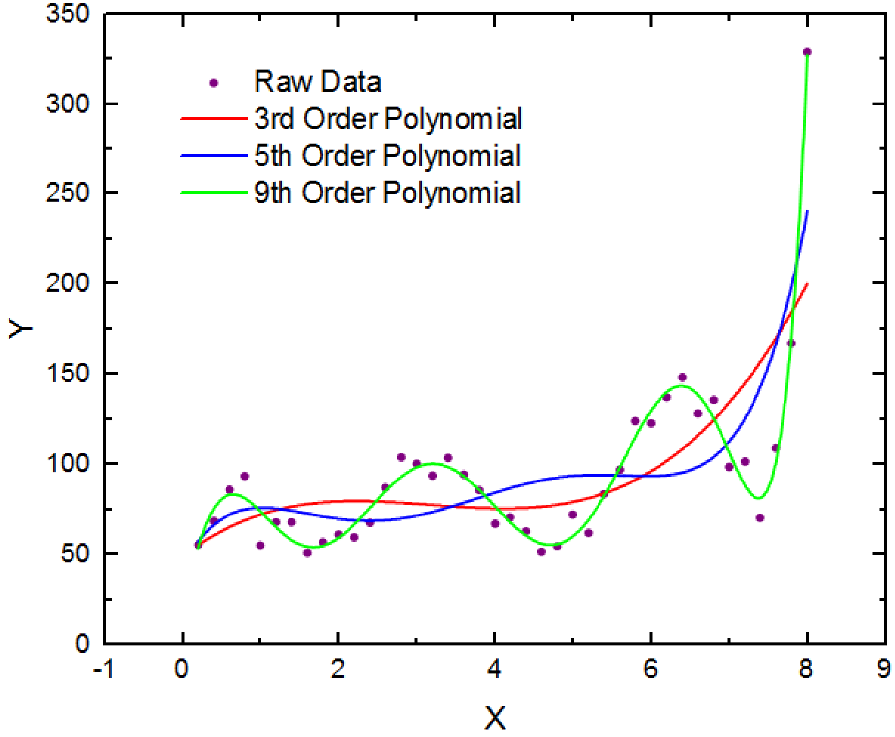
ECE 4252/8803: Fundamentals of Machine Learning (FunML)
Fall 2024

Lecture 9: Regularization and Performance Metrics



HW 3 and Exam 1

- HW 3 is released
  - Is due next Friday on both Gradescope and Canvas
- Midterms will be on Oct 2nd
  - Will cover all lecture materials until 30th Sept
  - Other logistics soon

Recap

Last Lecture

High-degree Polynomial Regression

Training by Gradient Descent

- The Algorithm
- Gradient Descent Alternatives
- Optimization techniques

Regularized Regression Models

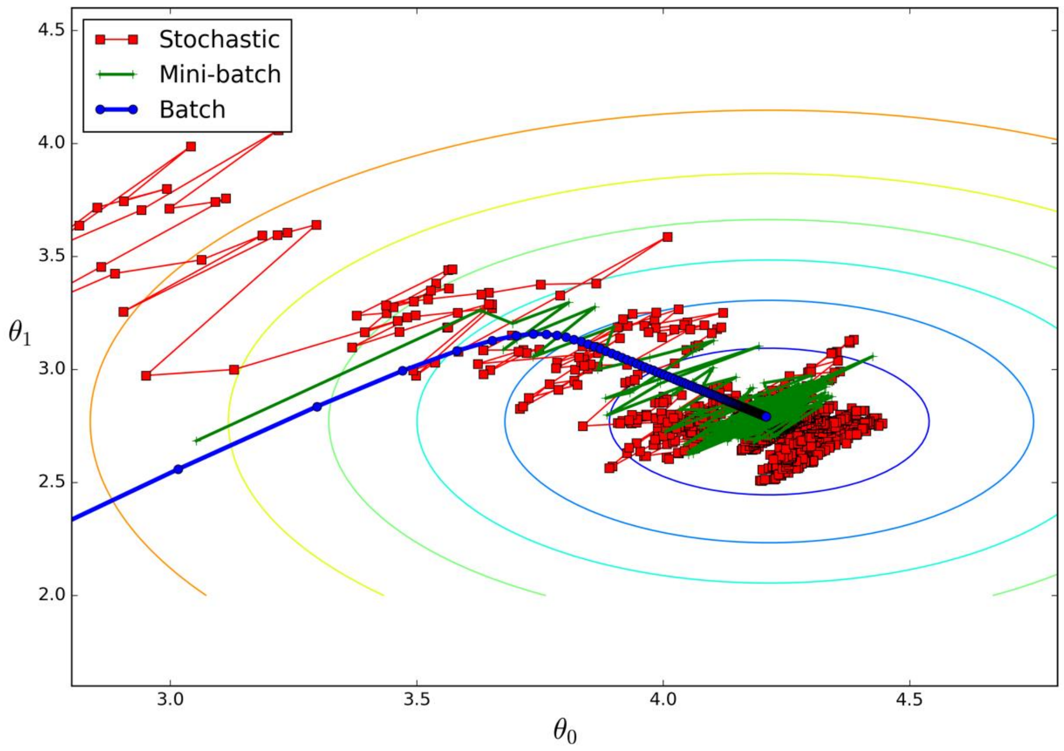
Performance Measures

Model Validation

Gradient Descent

Convergence

- All GD methods end up near the minimum
- Batch GD's path stops at the minimum
- Both Stochastic GD and Mini-batch GD continue to walk around.
- However, Batch GD takes a lot of time to take each step
- Stochastic GD and Mini-batch GD would also reach the minimum if a good learning schedule is used.



Gradient Descent

Comparisons

N is the number of training instances and P is the number of features

Algorithm	Performance with Large N	Memory Space Requirements	Performance with Large P	Normalization required?
Normal Equation	Fast	High	Slow	No
Batch GD	Slow	High	Fast	Yes
Stochastic GD	Fast	Minimum	Fast	Yes
Mini-batch GD	Fast	Relative to batch size	Fast	Yes

Gradient Descent

Newton’s Method

$$f(a)+\frac{f'(a)}{1!}(x-a)+\frac{f''(a)}{2!}(x-a)^2+\frac{f'''(a)}{3!}(x-a)^3+\cdots,$$

Newton's method is a local optimization scheme based on the ***second order*** Taylor series approximation

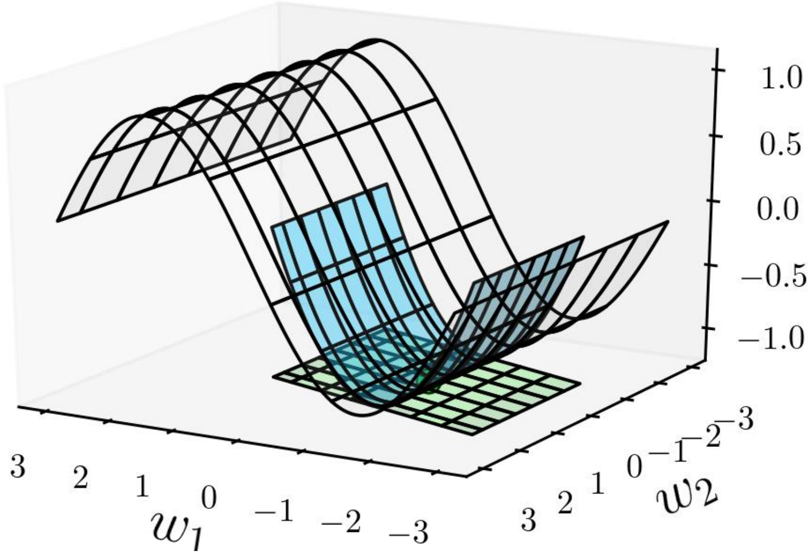
- The second order Taylor series approximation:

$$L(\boldsymbol{\theta}+\Delta\boldsymbol{\theta}) \approx L(\boldsymbol{\theta})+\Delta\boldsymbol{\theta}^T\nabla_\theta L(\theta)+\frac{1}{2}\overset{\boxed{\text{quadratic part}}}{\boxed{\Delta\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{H}(\boldsymbol{\theta})\Delta\boldsymbol{\theta}}}$$

- $\nabla_\theta L(\boldsymbol{\theta})$ is the gradient, $\boldsymbol{H}(\boldsymbol{\theta})=\nabla_\theta^2 L(\boldsymbol{\theta})$ is the second order derivatives of $L(\boldsymbol{\theta})$,which is also called Hessian matrix

$$\boldsymbol{H}(\boldsymbol{\theta})=\nabla_\theta^2 L(\boldsymbol{\theta})=\begin{pmatrix}\nabla_{\theta_1^2}^2 L(\boldsymbol{\theta}) & \dots & \nabla_{\theta_1\theta_P}^2 L(\boldsymbol{\theta})\\ \vdots & \vdots & \vdots\\ \nabla_{\theta_P\theta_1}^2 L(\boldsymbol{\theta}) & \dots & \nabla_{\theta_P^2}^2 L(\boldsymbol{\theta})\end{pmatrix}$$

- Assumes that $L(\boldsymbol{\theta})$ is *twice* differentiable
- Quadratic approximation
- Better local approximation of $L(\boldsymbol{\theta})$ than only using first order gradient.



Quadratic function (blue):

$$L(\boldsymbol{\theta})+\Delta\boldsymbol{\theta}^T\nabla_\theta L(\theta)+\frac{1}{2}\Delta\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{H}(\boldsymbol{\theta})\Delta\boldsymbol{\theta}$$

Materials adapted from Watt, J., Machine Learning Refined (2nd ed.)

