Bias / Variance Analysis, Kernel Methods

Professor Ameet Talwalkar

Outline

Announcements

- HW3 due now
- HW4 will online by Wednesday (due on 2/27)
- Midterm is on Wednesday

Midterm

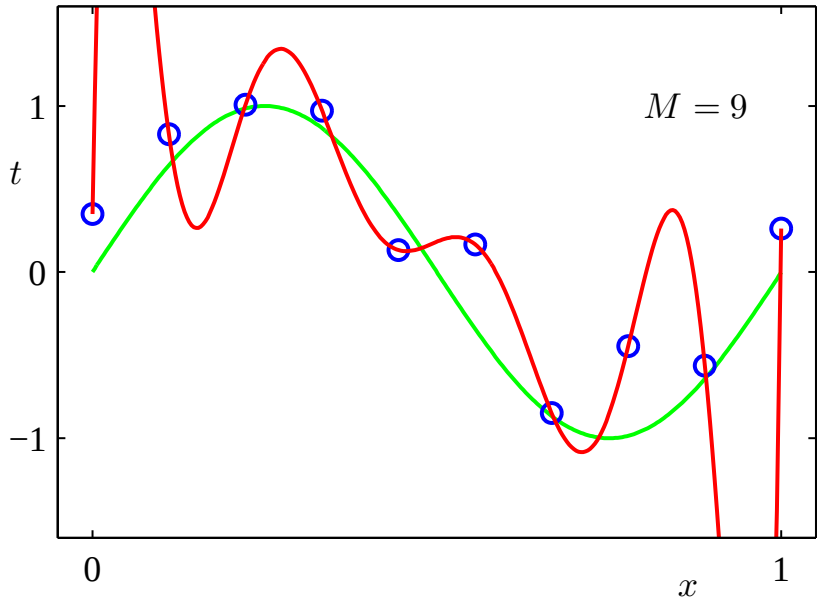
- In-class from 10am - 11:50am
- Completely closed-book (no notes allowed)
- 6 short answer questions and 3 long questions
  - Short questions should take 5 minutes on average
  - Long questions should take 15 minutes each
- Covers all material through (and including) last Wednesday's class (i.e., today's review material)
  - Goal is to test conceptual understanding of the course material
  - Suggestion: carefully review lecture notes and problem sets

Outline

Overfitting

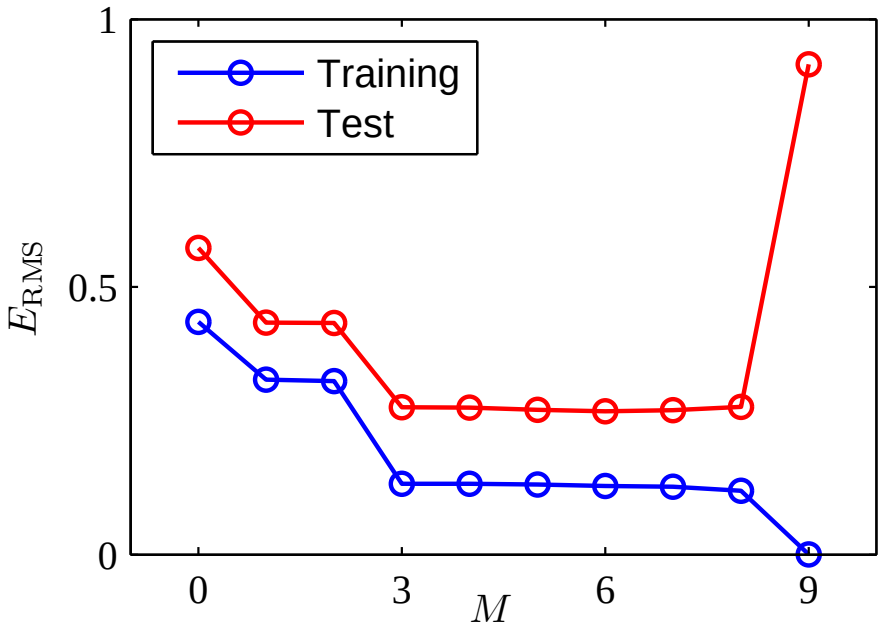
Example with regression, using polynomial basis functions

$$\boldsymbol{\phi}(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix} \Rightarrow f(x) = w_0 + \sum_{m=1}^{M} w_m x^m$$

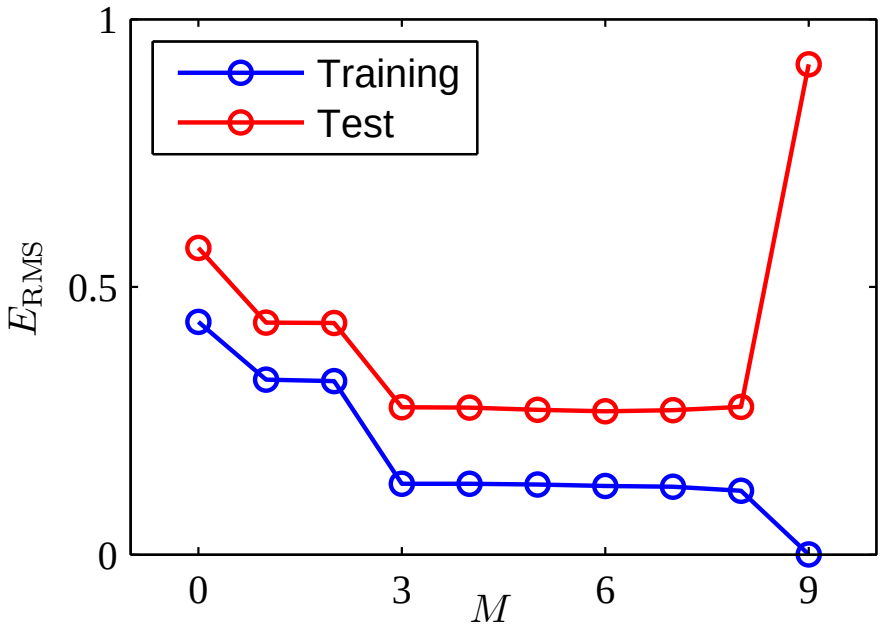


Can improve training accuracy at the expense of test accuracy

Visualizing overfitting



Visualizing overfitting

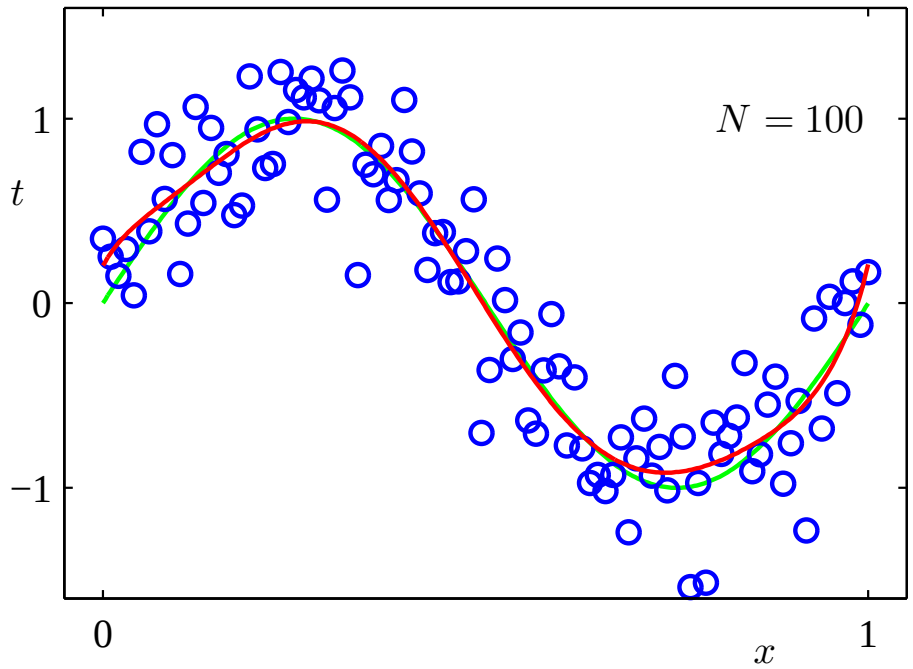


- As model becomes more complex, training error keeps improving while test error first improves then deteriorates

How to prevent overfitting?

