Artificial Intelligence and Machine Learning

Regularization

Slides from: https://harvard-iacs.github.io/2019-CS109A/lectures/lecture8/presentation/Lecture8a_Regularization.pptx

Regularization: An Overview

The idea of regularization revolves around modifying the loss function L; in particular, we add a regularization term that penalizes some specified properties of the model parameters

$$L_{reg}(\beta) = L(\beta) + \lambda R(\beta),$$

where λ is a scalar that gives the weight (or importance) of the regularization term.

Fitting the model using the modified loss function L_{reg} would result in model parameters with desirable properties (specified by R).

LASSO Regression

Since we wish to discourage extreme values in model parameter, we need to choose a regularization term that penalizes parameter magnitudes. For our loss function, we will again use MSE.

Together our regularized loss function is:

$$L_{LASSO}(\beta) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \boldsymbol{\beta}^\top \boldsymbol{x}_i|^2 + \lambda \sum_{j=1}^{J}|\beta_j|.$$

Note that $\sum_{j=1}^{J}|\beta_j|$ is the $\boldsymbol{l_1}$ norm of the vector $\boldsymbol{\beta}$

$$\sum_{j=1}^{J}|\beta_j| = \|\boldsymbol{\beta}\|_1$$

Ridge Regression

Alternatively, we can choose a regularization term that penalizes the squares of the parameter magnitudes. Then, our regularized loss function is:

$$L_{Ridge}(\beta) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \boldsymbol{\beta}^\top \boldsymbol{x}_i|^2 + \lambda \sum_{j=1}^{J} \beta_j^2.$$

Note that $\sum_{j=1}^{J} |\beta_j|^2$ is the square of the ***l*₂** norm of the vector $\boldsymbol{\beta}$

$$\sum_{j=1}^{J} \beta_j^2 = \|\boldsymbol{\beta}\|_2^2$$

Choosing λ

In both ridge and LASSO regression, we see that the larger our choice of the **regularization parameter** λ, the more heavily we penalize large values in $\boldsymbol{\beta}$,

- If λ is close to zero, we recover the MSE, i.e. ridge and LASSO regression is just ordinary regression.
- If λ is sufficiently large, the MSE term in the regularized loss function will be insignificant and the regularization term will force $\boldsymbol{\beta}_{\text{ridge}}$ and $\boldsymbol{\beta}_{\text{LASSO}}$ to be close to zero.

To avoid ad-hoc choices, we should select λ using cross-validation.

Ridge, LASSO - Computational complexity

Solution to ridge regression:

$$\beta = (X^T X + \lambda I)^{-1} X^T Y$$

The solution to the LASSO regression:

LASSO has no conventional analytical solution, as the L1 norm has no derivative at 0. We can, however, use the concept of subdifferential or subgradient to find a manageable expression. See a-sec2 for details.

Regularization Parameter with a Validation Seet

The solution of the Ridge/Lasso regression involves three steps:

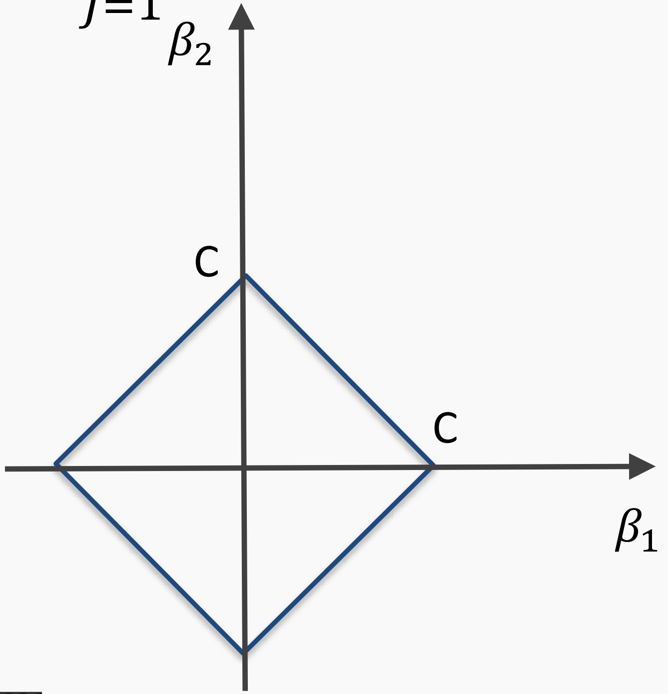
- Select λ
- Find the minimum of the ridge/Lasso regression loss function (using the formula for ridge) and record the *MSE* **on the validation set**.
- Find the λ that gives the smallest *MSE*

The Geometry of Regularization (LASSO)

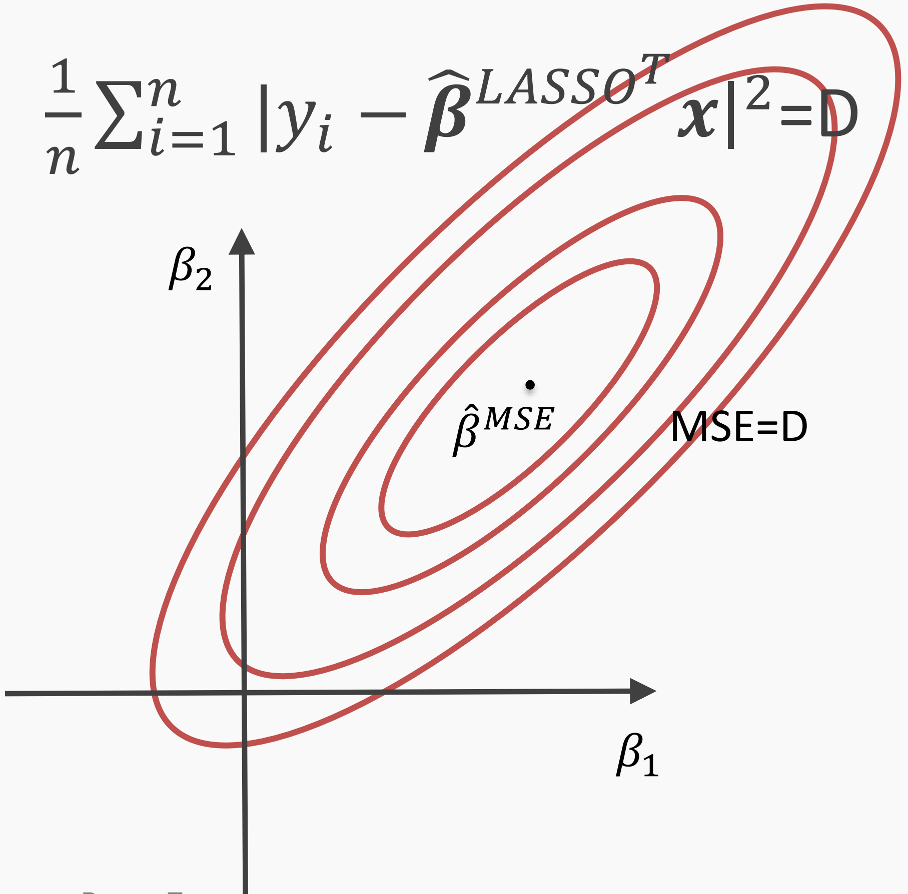
$$L_{LASSO}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} \left|y_i - \boldsymbol{\beta}^T\boldsymbol{x}\right|^2 + \lambda \sum_{j=1}^{J} |\beta_j|$$

$$\widehat{\boldsymbol{\beta}}^{LASSO} = \operatorname{argmin} L_{LASSO}(\boldsymbol{\beta})$$

$$\lambda \sum_{j=1}^{J} \left| \hat{\beta}_j^{LASSO} \right| = C$$



$$\frac{1}{n}\sum_{i=1}^{n} \left|y_i - \widehat{\boldsymbol{\beta}}^{LASSO^T}\boldsymbol{x}\right|^2 = D$$



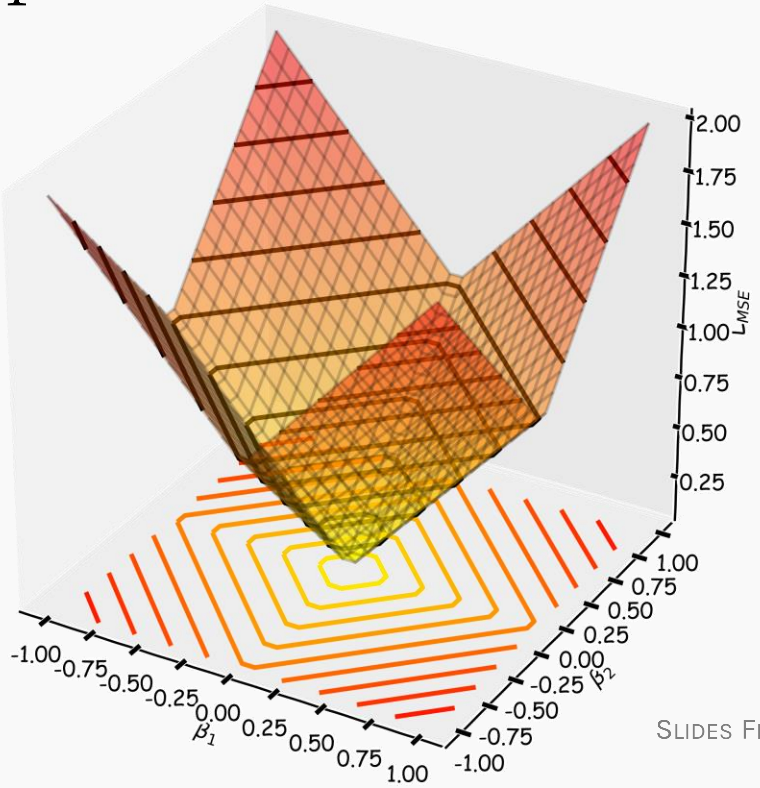
The Geometry of Regularization (LASSO)

