COMPSCI 689
Lecture 6: Basis Function Expansion and Regularization

Brendan O'Connor

College of Information and Computer Sciences
University of Massachusetts Amherst

Slides by Benjamin M. Marlin (marlin@cs.umass.edu).

Outline

Review

- We now have methods for learning basic linear models for both regression and classification.

Review

- We now have methods for learning basic linear models for both regression and classification.
- **Question:** What can we do when the data we are trying to model have non-linear relationships between inputs and outputs?

Outline

Basis Function Expansion

- A simple solution to the linearity problem is to apply a set of functions $\phi_1,...,\phi_K$ to the raw feature vector $\mathbf{x}$ to map it in to a new feature space:

$$\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$$

Basis Function Expansion

- A simple solution to the linearity problem is to apply a set of functions $\phi_1,...,\phi_K$ to the raw feature vector $\mathbf{x}$ to map it in to a new feature space:

$$\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$$

- This is called a *basis function expansion* since $K > D$ in general. This requires that we know the functions $\phi_1,...,\phi_K$ that we want to apply in advance. (*Feature engineering*: manually experimenting to develop ϕs)

Basis Function Expansion

- A simple solution to the linearity problem is to apply a set of functions $\phi_1,...,\phi_K$ to the raw feature vector $\mathbf{x}$ to map it in to a new feature space:

$$\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$$

- This is called a *basis function expansion* since $K > D$ in general. This requires that we know the functions $\phi_1,...,\phi_K$ that we want to apply in advance. (*Feature engineering*: manually experimenting to develop ϕs)
- Then define a linear model in this new feature space: $\theta \in \mathbb{R}^K$.

Basis Function Expansion

- A simple solution to the linearity problem is to apply a set of functions $\phi_1,...,\phi_K$ to the raw feature vector $\mathbf{x}$ to map it in to a new feature space:

$$\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$$

- This is called a *basis function expansion* since $K > D$ in general. This requires that we know the functions $\phi_1,...,\phi_K$ that we want to apply in advance. (*Feature engineering*: manually experimenting to develop ϕs)
- Then define a linear model in this new feature space: $\theta \in \mathbb{R}^K$.
- In the regression setting, we obtain the model $\hat{y} = \phi(\mathbf{x})\theta$.

Basis Function Expansion

- A simple solution to the linearity problem is to apply a set of functions $\phi_1,...,\phi_K$ to the raw feature vector $\mathbf{x}$ to map it in to a new feature space:

$$\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$$

- This is called a *basis function expansion* since $K > D$ in general. This requires that we know the functions $\phi_1,...,\phi_K$ that we want to apply in advance. (*Feature engineering*: manually experimenting to develop ϕs)
- Then define a linear model in this new feature space: $\theta \in \mathbb{R}^K$.
- In the regression setting, we obtain the model $\hat{y} = \phi(\mathbf{x})\theta$.
- In the classification setting, we obtain the model $\hat{y} = \text{sign}(\phi(\mathbf{x})\theta)$.

Basis Function Expansion Examples

- **Univariate Functions:** We can set $\phi_k(\mathbf{x})$ to any univariate function of a single x_d to obtain mappings like $\phi(\mathbf{x}) = [x_1, x_2, \sin(x_1), \exp(x_2)]$, etc.

Basis Function Expansion Examples

- **Univariate Functions:** We can set $\phi_k(\mathbf{x})$ to any univariate function of a single x_d to obtain mappings like $\phi(\mathbf{x}) = [x_1, x_2, \sin(x_1), \exp(x_2)]$, etc.
- **Degree 2 Polynomial Basis:** We include all single features x_d, their squares x_d^2, and all products of two distinct features $x_d x_{d'}$.

"Interaction terms"

Basis Function Expansion Examples

- **Univariate Functions:** We can set $\phi_k(\mathbf{x})$ to any univariate function of a single x_d to obtain mappings like $\phi(\mathbf{x}) = [x_1, x_2, \sin(x_1), \exp(x_2)]$, etc.
- **Degree 2 Polynomial Basis:** We include all single features x_d, their squares x_d^2, and all products of two distinct features $x_d x_{d'}$.
- **Degree B Polynomial Basis:** We include all single features x_d, and all unique products of between 2 and B features.