How to prevent overfitting?

Use more training data



Regularization: adding a term to the objective function

$$\lambda \|\boldsymbol{w}\|_2^2$$

that favors a small parameter vector $\boldsymbol{w}$.

Probabilistic interpretation of Ridge Regression

- Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$
  - $Y \sim N(\boldsymbol{w}^\top \boldsymbol{X}, \sigma_0^2)$ is a Gaussian random variable (as before)
  - $w_d \sim N(0, \sigma^2)$ are i.i.d. Gaussian random variables (unlike before)
  - Note that all w_d share the same variance σ^2

Probabilistic interpretation of Ridge Regression

- Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$
  - $Y \sim N(\boldsymbol{w}^\top \boldsymbol{X}, \sigma_0^2)$ is a Gaussian random variable (as before)
  - $w_d \sim N(0, \sigma^2)$ are i.i.d. Gaussian random variables (unlike before)
  - Note that all w_d share the same variance σ^2
- $\boldsymbol{w}$ is a random variable (*Bayesian* interpretation)
- To find $\boldsymbol{w}$ given data $\mathcal{D}$, we can compute posterior distribution of $\boldsymbol{w}$:

$$p(\boldsymbol{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w})}{p(\mathcal{D})}$$

Probabilistic interpretation of Ridge Regression

- Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$
  - $Y \sim N(\boldsymbol{w}^\top \boldsymbol{X}, \sigma_0^2)$ is a Gaussian random variable (as before)
  - $w_d \sim N(0, \sigma^2)$ are i.i.d. Gaussian random variables (unlike before)
  - Note that all w_d share the same variance σ^2
- $\boldsymbol{w}$ is a random variable (*Bayesian* interpretation)
- To find $\boldsymbol{w}$ given data $\mathcal{D}$, we can compute posterior distribution of $\boldsymbol{w}$:

$$p(\boldsymbol{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w})}{p(\mathcal{D})}$$

- Maximum a posterior (MAP) estimate:

$$\boldsymbol{w}^{\text{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\mathcal{D}) = \arg\max_{\boldsymbol{w}} p(\mathcal{D}, \boldsymbol{w})$$

Probabilistic interpretation of Ridge Regression

- Ridge Regression model: $Y = \boldsymbol{w}^\top \boldsymbol{X} + \eta$
  - $Y \sim N(\boldsymbol{w}^\top \boldsymbol{X}, \sigma_0^2)$ is a Gaussian random variable (as before)
  - $w_d \sim N(0, \sigma^2)$ are i.i.d. Gaussian random variables (unlike before)
  - Note that all w_d share the same variance σ^2
- $\boldsymbol{w}$ is a random variable (*Bayesian* interpretation)
- To find $\boldsymbol{w}$ given data $\mathcal{D}$, we can compute posterior distribution of $\boldsymbol{w}$:

$$p(\boldsymbol{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w})}{p(\mathcal{D})}$$

- Maximum a posterior (MAP) estimate:

$$\boldsymbol{w}^{\text{MAP}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\mathcal{D}) = \arg\max_{\boldsymbol{w}} p(\mathcal{D}, \boldsymbol{w})$$

- MAP reduces to MLE if we assume uniform prior for $p(\boldsymbol{w})$

Estimating $\boldsymbol{w}$

- Let $X_1, \ldots, X_N$ be IID with $y|\boldsymbol{w}, \boldsymbol{x} \sim N(\boldsymbol{w}^\top \boldsymbol{x}, \sigma_0^2)$
- Let w_d be IID with $w_d \sim N(0, \sigma^2)$

Joint likelihood of data and parameters (given σ_0, σ)

$$p(\mathcal{D}, \boldsymbol{w}) = p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w}) \quad = \prod_n p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) \prod_d p(w_d)$$

Estimating $\boldsymbol{w}$

- Let $X_1, \ldots, X_N$ be IID with $y|\boldsymbol{w}, \boldsymbol{x} \sim N(\boldsymbol{w}^\top \boldsymbol{x}, \sigma_0^2)$
- Let w_d be IID with $w_d \sim N(0, \sigma^2)$

Joint likelihood of data and parameters (given σ_0, σ)

$$p(\mathcal{D}, \boldsymbol{w}) = p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w}) \quad = \prod_n p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) \prod_d p(w_d)$$

Joint log likelihood Plugging in Gaussian PDF, we get:

$$\begin{aligned} \log p(\mathcal{D}, \boldsymbol{w}) &= \sum_n \log p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) + \sum_d \log p(w_d) \\ &= -\frac{\sum_n (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - y_n)^2}{2\sigma_0^2} - \sum_d \frac{1}{2\sigma^2} w_d^2 + \text{const} \end{aligned}$$

Estimating $\boldsymbol{w}$

- Let $X_1, \ldots, X_N$ be IID with $y|\boldsymbol{w}, \boldsymbol{x} \sim N(\boldsymbol{w}^\top \boldsymbol{x}, \sigma_0^2)$
- Let w_d be IID with $w_d \sim N(0, \sigma^2)$

Joint likelihood of data and parameters (given σ_0, σ)

$$p(\mathcal{D}, \boldsymbol{w}) = p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w}) \quad = \prod_n p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) \prod_d p(w_d)$$

Joint log likelihood Plugging in Gaussian PDF, we get:

$$\begin{aligned}\log p(\mathcal{D}, \boldsymbol{w}) &= \sum_n \log p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) + \sum_d \log p(w_d) \\ &= -\frac{\sum_n (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - y_n)^2}{2\sigma_0^2} - \sum_d \frac{1}{2\sigma^2} w_d^2 + \text{const}\end{aligned}$$

MAP estimate: $\boldsymbol{w}^{\mathrm{MAP}} = \arg\max_{\boldsymbol{w}} \log p(\mathcal{D}, \boldsymbol{w})$

- As with LMS, set gradient equal to zero and solve (for $\boldsymbol{w}$)

Maximum a posterior (MAP) estimate

Regularized linear regression: a new error to minimize

$$\mathcal{E}(\boldsymbol{w}) = \sum_n (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - y_n)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

where $\lambda > 0$ is used to denote σ_0^2/σ^2.

- What happens as $\lambda \to +\infty$?

Maximum a posterior (MAP) estimate

Regularized linear regression: a new error to minimize

$$\mathcal{E}(\boldsymbol{w}) = \sum_n (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - y_n)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

where $\lambda > 0$ is used to denote σ_0^2/σ^2.

- What happens as $\lambda \to +\infty$?
- What happens as $\lambda \to 0$?

Maximum a posterior (MAP) estimate

Regularized linear regression: a new error to minimize

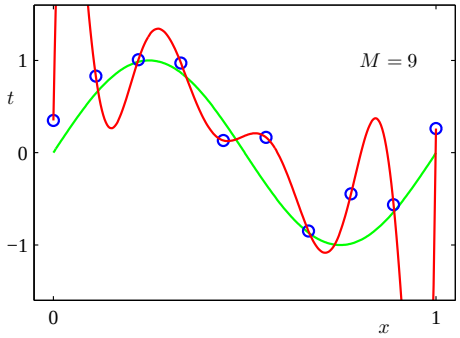
$$\mathcal{E}(\boldsymbol{w}) = \sum_n (\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n - y_n)^2 + \lambda \|\boldsymbol{w}\|_2^2$$

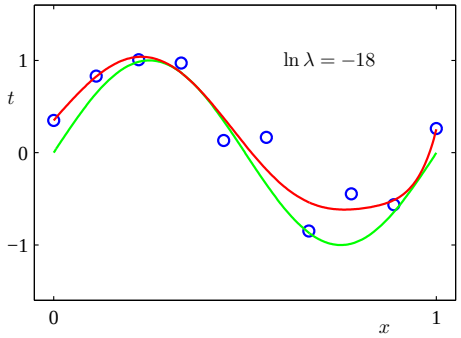
where $\lambda > 0$ is used to denote σ_0^2/σ^2.