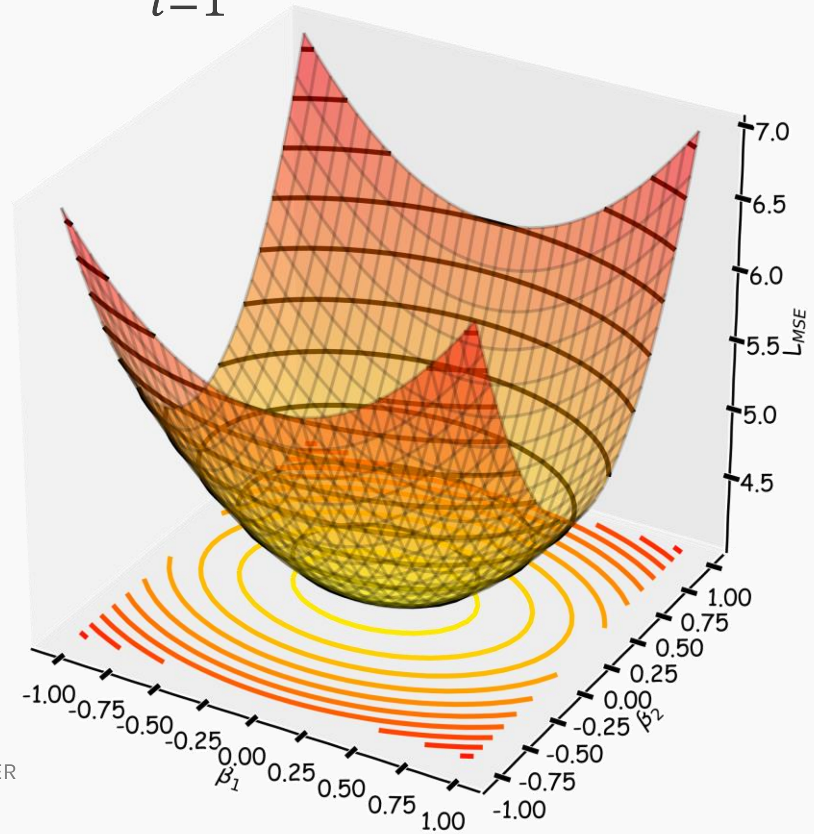
$$L_{LASSO}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \boldsymbol{\beta}^T\boldsymbol{x}\right|^2 + \lambda\sum_{j=1}^{J}|\beta_j|$$

$$\widehat{\boldsymbol{\beta}}^{LASSO} = \operatorname{argmin} L_{LASSO}(\boldsymbol{\beta})$$

$$L_1 = \lambda\sum_{j=1}^{J}\left|\hat{\beta}_j^{LASSO}\right|$$

$$L_{MSE}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \boldsymbol{\beta}^T\boldsymbol{x}\right|^2$$



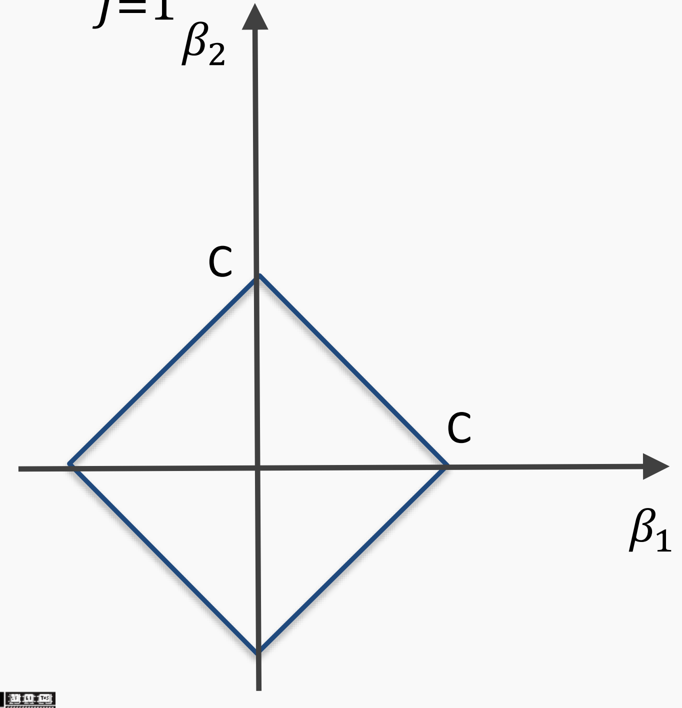


The Geometry of Regularization (LASSO)

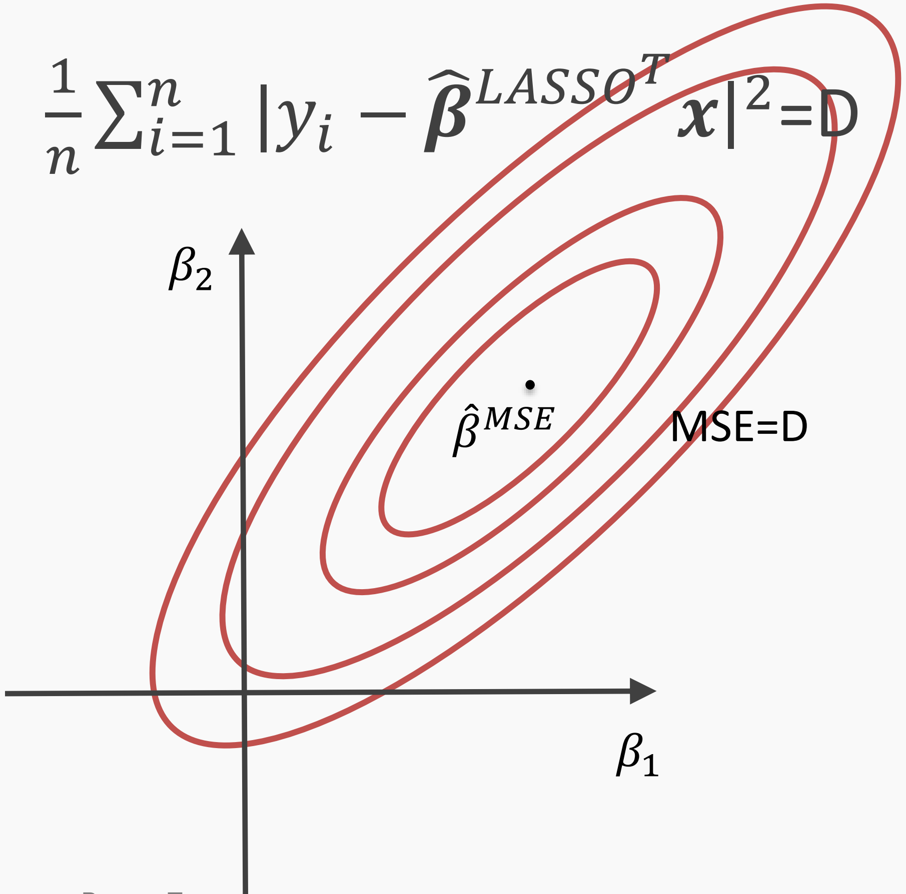
$$L_{LASSO}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} \left|y_i - \boldsymbol{\beta}^T \boldsymbol{x}\right|^2 + \lambda \sum_{j=1}^{J} |\beta_j|$$

$$\widehat{\boldsymbol{\beta}}^{LASSO} = \operatorname{argmin} L_{LASSO}(\boldsymbol{\beta})$$

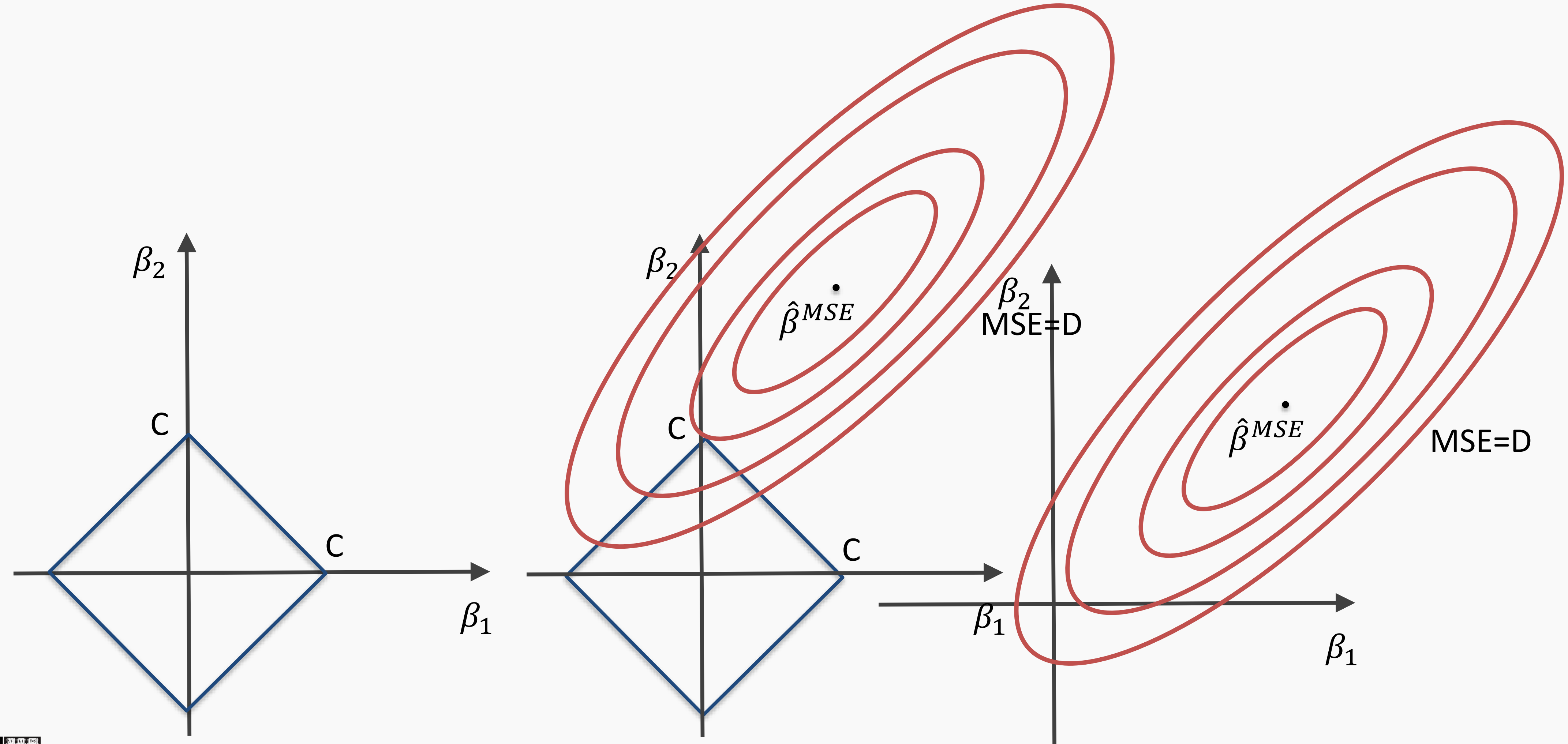
$$\lambda \sum_{j=1}^{J} \left| \hat{\beta}_j^{LASSO} \right| = C$$



$$\frac{1}{n}\sum_{i=1}^{n} |y_i - \widehat{\boldsymbol{\beta}}^{LASSO^T} \boldsymbol{x}|^2 = D$$