Newton’s Method

Jacobian and Hessian Matrices

- The **Jacobian matrix** is a matrix composed of the first-order partial derivatives of a multivariable function.
- The **Hessian matrix**, or simply **Hessian**, is an *nxn* square matrix composed of the second-order partial derivatives of a function of n variables.
- Example: $f(x,y) = x^2y + y^2x$

$$H(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

$$H_f(x,y) = \begin{pmatrix} 2y & 2x+2y \\ 2x+2y & 2x \end{pmatrix}$$

Newton's Method

Optimization

matrix is positive semi-definite when it is symmetric with non-negative eigenvalues

- Unlike a hyperplane, a quadratic function does not have a 'steepest descent direction' to move to minima. However, a quadratic has global minima when it is *convex*.
- Note: the Hessian matrix is positive semi-definite, i.e., $\Delta\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{H}(\boldsymbol{\theta})\Delta\boldsymbol{\theta} \geq 0$ for any$\Delta\boldsymbol{\theta}$. Thus, $L(\boldsymbol{\theta}+\Delta\boldsymbol{\theta})$ describes a *convex* parabola
- To find the minimum of convex $L(\boldsymbol{\theta}+\Delta\boldsymbol{\theta})$, we set its first derivative $\nabla_{\Delta\boldsymbol{\theta}}L(\boldsymbol{\theta}+\Delta\boldsymbol{\theta}) = \mathbf{0}$ and solve the optimal $\Delta\boldsymbol{\theta}$:

$$L(\boldsymbol{\theta}+\Delta\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}) + \Delta\boldsymbol{\theta}^{T}\nabla_{\theta}L(\boldsymbol{\theta}) + \frac{1}{2}\Delta\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{H}(\boldsymbol{\theta})\Delta\boldsymbol{\theta}$$

$$\nabla_{\Delta\boldsymbol{\theta}}L(\boldsymbol{\theta}+\Delta\boldsymbol{\theta}) = \nabla_{\theta}L(\theta) + \boldsymbol{H}(\boldsymbol{\theta})\Delta\boldsymbol{\theta} = 0$$

$$\Longrightarrow \Delta\boldsymbol{\theta} = -\left(\boldsymbol{H}(\boldsymbol{\theta})\right)^{-\mathbf{1}}\nabla_{\theta}L(\boldsymbol{\theta})$$

- Thus, the Newton's Method takes the form:

$$\boldsymbol{\theta}^{\boldsymbol{t}+\mathbf{1}} = \boldsymbol{\theta}^{\boldsymbol{t}} - \boldsymbol{\alpha}\left(\boldsymbol{H}(\boldsymbol{\theta})\right)^{-\mathbf{1}}\nabla_{\theta}L(\boldsymbol{\theta})$$

Gradient descent:

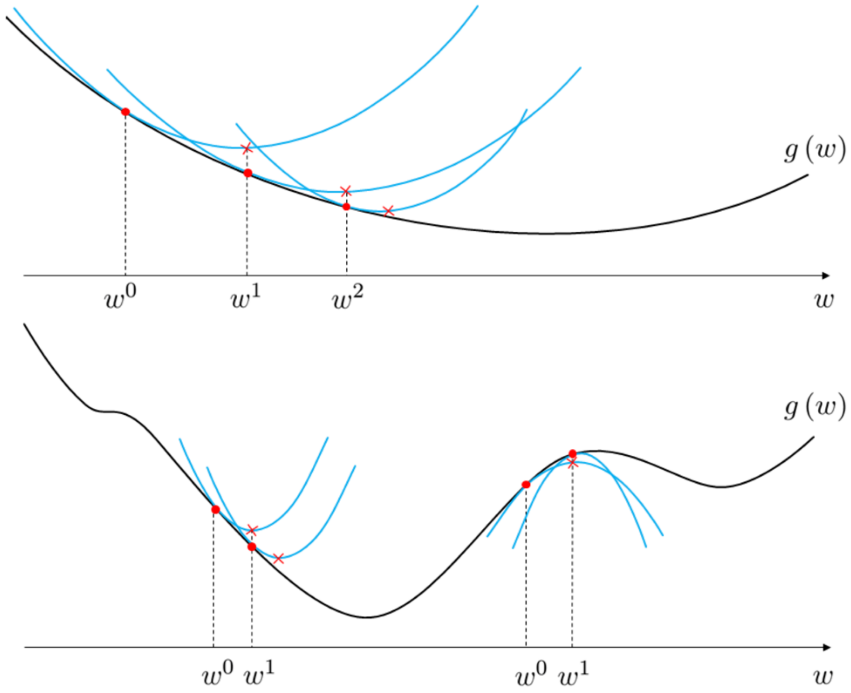
$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} - \alpha\nabla_{\theta}L(\boldsymbol{\theta}^{t})$$

Materials adapted from Watt, J., Machine Learning Refined (2nd ed.)

Newton’s Method

Convergence

- Newton's method is a powerful algorithm particularly for minimizing *convex* functions.
  - It uses quadratic, which is a more precise approximation than linear approximation at each step
  - it is often more effective than gradient descent as it requires fewer steps for convergence
- However, it is more difficult to use Newton's method to optimize non-convex functions.
  - At concave portions of such a function, the algorithm can climb to a local maximum or oscillate.



Materials adapted from Watt, J., Machine Learning Refined (2nd ed.)

Newton's Method

Computational Complexity

- Newton's method requires far more memory and computation than a first order algorithm.
  - The Hessian is a *P*×*P* matrix of second derivative. Computing its inverse-matrix causes scaling issues with input dimension

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} - \boldsymbol{\alpha}\big(\boldsymbol{H}(\boldsymbol{\theta})\big)^{-1} \nabla_{\theta} L(\boldsymbol{\theta})$$

Where $\boldsymbol{H}(\boldsymbol{\theta}) = \nabla_{\theta}^{2} L(\boldsymbol{\theta}) = \begin{pmatrix} \nabla_{\theta_1^2}^{2} L(\boldsymbol{\theta}) & \dots & \nabla_{\theta_1\theta_P}^{2} L(\boldsymbol{\theta}) \\ \vdots & \vdots & \vdots \\ \nabla_{\theta_P\theta_1}^{2} L(\boldsymbol{\theta}) & \dots & \nabla_{\theta_P^2}^{2} L(\boldsymbol{\theta}) \end{pmatrix}$

Materials adapted from Watt, J., Machine Learning Refined (2nd ed.)

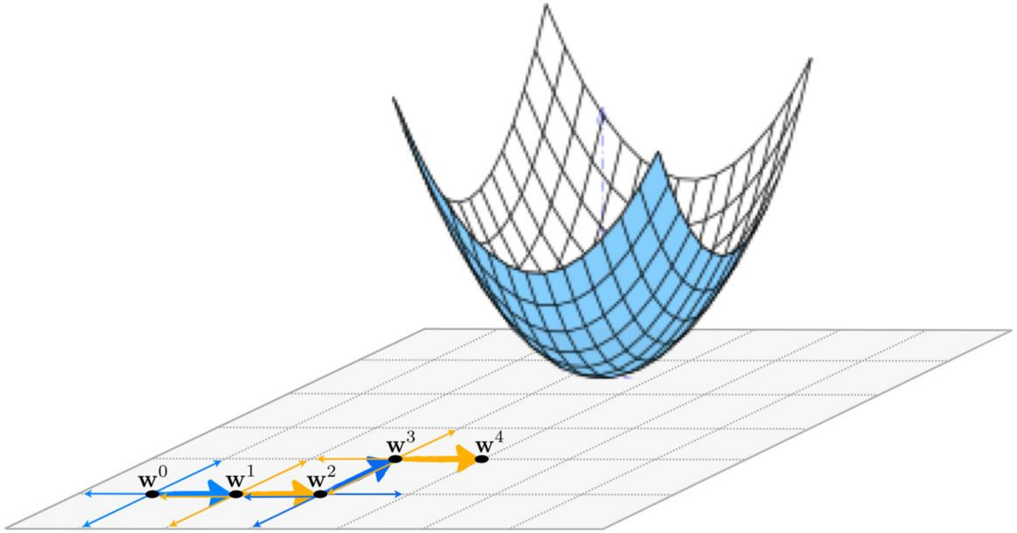
Coordinate Search

Optimization

- Other than first and second order approximation methods, there are simpler optimization techniques, searching through coordinate axes.
- The **coordinate search** algorithm takes the theme of descent direction search and searches through **all coordinate axes** (and their negatives) of the parameter space at each step.

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} \pm \alpha \boldsymbol{e}_j$$

Where $\boldsymbol{e}_j = [0, \ldots 1, \ldots 0]^{\boldsymbol{T}}$ is the j-th standard basis vector for the best descent direction among all P dimensions



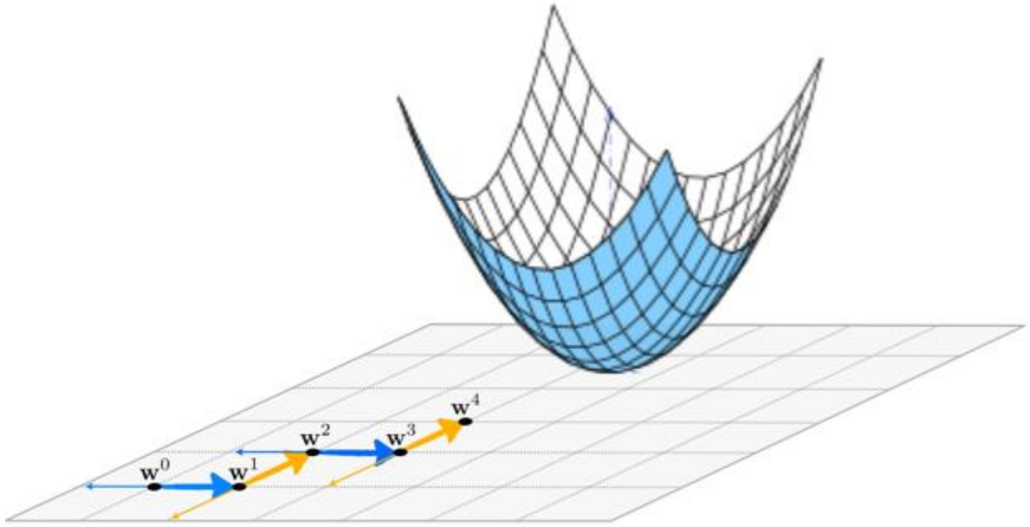
2-dimensional parameter space with 4=2x2 potential descent directions at each step

Materials adapted from Watt, J., Machine Learning Refined (2nd ed.)

Coordinate Descent

Optimization

- The **coordinate descent** algorithm examines **one coordinate axes** (and its negative) **at each step** and update in this direction if it produces descent.
- In the case of a continuously differentiable $L(\boldsymbol{\theta})$, a coordinate descent algorithm is:
  - Until convergence is reached:
    - Choose an index *j* from 1 to *P*.
    - Update $\theta_j^{t+1} = \theta_j^t - \alpha \nabla_{\theta_j} L(\boldsymbol{\theta})$