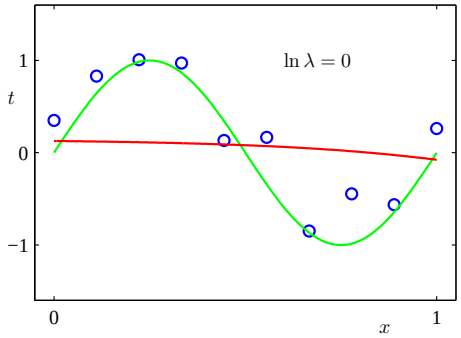
- What happens as $\lambda \to +\infty$?
- What happens as $\lambda \to 0$?
- What happens when $\lambda < 0$?

Overfitting in terms of λ

Overfitting is reduced as we increase the regularizer

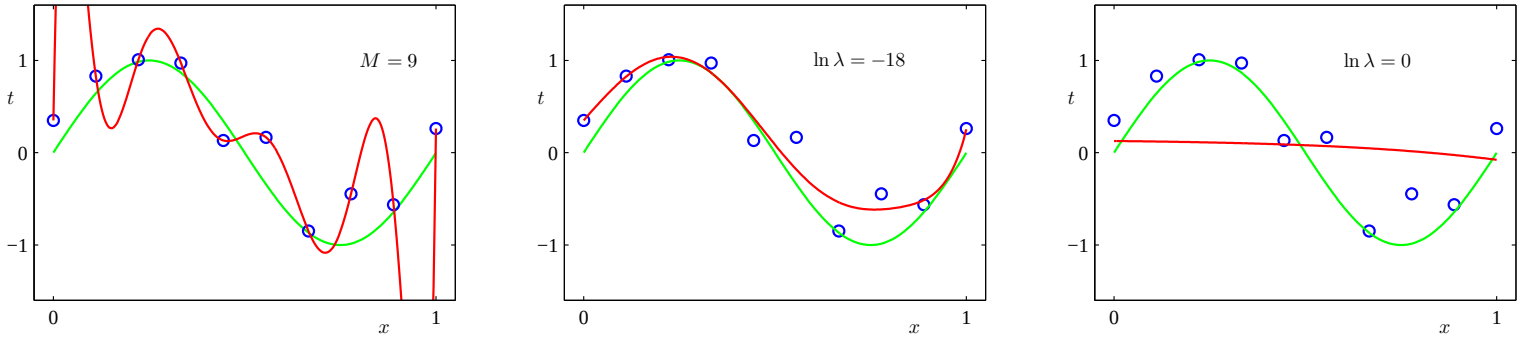




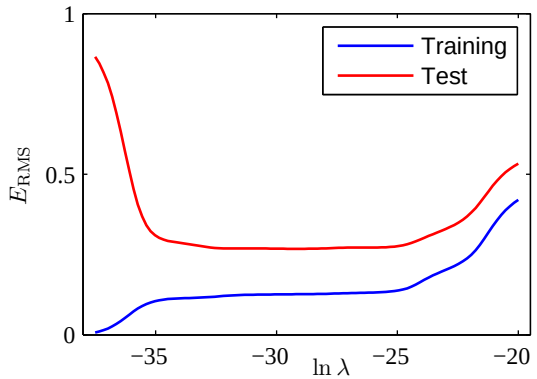


Overfitting in terms of λ

Overfitting is reduced as we increase the regularizer



λ **vs. residual error** shows the difference of the model performance on training and testing dataset



Outline

Basic and important machine learning concepts

Supervised learning

We aim to build a function $h(\boldsymbol{x})$ to predict the true value y associated with $\boldsymbol{x}$. If we make a mistake, we incur a *loss*

$$\ell(h(\boldsymbol{x}), y)$$

Basic and important machine learning concepts

Supervised learning

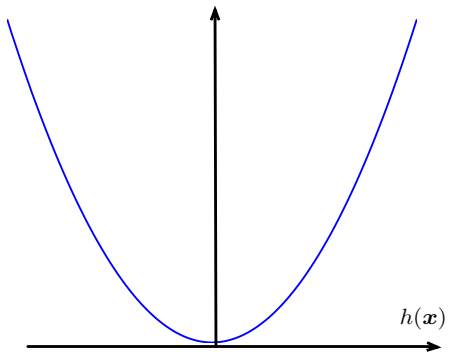
We aim to build a function $h(\boldsymbol{x})$ to predict the true value y associated with $\boldsymbol{x}$. If we make a mistake, we incur a *loss*

$$\ell(h(\boldsymbol{x}), y)$$

Example: quadratic loss function for regression when y is continuous

$$\ell(h(\boldsymbol{x}), y) = [h(\boldsymbol{x}) - y]^2$$

Ex: when $y = 0$

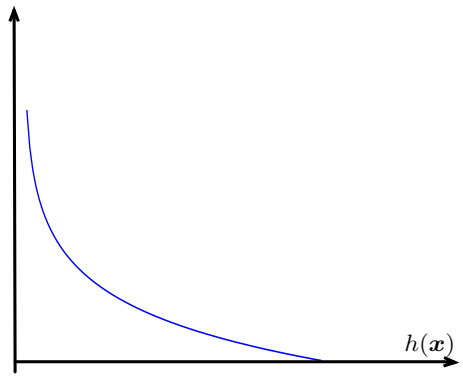


Other types of loss functions

For classification: cross-entropy loss (also called $logistic$ loss)

$$\ell(h(\boldsymbol{x}), y) = -y \log h(\boldsymbol{x}) - (1-y) \log[1-h(\boldsymbol{x})]$$

Ex: when $y = 1$



Measure how good our predictor is

Risk: assume we know the true distribution of data $p(\boldsymbol{x}, y)$, the *risk* is

$$R[h(\boldsymbol{x})] = \mathbb{E}_{(\boldsymbol{x},y)\sim p(\boldsymbol{x},y)}\ell(h(\boldsymbol{x}), y) = \int_{\boldsymbol{x},y} \ell(h(\boldsymbol{x}), y)p(\boldsymbol{x}, y)d\boldsymbol{x}d\, y$$

Measure how good our predictor is

Risk: assume we know the true distribution of data $p(\boldsymbol{x}, y)$, the *risk* is

$$R[h(\boldsymbol{x})] = \mathbb{E}_{(\boldsymbol{x},y)\sim p(\boldsymbol{x},y)}\ell(h(\boldsymbol{x}), y) = \int_{\boldsymbol{x},y} \ell(h(\boldsymbol{x}), y)p(\boldsymbol{x}, y)d\boldsymbol{x}d\, y$$

However, we cannot compute $R[h(\boldsymbol{x})]$, so we use *empirical risk*, given a training dataset $\mathcal{D}$

$$R^{\text{EMP}}[h(\boldsymbol{x})] = \frac{1}{N}\sum_n \ell(h(\boldsymbol{x}_n), y_n)$$

Measure how good our predictor is

Risk: assume we know the true distribution of data $p(\boldsymbol{x}, y)$, the *risk* is

$$R[h(\boldsymbol{x})] = \mathbb{E}_{(\boldsymbol{x},y)\sim p(\boldsymbol{x},y)}\ell(h(\boldsymbol{x}), y) = \int_{\boldsymbol{x},y} \ell(h(\boldsymbol{x}), y)p(\boldsymbol{x}, y)d\boldsymbol{x}d\, y$$

However, we cannot compute $R[h(\boldsymbol{x})]$, so we use *empirical risk*, given a training dataset $\mathcal{D}$

$$R^{\text{EMP}}[h(\boldsymbol{x})] = \frac{1}{N}\sum_n \ell(h(\boldsymbol{x}_n), y_n)$$

Intuitively, as $N \to +\infty$,

$$R^{\text{EMP}}[h(\boldsymbol{x})] \to R[h(\boldsymbol{x})]$$

How does this relate to what we have learned?

