Outline

1. Review of Hyperparameter Tuning and the Bias-Variance Trade-off
2. Classification Example: Spam Detection
3. Naive Bayes Model
4. Parameter Estimation

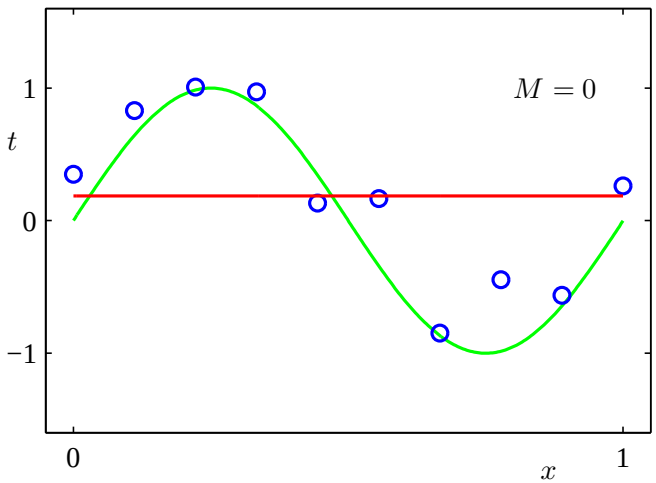
Overfitting and Regularization

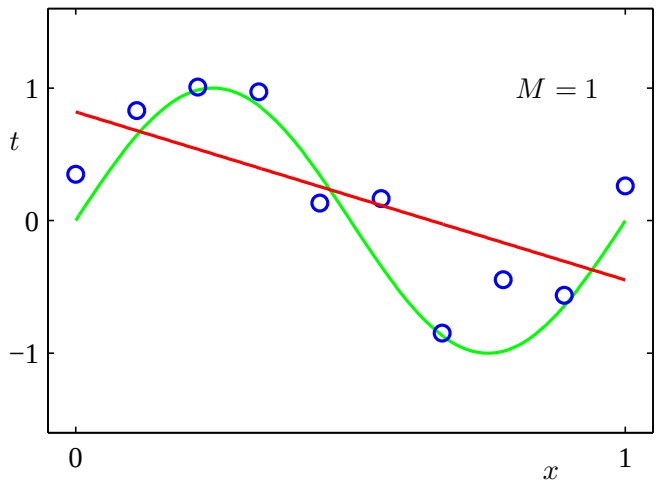
Non-linear basis functions: Polynomial regression

Polynomial basis functions

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

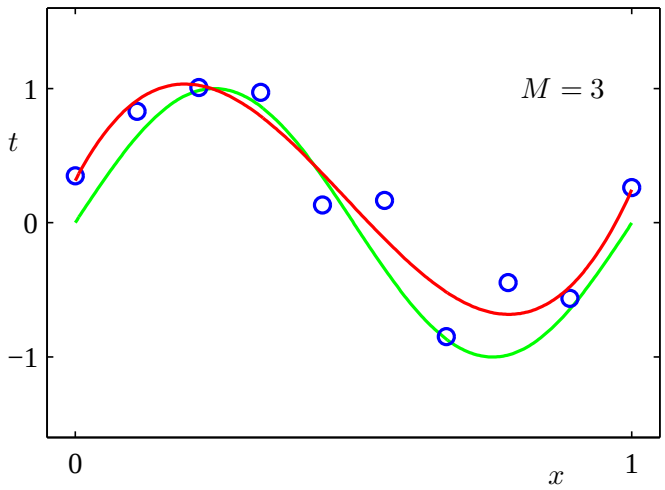
Fitting samples from a sine function:



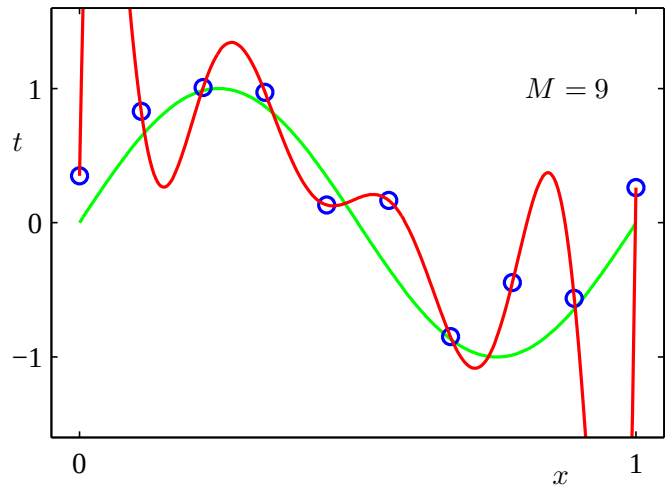


Adding high-order terms

M=3



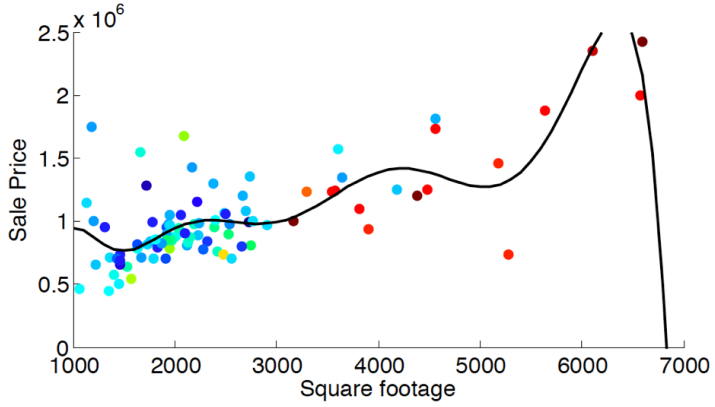
M=9: overfitting



More complex features lead to better results on the training data, but potentially worse results on new data, e.g., test data!