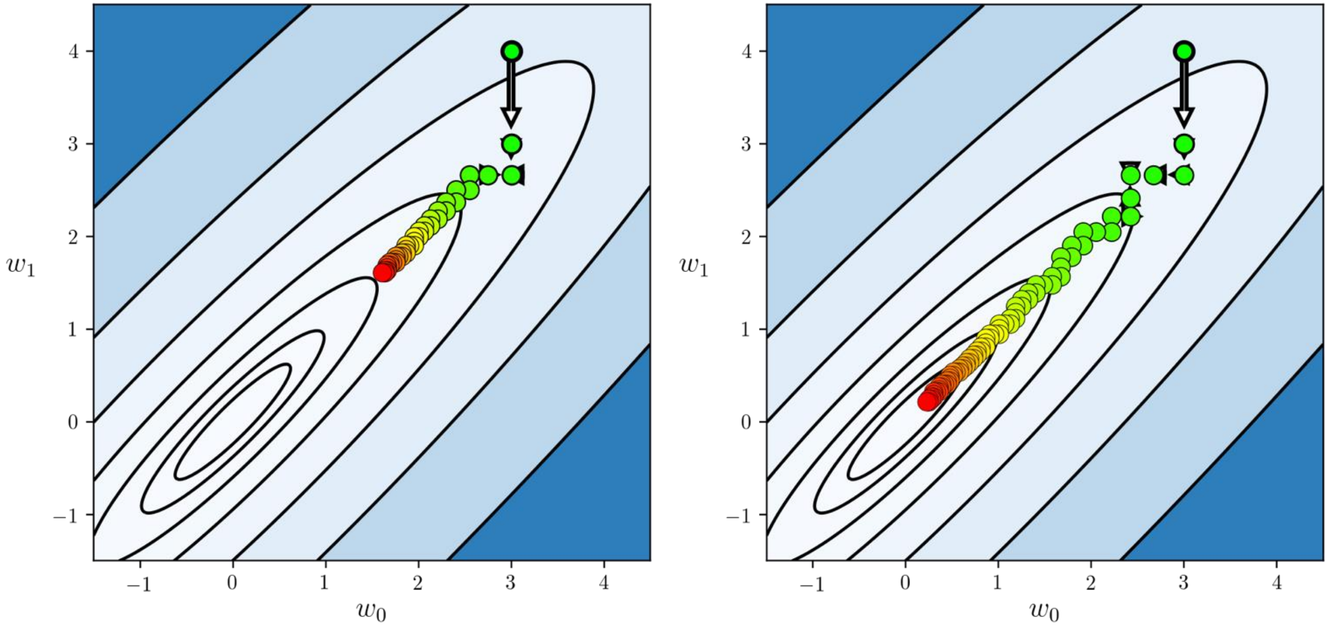
2-dimensional parameter space with 2=1x2 potential descent directions at each step

Materials adapted from Watt, J., Machine Learning Refined (2nd ed.)

Coordinate Descent

Convergence

- The first few steps of coordinate search are more effective because they search over all coordinate axes
- The coordinate descent quickly overtakes search finding a lower point on the cost function.

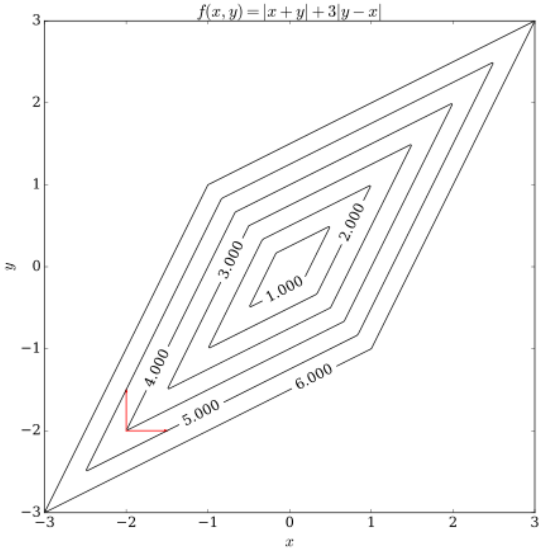


20 steps of coordinate search (left) and coordinate descent (right)

Materials adapted from Watt, J., Machine Learning Refined (2nd ed.)

Coordinate Descent

- Advantages:
  - Very simple and easy to implement
  - Scalable, less memory consumption compared to second order methods
- Limitation:
  - Difficult to optimize non-smooth multivariable function



Stuck at non-stationary point (-2,-2)

Overview

In this Lecture..

High-degree Polynomial Regression

Training by Gradient Descent

Regularized Linear Models

- Ridge Regression
- Lasso Regression
- Elastic Net Regression

Performance Measures

Model Validation

Regularization

Motivation

- Regularizations are constraints added to Linear Regression Models to
  - Reduce their sensitivity to noise
  - Reduce overfitting
- The Linear Model is typically regularized by constraining its coefficients.
- Three different ways to constrain coefficients in the Linear Model:
  - Ridge Regression
  - Lasso Regression
  - Elastic Net Regression

Ridge Regularization

Formulation

- Standard linear regression models attempt to fit data samples with minimum least squares cost. However, this is undesired when:
  - There is noise added to the features that causes significant change in coefficients
  - There are small amounts of available training data that cause overfitting
- Larger coefficients have higher relative impact on model predictions
- **Ridge Regression** (also called **Tikhonov regularization**) *shrinks* the regression coefficients $\boldsymbol{\theta}$ by imposing a penalty on the squared L2 norm $||\boldsymbol{\theta}||_2^2$ as an additional regularization term.
- During training, the regularization term is added to the cost function:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}_i - y_i)^2 + \frac{\gamma}{N}||\boldsymbol{\theta}||_{\mathbf{2}}^{\mathbf{2}}$$

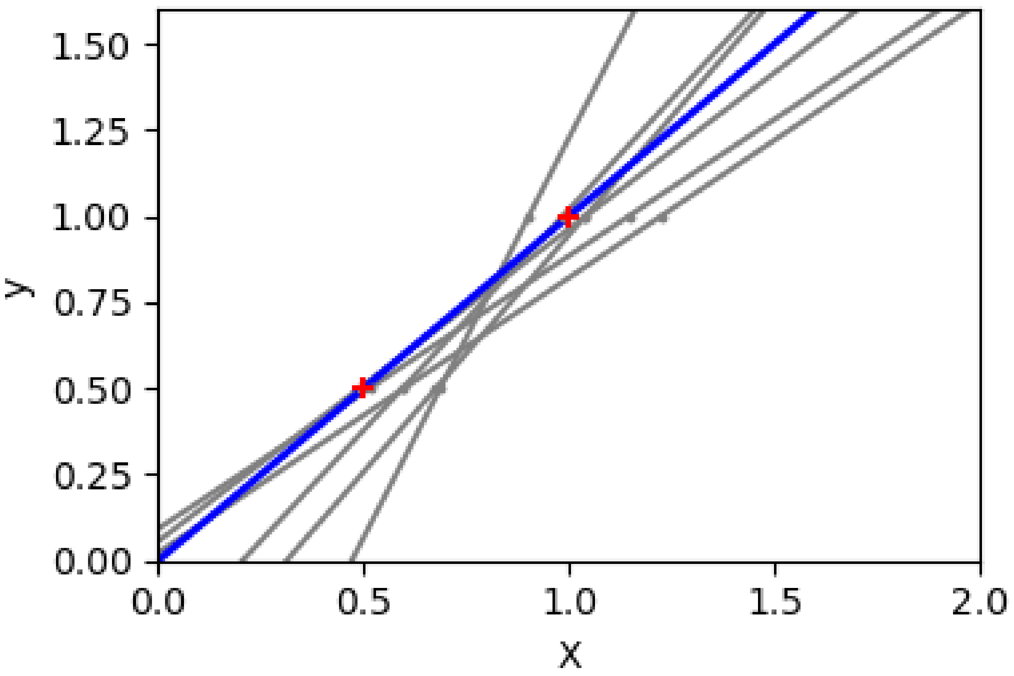
where $\gamma > 0$ is a hyperparameter that controls the amount of shrinkage

Note that the regularization term should only be added to the cost function during training. Once the model is trained, we evaluate the performance without regularization term.

Ridge Regularization

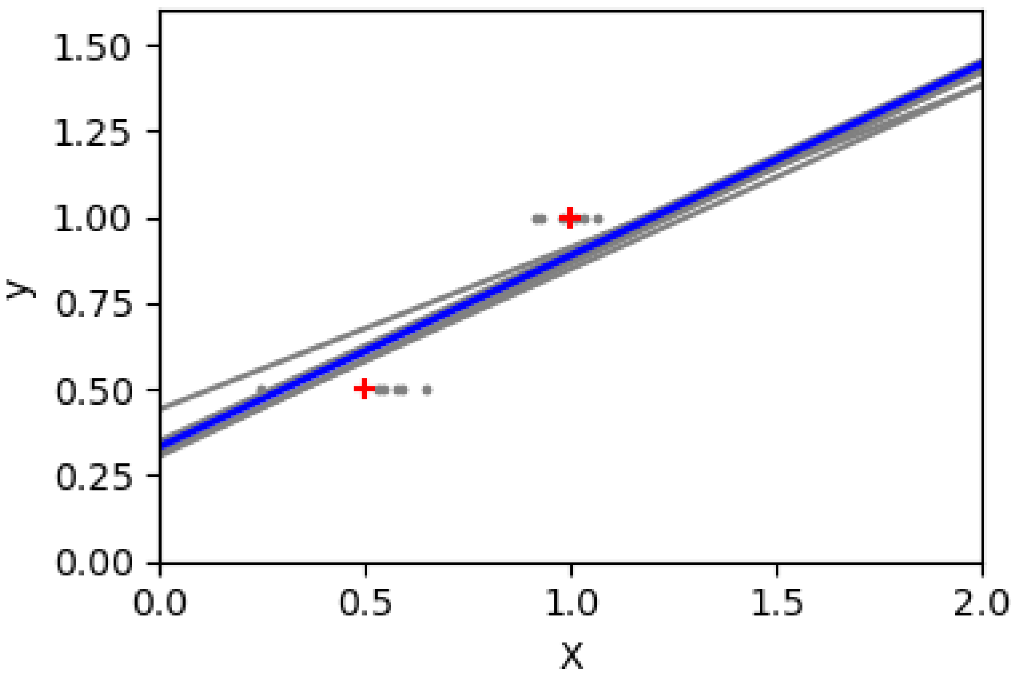
Robustness to Noise

Standard linear regression



Large variance of slope (coefficients) of linear regression models (gray lines) with induced noise samples (gray dots)