So far, we have been doing empirical risk minimization (ERM)

- For linear regression, $h(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$, and we use squared loss
- For logistic regression, $h(\boldsymbol{x}) = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$, and we use cross-entropy loss

How does this relate to what we have learned?

So far, we have been doing empirical risk minimization (ERM)

- For linear regression, $h(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$, and we use squared loss
- For logistic regression, $h(\boldsymbol{x}) = \sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$, and we use cross-entropy loss

ERM might be problematic

- If $h(\boldsymbol{x})$ is complicated enough,

$$R^{\mathrm{EMP}}[h(\boldsymbol{x})] \to 0$$

- But then $h(\boldsymbol{x})$ is unlikely to do well in predicting things out of the training dataset $\mathcal{D}$ (*poor generalization* or *overfitting*)
- We'll explore why regularization might work from the context of the bias-variance tradeoff, focusing on regression / squared loss

Bias/variance tradeoff (Looking ahead)

Error decomposes into 3 terms

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \text{VARIANCE} + \text{BIAS}^2 + \text{NOISE}$$

We will prove this result, and interpret what it means...

Bias/variance tradeoff for regression

Goal: to understand the sources of prediction errors

- $\mathcal{D}$: our training data

Bias/variance tradeoff for regression

Goal: to understand the sources of prediction errors

- $\mathcal{D}$: our training data
- $h_{\mathcal{D}}(\boldsymbol{x})$: our prediction function
  - subscript $\mathcal{D}$ indicates that function is learned on a specific training set

Bias/variance tradeoff for regression

Goal: to understand the sources of prediction errors

- $\mathcal{D}$: our training data
- $h_{\mathcal{D}}(\boldsymbol{x})$: our prediction function
  - subscript $\mathcal{D}$ indicates that function is learned on a specific training set
- $\ell(h(\boldsymbol{x}), y)$: squared loss function for regression

$$\ell(h_{\mathcal{D}}(\boldsymbol{x}), y) = [h_{\mathcal{D}}(\boldsymbol{x}) - y]^2$$

Bias/variance tradeoff for regression

Goal: to understand the sources of prediction errors

- $\mathcal{D}$: our training data
- $h_{\mathcal{D}}(\boldsymbol{x})$: our prediction function
  - subscript $\mathcal{D}$ indicates that function is learned on a specific training set
- $\ell(h(\boldsymbol{x}), y)$: squared loss function for regression

$$\ell(h_{\mathcal{D}}(\boldsymbol{x}), y) = [h_{\mathcal{D}}(\boldsymbol{x}) - y]^2$$

- Unknown joint distribution $p(\boldsymbol{x}, y)$

The effect of finite training samples

Risk of prediction function, $h_{\mathcal{D}}(x)$

$$R[h_{\mathcal{D}}(\boldsymbol{x})] = \mathbb{E}_{(\boldsymbol{x},y)\sim p(\boldsymbol{x},y)}\ell(h_{\mathcal{D}}(\boldsymbol{x}), y) = \int_{\boldsymbol{x}}\int_{y}[h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y)d\boldsymbol{x}dy$$

The effect of finite training samples

Risk of prediction function, $h_{\mathcal{D}}(x)$

$$R[h_{\mathcal{D}}(\boldsymbol{x})] = \mathbb{E}_{(\boldsymbol{x},y)\sim p(\boldsymbol{x},y)}\ell(h_{\mathcal{D}}(\boldsymbol{x}), y) = \int_{\boldsymbol{x}}\int_{y}[h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y)d\boldsymbol{x}dy$$

But, $\mathcal{D}$ is a random sample from the following joint distribution

$$\mathcal{D} \sim P(\mathcal{D}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n, y_n)$$

The effect of finite training samples

Risk of prediction function, $h_{\mathcal{D}}(x)$

$$R[h_{\mathcal{D}}(\boldsymbol{x})] = \mathbb{E}_{(\boldsymbol{x},y)\sim p(\boldsymbol{x},y)}\ell(h_{\mathcal{D}}(\boldsymbol{x}), y) = \int_{\boldsymbol{x}}\int_{y}[h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y)d\boldsymbol{x}dy$$

But, $\mathcal{D}$ is a random sample from the following joint distribution

$$\mathcal{D} \sim P(\mathcal{D}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n, y_n)$$

So, $h_{\mathcal{D}}(x)$ and $R[h_{\mathcal{D}}(x)]$ are also random w.r.t. $P(\mathcal{D})$

How can we disentangle the impact of the random sample $\mathcal{D}$ when assessing the quality of $h_{\mathcal{D}}(\cdot)$?

Average over the distribution of the training data

Averaged risk

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

Average over the distribution of the training data

Averaged risk

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

Namely, the randomness with respect to $\mathcal{D}$ is marginalized out.

Average over the distribution of the training data

Averaged risk

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

Namely, the randomness with respect to $\mathcal{D}$ is marginalized out.

Averaged prediction

$$\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) = \int_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) P(\mathcal{D}) d\mathcal{D}$$

Average over the distribution of the training data

Averaged risk

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

Namely, the randomness with respect to $\mathcal{D}$ is marginalized out.

Averaged prediction

$$\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) = \int_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) P(\mathcal{D}) d\mathcal{D}$$

Namely, if we have seen many training datasets, we predict with the average of our trained models learned on each training dataset.

Interpreting Averaged risk

We can add and subtract averaged prediction from averaged risk

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

Interpreting Averaged risk

We can add and subtract averaged prediction from averaged risk

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

$$= \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} \Bigg[\Big(h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) \Big)$$

$$+ \Big(\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y \Big) \Bigg]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

Interpreting Averaged risk

We can add and subtract averaged prediction from averaged risk

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

$$= \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} \Bigg[\Big(h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) \Big)$$

$$+ \Big(\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y \Big) \Bigg]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

$$= \underbrace{\int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}}_{\text{VARIANCE}}$$

Interpreting Averaged risk

We can add and subtract averaged prediction from averaged risk

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

$$= \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} \Bigg[\Big(h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) \Big)$$

$$+ \Big(\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y \Big) \Bigg]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

$$= \underbrace{\int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}}_{\text{VARIANCE}}$$

$$+ \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

Where does the cross-term go?

It is zero

$$\int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})][\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y] p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

Where does the cross-term go?

It is zero

$$\int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})][\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y] p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

$$= \int_{\boldsymbol{x}} \int_{y} \left\{ \int_{\mathcal{D}} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})] P(\mathcal{D}) d\mathcal{D} \right\} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y] p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

Where does the cross-term go?

It is zero

$$\int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})][\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y] p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

$$= \int_{\boldsymbol{x}} \int_{y} \left\{ \int_{\mathcal{D}} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})] P(\mathcal{D}) d\mathcal{D} \right\} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y] p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

$$= 0 \leftarrow \text{(the integral within the braces vanishes, by definition)}$$

Analyzing the variance

How can we reduce the variance?

$$\int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

Analyzing the variance

How can we reduce the variance?

$$\int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

- Use a lot of data (ie, increase the size of $\mathcal{D}$)
- Use a simple $h(\cdot)$ so that $h_{\mathcal{D}}(\boldsymbol{x})$ does not vary much across different training datasets.