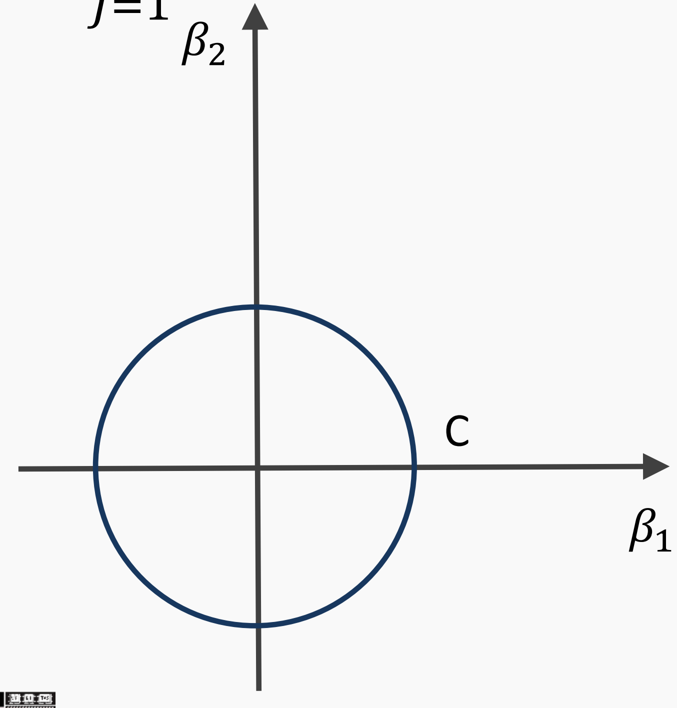
The Geometry of Regularization (LASSO)



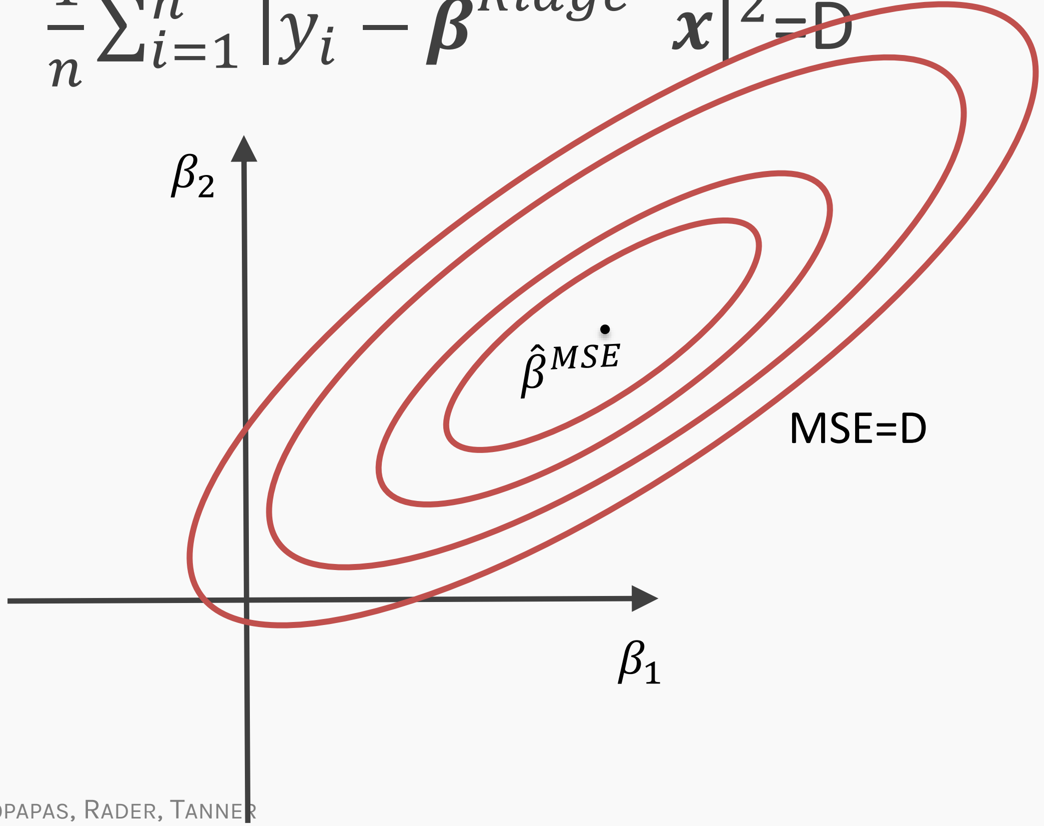
The Geometry of Regularization (Ridge)

$$L_{Ridge}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} \left|y_i - \boldsymbol{\beta}^T \boldsymbol{x}\right|^2 + \lambda \sum_{j=1}^{J} (\beta_j)^2 \qquad \widehat{\boldsymbol{\beta}}^{Ridge} = \operatorname{argmin} L_{Ridge}(\boldsymbol{\beta})$$

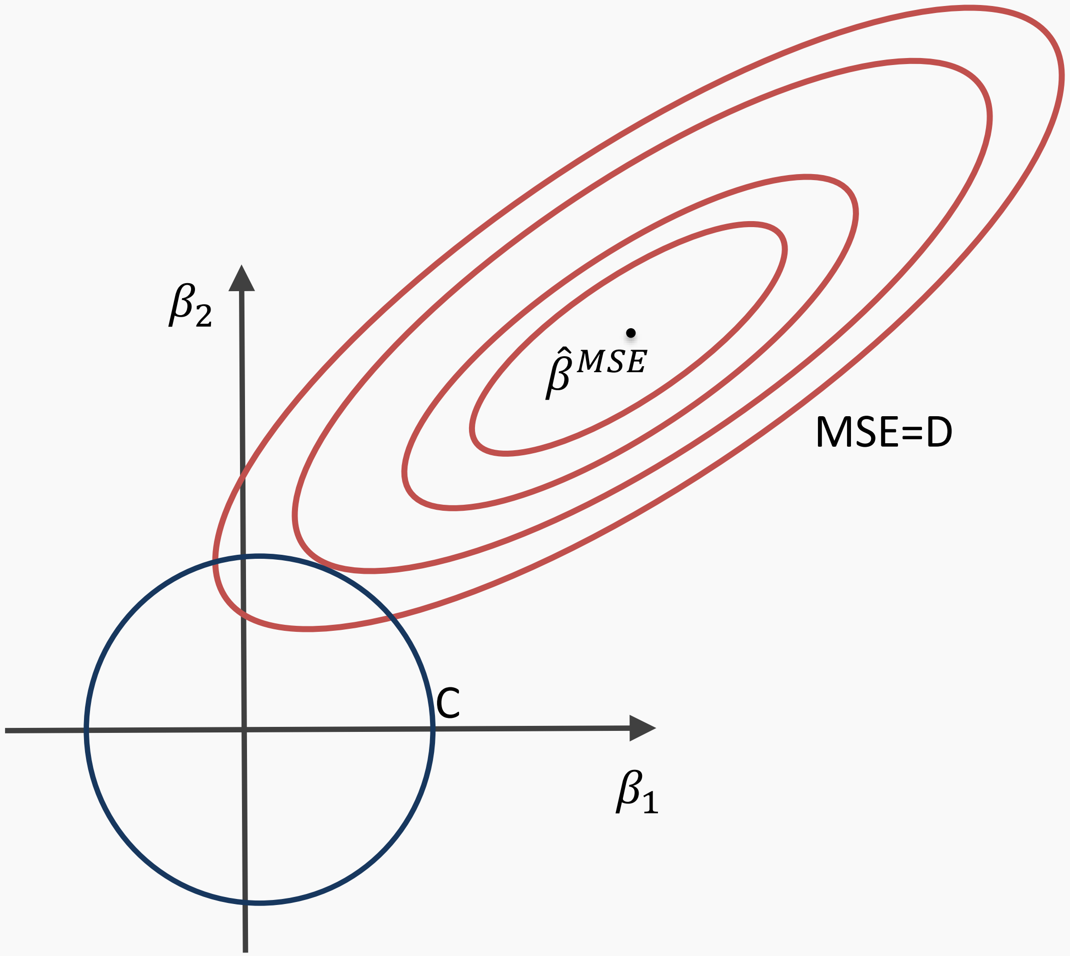
$$\lambda \sum_{j=1}^{J} \left|\hat{\beta}_j^{Ridge}\right|^2 = C$$