Ridge regression

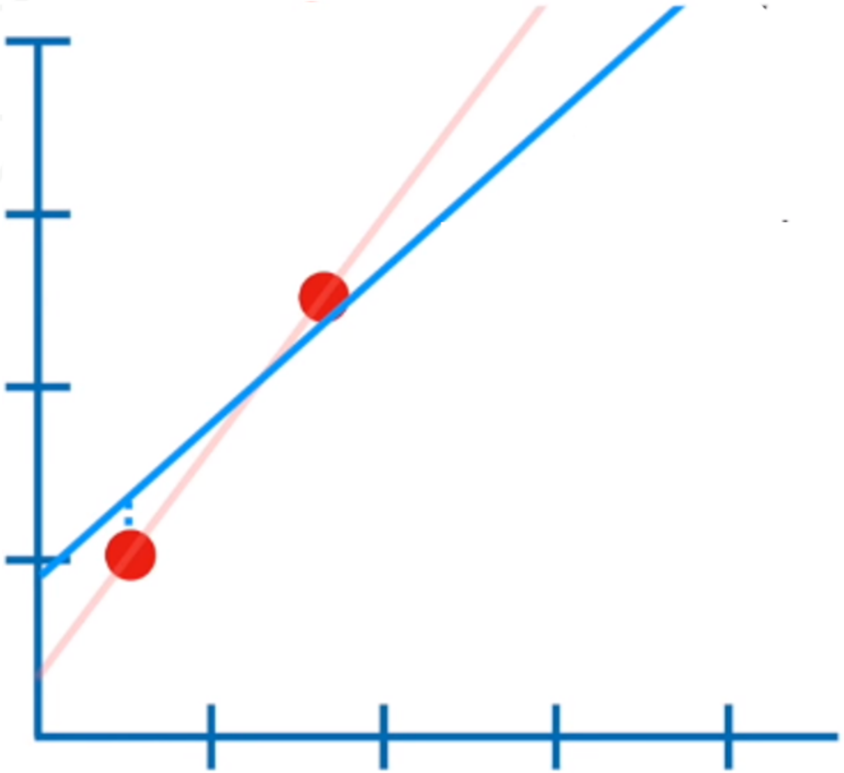


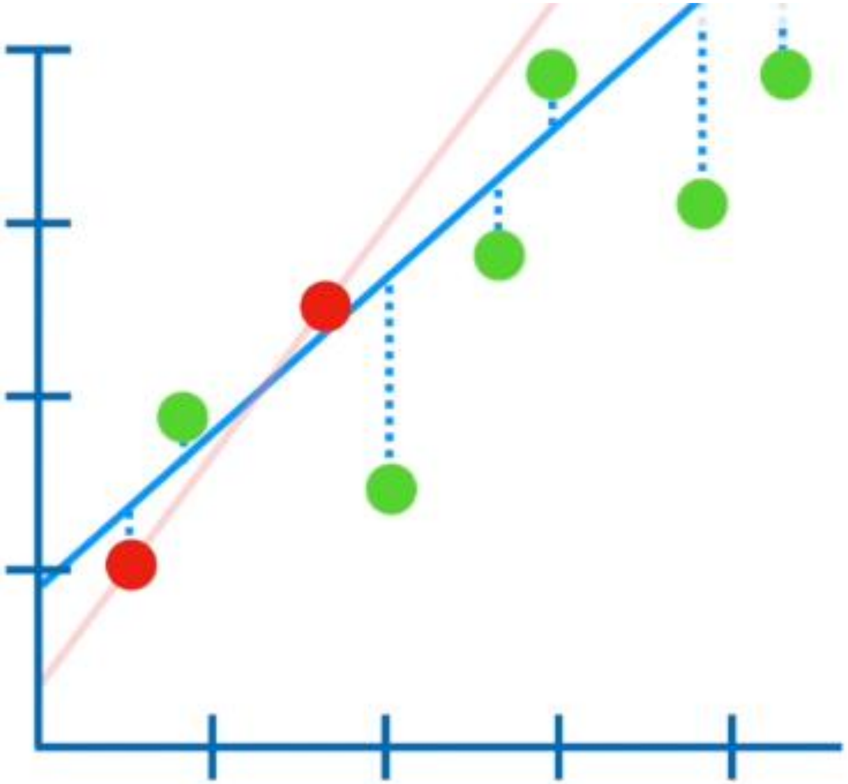
Reduced variance of ridge regression models (gray lines) with induced noise samples (gray dots)

Ridge Regularization

Reduce Overfitting to Limited Training Data

The linear regression model (red line) overfits to the small amount of training data (red dots), while ridge regression (blue line) generalizes better to the test data (green dots) with lower variance



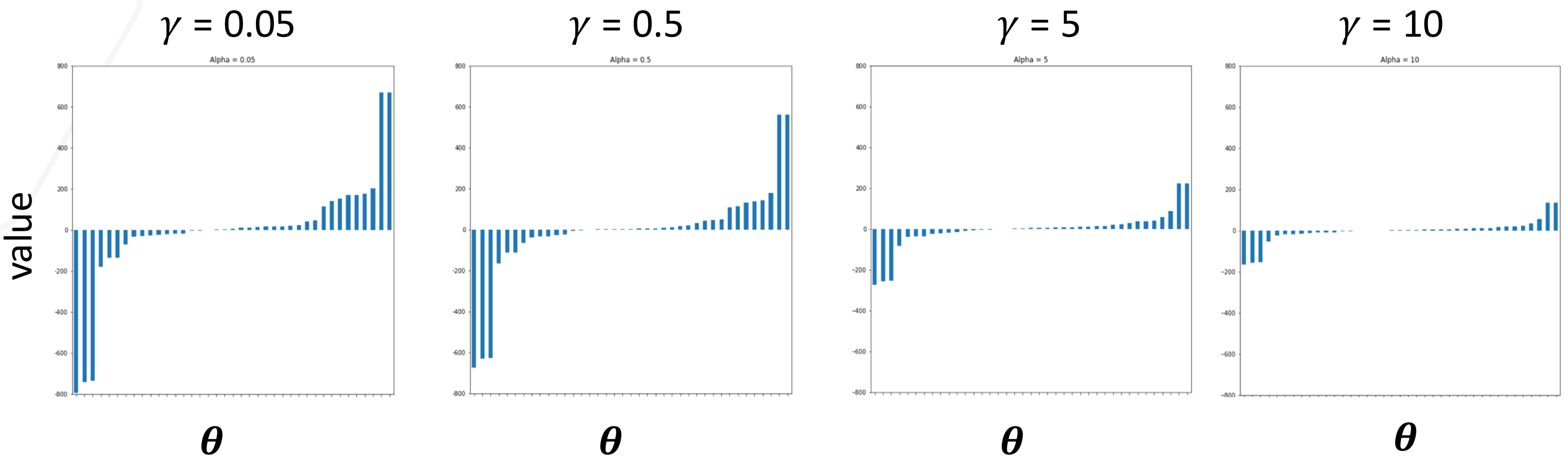


Ridge Regularization

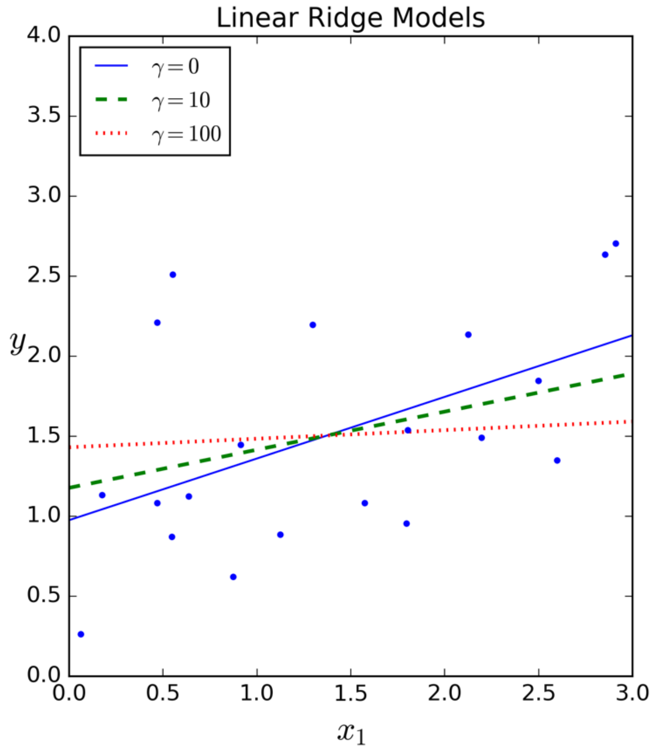
Effect of γ on θ Magnitudes

As $\gamma \downarrow$, the model gets closer to the standard linear regression

As $\gamma \uparrow$, the model gets closer to a flat line with high bias (mean of data)



γ = 0.05 γ = 0.5 γ = 5 γ = 10



Ridge Regularization

Normal Equation

- We can derive the optimal $\boldsymbol{\theta}$ of ridge regression by solving the normal equation.
- The cost function for ridge regression $L(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}_i - y_i)^2 + \frac{\gamma}{N}||\boldsymbol{\theta}||_{\mathbf{2}}^{\mathbf{2}}$ can be written in the matrix form:

$$L(\boldsymbol{\theta}) = \frac{1}{N}(\boldsymbol{X\theta} - \boldsymbol{y})^T(\boldsymbol{X\theta} - \boldsymbol{y}) + \frac{\gamma}{N}\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{\theta}$$

- The derivative is:

$$\frac{\partial L_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} = 2\boldsymbol{X}^T\boldsymbol{X\theta} - 2\boldsymbol{X}^T\boldsymbol{y} + 2\gamma\boldsymbol{\theta}$$

- when $\frac{\partial L_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} = \mathbf{0}, (\boldsymbol{X}^T\boldsymbol{X} + \gamma\mathbf{I})\boldsymbol{\theta} = \boldsymbol{X}^T\boldsymbol{y},$ we derive the optimal $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = \boxed{(\boldsymbol{X}^T\boldsymbol{X} + \gamma\mathbf{I})}^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Recall the solution for the case with no regularization: $\theta = \left(\phi(X)^T\phi(X)\right)^{-1}\phi(X)^T y$

$\boldsymbol{X}^T\boldsymbol{X} + \gamma\mathbf{I}$ is **non-singular** and always invertible for γ>0

where $\mathbf{I}$ is an identity matrix

- The above form is the direct solution of optimal coefficient vector $\boldsymbol{\theta}$ given by the Normal Equation of ridge regression

Ridge Regularization

Gradient Descent

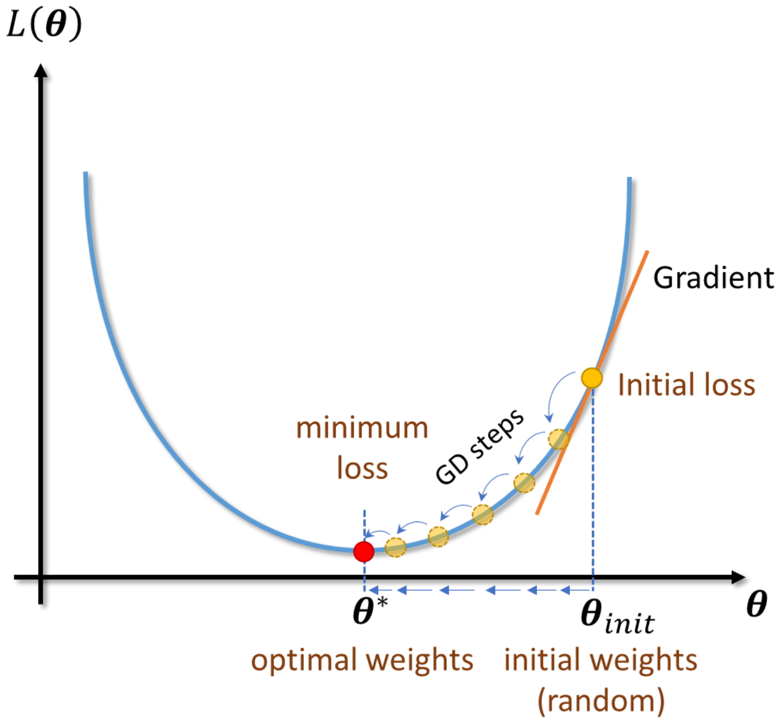
- We can also employ gradient descent algorithm to obtain the optimal $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \frac{\partial L_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}}$$

Where

$$\frac{\partial L_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} = -\frac{2}{N} \sum_{i=1}^{N} (y_i - \boldsymbol{\theta}^T \boldsymbol{x}_i)\boldsymbol{x}_i + \frac{2\gamma}{N} \boldsymbol{\theta}$$

- γ is the regularization hyper-parameter. It controls the trade-off between fitting the training data well versus penalizing the parameters to avoid overfitting.