Analyzing the variance

How can we reduce the variance?

$$\int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

- Use a lot of data (ie, increase the size of $\mathcal{D}$)
- Use a simple $h(\cdot)$ so that $h_{\mathcal{D}}(\boldsymbol{x})$ does not vary much across different training datasets.
  Ex: $h(\boldsymbol{x}) = \text{const}$

The remaining item

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

$$+ \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

The remaining item

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

$$+ \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

The integrand has no dependency on $\mathcal{D}$ anymore and simplifies to

$$\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

The remaining item

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

$$+ \int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}$$

The integrand has no dependency on $\mathcal{D}$ anymore and simplifies to

$$\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

We will apply a similar trick, by using the averaged target y

$$\mathbb{E}_y[y] = \int_y y p(y|\boldsymbol{x}) dy$$

Bias and noise

Decompose again

$$\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

$$= \int_{\boldsymbol{x}} \int_{y} \left[\left(\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_y[y] \right) + \left(\mathbb{E}_y[y] - y \right) \right]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

Bias and noise

Decompose again

$$\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

$$= \int_{\boldsymbol{x}} \int_{y} \left[\left(\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_y[y] \right) + \left(\mathbb{E}_y[y] - y \right) \right]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

$$= \underbrace{\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_y[y]]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy}_{\text{BIAS}^2}$$

$$+ \underbrace{\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_y[y] - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy}_{\text{NOISE}}$$

Where is the cross-term?

Left as a take-home exercise

Analyzing the noise

How can we reduce noise

$$\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_y[y] - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy = \int_{\boldsymbol{x}} \left(\int_{y} [\mathbb{E}_y[y] - y]^2 p(y|\boldsymbol{x}) dy \right) p(\boldsymbol{x}) d\boldsymbol{x}$$

Analyzing the noise

How can we reduce noise

$$\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_y[y] - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy = \int_{\boldsymbol{x}} \left(\int_{y} [\mathbb{E}_y[y] - y]^2 p(y|\boldsymbol{x}) dy \right) p(\boldsymbol{x}) d\boldsymbol{x}$$

There is *nothing* we can do. This quantity depends on $p(\boldsymbol{x}, y)$ only; choosing $h(\cdot)$ or the training dataset $\mathcal{D}$ will not affect it.

Analyzing the noise

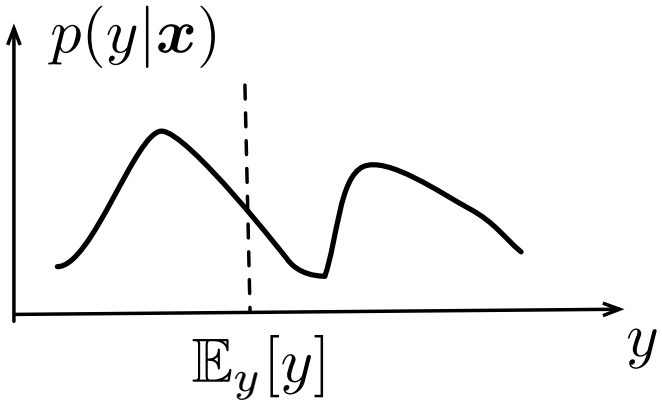
How can we reduce noise

$$\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_y[y] - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy = \int_{\boldsymbol{x}} \left(\int_{y} [\mathbb{E}_y[y] - y]^2 p(y|\boldsymbol{x}) dy \right) p(\boldsymbol{x}) d\boldsymbol{x}$$

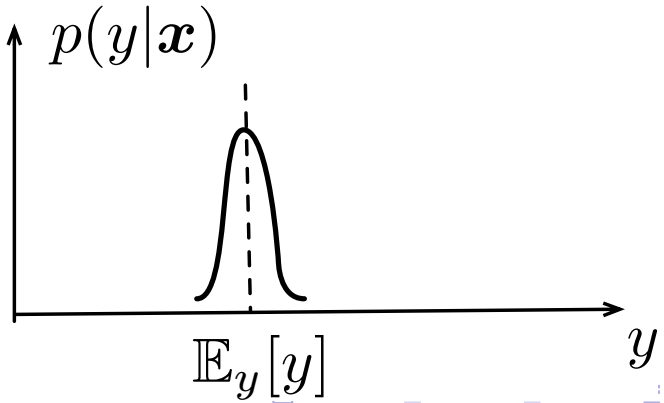
There is *nothing* we can do. This quantity depends on $p(\boldsymbol{x}, y)$ only; choosing $h(\cdot)$ or the training dataset $\mathcal{D}$ will not affect it.

Note that the integral inside the parentheses is the *variance* (noise) of the distribution $p(y|\boldsymbol{x})$ at the given $\boldsymbol{x}$.

More difficult



Easier



Analyzing the bias term

How can we reduce bias?

$$\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{y}[y]]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy = \int_{\boldsymbol{x}} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{y}[y]]^2 p(\boldsymbol{x}) d\boldsymbol{x}$$

Analyzing the bias term

How can we reduce bias?

$$\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{y}[y]]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy = \int_{\boldsymbol{x}} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{y}[y]]^2 p(\boldsymbol{x}) d\boldsymbol{x}$$

By using richer / more complex models to better approximate $\mathbb{E}_y[y]$

Analyzing the bias term

How can we reduce bias?

$$\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{y}[y]]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy = \int_{\boldsymbol{x}} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{y}[y]]^2 p(\boldsymbol{x}) d\boldsymbol{x}$$

By using richer / more complex models to better approximate $\mathbb{E}_y[y]$

However, this increased complexity will increase the VARIANCE term (as we can potentially overfit)

Bias/variance tradeoff

Error decomposes into 3 terms

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \text{VARIANCE} + \text{BIAS}^2 + \text{NOISE}$$

where the first and the second term are inherently in conflict in terms of choosing what kind of $h(\boldsymbol{x})$ we should use (unless we have an infinite amount of data)

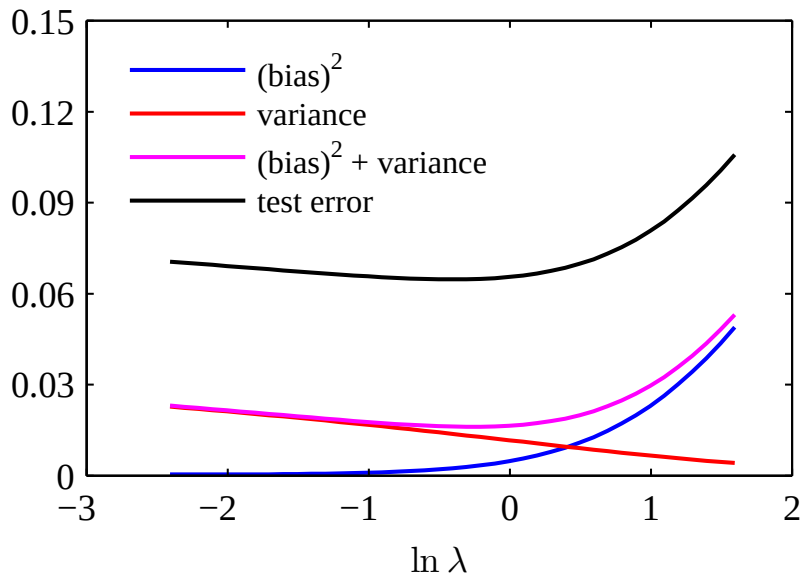
Bias/variance tradeoff

Error decomposes into 3 terms

$$\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})] = \text{VARIANCE} + \text{BIAS}^2 + \text{NOISE}$$

where the first and the second term are inherently in conflict in terms of choosing what kind of $h(\boldsymbol{x})$ we should use (unless we have an infinite amount of data)

If we can compute all terms analytically, they will look like this



Outline

Motivation

How to choose nonlinear basis function for regression?

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$$

- $\boldsymbol{\phi}(\cdot)$ maps the original feature vector $\boldsymbol{x}$ to a *new* M-dimensional feature vector
- We can sidestep the issue of choosing which $\boldsymbol{\phi}(\cdot)$ to use by *equivalently* choosing a *kernel function*

Motivation

How to choose nonlinear basis function for regression?