Basis Function Expansion Examples

- **Univariate Functions:** We can set $\phi_k(\mathbf{x})$ to any univariate function of a single x_d to obtain mappings like $\phi(\mathbf{x}) = [x_1, x_2, \sin(x_1), \exp(x_2)]$, etc.
- **Degree 2 Polynomial Basis:** We include all single features x_d, their squares x_d^2, and all products of two distinct features $x_d x_{d'}$.
- **Degree B Polynomial Basis:** We include all single features x_d, and all unique products of between 2 and B features.
- Don't forget that basis expansion still requires a bias term in the model!

Basis Function Expansion Examples

- **Univariate Functions:** We can set $\phi_k(\mathbf{x})$ to any univariate function of a single x_d to obtain mappings like $\phi(\mathbf{x}) = [x_1, x_2, \sin(x_1), \exp(x_2)]$, etc.
- **Degree 2 Polynomial Basis:** We include all single features x_d, their squares x_d^2, and all products of two distinct features $x_d x_{d'}$.
- **Degree B Polynomial Basis:** We include all single features x_d, and all unique products of between 2 and B features.
- Don't forget that basis expansion still requires a bias term in the model!
- A key question is how complex should we let the basis expanded model be?

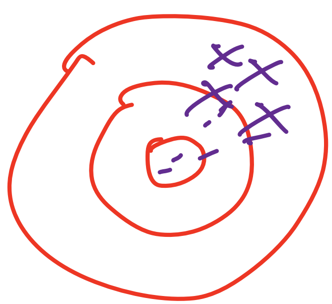
Bias-Variance Trade-Off

- **Bias:** A model is said to have low *bias* if the optimal prediction function can be approximated closely by the model.

Bias-Variance Trade-Off

- **Bias:** A model is said to have low *bias* if the optimal prediction function can be approximated closely by the model.

- **Variance:** A model is said to have low *variance* if the prediction function it constructs is stable with respect to small changes to the training data.
  [Illustration:]



High bias
Low var

Low bias
High var

Bias-Variance Trade-Off

- **Bias:** A model is said to have low *bias* if the optimal prediction function can be approximated closely by the model.
- **Variance:** A model is said to have low *variance* if the prediction function it constructs is stable with respect to small changes to the training data.
  [Illustration:]
- **Bias-Variance Dilemma:** To achieve low generalization error, we need models that are low-bias and low-variance, but this isn't possible for complex data.

Overfitting and Underfitting

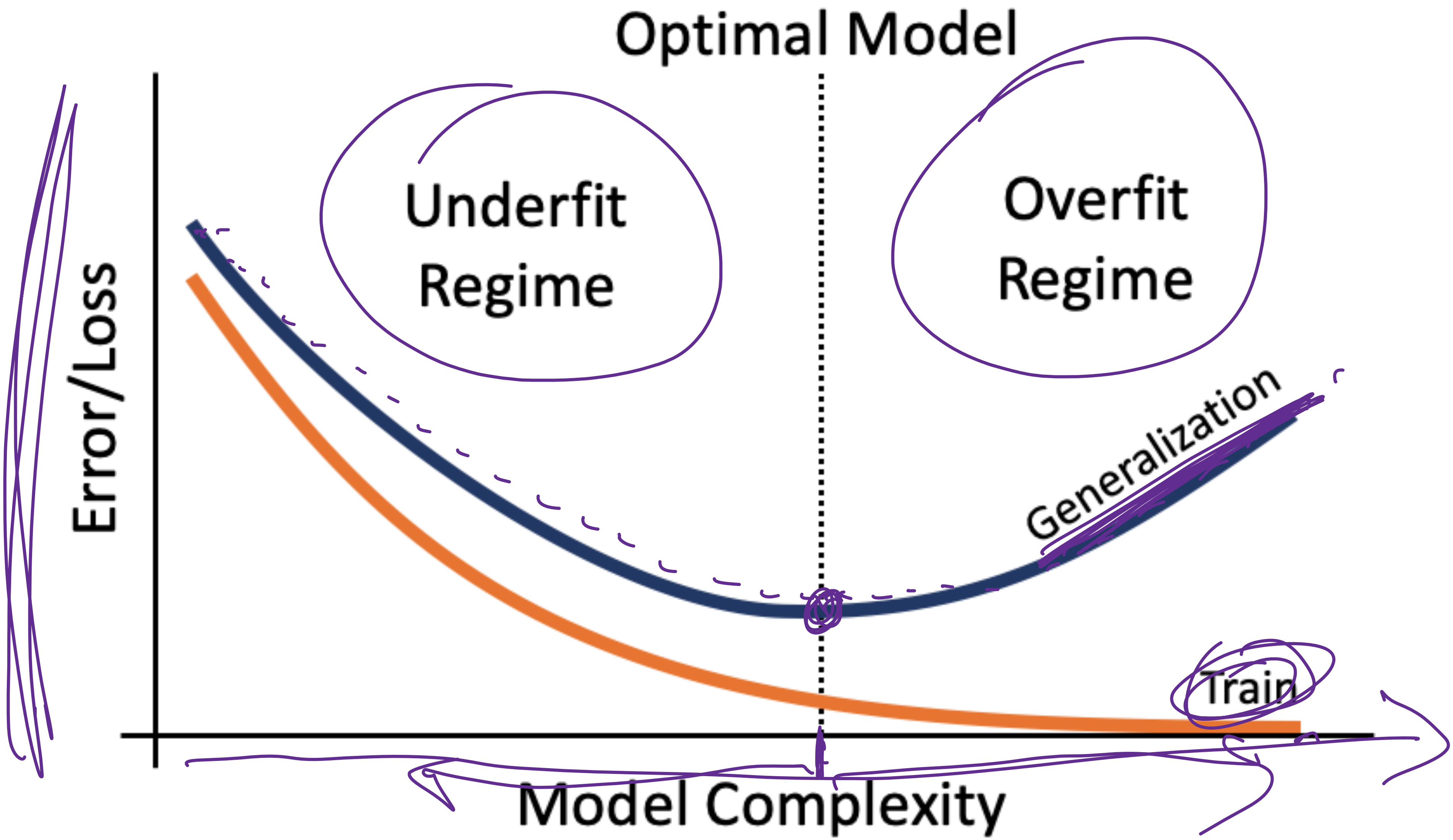
- **Underfitting:** Occurs when the complexity of the model is too low to capture the actual structure in the data, leading to both high training loss and high generalization loss.