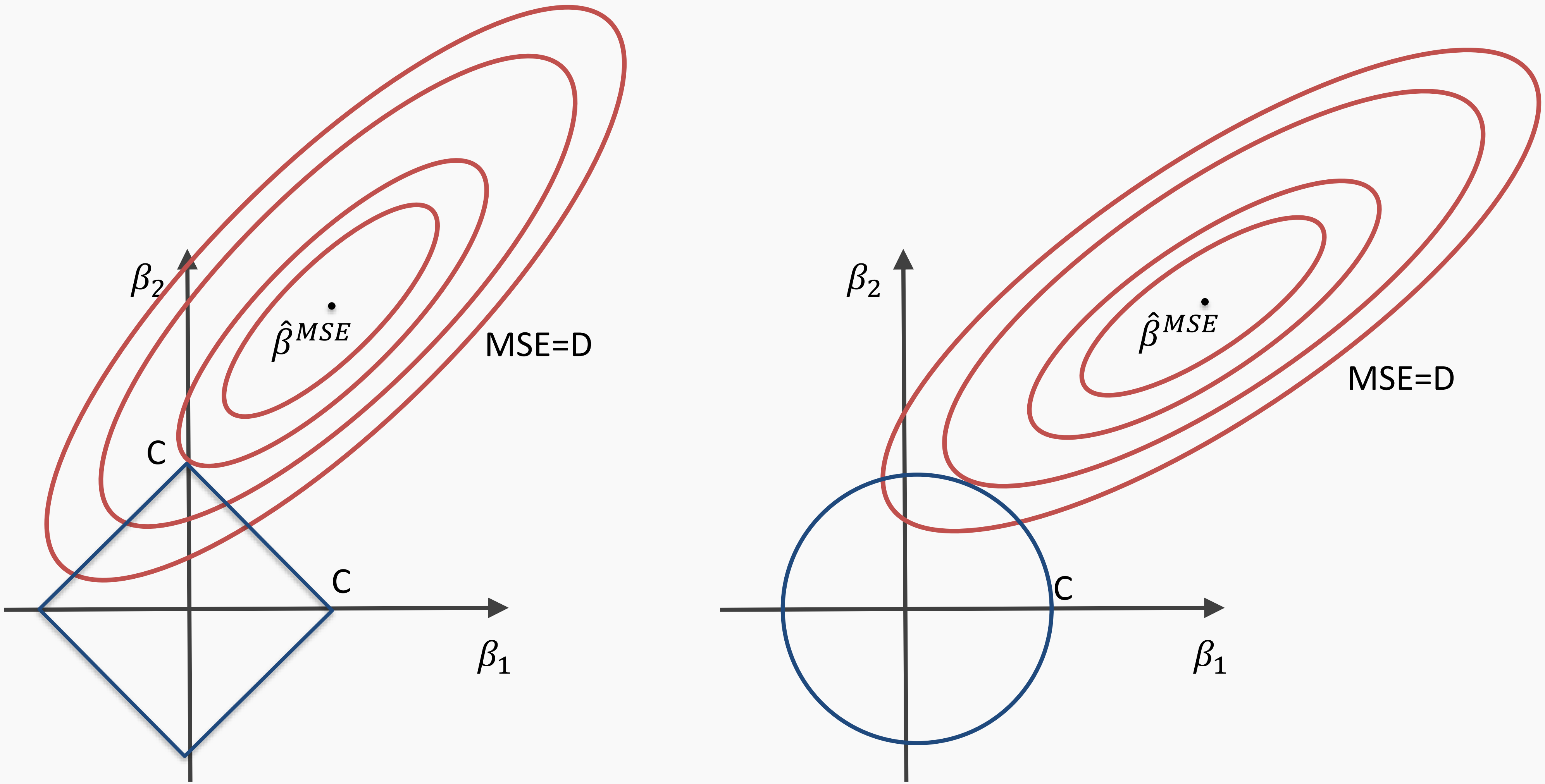
$$\frac{1}{n}\sum_{i=1}^{n} \left|y_i - \widehat{\boldsymbol{\beta}}^{Ridge^T} \boldsymbol{x}\right|^2 = \mathrm{D}$$



The Geometry of Regularization (Ridge)



The Geometry of Regularization



Examples

```
In [ ]: from sklearn.linear_model import Lasso
```

```
In [22]: lasso_regression = Lasso(alpha=1.0, fit_intercept=True)
         lasso_regression.fit(np.vstack((X_train, X_val)), np.hstack((y_train, y_val)))

         print('Lasso regression model:\n {} + {}^T . x'.format(lasso_regression.intercept_, lasso_regression.coe
```

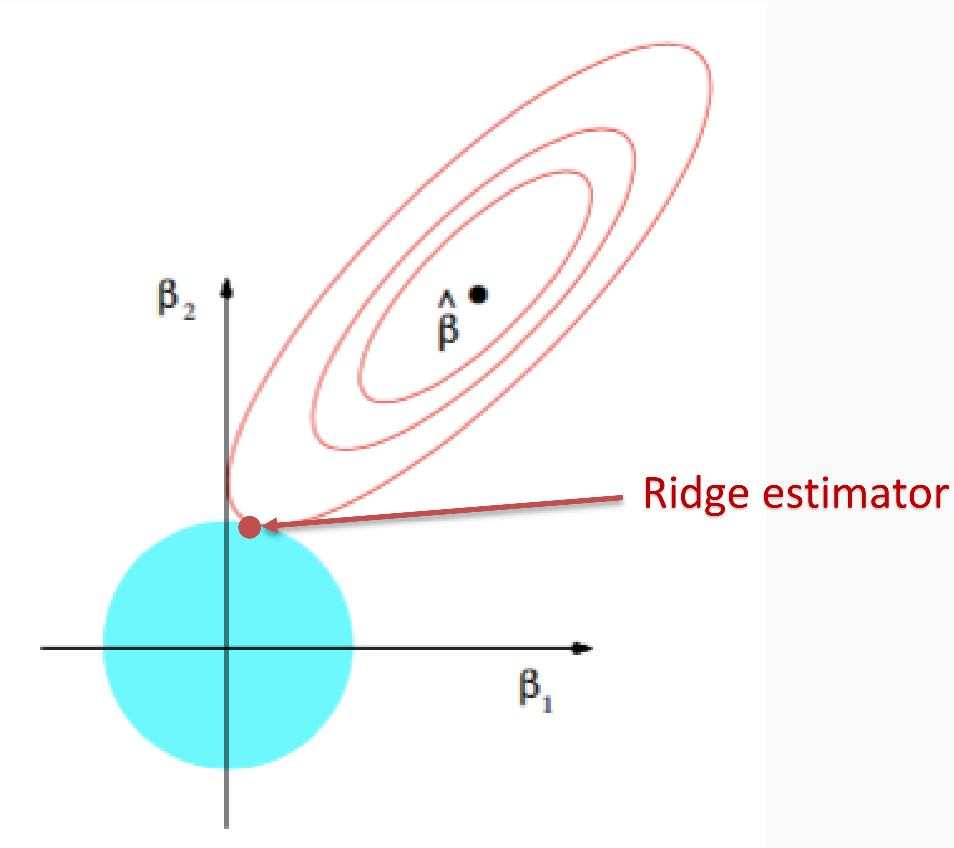
```
Lasso regression model:
 10.424895873901445 + [ 0.24482603  3.48164594  1.84836859 -0.06864603 -0.          -0.
 -0.02249766 -0.          0.          0.          0.          0.        ]^T . x
```

```
In [ ]: from sklearn.linear_model import Ridge
```

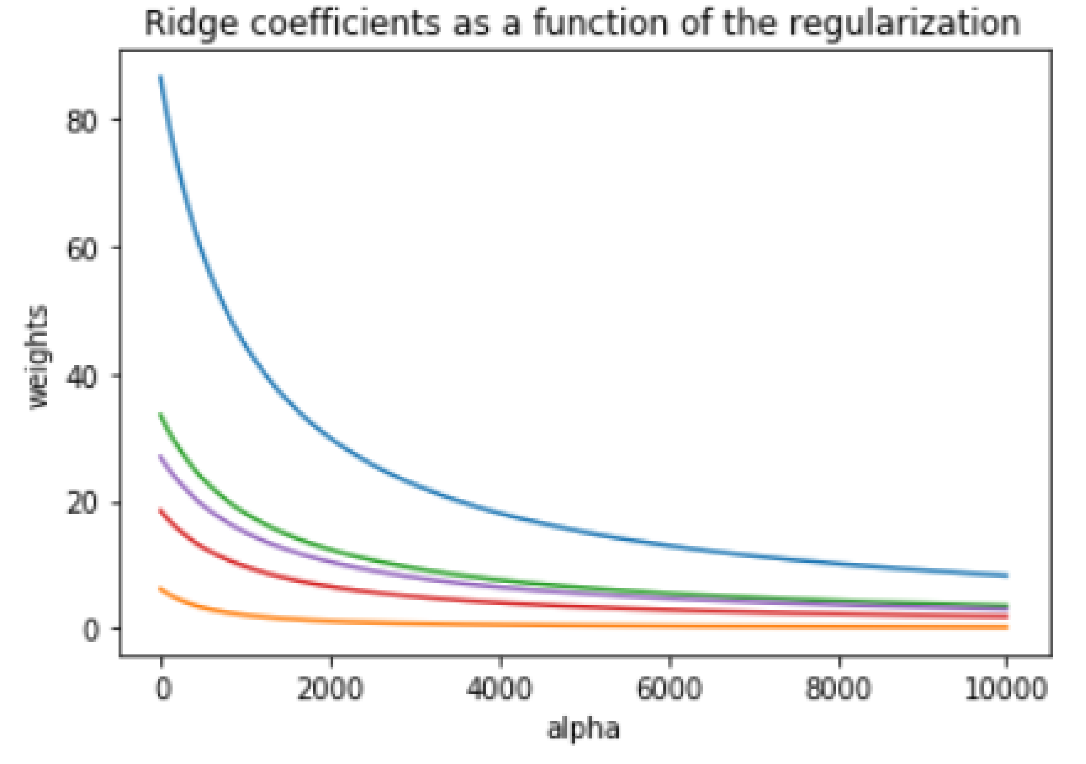
```
In [20]: X_train = train[all_predictors].values
         X_val = validation[all_predictors].values
         X_test = test[all_predictors].values

         ridge_regression = Ridge(alpha=1.0, fit_intercept=True)
         ridge_regression.fit(np.vstack((X_train, X_val)), np.hstack((y_train, y_val)))

         print('Ridge regression model:\n {} + {}^T . x'.format(ridge_regression.intercept_, ridge_regression.coe
```

```
Ridge regression model:
 -525.7662550875951 + [ 0.24007312  8.42566029  2.04098593 -0.04449172 -0.01227935  0.41902475
 -0.50397312 -4.47065168  4.99834262  0.          0.          0.29892679]^T . x
```