Lasso Regularization

Formulation

- Least Absolute Shrinkage and Selection Operator (Lasso) Regression also solves a regularized least squares problem, but a different cost function via L_1 regularization:

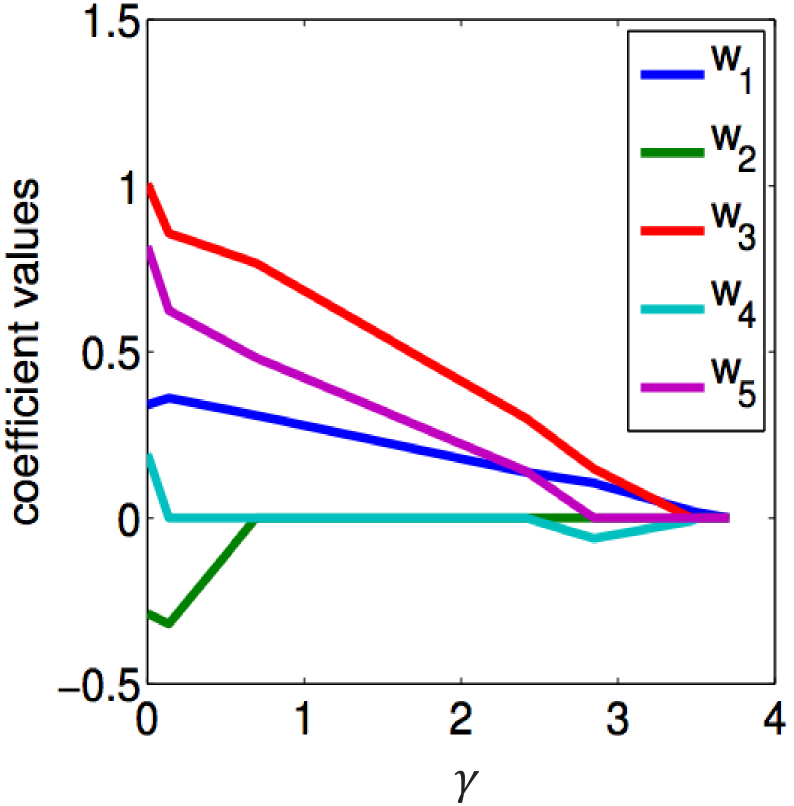
$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{\theta}^T \boldsymbol{x}_i - y_i)^2 + \gamma \sum_{j=1}^{P} |\theta_j|$$

where $\gamma > 0$ is a hyperparameter that controls the amount of regularization

- To minimize the above cost function, some of the θ_j's will be penalized to *zero exactly*. That is, Lasso produces a ***sparse model***, i.e., with few nonzero coefficients.
- Different from ridge regression, Lasso regression can perform both **feature selection** and regularization. As γ increases, more coefficients of *less important* features are set to zero.

Lasso Regularization

Feature Selection



As γ increases, coefficients (w2 and w4) of *less important* features are set to zero earlier than the coefficients (w1, w3 and w5) of *more important* features.

Lasso Regularization

Gradient Descent

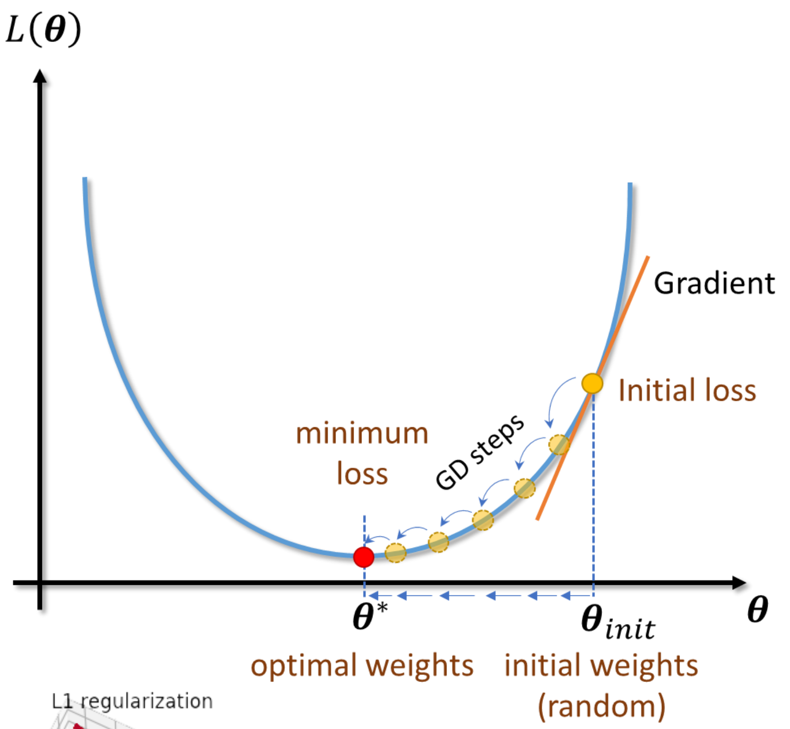
- L_1 regularization term $\gamma \sum_{j=1}^{P} |\theta_j|$ is not differentiable at $\theta_j = 0$ (for $i = 1, 2, \cdots, P$). Thus, Lasso does not have closed-form solution as Normal Equation
- Alternatively, we can employ gradient descent algorithm to obtain the optimal $\boldsymbol{\theta}$:

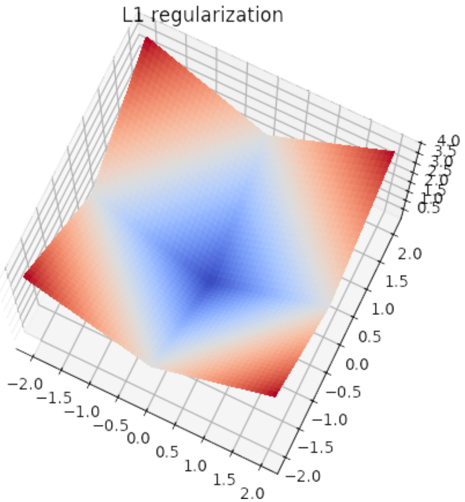
$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \frac{\partial L_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}}$$

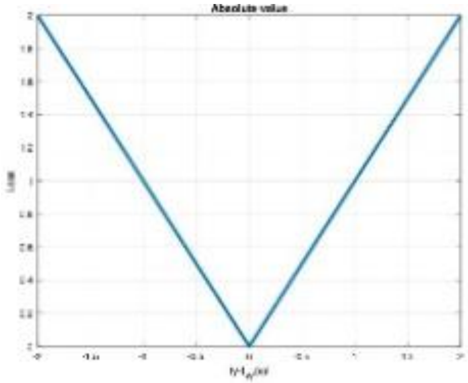
$$\frac{\partial L_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} = -\frac{2}{N} \sum_{i=1}^{N} (y_i - \boldsymbol{\theta}^T \boldsymbol{x_i}) \boldsymbol{x_i} + \gamma \begin{pmatrix} sign(\theta_1) \\ sign(\theta_2) \\ \vdots \\ sign(\theta_P) \end{pmatrix}$$

where $sign(\theta_j) = \begin{cases} 1 & \theta_j > 0 \\ 0 & \theta_j = 0 \\ -1 & \theta_j < 0 \end{cases}$

- γ is the regularization hyper-parameter. It controls the trade-off between fitting the training data well versus penalizing the parameters for feature selection and avoid overfitting.







Lasso Regularization

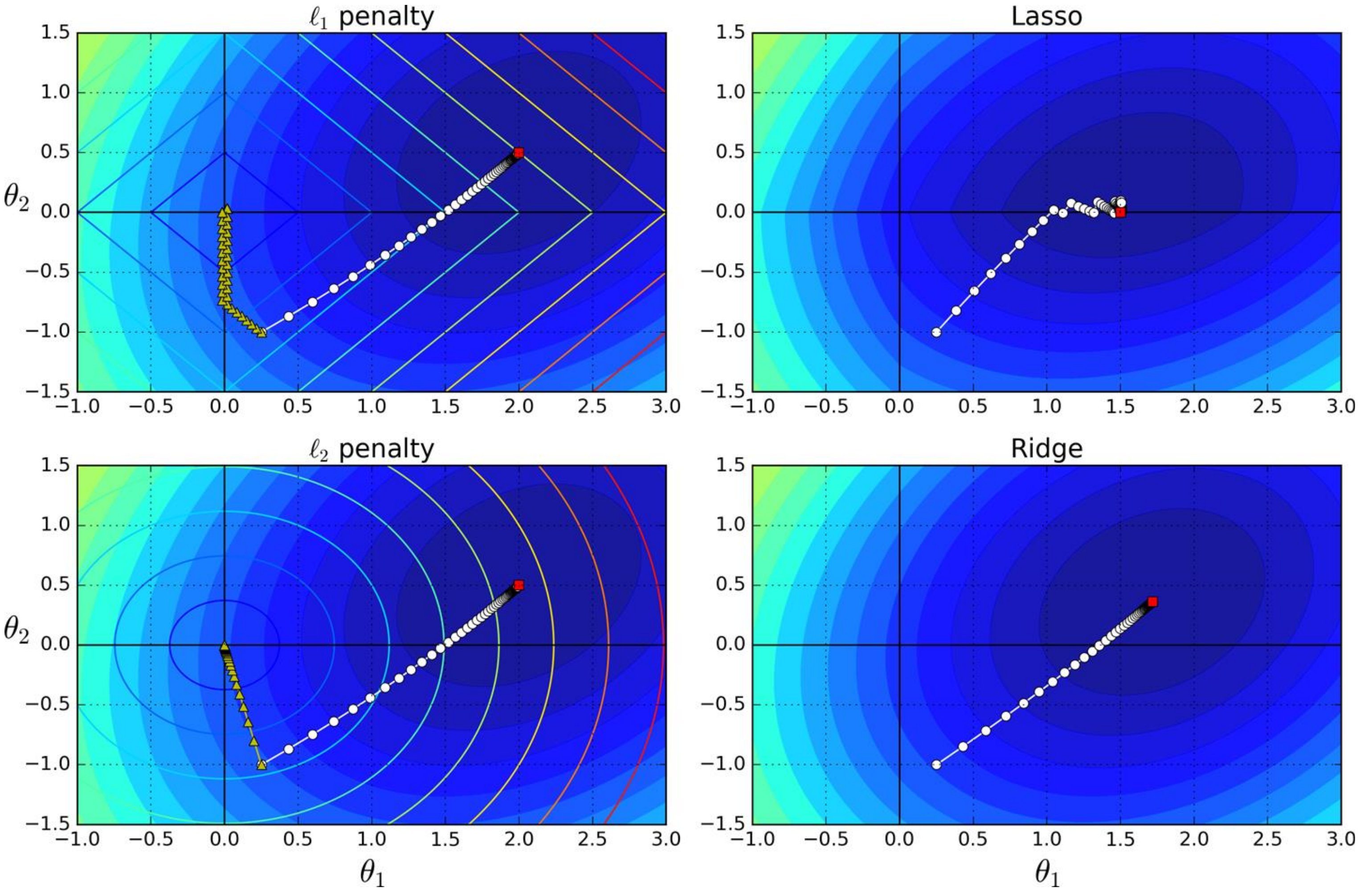
Lasso vs Ridge Regression

ℓ_1 cost only (diamonds)