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$$

- $\boldsymbol{\phi}(\cdot)$ maps the original feature vector $\boldsymbol{x}$ to a *new* M-dimensional feature vector
- We can sidestep the issue of choosing which $\boldsymbol{\phi}(\cdot)$ to use by *equivalently* choosing a *kernel function*

Regularized least squares

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{n}(y_n - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n))^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

Its solution $\boldsymbol{w}^{\mathrm{MAP}}$ is given by

$$\frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} = \sum_{n}(y_n - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n))(-\boldsymbol{\phi}(\boldsymbol{x}_n)) + \lambda\boldsymbol{w} = 0$$

MAP Solution

The optimal parameter vector is a linear combination of features

$$\boldsymbol{w}^{\mathrm{MAP}} = \sum_n \frac{1}{\lambda}(y_n - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n))\boldsymbol{\phi}(\boldsymbol{x}_n)$$

MAP Solution

The optimal parameter vector is a linear combination of features

$$\boldsymbol{w}^{\text{MAP}} = \sum_n \frac{1}{\lambda}(y_n - \boldsymbol{w}^{\text{T}}\boldsymbol{\phi}(\boldsymbol{x}_n))\boldsymbol{\phi}(\boldsymbol{x}_n) = \sum_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) = \boldsymbol{\Phi}^{\text{T}}\boldsymbol{\alpha}$$

- $\alpha_n = \frac{1}{\lambda}(y_n - \boldsymbol{w}^{\text{T}}\boldsymbol{\phi}(\boldsymbol{x}_n))$
- $\boldsymbol{\Phi}$ is the *design matrix* of *transformed* features
- Its transpose is made of column vectors and is given by

$$\boldsymbol{\Phi}^{\text{T}} = (\boldsymbol{\phi}(\boldsymbol{x}_1)\ \boldsymbol{\phi}(\boldsymbol{x}_2)\ \cdots\ \boldsymbol{\phi}(\boldsymbol{x}_N)) \in \mathbb{R}^{M \times N}$$

MAP Solution

The optimal parameter vector is a linear combination of features

$$\boldsymbol{w}^{\text{MAP}} = \sum_n \frac{1}{\lambda}(y_n - \boldsymbol{w}^{\text{T}}\boldsymbol{\phi}(\boldsymbol{x}_n))\boldsymbol{\phi}(\boldsymbol{x}_n) = \sum_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) = \boldsymbol{\Phi}^{\text{T}}\boldsymbol{\alpha}$$

- $\alpha_n = \frac{1}{\lambda}(y_n - \boldsymbol{w}^{\text{T}}\boldsymbol{\phi}(\boldsymbol{x}_n))$
- $\boldsymbol{\Phi}$ is the *design matrix* of *transformed* features
- Its transpose is made of column vectors and is given by

$$\boldsymbol{\Phi}^{\text{T}} = (\boldsymbol{\phi}(\boldsymbol{x}_1)\ \boldsymbol{\phi}(\boldsymbol{x}_2)\ \cdots\ \boldsymbol{\phi}(\boldsymbol{x}_N)) \in \mathbb{R}^{M \times N}$$

Of course, we don't know what $\boldsymbol{\alpha}$ (the vector of α_n) corresponds to $\boldsymbol{w}^{\text{MAP}}$!

Dual formulation

Regularized least squares: $J(\boldsymbol{w}) = \frac{1}{2}\sum_n (y_n - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n))^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$

Substitute $\boldsymbol{w}^{\mathrm{MAP}} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}$ into $J(\boldsymbol{w})$ to obtain a function of $\boldsymbol{\alpha}$:

$$J(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} - (\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y})^{\mathrm{T}}\boldsymbol{\alpha} + \frac{\lambda}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}$$

Dual formulation

Regularized least squares: $J(\boldsymbol{w}) = \frac{1}{2}\sum_n (y_n - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n))^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$

Substitute $\boldsymbol{w}^{\mathrm{MAP}} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}$ into $J(\boldsymbol{w})$ to obtain a function of $\boldsymbol{\alpha}$:

$$J(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} - (\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y})^{\mathrm{T}}\boldsymbol{\alpha} + \frac{\lambda}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}$$

Before we show how $J(\boldsymbol{\alpha})$ is derived, we make an important observation

Dual formulation

Regularized least squares: $J(\boldsymbol{w}) = \frac{1}{2}\sum_n (y_n - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n))^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$

Substitute $\boldsymbol{w}^{\mathrm{MAP}} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}$ into $J(\boldsymbol{w})$ to obtain a function of $\boldsymbol{\alpha}$:

$$J(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} - (\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y})^{\mathrm{T}}\boldsymbol{\alpha} + \frac{\lambda}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}$$

Before we show how $J(\boldsymbol{\alpha})$ is derived, we make an important observation

The *Gram matrix* or *kernel matrix* — $\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}$ — appears multiple times

$$\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}$$

$$= \begin{pmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_1) & \boldsymbol{\phi}(\boldsymbol{x}_1)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_2) & \cdots & \boldsymbol{\phi}(\boldsymbol{x}_1)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_N) \\ \boldsymbol{\phi}(\boldsymbol{x}_2)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_1) & \boldsymbol{\phi}(\boldsymbol{x}_2)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_2) & \cdots & \boldsymbol{\phi}(\boldsymbol{x}_2)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_N) \\ \cdots & \cdots & \cdots & \cdots \\ \boldsymbol{\phi}(\boldsymbol{x}_N)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_1) & \boldsymbol{\phi}(\boldsymbol{x}_N)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_2) & \cdots & \boldsymbol{\phi}(\boldsymbol{x}_N)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_N) \end{pmatrix} \in \mathbb{R}^{N\times N}$$

Properties of the matrix $\boldsymbol{K}$

- Symmetric

$$K_{mn} = \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) = \boldsymbol{\phi}(\boldsymbol{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_m) = K_{nm}$$

- Positive semidefinite: for any vector $\boldsymbol{a}$

$$\boldsymbol{a}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{a} = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{a})^{\mathrm{T}}(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{a}) \geq 0$$

Properties of the matrix $\boldsymbol{K}$

- Symmetric

$$K_{mn} = \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) = \boldsymbol{\phi}(\boldsymbol{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_m) = K_{nm}$$

- Positive semidefinite: for any vector $\boldsymbol{a}$

$$\boldsymbol{a}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{a} = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{a})^{\mathrm{T}}(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{a}) \geq 0$$

- Not the same as the second-moment (covariance) matrix $\boldsymbol{C} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}$
  - $\boldsymbol{C}$ has a size of $M \times M$ while $\boldsymbol{K}$ is $N \times N$
  - When $N \leq M$, using $\boldsymbol{K}$ is more computationally advantageous

The derivation of $J(\boldsymbol{\alpha})$

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_n (y - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n))^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

The derivation of $J(\boldsymbol{\alpha})$

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_n (y - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n))^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

$$= \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \qquad (\boldsymbol{y}, \boldsymbol{\Phi}\boldsymbol{w} \in \mathbb{R}^N)$$

The derivation of $J(\boldsymbol{\alpha})$

$$
\begin{aligned}
J(\boldsymbol{w}) &= \frac{1}{2}\sum_n (y - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n))^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \\
&= \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 && (\boldsymbol{y}, \boldsymbol{\Phi}\boldsymbol{w} \in \mathbb{R}^N) \\
&= \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2 && (\boldsymbol{w}^{\mathrm{MAP}} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha})
\end{aligned}
$$

The derivation of $J(\boldsymbol{\alpha})$

$$
\begin{aligned}
J(\boldsymbol{w}) &= \frac{1}{2}\sum_n (y - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n))^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \\
&= \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 && (\boldsymbol{y}, \boldsymbol{\Phi}\boldsymbol{w} \in \mathbb{R}^N) \\
&= \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2 && (\boldsymbol{w}^{\mathrm{MAP}} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}) \\
&= \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha}\|_2^2 + \frac{\lambda}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} && (\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}, \|\boldsymbol{v}\|_2^2 = \boldsymbol{v}^{\top}\boldsymbol{v})
\end{aligned}
$$