Ridge visualized

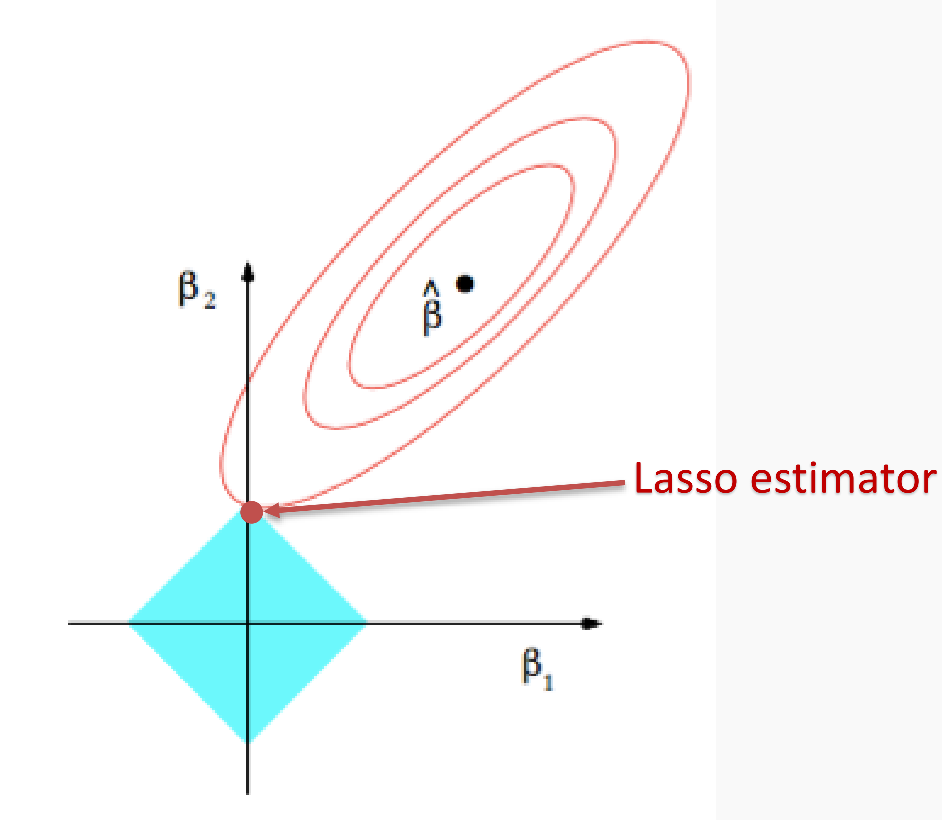


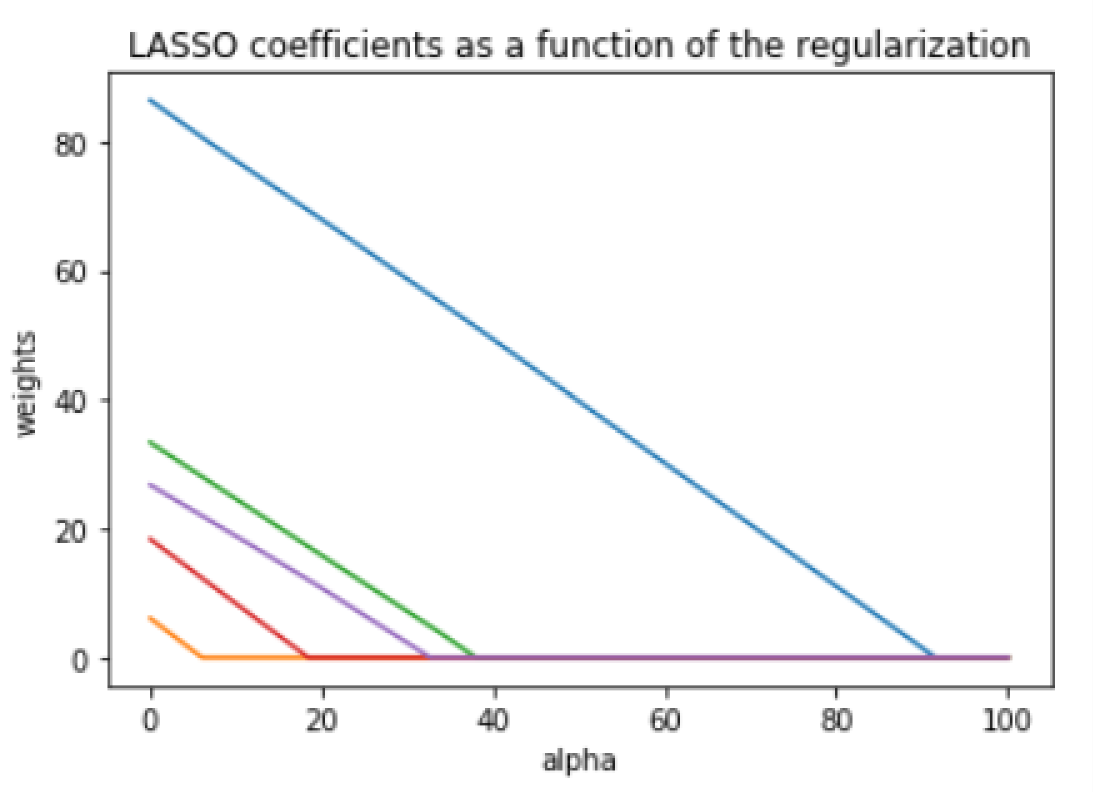
The ridge estimator is where the constraint and the loss intersect.



The values of the coefficients decrease as lambda increases, but they are not nullified.

LASSO visualized





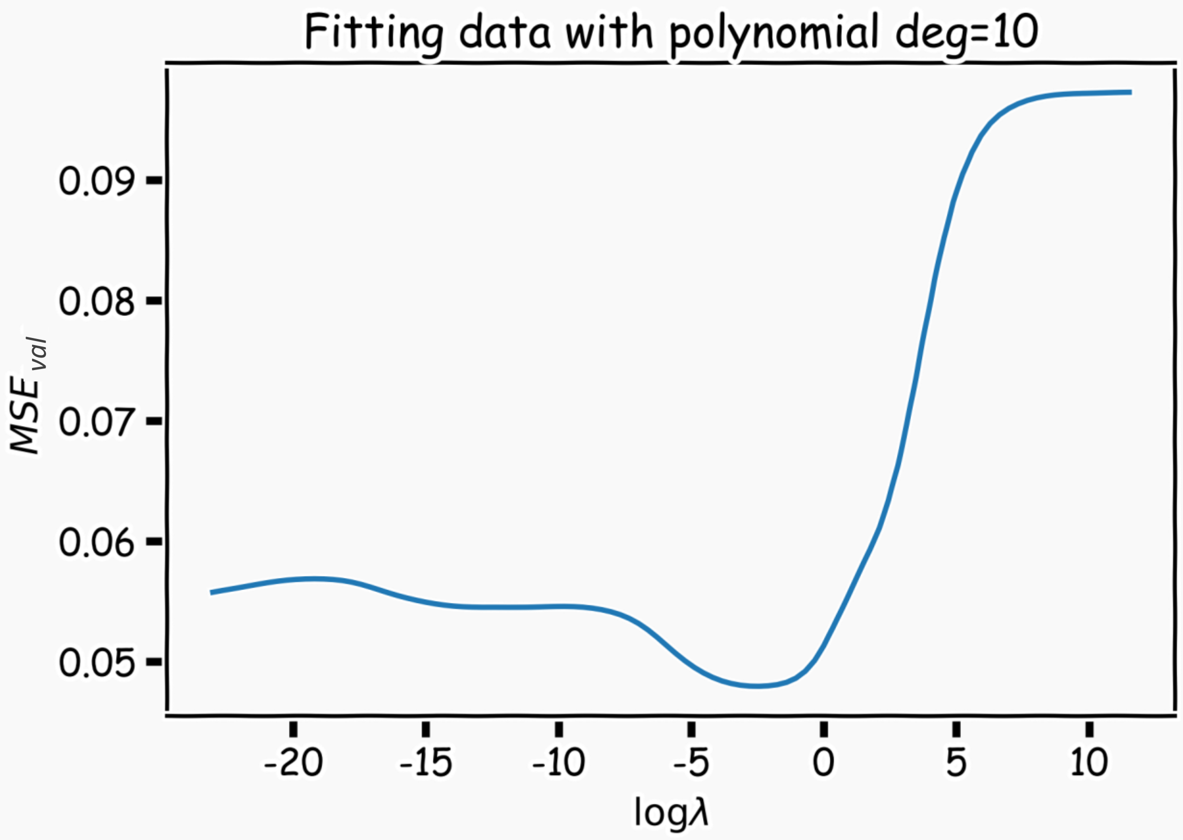
The Lasso estimator tends to zero out parameters as the OLS loss can easily intersect with the constraint on one of the axis.

The values of the coefficients decrease as lambda increases, and are nullified fast.

Ridge regularization with only **validation** : step by step

1. split data into $\{\{X,Y\}_{train}, \{X,Y\}_{validation}, \{X,Y\}_{test}\}$
2. for λ in $\{\lambda_{min}, \dots \lambda_{max}\}$:
   1. determine the β that minimizes the L_{ridge}, $\hat{\beta}_{Ridge}(\lambda) = \left(\mathrm{X}^{\mathrm{T}}\mathrm{X} + \lambda I\right)^{-1} X^T Y$, using the train data.
   2. record $L_{MSE}(\lambda)$ using validation data.
3. select the λ that minimizes the loss on the validation data,
$$\lambda_{ridge} = \mathrm{argmin}_{\lambda}\, L_{MSE}(\lambda)$$
4. Refit the model using both train and validation data, $\{\{X,Y\}_{train}, \{X,Y\}_{validation}\}$, resulting to $\hat{\beta}_{ridge}(\lambda_{ridge})$
5. report MSE or R^2 on $\{X,Y\}_{test}$ given the $\hat{\beta}_{ridge}(\lambda_{ridge})$

Ridge regularization with **validation** only: step by step



Lasso regularization with **validation** only: step by step

1. split data into $\{\{X,Y\}_{train}, \{X,Y\}_{validation}, \{X,Y\}_{test}\}$
2. for λ in $\{\lambda_{min}, \ldots \lambda_{max}\}$:
   A. determine the β that minimizes the L_{lasso}, $\hat{\beta}_{lasso}(\lambda)$, using the train data. **This is done using a solver.**
   B. record $L_{MSE}(\lambda)$ using validation data
3. select the λ that minimizes the loss on the validation data,
   $$\lambda_{lasso} = \operatorname{argmin}_{\lambda} L_{MSE}(\lambda)$$
4. Refit the model using both train and validation data, $\{\{X,Y\}_{train}, \{X,Y\}_{validation}\}$, resulting to $\hat{\beta}_{lasso}(\lambda_{lasso})$
5. report MSE or R^2 on $\{X,Y\}_{test}$ given the $\hat{\beta}_{lasso}(\lambda_{lasso})$

