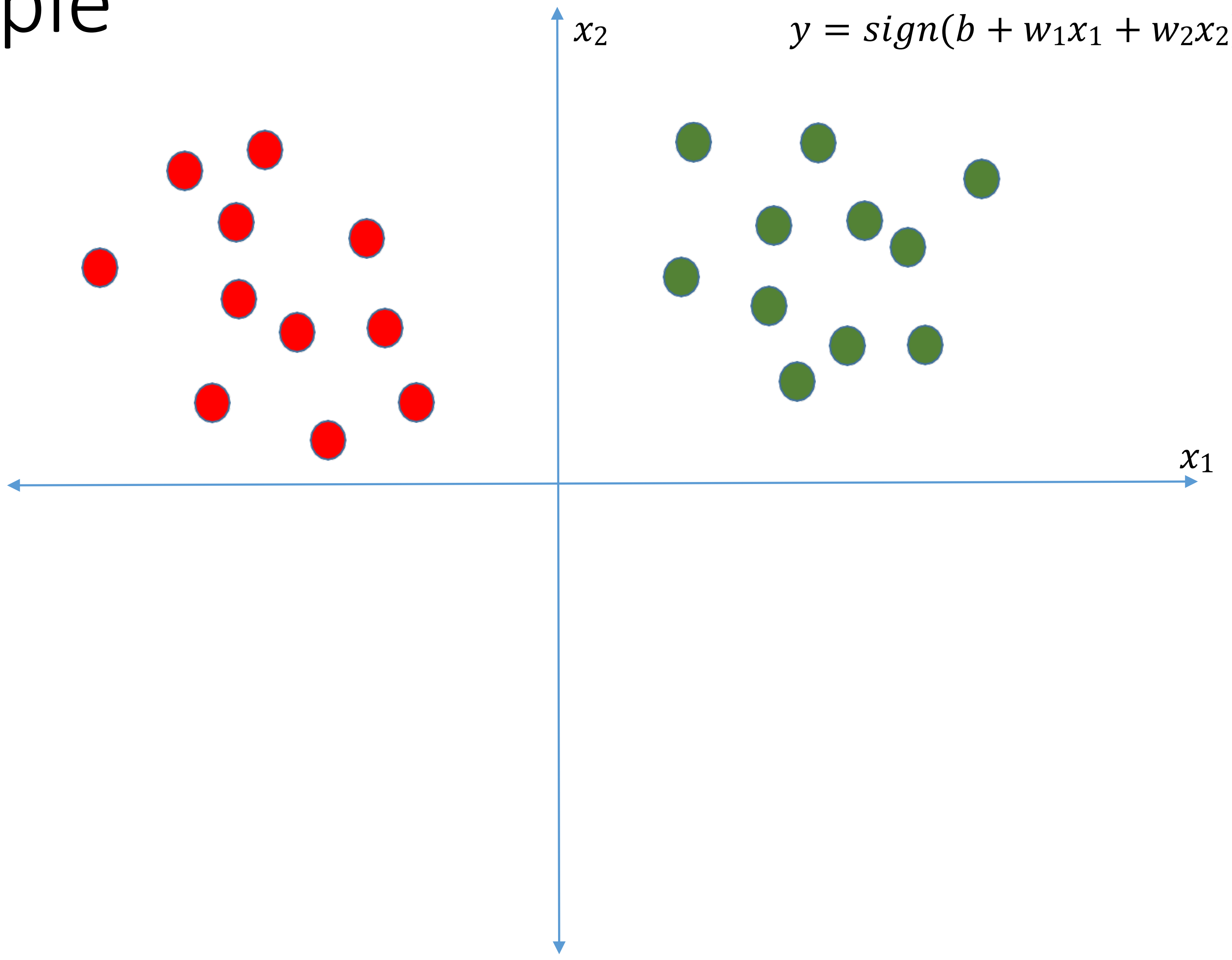
1. L2 regularization
2. L1 regularization

Application of Regularization - Feature Crosses/Extractions

- A feature cross is a synthetic feature formed by multiplying (crossing) two or more features.
- Crossing combinations of features can provide predictive abilities beyond what those features can provide individually.
- A cross feature E derived from features A, B, C, and D would be E=AxBxCxDxE
- When Features represent Boolean values, the resulting crosses can be extremely sparse.