Overfitting can be quite disastrous

Fitting the housing price data with large M:



Predicted price goes to zero (and is ultimately negative) if you buy a big enough house!

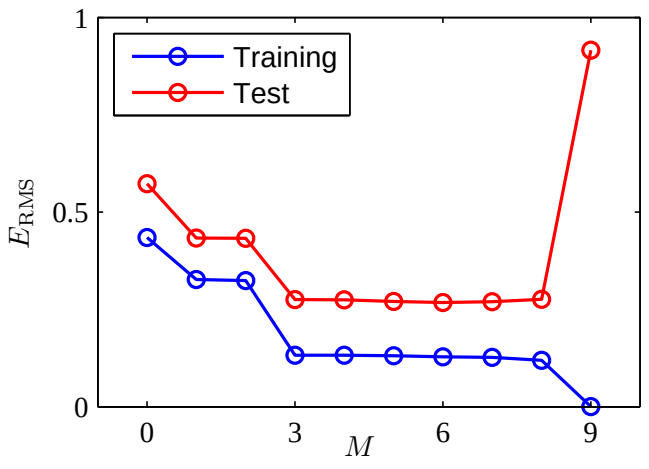
This is called poor generalization/overfitting.

Detecting overfitting

Plot model complexity versus objective function:

- X axis: model complexity, e.g., M
- Y axis: error, e.g., RSS, RMS (square root of RSS), 0-1 loss

Compute the objective on a training and test dataset.

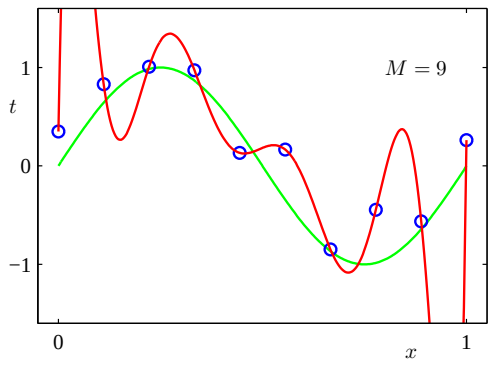


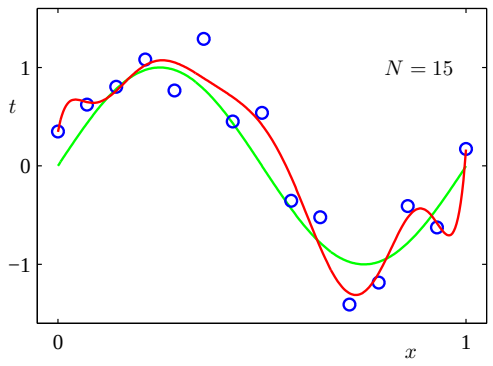
As a model increases in complexity:

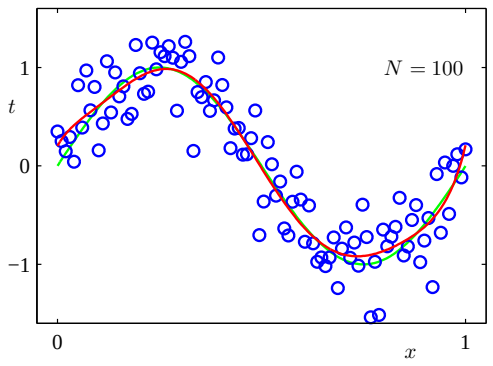
- Training error keeps improving
- Test error may first improve but eventually will deteriorate