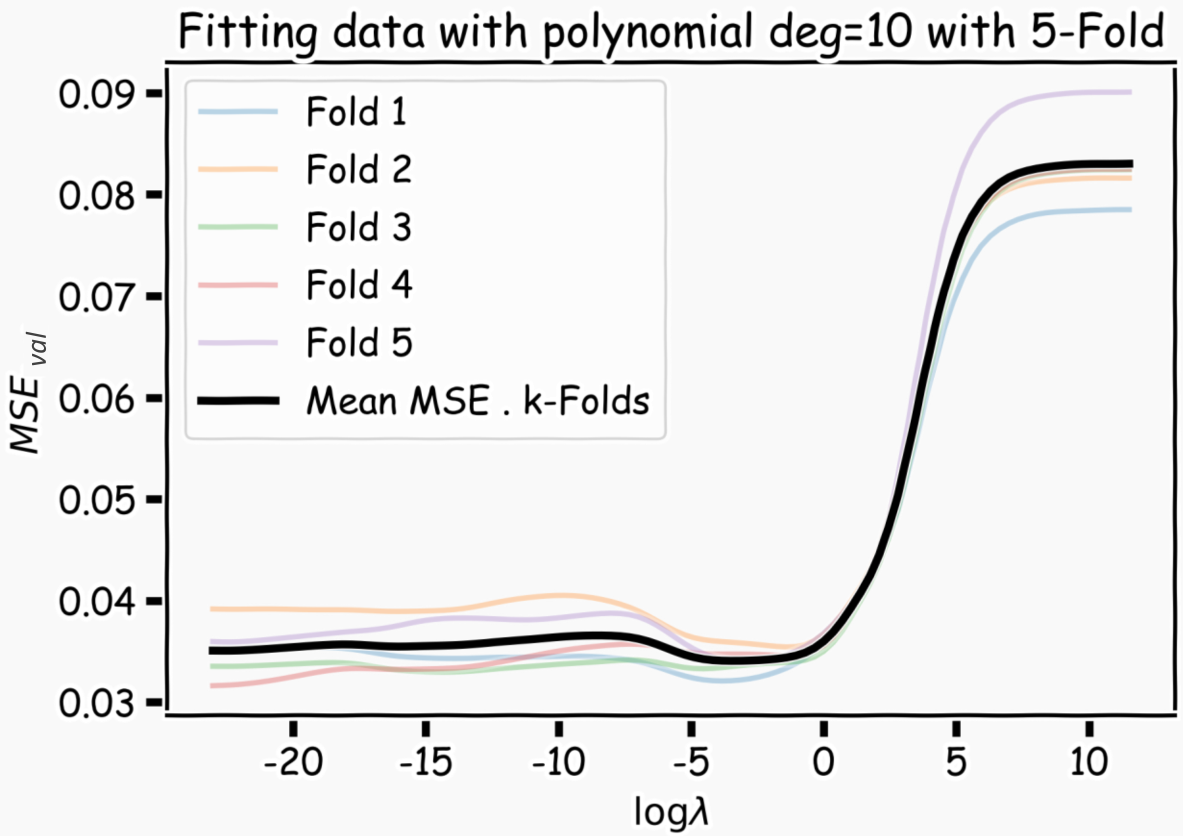
Ridge regularization with **CV**: step by step

	λ_1	λ_2	…	λ_n
k_1	L_{11}	L_{12}	..	…
k_2	L_{21}	…	..	…
…	..	…	..	…
k_n	…	…	…	…
E[]	$\bar{L}_1$	$\bar{L}_2$	…	$\bar{L}_n$

1. remove $\{X, Y\}_{test}$ from data
2. split the rest of data into K folds, $\{\{X, Y\}_{train}^{-k}, \{X, Y\}_{val}^{k}\}$
3. for k in $\{1, \ldots, K\}$
   1. for λ in $\{\lambda_0, \ldots, \lambda_n\}$:
      A. determine the β that minimizes the L_{ridge}, $\hat{\beta}_{ridge}(\lambda, k) = \left(\mathrm{X}^{\mathrm{T}}\mathrm{X} + \lambda I\right)^{-1} X^T Y$, using the train data of the fold, $\{X, Y\}_{train}^{-k}$.
      B. record $L_{MSE}(\lambda, k)$ using the validation data of the fold $\{X, Y\}_{val}^{k}$

      At this point we have a 2-D matrix, rows are for different k, and columns are for different λ values.
4. Average the $L_{MSE}(\lambda, k)$ for each λ, $\bar{L}_{MSE}(\lambda)$.
5. Find the λ that minimizes the $\bar{L}_{MSE}(\lambda)$, resulting to λ_{ridge}.
6. Refit the model using the full training data, $\{\{X, Y\}_{train}, \{X, Y\}_{val}\}$, resulting to $\hat{\beta}_{ridge}(\lambda_{ridge})$
7. report MSE or R^2 on $\{X, Y\}_{test}$ given the $\hat{\beta}_{ridge}(\lambda_{ridge})$

Ridge regularization with **validation** only: step by step



Variable Selection as Regularization

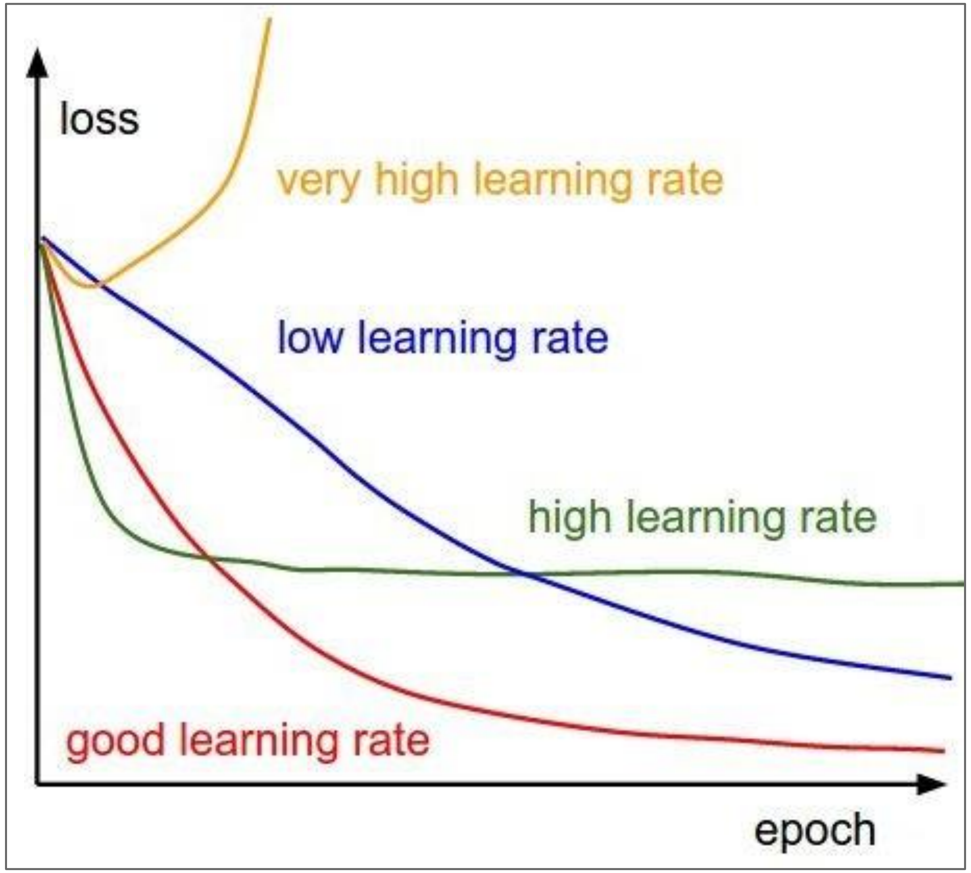
Since LASSO regression tend to produce zero estimates for a number of model parameters - we say that LASSO solutions are **sparse** - we consider LASSO to be a method for variable selection.

Many prefer using LASSO for variable selection (as well as for suppressing extreme parameter values) rather than stepwise selection, as LASSO avoids the statistic problems that arises in stepwise selection.

Question: What are the pros and cons of the two approaches?

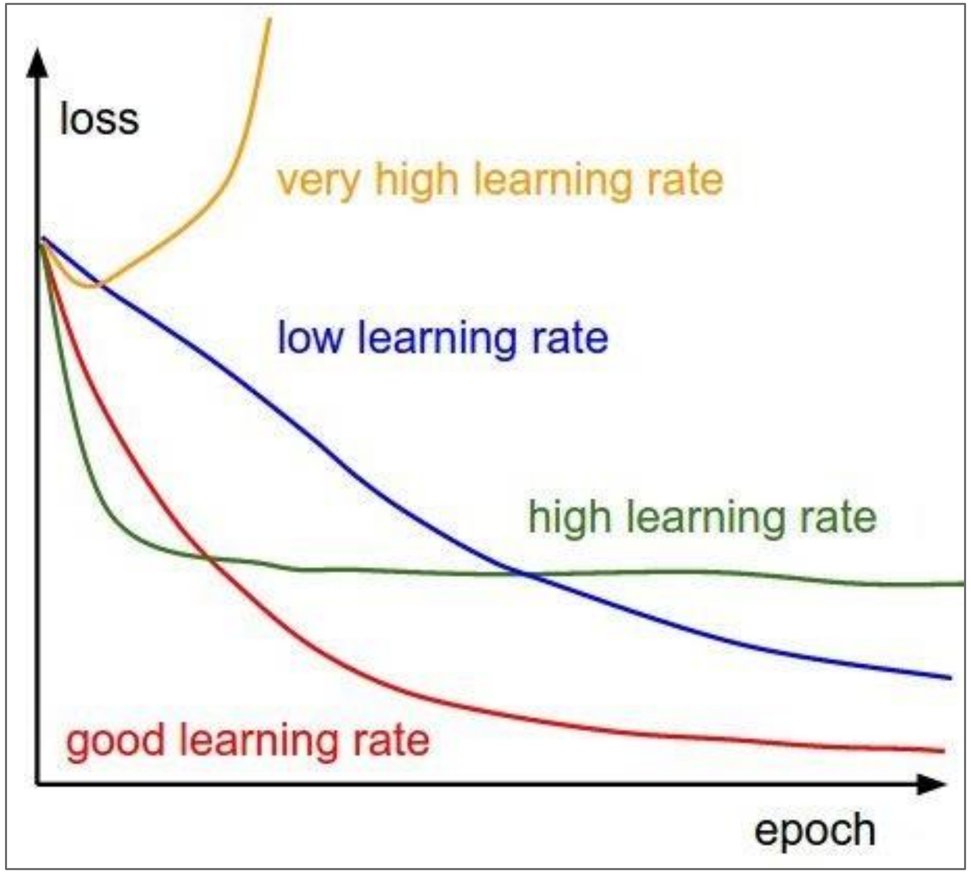
Learning rate schedules

Optimizers have **learning rate** as a hyperparameter



Q: Which one of these learning rates is best to use?