Overfitting and Underfitting

- **Underfitting:** Occurs when the complexity of the model is too low to capture the actual structure in the data, leading to both high training loss and high generalization loss.

- **Overfitting:** Occurs when the complexity of the model is too high so the model is able to closely fit random variations in the data. Results is low training loss and much higher generalization loss.

Overfitting and Underfitting



Outline

Controlling Complexity

- There are two basic ways to control the complexity of a basis expanded model.

Controlling Complexity

- There are two basic ways to control the complexity of a basis expanded model.
- The first approach is to control the complexity of the terms used in the basis expansion.

Controlling Complexity

- There are two basic ways to control the complexity of a basis expanded model.
- The first approach is to control the complexity of the terms used in the basis expansion.
- The second approach is to control the magnitude of the weights.

Control of the Basis Expansions

- Assume that the elements in a candidate basis expansion $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$ are ordered in terms of increasing complexity (for example, increasing polynomial powers).

Control of the Basis Expansions

- Assume that the elements in a candidate basis expansion $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_K(\mathbf{x})]$ are ordered in terms of increasing complexity (for example, increasing polynomial powers).

- To select an appropriate complexity for the basis expanded model, we construct a sequence of models $f_\theta^1, \ldots, f_\theta^K$ where model f_θ^j uses the partial basis expansion $\phi^j(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_j(\mathbf{x})]$.

Control of the Basis Expansions

- Assume that the elements in a candidate basis expansion $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_K(\mathbf{x})]$ are ordered in terms of increasing complexity (for example, increasing polynomial powers).

- To select an appropriate complexity for the basis expanded model, we construct a sequence of models $f_\theta^1, \ldots, f_\theta^K$ where model f_θ^j uses the partial basis expansion $\phi^j(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_j(\mathbf{x})]$.

- We then select the model f_θ^k from this set of candidates that exhibits the best generalization loss.