Dealing with overfitting: Option 1

Try to use more training data



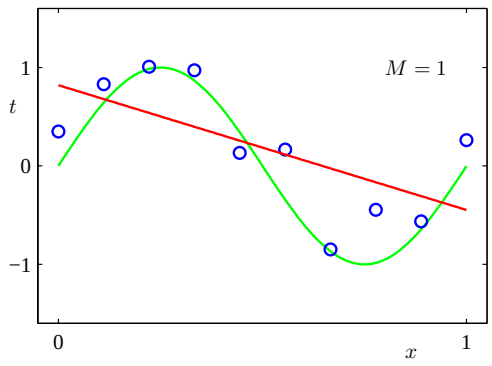


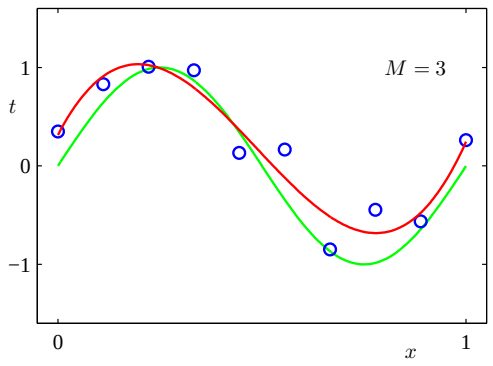


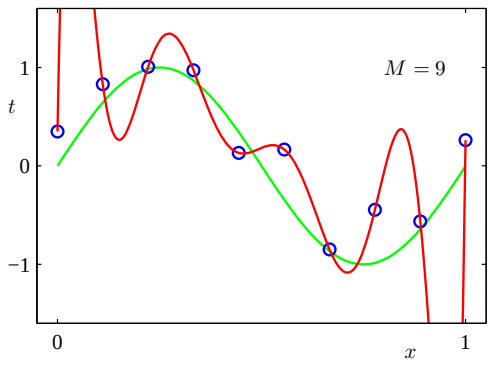
But getting a lot of data can be expensive and time-consuming

Dealing with overfitting: Option 2

Reduce the Number of Features







May not know which and how many features to remove

Dealing with overfitting: Option 3

Regularization Methods: Give preference to 'simpler' models

- How do we define a simple linear regression model — $\boldsymbol{w}^\top \boldsymbol{x}$?
- Intuitively, the weights corresponding to higher order terms should not be "too large"

	$M=0$	$M=1$	$M=3$	$M=9$
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18
w_8				-557682.99
w_9				125201.43

Regularization methods

Add a term to the objective function.

Choose the parameters to not just minimize risk, but avoid being large.

$$\frac{1}{2}\left\{\boldsymbol{w}^\top \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{w} - 2\left(\boldsymbol{X}^\top \boldsymbol{y}\right)^\top \boldsymbol{w}\right\} + \frac{1}{2}\lambda\|w\|_2^2$$

Ridge regression is just regularized linear regression.

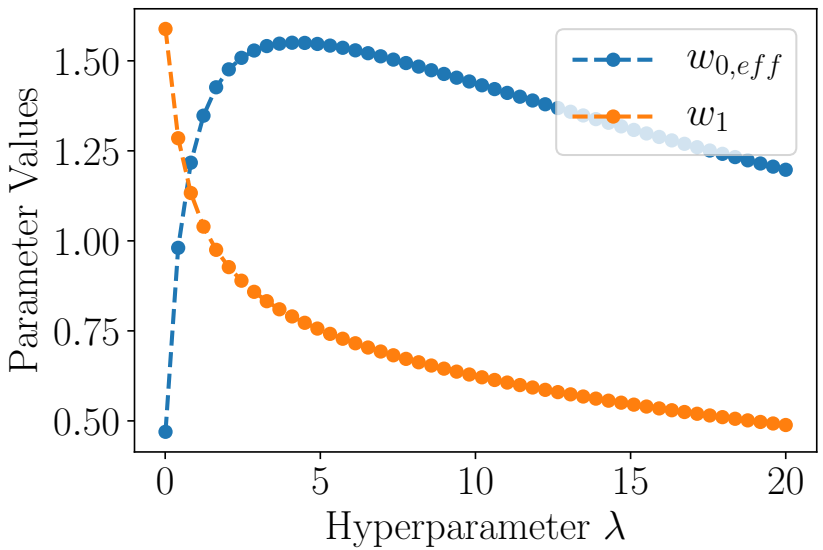
Advantages

- Forces the magnitude of $\boldsymbol{w}$ to be small
- Tries to find a simple model with few parameters
- Generalizes well to new data points

Ridge regression as regularization

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I} \right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

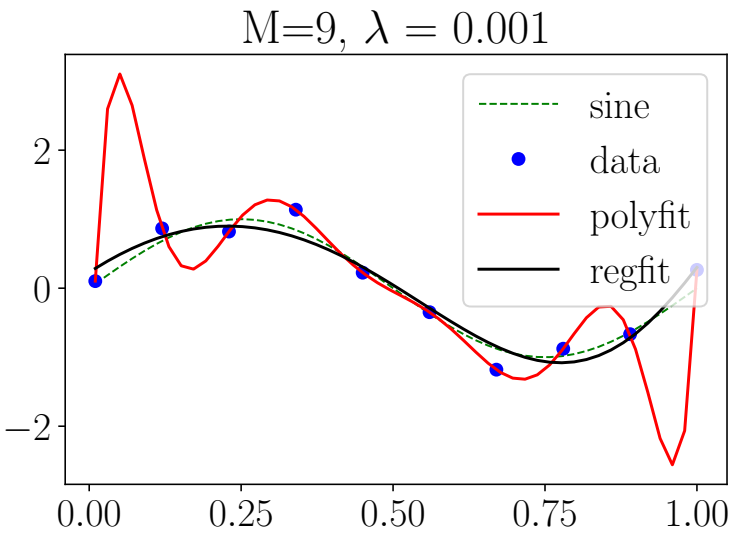
Let us plot $w_o' = w_0 + 2w_2$ and w_1 for different $\lambda \in [0.01, 20]$