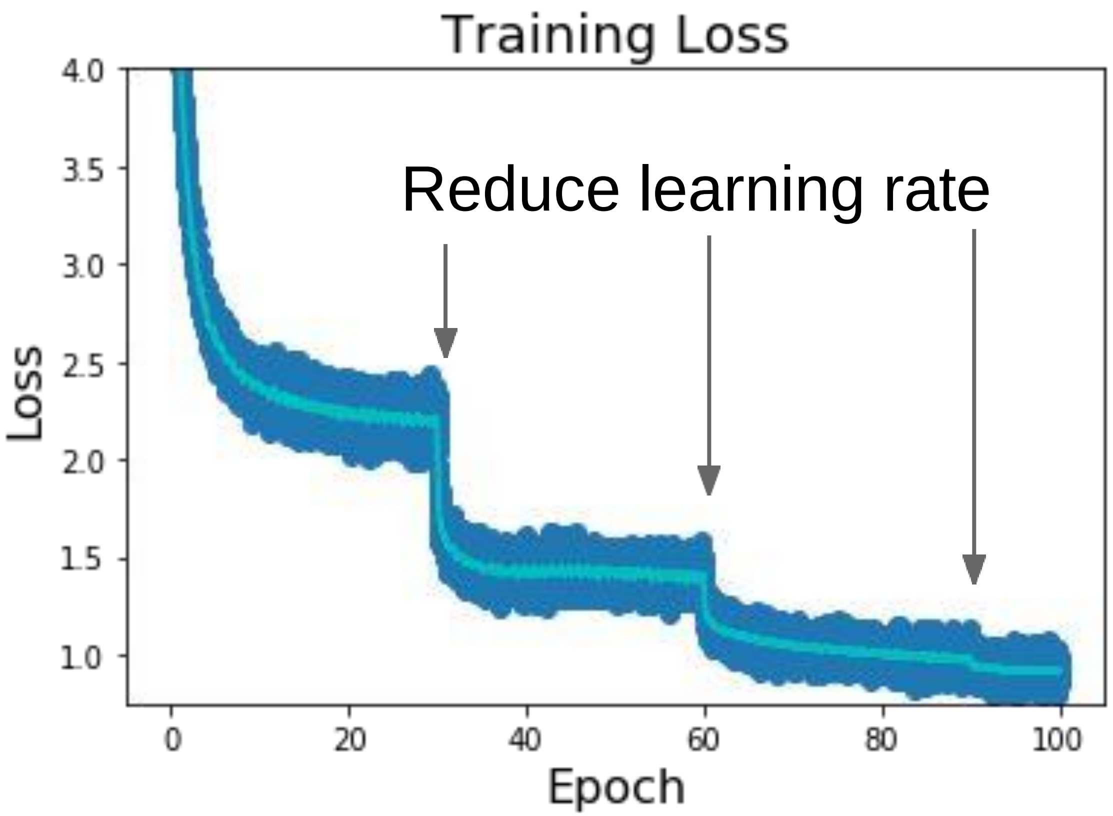
Optimizers have **learning rate** as a hyperparameter



Q: Which one of these learning rates is best to use?

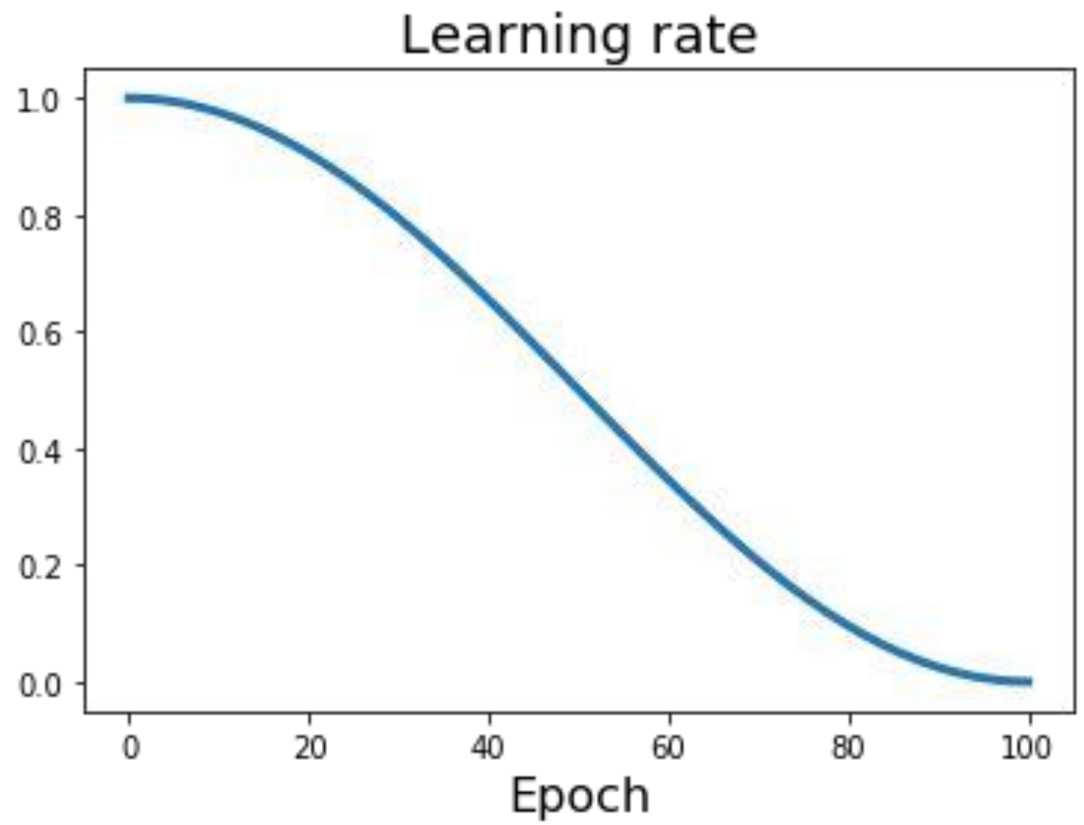
A: In reality, all of these are good learning rates.

Learning rate decays over time



Step: Reduce learning rate at a few fixed points. E.g. for ResNets, multiply LR by 0.1 after epochs 30, 60, and 90.

Learning Rate Decay



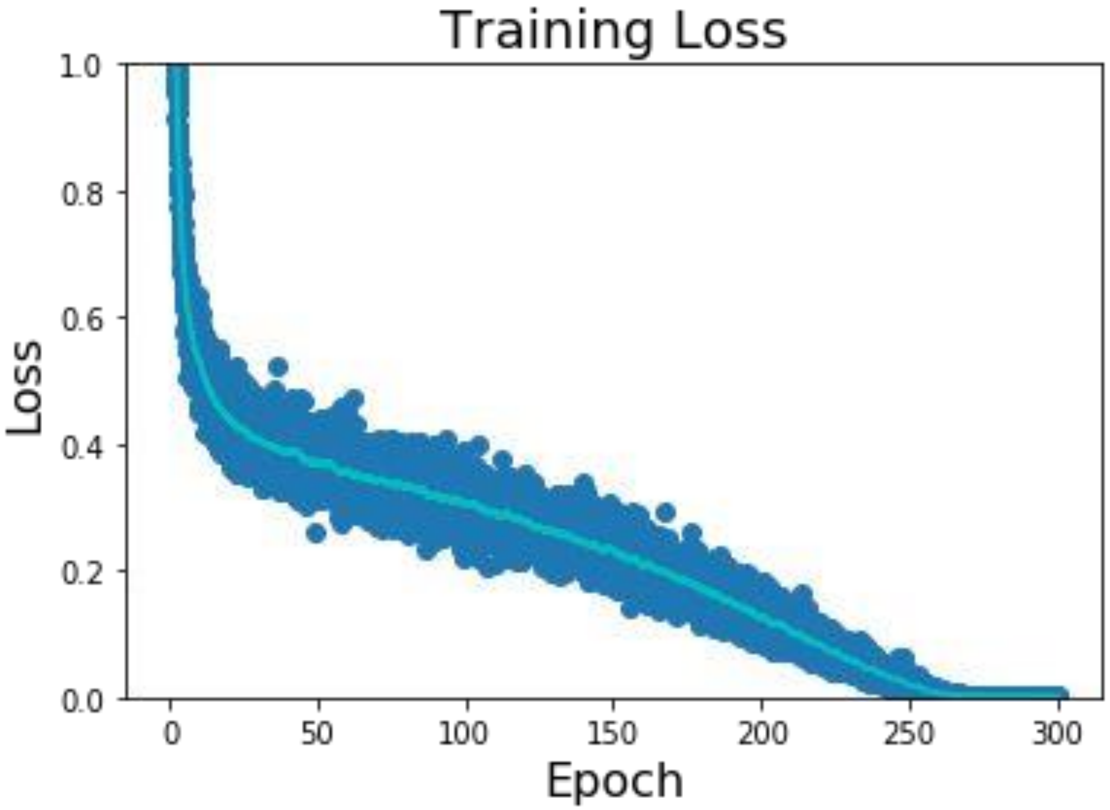
- **Step:** Reduce learning rate at a few fixed points. E.g. for ResNets, multiply LR by 0.1 after epochs 30, 60, and 90.