Control of the Basis Expansions

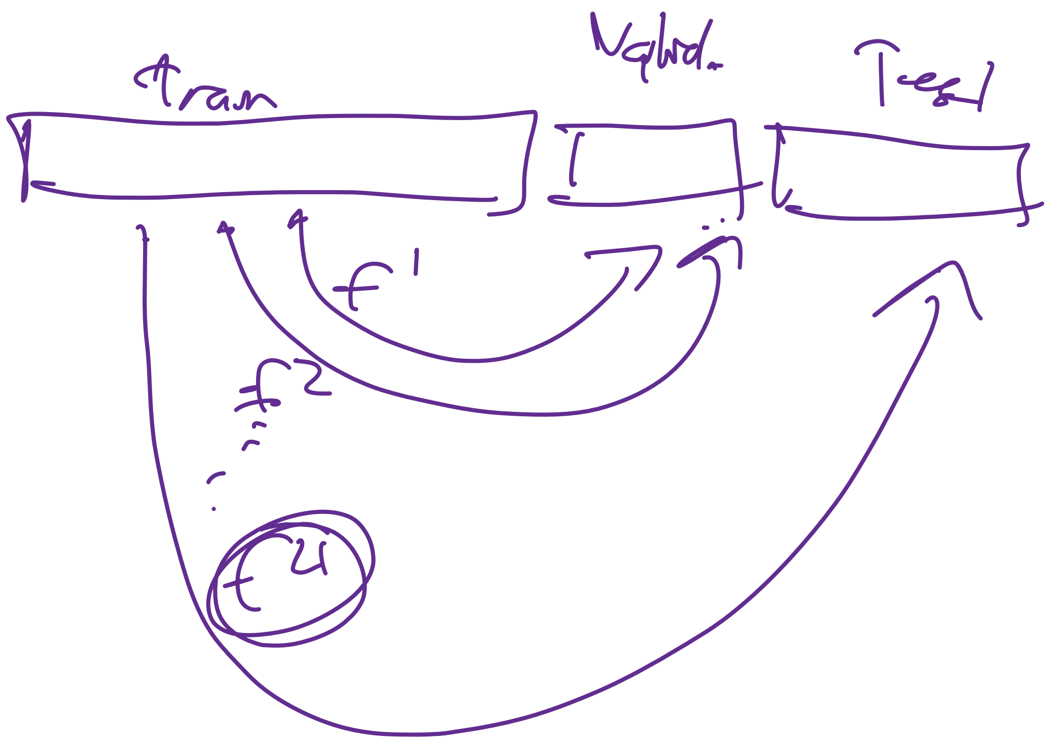
- Assume that the elements in a candidate basis expansion $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$ are ordered in terms of increasing complexity (for example, increasing polynomial powers).
- To select an appropriate complexity for the basis expanded model, we construct a sequence of models $f_\theta^1, ..., f_\theta^K$ where model f_θ^j uses the partial basis expansion $\phi^j(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_j(\mathbf{x})]$.
- We then select the model f_θ^k from this set of candidates that exhibits the best generalization loss.
- We refer to the parameter k as a model complexity *hyper-parameter*. The problem of selecting the optimal value of k is referred to as a *hyper-parameter* selection problem.

Hyper-Parameter Selection

- We previously saw how to estimate generalization performance using a basic train-test experiment.

Hyper-Parameter Selection

- We previously saw how to estimate generalization performance using a basic train-test experiment.
- To select the optimal hyper-parameter value, we split the data set into a training set and a *validation* set.



Hyper-Parameter Selection

- We previously saw how to estimate generalization performance using a basic train-test experiment.
- To select the optimal hyper-parameter value, we split the data set into a training set and a *validation* set.
- We learn the model parameters using the training set for each model f_θ^j.

Hyper-Parameter Selection

- We previously saw how to estimate generalization performance using a basic train-test experiment.
- To select the optimal hyper-parameter value, we split the data set into a training set and a *validation* set.
- We learn the model parameters using the training set for each model f^j_θ.
- We then estimate the generalization loss of each fit model $f^j_{\hat{\theta}}$ using the validation set and select the value of j with the lowest estimated value of the generalization loss.

Hyper-Parameter Selection and Performance Estimation

- To get an unbiased estimate of the generalization loss of the model with the optimal hyper-parameters, we need to use a separate test set to estimate its generalization error.

Hyper-Parameter Selection and Performance Estimation

- To get an unbiased estimate of the generalization loss of the model with the optimal hyper-parameters, we need to use a separate test set to estimate its generalization error.
- The basic experiment design to accomplish this requires three data sets: a training set to estimate model parameters, the validation set to select the optimal hyper-parameters, and a test set to estimate the generalization performance of the model with the optimal hyper-parameters.