GD path for ℓ_1 (triangles) first reaches $\theta_1 = 0$, then reaches $\theta_2 = 0$

ℓ_2 cost only (ovals)

GD path for ℓ_2 (triangles) moves towards 0 through radial axis



GD path for *Lasso* cost

The global minimum is on the $\theta_2 = 0$ axis

GD path for *Ridge* cost

The global minimum is closer to 0, the weights do not get fully eliminated

Elastic Net Regularization

Formulation

- The Elastic Net regression is a regularized regression method that *linearly combines* the L_1 and L_2 penalties of the lasso and ridge regression.
- The linear combination can be controlled by the ratio r

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}_i - y_i)^2 + r\gamma\sum_{j=1}^{P}|\theta_j| + (1-r)\gamma\|\boldsymbol{\theta}\|_{\boldsymbol{2}}^{\boldsymbol{2}}$$

- Elastic Net combines characteristics of both lasso and ridge. Elastic Net reduces the impact of different features to avoid overfitting while eliminating many of the features.

Overview

In This Lecture..

High-degree Polynomial Regression

Training by Gradient Descent

Regularized Regression Models

Performance Measures

- MSE and MAD
- Coefficient of Determination

Model Validation

Performance Metrics

Distance Measures

- Mean Squared Error:
  - $MSE(\boldsymbol{X},\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(\widehat{\boldsymbol{\theta}}^{\boldsymbol{T}}\boldsymbol{x}_i - y_i)^2$
- RMSE:
  - $RMSE(\boldsymbol{X},\boldsymbol{\theta}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\widehat{\boldsymbol{\theta}}^{\boldsymbol{T}}\boldsymbol{x}_i - y_i)^2}$
- Mean Absolute Error:
  - $MAD(\boldsymbol{X},\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left|\widehat{\boldsymbol{\theta}}^{\boldsymbol{T}}\boldsymbol{x}_i - y_i\right|$

- Mean Square Error (MSE): Larger errors are well noted (better than MAE). But the disadvantage is that it also squares up the units of feature as well.
- Root Mean Squared Error (RMSE): Solves the problem of squaring the units.
- Mean Absolute Error (MAE): Fails to punish large errors in prediction.
- The lower these metrics, the better the quality of the corresponding trained regression model.

Performance Metrics

Distance Measures

- Coefficient of Determination (R^2 measure)

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

where SS_{res} is the sum of squares of residuals

$$SS_{res} = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

and SS_{tot} is the total sum of squares

$$SS_{tot} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

where $\bar{y}$ is the mean of all y

- R^2 is a number in [0,1]. The closer to 1 the better
- R^2 is a measure of the ratio of variability that the model can capture versus the natural variability in the target variable.

- If $R^2 = 1$, all the data points fall perfectly on the regression line.
- If $R^2 = 0$, the estimated regression line is perfectly horizontal (mean of all y)

Overview

In This Lecture..

- High-degree Polynomial Regression
- Training by Gradient Descent
- Regularized Regression Models
- Performance Measures
- Model Validation
  - Test/Train/Validation Split
  - Learning Curves
  - Cross-Validation
  - The Bias-Variance Tradeoff

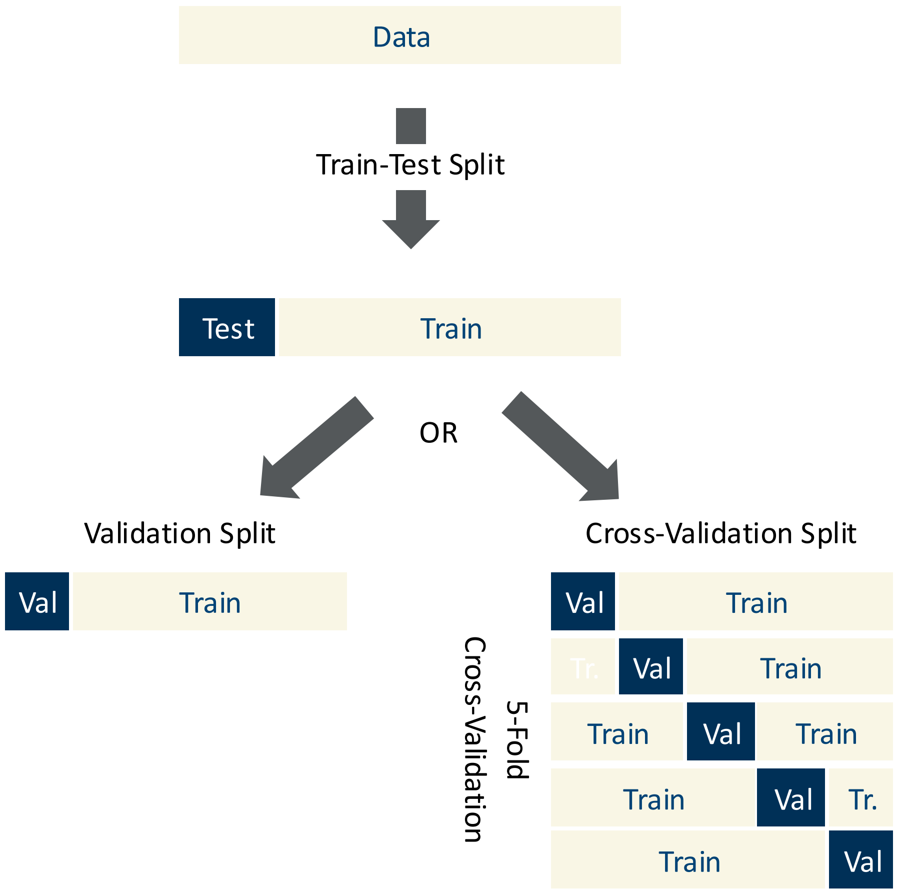
Model Validation

- Training/Validation/Test Split
  - Testing subset kept away and never used during model training
  - Training subset used for training and typically split further into Training/Validation
  - Validation subset for model validation
- Learning Curves
  - Show how *training* and *validation* scores compare as a function of increasing training set size
  - Gives insight on a model's generalization ability
- Cross-Validation
  - Used for assessing how the training results of a model will generalize to unseen data

Model Validation

Train/Test/Validation

- Test Dataset
  - Only used once a model is completely trained (using the train and validation sets)
  - Generally used to evaluate competing models
  - Contains data that spans the various classes in the real world.
- Training Dataset
  - The actual dataset use to train the model.
  - The model sees and learns from this data
- Validation Dataset
  - Used to evaluate a given model and fine-tune its *hyperparameters*
  - The model sees this data during training but never learns directly from it
  - Typically implemented as cross-validation
- Split ratio
  - Typically: %20-30% test, and the remaining training/validation



Data

Train-Test Split

Test | Train

OR

Validation Split

Val | Train

Cross-Validation Split

5-Fold Cross-Validation

Val | Train

Tr. | Val | Train

Train | Val | Train

Train | Val | Tr.

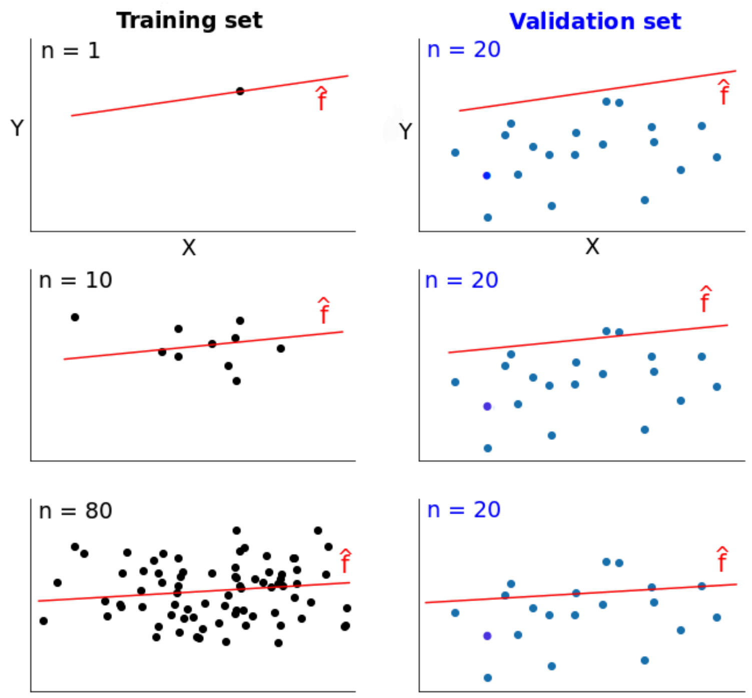
Train | Val

Model Validation

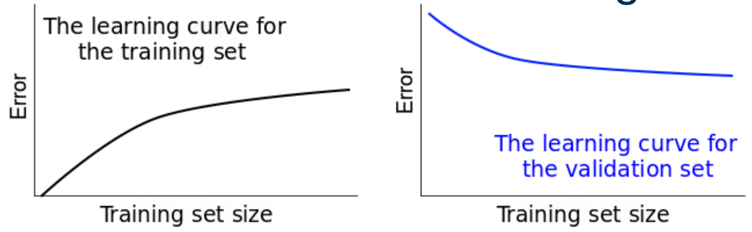
Learning Curves

- Training error starts very low when training set is small, and increases as more training data added
- Validation error starts high and decreases as more training data added.
- The general procedure for generating learning curves is as follows (assuming a dataset of size $n = 100$ samples):

1. Set aside validation set (e.g., $v = 20$ samples)
2. For $k = 1$ to $n - v$
    1. Take the first k samples as one training dataset
    2. Fit the model on the training set and evaluate it on the validation set
    3. Retain the training score and the evaluation score and discard the model
3. Plot the training and evaluation scores recorded in the iterations above against training set sizes