- **Cosine**: $\alpha_t = \frac{1}{2}\alpha_0\left(1 + \cos(t\pi/T)\right)$

α_0 : Initial learning rate
α_t : Learning rate at epoch t
T : Total number of epochs

Loshchilov and Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts", ICLR 2017
Radford et al, "Improving Language Understanding by Generative Pre-Training", 2018
Feichtenhofer et al, "SlowFast Networks for Video Recognition", arXiv 2018
Child at al, "Generating Long Sequences with Sparse Transformers", arXiv 2019

Learning Rate Decay



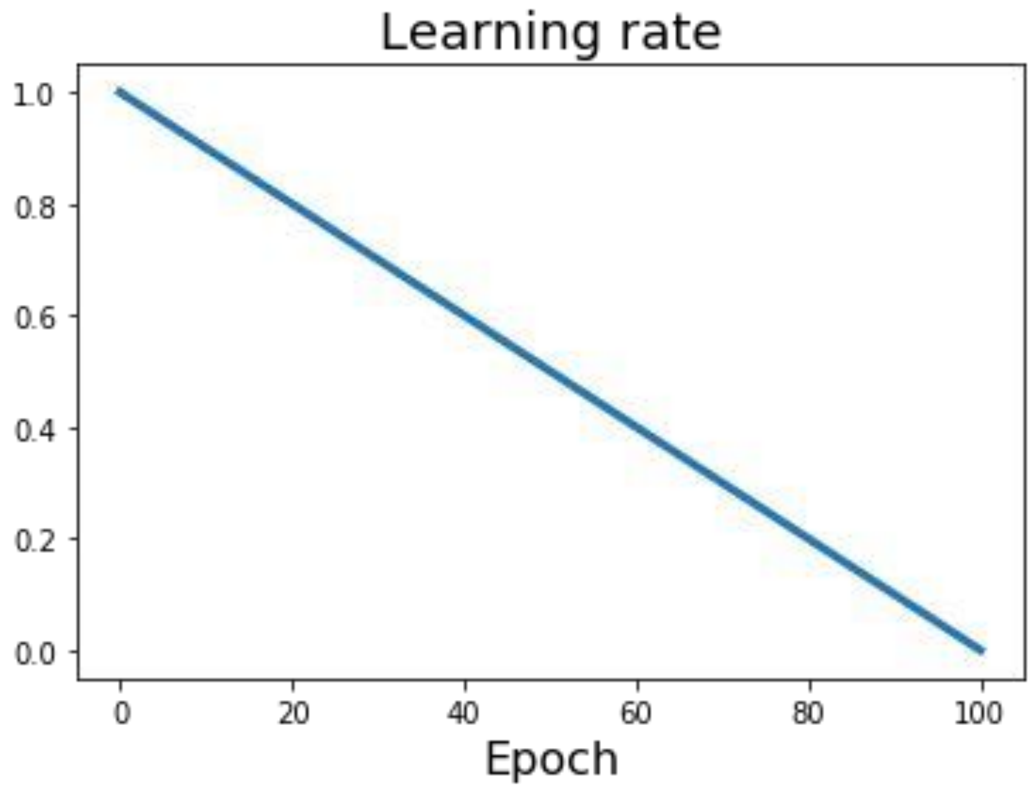
- **Step:** Reduce learning rate at a few fixed points. E.g. for ResNets, multiply LR by 0.1 after epochs 30, 60, and 90.

- **Cosine**: $\alpha_t = \frac{1}{2}\alpha_0\left(1 + \cos(t\pi/T)\right)$

α_0 : Initial learning rate
α_t : Learning rate at epoch t
T : Total number of epochs

Loshchilov and Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts", ICLR 2017
Radford et al, "Improving Language Understanding by Generative Pre-Training", 2018
Feichtenhofer et al, "SlowFast Networks for Video Recognition", arXiv 2018
Child at al, "Generating Long Sequences with Sparse Transformers", arXiv 2019

Learning Rate Decay

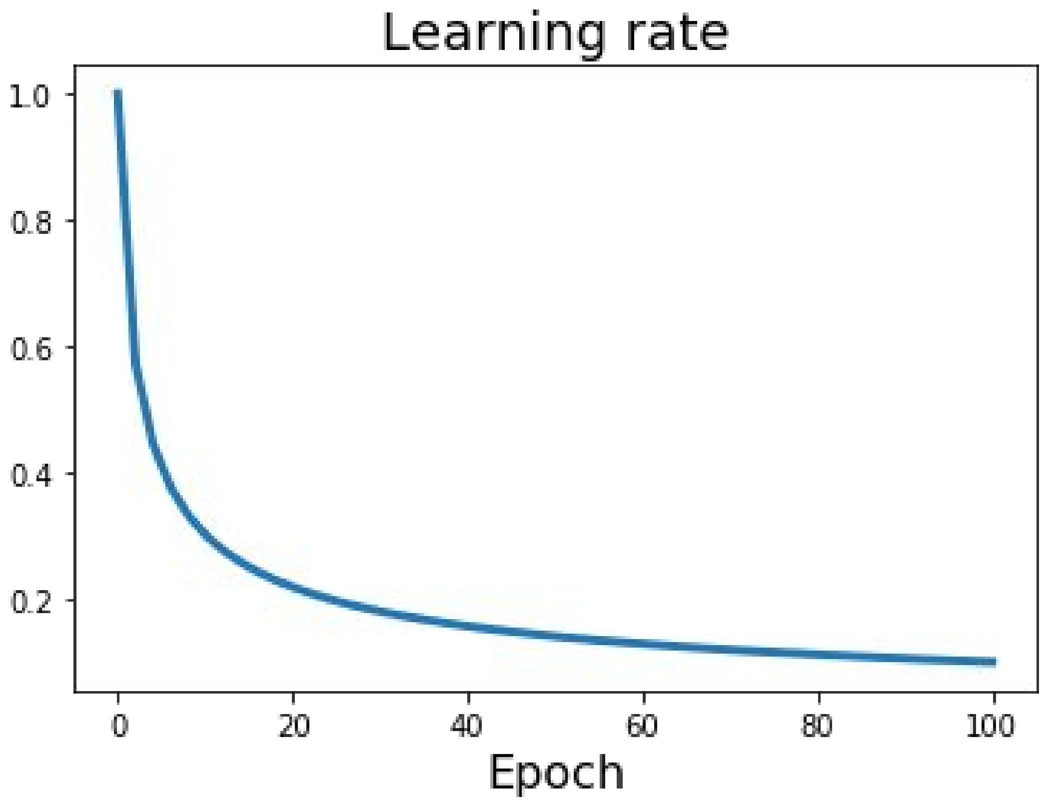


Devlin et al, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, 2018

- **Step:** Reduce learning rate at a few fixed points. E.g. for ResNets, multiply LR by 0.1 after epochs 30, 60, and 90.
- **Cosine**: $\alpha_t = \frac{1}{2}\alpha_0\left(1 + \cos(t\pi/T)\right)$
- **Linear:** $\alpha_t = \alpha_0(1 - t/T)$

α_0 : Initial learning rate
α_t : Learning rate at epoch t
T : Total number of epochs

Learning Rate Decay

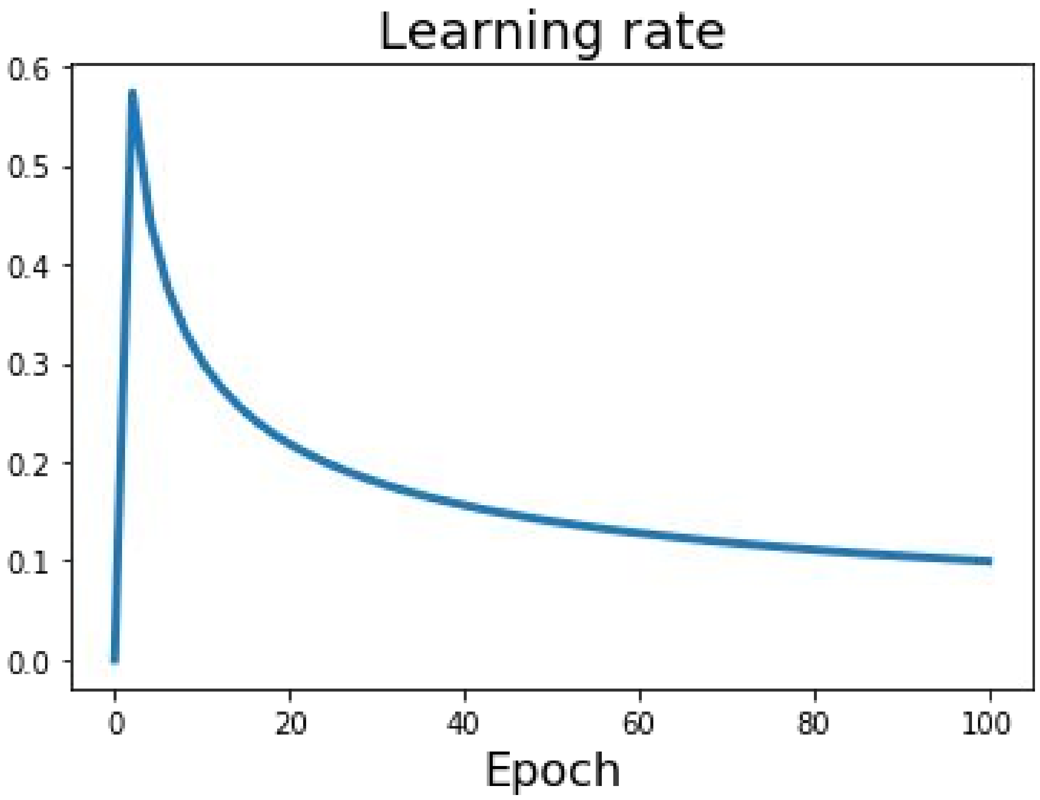


Vaswani et al, "Attention is all you need", NIPS 2017

- **Step:** Reduce learning rate at a few fixed points. E.g. for ResNets, multiply LR by 0.1 after epochs 30, 60, and 90.
- **Cosine**: $\alpha_t = \frac{1}{2}\alpha_0\left(1 + \cos(t\pi/T)\right)$
- **Linear:** $\alpha_t = \alpha_0(1 - t/T)$
- **Sqrt:** $\alpha_t = \alpha_0/\sqrt{t}$

α_0 : Initial learning rate
α_t : Learning rate at epoch t
T : Total number of epochs

Learning Rate Decay: Linear Warmup



High initial learning rates can make loss explode; linearly increasing learning rate from 0 over the first ~5000 iterations can prevent this

Empirical rule of thumb: If you increase the batch size by N, also scale the initial learning rate by N

Goyal et al, “Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour”, arXiv 2017

أكاديمية كاوست
KAUST ACADEMY