Hyper-Parameter Selection and Performance Estimation

- To get an unbiased estimate of the generalization loss of the model with the optimal hyper-parameters, we need to use a separate test set to estimate its generalization error.
- The basic experiment design to accomplish this requires three data sets: a training set to estimate model parameters, the validation set to select the optimal hyper-parameters, and a test set to estimate the generalization performance of the model with the optimal hyper-parameters.
- This experiment design is referred to as *Train-Validation-Test* experiment.

Controlling Weights

- The other approach to controlling the complexity of basis expanded linear models (and many other models) is to control the magnitude of their weights.

Controlling Weights

- The other approach to controlling the complexity of basis expanded linear models (and many other models) is to control the magnitude of their weights.
- In linear model, smaller magnitude weights usually represent “smoother" functions.

Controlling Weights

- The other approach to controlling the complexity of basis expanded linear models (and many other models) is to control the magnitude of their weights.
- In linear model, smaller magnitude weights usually represent “smoother" functions.
- Controlling model complexity via weight magnitudes gives us a complementary approach to selecting from a discrete set of models of different complexity.

Regularization

- One way to control the magnitude of the weights in a basis expanded linear model (and many other models) is to add a term to the optimization objective function that penalizes solutions where the weights have large magnitudes.

Regularization

- One way to control the magnitude of the weights in a basis expanded linear model (and many other models) is to add a term to the optimization objective function that penalizes solutions where the weights have large magnitudes.

- The two most common ways to measure the magnitude of the weights are via the ℓ_1 norm $\|\mathbf{w}\|_1$ and squared ℓ_2 norm of the weights $\|\mathbf{w}\|_2^2$.

$\sum_d |w_d|$

$\sum_d w_d^2$

Regularization

- One way to control the magnitude of the weights in a basis expanded linear model (and many other models) is to add a term to the optimization objective function that penalizes solutions where the weights have large magnitudes.

- The two most common ways to measure the magnitude of the weights are via the ℓ_1 norm $\|\mathbf{w}\|_1$ and squared ℓ_2 norm of the weights $\|\mathbf{w}\|_2^2$.

- These regularization functions can equivalently be thought of as penalizing the distance (under the corresponding norm) of the weights from 0.

Regularization

- One way to control the magnitude of the weights in a basis expanded linear model (and many other models) is to add a term to the optimization objective function that penalizes solutions where the weights have large magnitudes.

- The two most common ways to measure the magnitude of the weights are via the ℓ_1 norm $\|\mathbf{w}\|_1$ and squared ℓ_2 norm of the weights $\|\mathbf{w}\|_2^2$.

- These regularization functions can equivalently be thought of as penalizing the distance (under the corresponding norm) of the weights from 0.

- Note that we typically do not regularize the bias term in the model.

Visualization: L_p norm regularization

Visualization: L_p norm regularization

Regularized Risk Minimization

- When regularization is applied in the expected risk minimization framework, the resulting framework is referred to as *regularized risk minimization* (RRM).

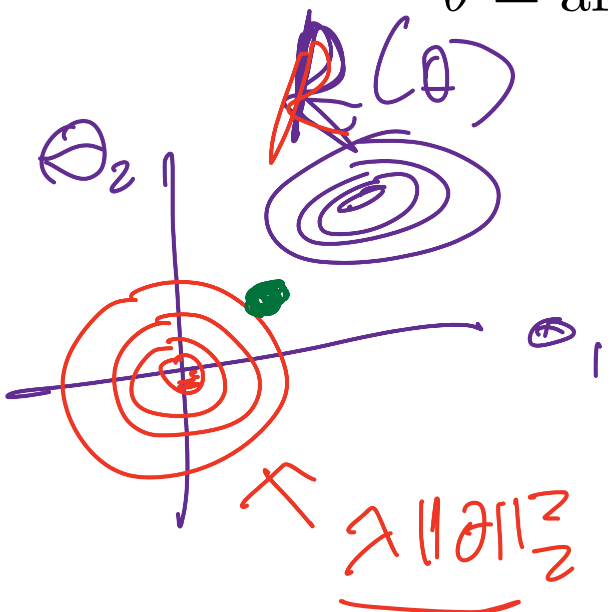
Regularized Risk Minimization

- When regularization is applied in the expected risk minimization framework, the resulting framework is referred to as *regularized risk minimization* (RRM).