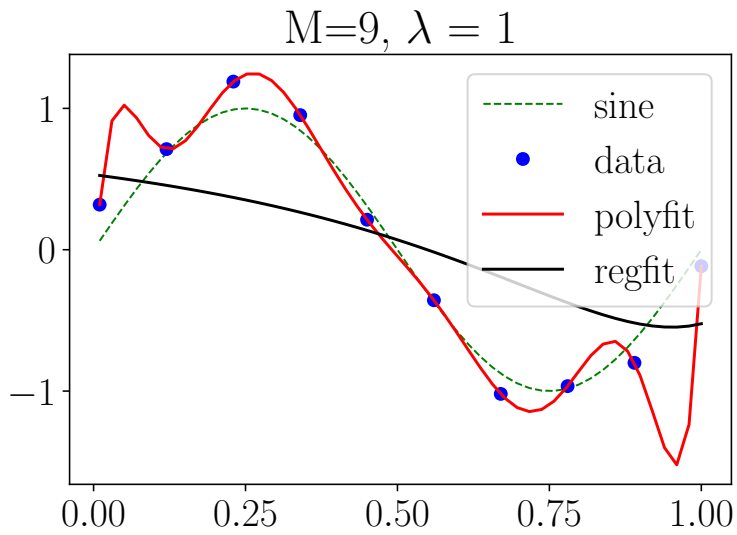


Setting small λ gives almost the least-squares solution, but it can cause numerical instability in the inversion

Example: Effect of regularization

- Regularization makes the higher order w_i's smaller
- Regularized polynomial fit will generalize much better
- As λ increases, the model becomes simpler





Probabilistic interpretation of regularization

Regularized 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)
- We first choose the weight for each feature d, w_d, from $N(0, \sigma^2)$. Then for each input vector $\boldsymbol{x}_n$, draw y_n from the distribution $N(\boldsymbol{w}^\top \boldsymbol{x}_n, \sigma_0^2)$.

How do we estimate $\boldsymbol{w}$ for this model?

Maximum a posterior (MAP) estimate:

$$\begin{aligned} \boldsymbol{w}^{\text{MAP}} &= \arg\max_{\boldsymbol{w}} p(\boldsymbol{w}|\mathcal{D}) = \arg\max_{\boldsymbol{w}} \frac{p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w})}{p(\mathcal{D})} \\ &= \arg\max_{\boldsymbol{w}} p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w}) \end{aligned}$$

Estimating w

Let $\mathbf{x}_1, \ldots, \mathbf{x}_N$ be i.i.d. with $y|\boldsymbol{w}, \boldsymbol{x} \sim N(\boldsymbol{w}^\top \boldsymbol{x}, \sigma_0^2)$; $w_d \sim N(0, \sigma^2)$.

Given σ_0, σ, we choose $\boldsymbol{w}$ so as to maximize:

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

Now we know $p(w_d) \propto \exp\left(\frac{-w_d^2}{2\sigma^2}\right)$ and $p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) \propto \exp\left(\frac{-(\boldsymbol{w}^T \boldsymbol{x}_n - y_n)^2}{2\sigma_0^2}\right)$:

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

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

$$\boldsymbol{w}^{\text{MAP}} = \text{argmin}_{\boldsymbol{w}} \frac{\sum_n (\boldsymbol{w}^\top \boldsymbol{x}_n - y_n)^2}{2\sigma_0^2} + \frac{1}{2\sigma^2} \|\boldsymbol{w}\|_2^2$$

Maximum a posterior (MAP) estimate

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

where $\lambda > 0$ is used to denote σ_0^2/σ^2. This extra term $\|\boldsymbol{w}\|_2^2$ is called regularization/regularizer and controls the magnitude of $\boldsymbol{w}$.

- If $\lambda \to +\infty$, then $\sigma_0^2 \gg \sigma^2$: the variance of noise is far greater than what our prior model can allow for $\boldsymbol{w}$. In this case, our prior model on $\boldsymbol{w}$ will give a simpler model. Numerically,

$$\boldsymbol{w}^{\mathrm{MAP}} \to \mathbf{0}$$

- If $\lambda \to 0$, then we trust our data more. Numerically,

$$\boldsymbol{w}^{\mathrm{MAP}} \to \boldsymbol{w}^{\mathrm{LMS}} = \operatorname{argmin} \sum_n (\boldsymbol{w}^\top \boldsymbol{x}_n - y_n)^2$$

Hyperparameter Tuning and Cross-Validation

How should we choose the right amount of regularization?

Can we tune λ on the training dataset?

No: as this will always set λ to zero, i.e., no regularization, defeating our intention of controlling model complexity

λ is thus a hyperparameter. To tune it,

- We can use a validation set or do cross validation.
- Pick the value of λ that yields lowest error on the testing dataset.

Similar idea applies to tuning learning rate η (or any other hyperparameter) as well.

Tuning by using a validation dataset

Training data are used to learn $f(\cdot)$.