Example

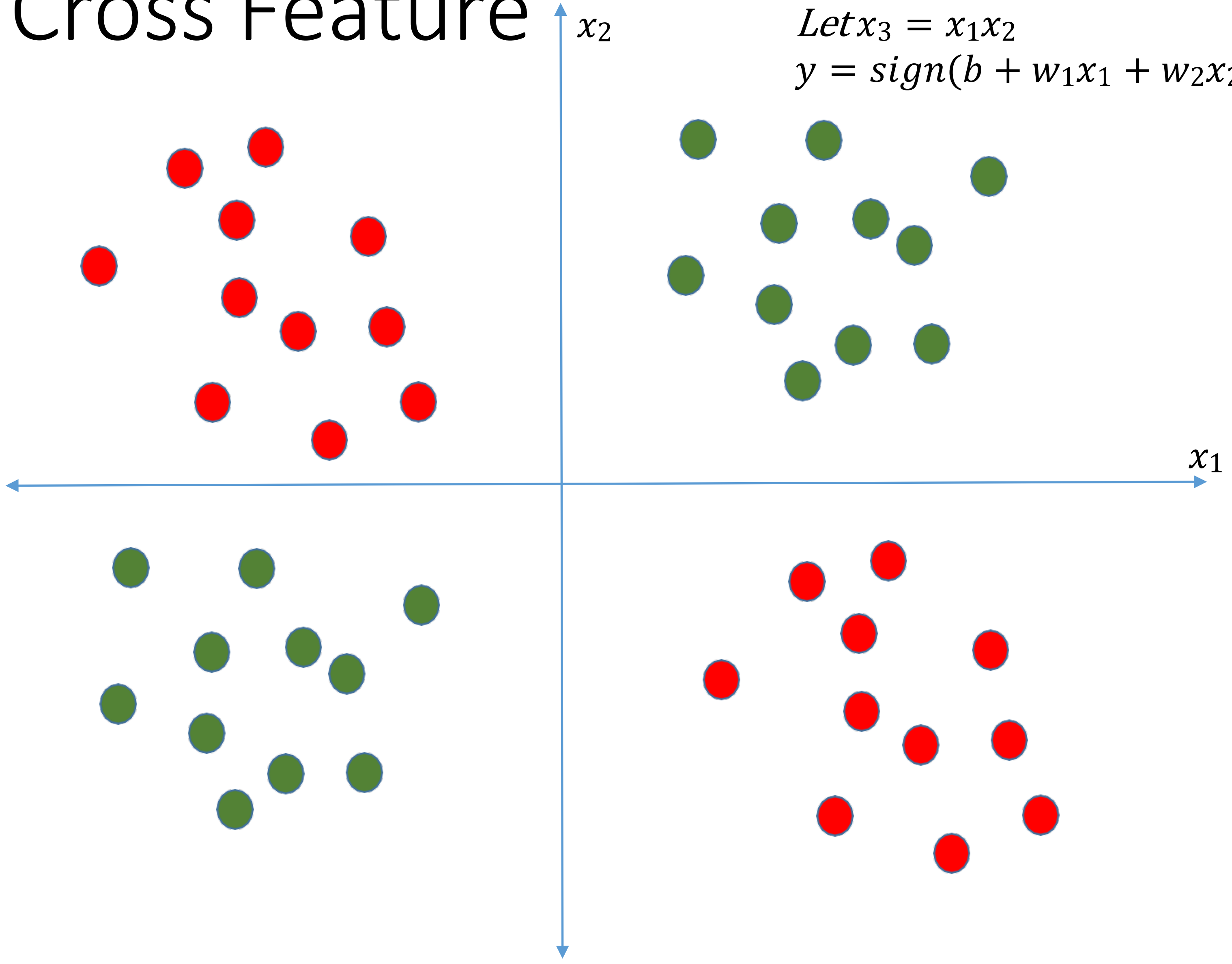
$y = sign(b + w_1 x_1 + w_2 x_2)$



Add a Cross Feature

$Let\, x_3 = x_1 x_2$

$y = sign(b + w_1 x_1 + w_2 x_2 + w_3 x_3)$



Why Feature Crosses

- Allow nonlinear learners using linear models
- Such learners scale well to massive data
- Without feature crosses, the expressivity of these models would be limited
- Using feature crosses and massive data is one efficient strategy for learning highly complex models.

Sparse Vectors

- Sparse vectors often contain many dimensions
- Creating a feature cross results in even more dimensions.
- Given such high dimensional feature vectors, model size may become huge and require huge amounts of memory.
- In high dimensional sparse vector, it would be nice to encourage weights to drop to exactly 0 where possible.
- A weight of exactly 0 essentially removes the corresponding feature from the model. Zeroing out features will save memory and may reduce noise in the model.

Example – Housing Data

- Consider a housing dataset that covers not just Georgia but the entire globe.
- Bucketing global latitude at the minute level (60 minutes per degree) gives about 10,000 dimensions as in a sparse encoding.
- Global longitude at the minute level gives about 20,000 dimensions.
- A feature cross of these two features would result in roughly 200,000,000 dimensions.
- Many of the those 200,000,000 dimensions represent areas of such limited residence (e.g., middle of ocean) that it would be difficult to use that data to generalize effectively.

Cont.

- It would be silly to pay the RAM cost of storing these unneeded dimensions.
- It would be good to encourage the weights for the meaningless dimensions to drop to exactly 0, which allow us to avoid paying for the storage cost of these model coefficients at inference time.

Sparse Feature Crosses

- Sparse feature crosses may significantly increase feature space.
- Passible issues:
  - Model size (RAM) may become huge
  - “Noise” coefficients causes overfitting