- Below, we show the RRM objective function for the case where the regularization function is based on the squared two norm:

hyper

$$\hat{\theta} = \underset{\theta \in \Theta}{\arg\min} \left(\underbrace{\left(\sum_{n=1}^{N} L(y_n, f_\theta(\mathbf{x}_n)) \right)}_{\text{Train loss}} + \lambda \sum_{k=1}^{K} w_k^2 \right)$$



$\lambda = 0 \Rightarrow$ no regul. (OLS)

$\lambda = 1000 \Rightarrow \vec{w} \approx 0$

Regularized Risk Minimization

- When regularization is applied in the expected risk minimization framework, the resulting framework is referred to as *regularized risk minimization* (RRM).
- Below, we show the RRM objective function for the case where the regularization function is based on the squared two norm:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \left(\left(\sum_{n=1}^{N} L(y_n, f_\theta(\mathbf{x}_n)) \right) + \lambda \sum_{k=1}^{K} w_k^2 \right)$$

- Note that λ serves as the model complexity or regularization hyper-parameter. When λ is large, the regularization term encourages the weights to be small and they will eventually be driven to 0.

Regularized Risk Minimization

- When regularization is applied in the expected risk minimization framework, the resulting framework is referred to as *regularized risk minimization* (RRM).
- Below, we show the RRM objective function for the case where the regularization function is based on the squared two norm:

$$\hat{\theta} = \underset{\theta \in \Theta}{\arg\min} \left(\left(\sum_{n=1}^{N} L(y_n, f_\theta(\mathbf{x}_n)) \right) + \lambda \sum_{k=1}^{K} w_k^2 \right)$$

- Note that λ serves as the model complexity or regularization hyper-parameter. When λ is large, the regularization term encourages the weights to be small and they will eventually be driven to 0.
- When $\lambda = 0$, we recover ERM and the weights are free to take arbitrary values.

Selecting the Regularization Hyper-Parameter

- To select a regularization hyper-parameter, we can again apply a Train-Validation procedure.

Selecting the Regularization Hyper-Parameter

- To select a regularization hyper-parameter, we can again apply a Train-Validation procedure.

- It is typical to use an exponential grid of values for λ such as $\lambda \in \{10^{-4}, 10^{-3}, ..., 10^{1}, 10^{2}\}$.

Selecting the Regularization Hyper-Parameter

- To select a regularization hyper-parameter, we can again apply a Train-Validation procedure.
- It is typical to use an exponential grid of values for λ such as $\lambda \in \{10^{-4}, 10^{-3}, ..., 10^{1}, 10^{2}\}$.
- When running such an experiment with a given range of values for λ, it is important to check that the optimal value over the range is not achieved at the minimum or maximum value tested (if it is, the range needs to be extended and the experiment re-run).

Selecting the Regularization Hyper-Parameter

- To select a regularization hyper-parameter, we can again apply a Train-Validation procedure.
- It is typical to use an exponential grid of values for λ such as $\lambda \in \{10^{-4}, 10^{-3}, ..., 10^{1}, 10^{2}\}$.
- When running such an experiment with a given range of values for λ, it is important to check that the optimal value over the range is not achieved at the minimum or maximum value tested (if it is, the range needs to be extended and the experiment re-run).
- As noted previously, to get a valid estimate of the generalization performance of the model with the optimal hyper-parameters, it is necessary to run a Train-Validation-Test experiment.

Nested optimization

When considering held-out loss and parameters vs. hyperparameters, the broader learning problem is quite complex. Typically only the parameters for training loss are differentiable.

Outline

Summary

- Basis expansion gives us a powerful and useful tool for constructing models that are non-linear in their original feature space while remaining linear in their parameters.

Summary

- Basis expansion gives us a powerful and useful tool for constructing models that are non-linear in their original feature space while remaining linear in their parameters.

- A key property of this approach is that we need to have prior knowledge of a good set of potential basis expansion elements.

Summary

- Basis expansion gives us a powerful and useful tool for constructing models that are non-linear in their original feature space while remaining linear in their parameters.
- A key property of this approach is that we need to have prior knowledge of a good set of potential basis expansion elements.
- Importantly, we also need to apply model complexity control methodology to balance bias and variance and thus optimize generalization performance.

Summary

- Basis expansion gives us a powerful and useful tool for constructing models that are non-linear in their original feature space while remaining linear in their parameters.
- A key property of this approach is that we need to have prior knowledge of a good set of potential basis expansion elements.
- Importantly, we also need to apply model complexity control methodology to balance bias and variance and thus optimize generalization performance.
- The main drawback with this approach is that while it can select from pre-specified elements, it can not learn useful basis expansion elements directly from data.