N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{N}}, y_{\text{N}})\}$

Test data are used to assess the prediction error.

- M samples/instances: $\mathcal{D}^{\text{TEST}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{M}}, y_{\text{M}})\}$
- They are used for assessing how well $f(\cdot)$ will do in predicting an unseen $\boldsymbol{x} \notin \mathcal{D}^{\text{TRAIN}}$

Validation data are used to optimize hyperparameter(s).

L samples/instances: $\mathcal{D}^{\text{VAL}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{L}}, y_{\text{L}})\}$

Training data, validation and test data should not overlap!

Recipe

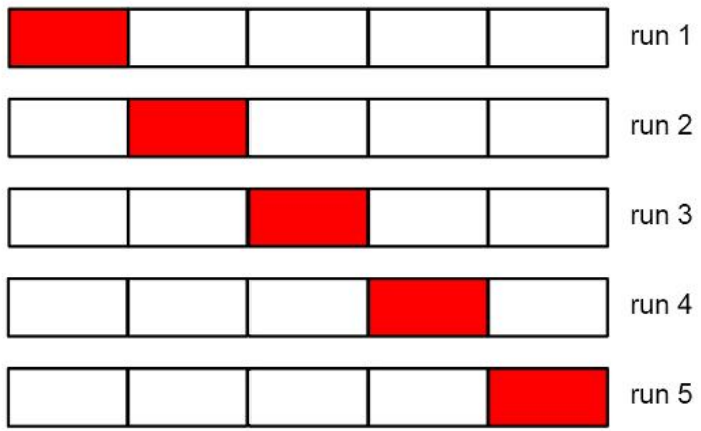
- For each possible value of the hyperparameter (say $\lambda = 1, 3, \cdots, 100$)
  - Train a model using $\mathcal{D}^{\text{TRAIN}}$
  - Evaluate the performance of the model on $\mathcal{D}^{\text{VAL}}$
- Choose the model with the best performance on $\mathcal{D}^{\text{VAL}}$
- Evaluate this model on $\mathcal{D}^{\text{TEST}}$ to get the final prediction error

Cross-validation

What if we do not have validation data?

- We split the training data into S equal parts.
- We use each part in turn as a validation dataset and use the others as a training dataset.
- We choose the hyperparameter such that the model performs the best (based on average, variance, etc.)

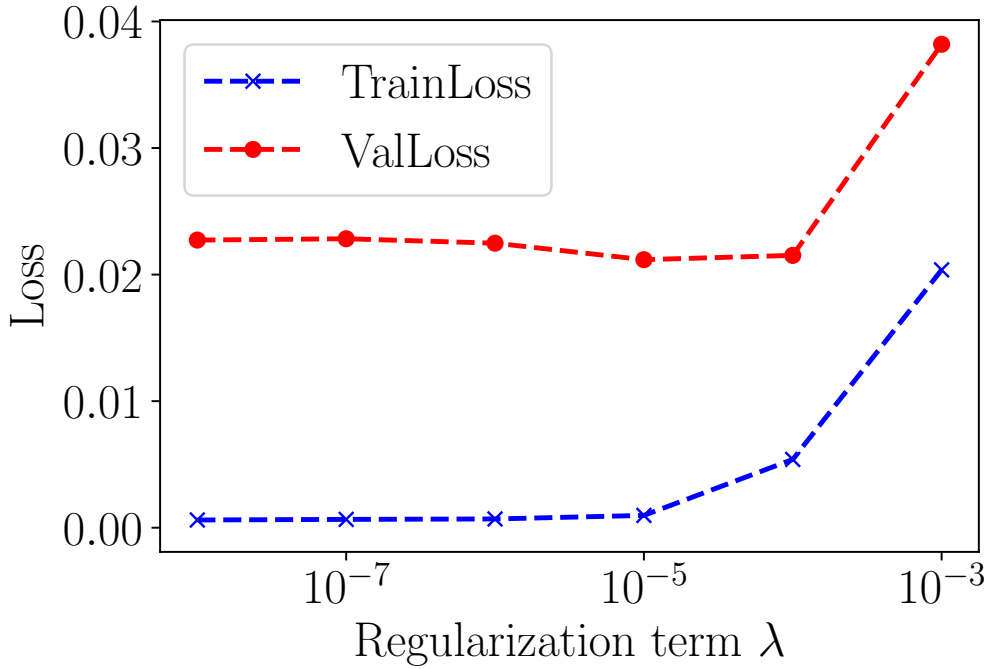
Figure 5: S = 5: 5-fold cross validation



Special case: when S = N, this will be leave-one-out.