Conclusion: Underfitting

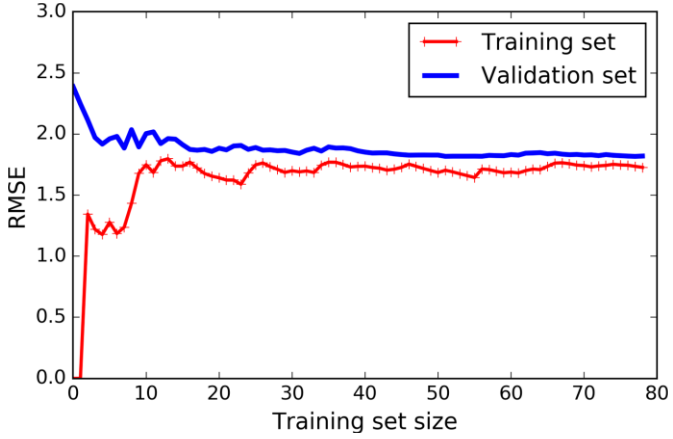


Model Validation

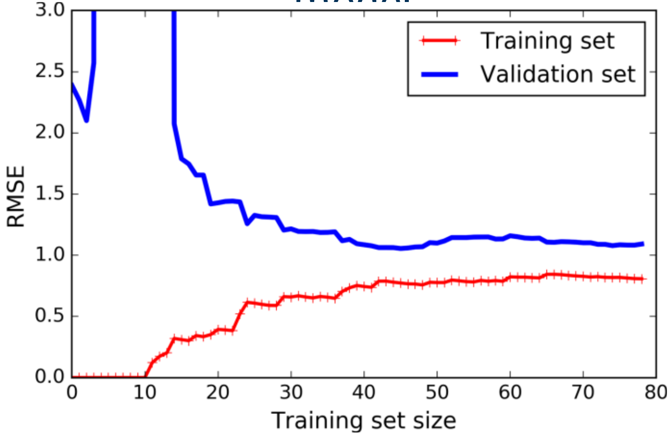
Learning Curves

- Examples of two models trained on the same data.
- Evaluation metric: RMSE
- Linear regression model (top)
  - High error rates
  - Quickly reaches a plateau both in training and validation (underfitting)
- Polynomial regression model
  - Has lower error than linear model,
  - Wider gap between the curves (overfitting)
- How to combat underfitting? Increase complexity of model
- How to combat overfitting? More training data

Linear Regression model



a 10th-degree Polynomial model

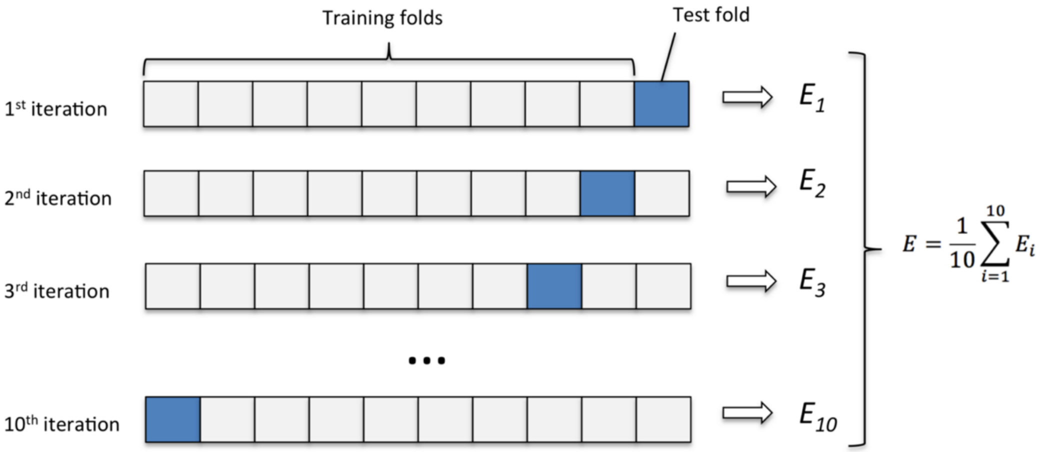


Model Validation

Cross Validation

- Helps determine hyperparameters
- More rigorous and randomized than single validation split
- A cross-validation procedure is performed for each combination of hyperparameters
- The general procedure is as follow:

```
1. Shuffle the dataset randomly
2. Split the dataset into k groups
3. For each group
   1) Take that group as a hold-out or
      validation dataset
   2) Combine the remaining k-1 groups as
      one training dataset
   3) Fit a model on the training set and
      evaluate it on the validation (hold-
      out) set
   4) Retain the evaluation scores and
      discard the model
4. Average the scores of the model to get a
   single k-fold validation score
```

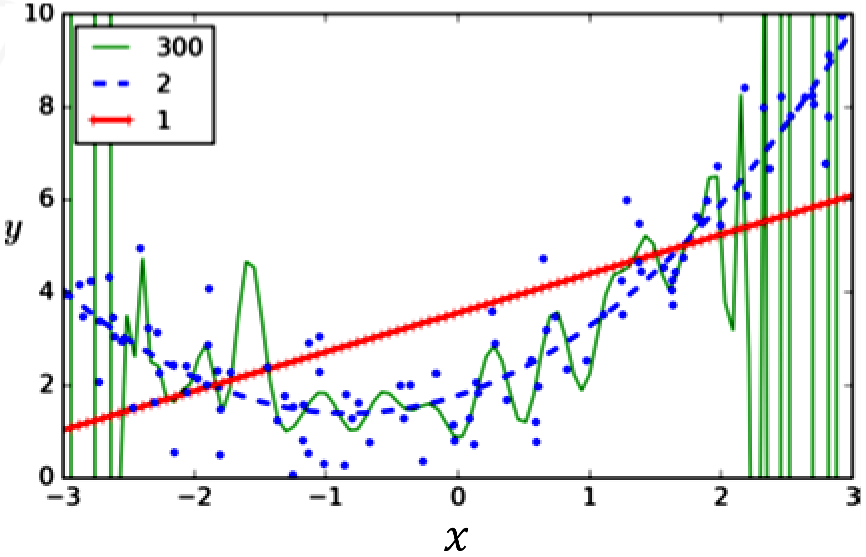


E_i is validation error

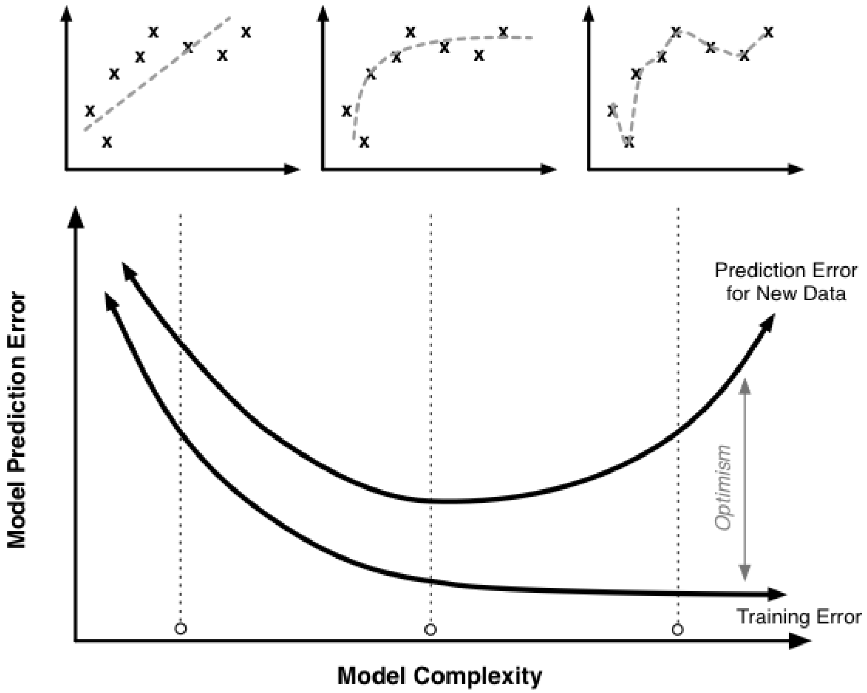
Model Validation

Model Complexity vs Prediction Error

- Model complexity increases with increasing degrees of freedom, e.g., the degree of polynomial regression
- Increasing model complexity:
  - consistently decreases training error at first
  - model starts to overfit to the training data after testing error decreases to a certain complexity level



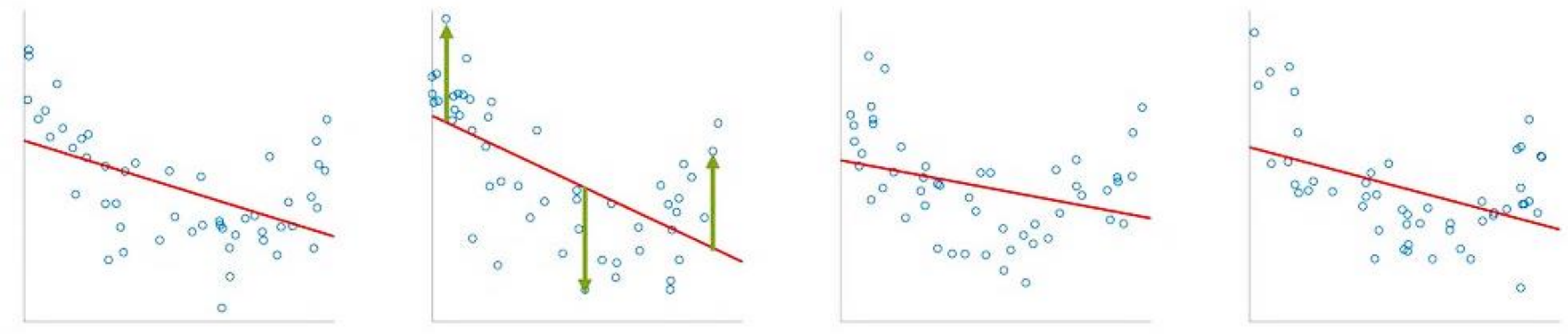
Linear regression (red)
2-degree polynomial regression (blue)
300-degree polynomial regression (green)



Model Validation

Bias-Variance Tradeoff

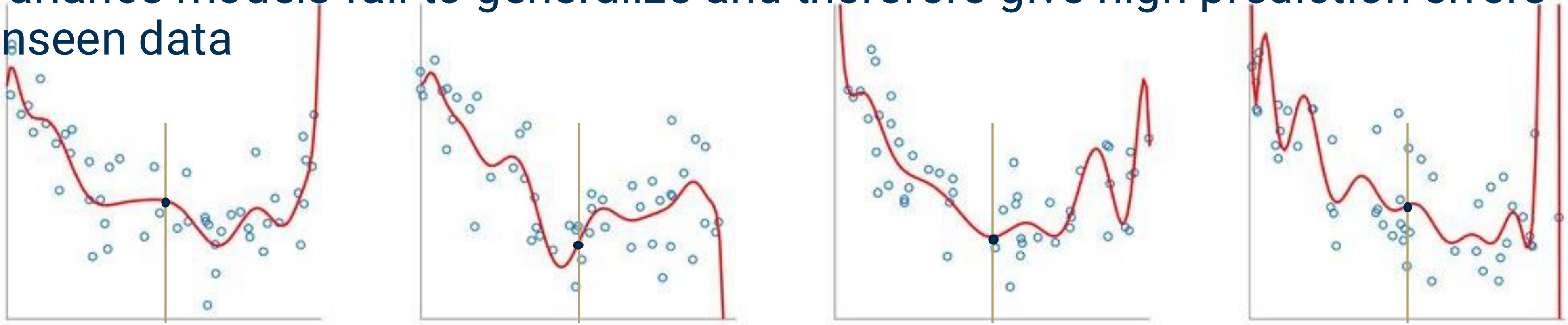
- Assuming you could train a Linear Regression model more than once, each time based on new data.
- The resulting models will have a range of prediction scores.
- Bias is the average prediction scores of these models.
- Models that give high average error are considered to have high-bias, and this is directly related to the model’s low complexity, and lack of flexibility