The derivation of $J(\boldsymbol{\alpha})$

$$
\begin{aligned}
J(\boldsymbol{w}) &= \frac{1}{2}\sum_n (y - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n))^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \\
&= \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 && (\boldsymbol{y}, \boldsymbol{\Phi}\boldsymbol{w} \in \mathbb{R}^N) \\
&= \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2 && (\boldsymbol{w}^{\mathrm{MAP}} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}) \\
&= \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha}\|_2^2 + \frac{\lambda}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} && (\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}, \|\boldsymbol{v}\|_2^2 = \boldsymbol{v}^{\top}\boldsymbol{v}) \\
&\propto \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \frac{\lambda}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha}
\end{aligned}
$$

The derivation of $J(\boldsymbol{\alpha})$

$$
\begin{aligned}
J(\boldsymbol{w}) &= \frac{1}{2}\sum_n (y - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n))^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \\
&= \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 && (\boldsymbol{y}, \boldsymbol{\Phi}\boldsymbol{w} \in \mathbb{R}^N) \\
&= \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2 && (\boldsymbol{w}^{\mathrm{MAP}} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}) \\
&= \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha}\|_2^2 + \frac{\lambda}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} && (\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}, \|\boldsymbol{v}\|_2^2 = \boldsymbol{v}^{\top}\boldsymbol{v}) \\
&\propto \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \frac{\lambda}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} \\
&= \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}^2\boldsymbol{\alpha} - (\boldsymbol{K}\boldsymbol{y})^{\mathrm{T}}\boldsymbol{\alpha} + \frac{\lambda}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} && (\boldsymbol{K} \text{ is symmetric}) \\
&= J(\boldsymbol{\alpha})
\end{aligned}
$$

Optimal $\boldsymbol{\alpha}$

$$\frac{\partial J(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \boldsymbol{K}^2\boldsymbol{\alpha} - \boldsymbol{K}\boldsymbol{y} + \lambda \boldsymbol{K}\boldsymbol{\alpha} = 0$$

which leads to (assuming that $\boldsymbol{K}$ is invertible)

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$$

- We only need to know $\boldsymbol{K}$ in order to compute $\boldsymbol{\alpha}$!
- Solution doesn't involve $\boldsymbol{\phi}(\cdot)$, but instead inner products $\boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)$
- This observation will give rise to the use of *kernel functions*

Optimal $\boldsymbol{\alpha}$

$$\frac{\partial J(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \boldsymbol{K}^2\boldsymbol{\alpha} - \boldsymbol{K}\boldsymbol{y} + \lambda \boldsymbol{K}\boldsymbol{\alpha} = 0$$

which leads to (assuming that $\boldsymbol{K}$ is invertible)

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$$

- We only need to know $\boldsymbol{K}$ in order to compute $\boldsymbol{\alpha}$!
- Solution doesn't involve $\boldsymbol{\phi}(\cdot)$, but instead inner products $\boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)$
- This observation will give rise to the use of *kernel functions*

Note that the initial parameter vector does require knowledge of $\boldsymbol{\Phi}$

$$\boldsymbol{w}^{\mathrm{MAP}} = \boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$$

Computing prediction needs only inner products too!

Since $\boldsymbol{w}^{\text{MAP}} = \boldsymbol{\Phi}^{\text{T}}(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$, at test time we compute:

$$\boldsymbol{w}^{\text{T}}\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{y}^{\text{T}}(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}\boldsymbol{\phi}(\boldsymbol{x})$$

Computing prediction needs only inner products too!

Since $\boldsymbol{w}^{\text{MAP}} = \boldsymbol{\Phi}^{\text{T}}(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$, at test time we compute:

$$\boldsymbol{w}^{\text{T}}\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{y}^{\text{T}}(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}\boldsymbol{\phi}(\boldsymbol{x})$$

$$= \boldsymbol{y}^{\text{T}}(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1} \begin{pmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^{\text{T}}\boldsymbol{\phi}(\boldsymbol{x}) \\ \boldsymbol{\phi}(\boldsymbol{x}_2)^{\text{T}}\boldsymbol{\phi}(\boldsymbol{x}) \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_N)^{\text{T}}\boldsymbol{\phi}(\boldsymbol{x}) \end{pmatrix} = \boldsymbol{y}^{\text{T}}(\boldsymbol{K} + \lambda I)^{-1}\boldsymbol{k}_{\boldsymbol{x}}$$

- We used the property that $(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}$ is symmetric (as $\boldsymbol{K}$ is)
- $\boldsymbol{k}_{\boldsymbol{x}}$ is shorthand notation for the column vector of inner products between training set and test point
- To make a prediction we *only need* $\boldsymbol{\phi}(\boldsymbol{x}_n)^T\boldsymbol{\phi}(\boldsymbol{x})$!

E.g.: Dot product between degree two polynomial features

Consider the following $\boldsymbol{\phi}(\boldsymbol{x})$:

$$\boldsymbol{\phi} : \boldsymbol{x} \rightarrow \boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

E.g.: Dot product between degree two polynomial features

Consider the following $\boldsymbol{\phi}(\boldsymbol{x})$:

$$\boldsymbol{\phi} : \boldsymbol{x} \rightarrow \boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

This gives rise to an inner product in a special form,

$$\begin{aligned} \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) &= x_{m1}^2x_{n1}^2 + 2x_{m1}x_{m2}x_{n1}x_{n2} + x_{m2}^2x_{n2}^2 \\ &= (x_{m1}x_{n1} + x_{m2}x_{n2})^2 = (\boldsymbol{x}_m^{\mathrm{T}}\boldsymbol{x}_n)^2 \end{aligned}$$

E.g.: Dot product between degree two polynomial features

Consider the following $\boldsymbol{\phi}(\boldsymbol{x})$:

$$\boldsymbol{\phi} : \boldsymbol{x} \rightarrow \boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

This gives rise to an inner product in a special form,

$$\begin{aligned} \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) &= x_{m1}^2x_{n1}^2 + 2x_{m1}x_{m2}x_{n1}x_{n2} + x_{m2}^2x_{n2}^2 \\ &= (x_{m1}x_{n1} + x_{m2}x_{n2})^2 = (\boldsymbol{x}_m^{\mathrm{T}}\boldsymbol{x}_n)^2 \end{aligned}$$

Namely, inner product can be computed by a function $(\boldsymbol{x}_m^{\mathrm{T}}\boldsymbol{x}_n)^2$ defined in terms of the original features, *without knowing $\boldsymbol{\phi}(\cdot)$!*

Common kernel functions

Polynomial kernel function with degree of d

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = (\boldsymbol{x}_m^{\mathrm{T}} \boldsymbol{x}_n + c)^d$$

for $c \geq 0$ and d is a positive integer.

Common kernel functions

Polynomial kernel function with degree of d

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = (\boldsymbol{x}_m^{\mathrm{T}} \boldsymbol{x}_n + c)^d$$

for $c \geq 0$ and d is a positive integer.

Gaussian kernel, RBF kernel, or Gaussian RBF kernel

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = e^{-\|\boldsymbol{x}_m - \boldsymbol{x}_n\|_2^2 / 2\sigma^2}$$

- Shift-invariant kernel (only depends on difference between two inputs)
- Corresponds to a feature space with *infinite* dimensions (but we can work directly with the original features)

Common kernel functions

Polynomial kernel function with degree of d

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = (\boldsymbol{x}_m^{\mathrm{T}} \boldsymbol{x}_n + c)^d$$

for $c \geq 0$ and d is a positive integer.

Gaussian kernel, RBF kernel, or Gaussian RBF kernel

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = e^{-\|\boldsymbol{x}_m - \boldsymbol{x}_n\|_2^2 / 2\sigma^2}$$

- Shift-invariant kernel (only depends on difference between two inputs)
- Corresponds to a feature space with *infinite* dimensions (but we can work directly with the original features)

These kernels have hyperparameters to be tuned: d, c, σ^2

Kernel functions

Definition: a (positive semidefinite) kernel function $k(\cdot, \cdot)$ is a bivariate function that satisfies the following properties. For any $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$,

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = k(\boldsymbol{x}_n, \boldsymbol{x}_m) \text{ and } k(\boldsymbol{x}_m, \boldsymbol{x}_n) = \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n)$$

for *some* function $\boldsymbol{\phi}(\cdot)$.