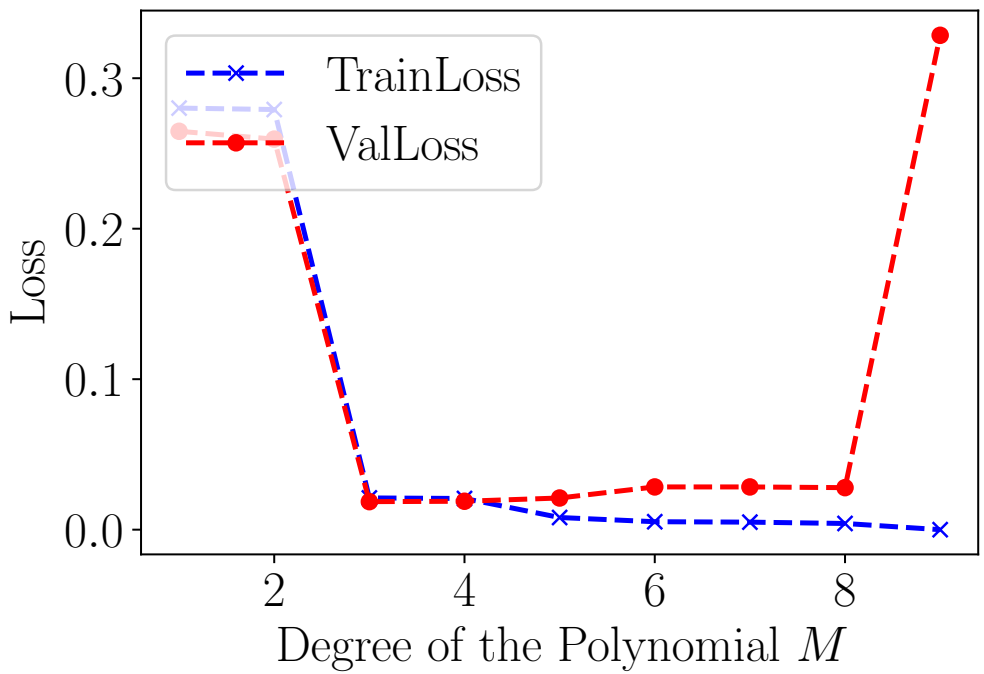
Example: Hyper-parameter Tuning λ

- $\lambda = 10^{-4}$ gives the smallest validation loss
- Strikes a balance between over- and under-fitting



Example: Hyper-parameter Tuning M

- Considering polynomial regression without regularization
- $M = 3$ or $M = 4$ gives the smallest validation loss
- Strikes a balance between over- and under-fitting



Bias-Variance Trade-off

Empirical Risk Minimization

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

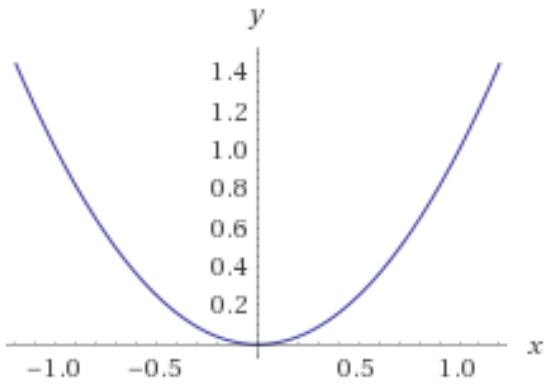
$$\ell(h(\boldsymbol{x}), y)$$

Example:

Quadratic loss function for regression when y is continuous:

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

Ex: when $y = 0$



How good is our predictor?

Risk:

Given the true distribution of data $p(\boldsymbol{x}, y)$, the risk of a given predictor $h(\boldsymbol{x})$ is its expected loss ℓ:

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

However, we cannot compute $R[h(\boldsymbol{x})]$ (we do not know p), so we use the empirical risk, given a training dataset $\mathcal{D}$:

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

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

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

How could this go wrong?

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

For linear regression, $h(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$, and we use squared loss ℓ.

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

$$R[h(\boldsymbol{x})] = \int_{\boldsymbol{x},y} \ell(h(\boldsymbol{x}), y) p(\boldsymbol{x}, y) d\boldsymbol{x} dy$$

What could go wrong with ERM?

- Limited Function Class: The function $h(\boldsymbol{x})$ is restricted to a limited class (e.g. linear functions), which does not allow us to perfectly fit y, even if we had infinitely many training data points.
- Limited Data: We don't know $p(\boldsymbol{x}, y)$, so we must hope that we have enough training data that the empirical risk approximates the real risk. Otherwise, we will overfit to the training data.

Bias-Variance Trade-off: Intuition

- High Bias: Model is not rich enough to fit the training dataset and achieve low training loss
- High Variance: If the training dataset changes slightly, the model changes a lot
- Regularization helps find a middle ground