Model Validation

Bias-Variance Tradeoff

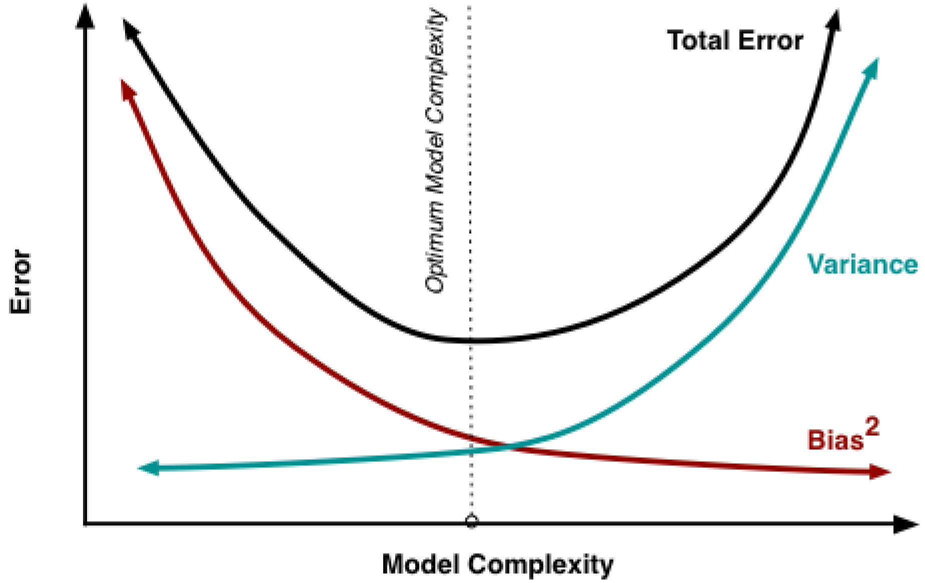
- Assume the same with a high-degree Polynomial Regression model
- Average prediction error (bias) of the resulting polynomial models will be much lower.
- However, the predictions for a given test point will vary largely between the models.
- **High-variance** occurs when a model is **too flexible** that it **overfits** to the specific data it was trained on.
- High-variance models fail to generalize and therefore give high prediction errors with unseen data



Model Validation

Bias-Variance Tradeoff

- A model's error score and its failure to generalize is directly related to its complexity (flexibility).
- Optimal model complexity is a tradeoff between Bias and Variance
  - More complexity decreases Bias
  - More complexity increases Variance

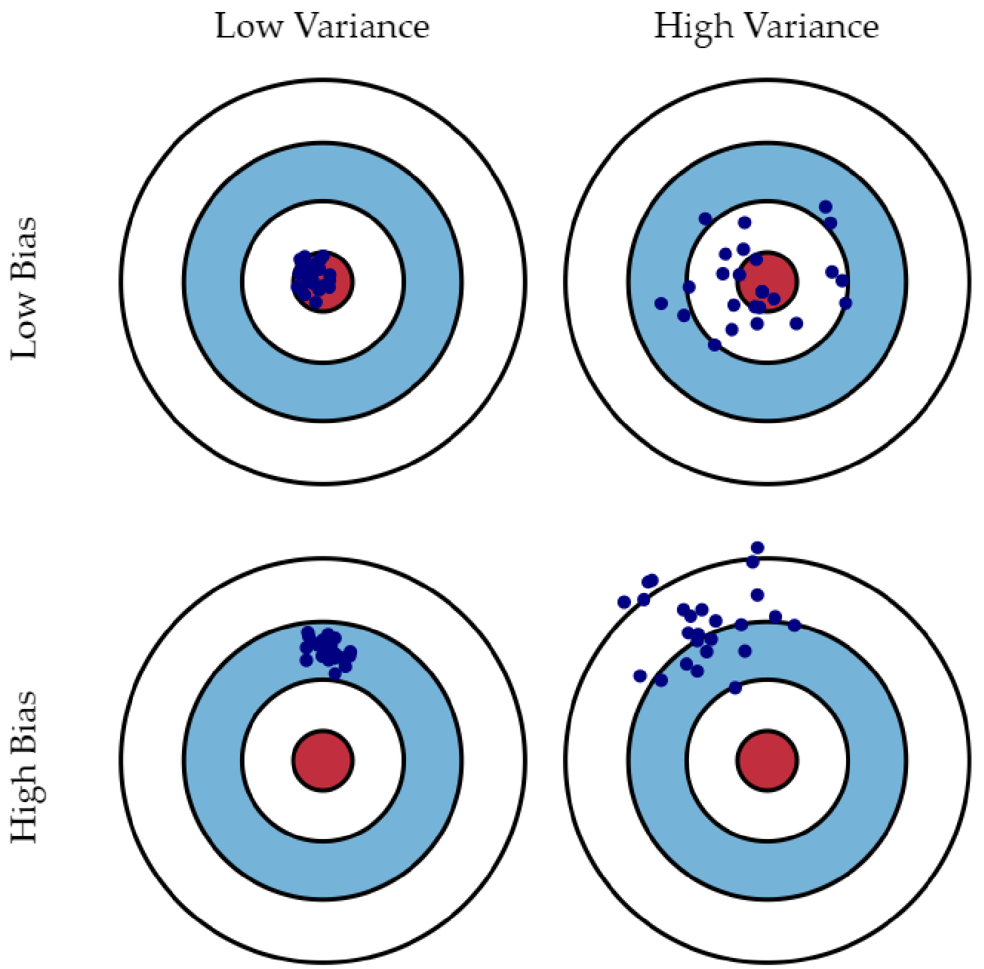


Model Validation

Bias-Variance Tradeoff

predictions close to target and don't vary much

Predictions in the vicinity of target but vary widely

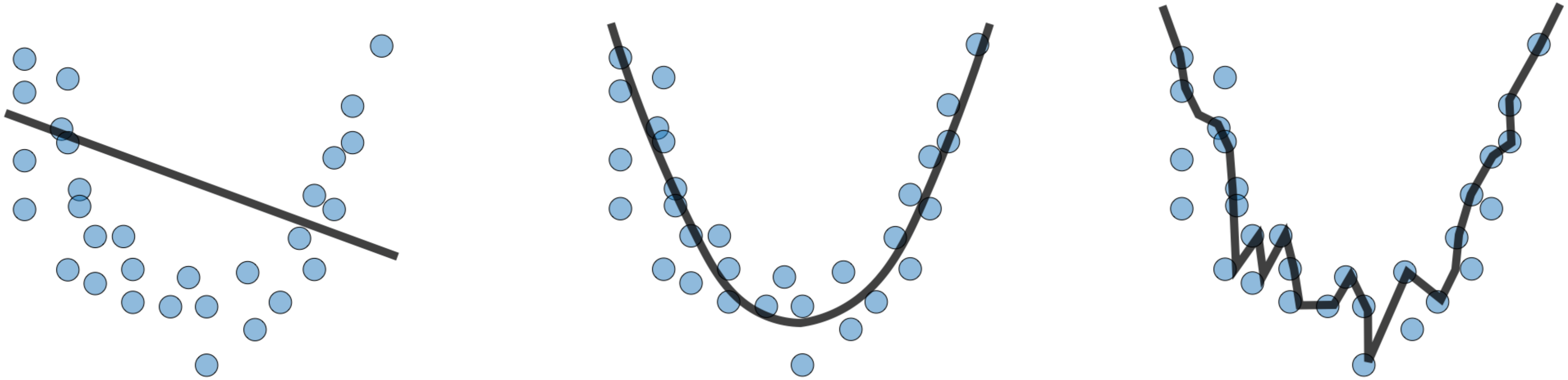


Predictions away from target but don't vary

Predictions far away from target and vary widely

Model Validation

Bias-Variance Tradeoff



	Linear Model	Low-degree Polynomial	High-degree Polynomial
Bias	VERY HIGH Linear model cannot fit non-linear data. *Underfitting*	LOW Training error slightly lower than test error	VERY LOW Model is well fixed on training data. *Overfitting*
Variance	LOW No significant difference in error when measured across different datasets	LOW Model performance on other datasets will not vary significantly.	HIGH Model will show variance performance on other datasets.

Model Validation

Bias-Variance Tradeoff

- Assume $\boldsymbol{x}$ is a test data sample and $f(\boldsymbol{x})$ its true target
- $\hat{f}(\boldsymbol{x})$ is a model's prediction of $\boldsymbol{x}$
- $E\left[\hat{f}(\boldsymbol{x})\right]$ is the average of target predictions given by the trained models
- Bias is the square of the difference between predicted and true target

$$\left(E[\hat{f}(\boldsymbol{x})] - f(\boldsymbol{x})\right)^2$$

- Variance is variance of all predictions made by the different models

$$E\left[\left(\hat{f}(\boldsymbol{x}) - E[\hat{f}(\boldsymbol{x})]\right)^2\right]$$

- The prediction error made by a model is decomposed as:

$$Error(\boldsymbol{x}) = E\left[\left(f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})\right)^2\right] = \underbrace{\left(E[\hat{f}(\boldsymbol{x})] - f(\boldsymbol{x})\right)^2}_{\text{Bias}^2} + \underbrace{E\left[\left(\hat{f}(\boldsymbol{x}) - E[\hat{f}(\boldsymbol{x})]\right)^2\right]}_{\text{Variance}} + \sigma_e^2$$

Irreducible Error

Regression Demo

The following is a visualization of how fitting a set of data points to the best linear or polynomial curve would look in 2D.

Data:

X	Y
-4	1
-2	5
5	2
6	4
13	12

Linear Regression Visualization

Polynomial Regression Vizualization

Appendix A: Notations

- x_i: a single feature
- $\boldsymbol{x}_i$: feature vector (a data sample)
- $\boldsymbol{x}_{:,i}$: feature vector of all data samples
- $\boldsymbol{X}$: matrix of feature vectors (dataset)
- N: number of data samples
- m: degree of polynomial
- P: number of features in a feature vector
- θ_i: a single model coefficient (parameter)
- $\boldsymbol{\theta}$: coefficient vector
- ε: error margin
- α: learning rate
- γ: bias factor
- Bold letter/symbol: vector
- Bold capital letters/symbol: matrix