Kernel functions

Definition: a (positive semidefinite) kernel function $k(\cdot, \cdot)$ is a bivariate function that satisfies the following properties. For any $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$,

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = k(\boldsymbol{x}_n, \boldsymbol{x}_m) \text{ and } k(\boldsymbol{x}_m, \boldsymbol{x}_n) = \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n)$$

for *some* function $\boldsymbol{\phi}(\cdot)$.

Examples we have seen

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = (\boldsymbol{x}_m^{\mathrm{T}} \boldsymbol{x}_n)^2$$

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = e^{-\|\boldsymbol{x}_m - \boldsymbol{x}_n\|_2^2 / 2\sigma^2}$$

Kernel functions

Definition: a (positive semidefinite) kernel function $k(\cdot, \cdot)$ is a bivariate function that satisfies the following properties. For any $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$,

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = k(\boldsymbol{x}_n, \boldsymbol{x}_m) \text{ and } k(\boldsymbol{x}_m, \boldsymbol{x}_n) = \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n)$$

for *some* function $\boldsymbol{\phi}(\cdot)$.

Examples we have seen

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = (\boldsymbol{x}_m^{\mathrm{T}} \boldsymbol{x}_n)^2$$
$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = e^{-\|\boldsymbol{x}_m - \boldsymbol{x}_n\|_2^2 / 2\sigma^2}$$

Example that is not a kernel

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = \|\boldsymbol{x}_m - \boldsymbol{x}_n\|_2^2$$

(we'll see why later)

Conditions for being a positive semidefinite kernel function

Mercer theorem (loosely), a bivariate function $k(\cdot, \cdot)$ is a positive semidefinite kernel function, if and only if, for *any* N *and any* $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ..., *and* $\boldsymbol{x}_N$, the matrix

$$\boldsymbol{K} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & k(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

is positive semidefinite. We also refer $k(\cdot, \cdot)$ as a positive semidefinite kernel.

Why $\|\boldsymbol{x}_m - \boldsymbol{x}_n\|_2^2$ is not a positive semidefinite kernel?

Use the definition of positive semidefinite kernel function. We choose $N = 2$, and compute the matrix

$$\boldsymbol{K} = \begin{pmatrix} 0 & \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2 \\ \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2 & 0 \end{pmatrix}$$

Why $\|\boldsymbol{x}_m - \boldsymbol{x}_n\|_2^2$ is not a positive semidefinite kernel?

Use the definition of positive semidefinite kernel function. We choose $N = 2$, and compute the matrix

$$\boldsymbol{K} = \begin{pmatrix} 0 & \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2 \\ \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2 & 0 \end{pmatrix}$$

- PSD matrices have only non-negative eigenvalues
- This matrix has both *negative* and positive eigenvalues
- *Trace* of a matrix equals the sum of the diagonal elements, and also equals to the sum of the matrix's eigenvalues
  - In our case, the trace is zero

Why use kernel functions?

Can define kernel matrix without specifying $\phi(\cdot)$

$$\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & k(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

'Kernel trick'

- Many learning methods rely on training and test data only in the form of *inner products*, e.g., regularized least squares, nearest neighbors
- We can use a kernel function to introduce nonlinearity, i.e., *"kernelizing"* the methods
- We will show this "trick" by kernelizing the nearest neighbor classifier
- We will see this again when we talk about support vector machines

Kernelized nearest neighbors classifier (NNC)

Fundamental quantity is (squared) distance between two data points

$$d(\boldsymbol{x}_m, \boldsymbol{x}_n) = \|\boldsymbol{x}_m - \boldsymbol{x}_n\|_2^2 = \boldsymbol{x}_m^{\mathrm{T}}\boldsymbol{x}_m + \boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{x}_n - 2\boldsymbol{x}_m^{\mathrm{T}}\boldsymbol{x}_n$$

Kernelized nearest neighbors classifier (NNC)

Fundamental quantity is (squared) distance between two data points

$$d(\boldsymbol{x}_m, \boldsymbol{x}_n) = \|\boldsymbol{x}_m - \boldsymbol{x}_n\|_2^2 = \boldsymbol{x}_m^{\mathrm{T}}\boldsymbol{x}_m + \boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{x}_n - 2\boldsymbol{x}_m^{\mathrm{T}}\boldsymbol{x}_n$$

Can replace dot products by a kernel function $k(\cdot, \cdot)$:

$$d^{\mathrm{KERNEL}}(\boldsymbol{x}_m, \boldsymbol{x}_n) = k(\boldsymbol{x}_m, \boldsymbol{x}_m) + k(\boldsymbol{x}_n, \boldsymbol{x}_n) - 2k(\boldsymbol{x}_m, \boldsymbol{x}_n)$$

Kernelized nearest neighbors classifier (NNC)

Fundamental quantity is (squared) distance between two data points

$$d(\boldsymbol{x}_m, \boldsymbol{x}_n) = \|\boldsymbol{x}_m - \boldsymbol{x}_n\|_2^2 = \boldsymbol{x}_m^{\mathrm{T}}\boldsymbol{x}_m + \boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{x}_n - 2\boldsymbol{x}_m^{\mathrm{T}}\boldsymbol{x}_n$$

Can replace dot products by a kernel function $k(\cdot, \cdot)$:

$$d^{\mathrm{KERNEL}}(\boldsymbol{x}_m, \boldsymbol{x}_n) = k(\boldsymbol{x}_m, \boldsymbol{x}_m) + k(\boldsymbol{x}_n, \boldsymbol{x}_n) - 2k(\boldsymbol{x}_m, \boldsymbol{x}_n) = d(\boldsymbol{\phi}(\boldsymbol{x}_m), \boldsymbol{\phi}(\boldsymbol{x}_n))$$

- d^{KERNEL} equivalent to distance between $\boldsymbol{\phi}(\boldsymbol{x}_m)$ and $\boldsymbol{\phi}(\boldsymbol{x}_n)$ where $\boldsymbol{\phi}(\cdot)$ is the nonlinear mapping implied by the kernel function
- The nearest neighbor of a test point $\boldsymbol{x}$ is found via

$$\arg\min_n d^{\mathrm{KERNEL}}(\boldsymbol{x}, \boldsymbol{x}_n)$$

There are infinite numbers of kernels to use!

Rules of composing kernels (this is just a partial list)

- if $k(\boldsymbol{x}_m, \boldsymbol{x}_n)$ is a kernel, then $ck(\boldsymbol{x}_m, \boldsymbol{x}_n)$ is also if $c > 0$.
- if both $k_1(\boldsymbol{x}_m, \boldsymbol{x}_n)$ and $k_2(\boldsymbol{x}_m, \boldsymbol{x}_n)$ are kernels, then $\alpha k_1(\boldsymbol{x}_m, \boldsymbol{x}_n) + \beta k_2(\boldsymbol{x}_m, \boldsymbol{x}_n)$ are also if $\alpha, \beta \geq 0$
- if both $k_1(\boldsymbol{x}_m, \boldsymbol{x}_n)$ and $k_2(\boldsymbol{x}_m, \boldsymbol{x}_n)$ are kernels, then $k_1(\boldsymbol{x}_m, \boldsymbol{x}_n)k_2(\boldsymbol{x}_m, \boldsymbol{x}_n)$ are also.
- if $k(\boldsymbol{x}_m, \boldsymbol{x}_n)$ is a kernel, then $e^{k(\boldsymbol{x}_m, \boldsymbol{x}_n)}$ is also.
- $\ldots$

In practice, choosing an appropriate kernel is an "art"

People typically start with polynomial and Gaussian RBF kernels or incorporate domain knowledge.