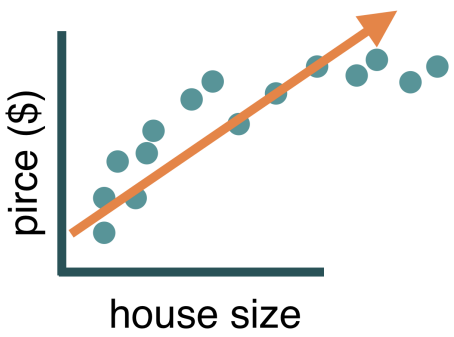


Figure 6: High Bias

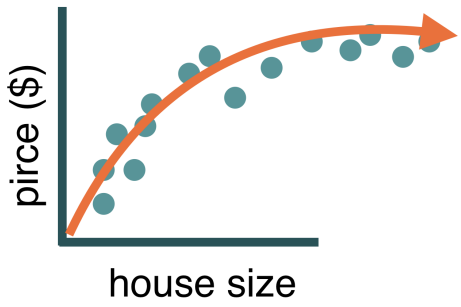


Figure 7: Just Right

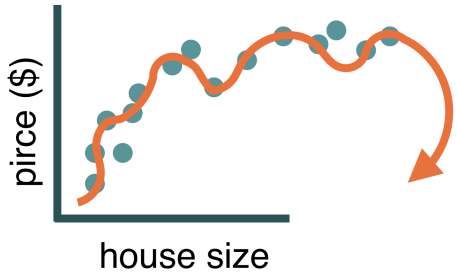


Figure 8: High Variance

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
  We are using the subscript $\mathcal{D}$ to indicate that the prediction function is learned on the specific set of training data $\mathcal{D}$
- $\ell(h(\boldsymbol{x}), y)$: our square 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

Every training sample $\mathcal{D}$ is a sample from the following joint distribution of all possible training datasets

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

Thus, the prediction function $h_{\mathcal{D}}(\boldsymbol{x})$ is a random function with respect to this distribution of possible training datasets. So is also its risk

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

We will now evaluate the expected risk $\mathbb{E}_{\mathcal{D}} R[h_{\mathcal{D}}(\boldsymbol{x})]$: the average risk over the distribution of possible training datasets, $P(\mathcal{D})$.

Bias-Variance Trade-off: Intuition

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...

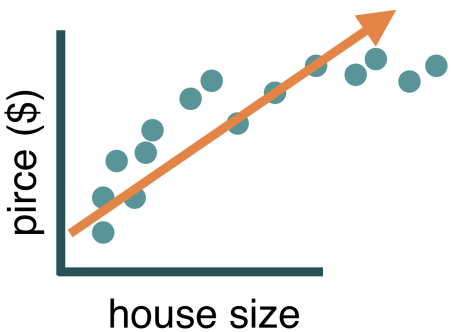


Figure 9: High Bias

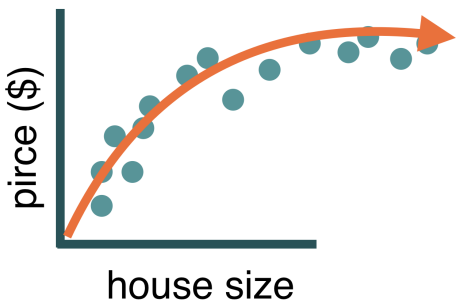


Figure 10: Just Right

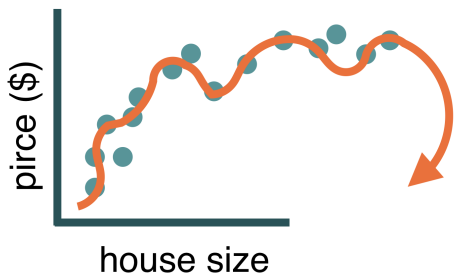


Figure 11: High Variance

Average over the distribution of the training data

Expected risk

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

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 the prediction functions learned on each training dataset.

Variance

We will subtract the averaged prediction from the averaged risk

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

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

Understanding the variance

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

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

For each $(\boldsymbol{x}, y)$ pair, we compute the squared difference of $h_{\mathcal{D}}(\boldsymbol{x})$ (the prediction with training dataset $\mathcal{D}$) and the averaged prediction $\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})$: the average (over all $(\boldsymbol{x}, y) \sim p$) variance of the prediction over $\mathcal{D}$.

How can we reduce the variance?

- 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. An extreme example is $h(\boldsymbol{x}) = \text{const}$.

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 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 add-and-subtract trick, by using an averaged target y (what we want to predict from $\boldsymbol{x}$):

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

Bias and noise

Decompose again

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

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

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

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

Where is the cross-term?

Take-home exercise: Show that it is zero

Analyzing the noise

How can we reduce noise?

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

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 posterior distribution $p(y|\boldsymbol{x})$ at the given $\boldsymbol{x}$.

Figure 12: Somewhat difficult posterior

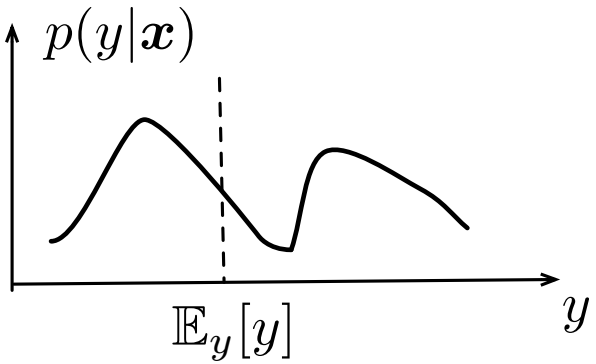
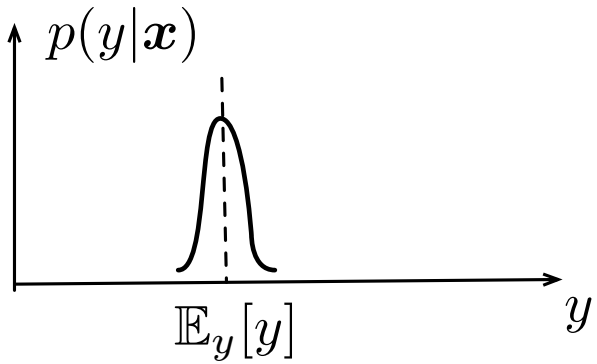


Figure 13: Somewhat easy posterior



Analyzing the bias term

Understanding the bias

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

For each $(\boldsymbol{x}, y)$ pair, we compute the loss of our averaged prediction $\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})$ compared to the expected value of y given $\boldsymbol{x}$, which we compute as $\mathbb{E}_{y}[y|\boldsymbol{x}] = \int_{y} y p(y|\boldsymbol{x}) dy$. Then we take the average over all pairs $(\boldsymbol{x}, y) \sim p(\boldsymbol{x}, y)$.

How can we reduce the bias?

It can be reduced by using more complex models. We shall choose $h(\cdot)$ to be as flexible as possible: the better $h(\cdot)$ approximates $\mathbb{E}_{y}[y|\boldsymbol{x}]$, the smaller the bias. However, this will increase the VARIANCE term.

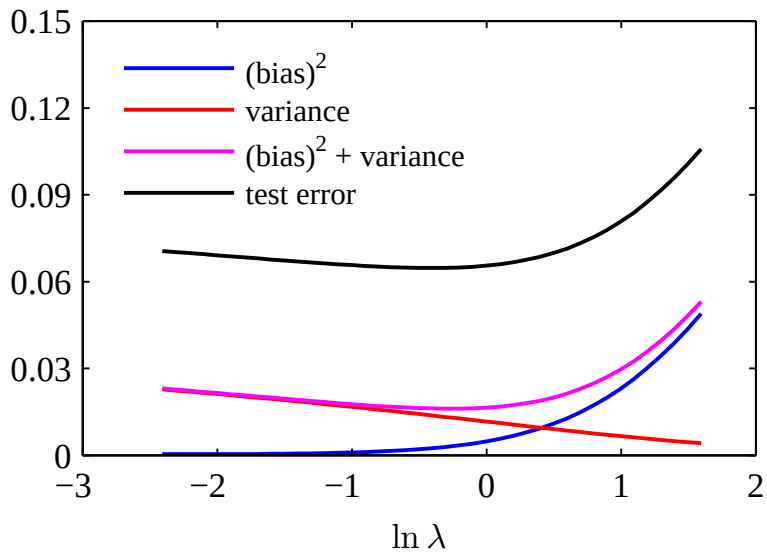
Bias/variance tradeoff

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



Summary of risk components

The average risk (with quadratic loss) can be decomposed as:

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

$$+ \underbrace{\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{y}[y|\boldsymbol{x}]]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy}_{\text{BIAS}^2\text{: error due to the model approximation}}$$

$$+ \underbrace{\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{y}[y|\boldsymbol{x}] - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy}_{\text{NOISE: error due to randomness of } y}$$

Here we define: $h_{\mathcal{D}}(\boldsymbol{x})$ as the output of the model trained on $\mathcal{D}$, $\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})$ as the expectation of the model over all datasets $\mathcal{D}$, and $\mathbb{E}_{y}[y|\boldsymbol{x}]$ as the expected value of y.

Example: Why regularized linear regression could be helpful?

Model

$$h(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$$

Consider the best possible (linear) $h^*(\boldsymbol{x})$

$$\boldsymbol{w}^* = \text{argmin}_{\boldsymbol{w}} \int_{\boldsymbol{x}} [\mathbb{E}_y[y|\boldsymbol{x}] - \boldsymbol{w}^\top \boldsymbol{x}]^2 p(\boldsymbol{x}) d\boldsymbol{x}$$

Note that this linear model assumes the knowledge of joint distribution, thus, not achievable. Intuitively, it is the *best* linear model that can predict the data most accurately.

More refined decomposition of the bias

$$\int_{\boldsymbol{x}} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_y[y|\boldsymbol{x}]]^2 p(\boldsymbol{x}) d\boldsymbol{x} = \int_{\boldsymbol{x}} [h^*(\boldsymbol{x}) - \mathbb{E}_y[y|\boldsymbol{x}]]^2 p(\boldsymbol{x}) d\boldsymbol{x} + \int_{\boldsymbol{x}} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - h^*(\boldsymbol{x})]^2 p(\boldsymbol{x}) d\boldsymbol{x}$$

- Model bias: the price we pay for choosing linear functions to model data. This is the difference between the prediction of the best possible linear model and the actual target.
- Estimation bias: the difference between the optimal model and the estimated model.

Normally, the estimation bias is zero if we do not regularize.

Bias/variance tradeoff for regularized linear regression

We can only adjust estimation bias

$$\int_{\boldsymbol{x}} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}; \lambda) - h^*(\boldsymbol{x})]^2 p(\boldsymbol{x}) d\boldsymbol{x}$$

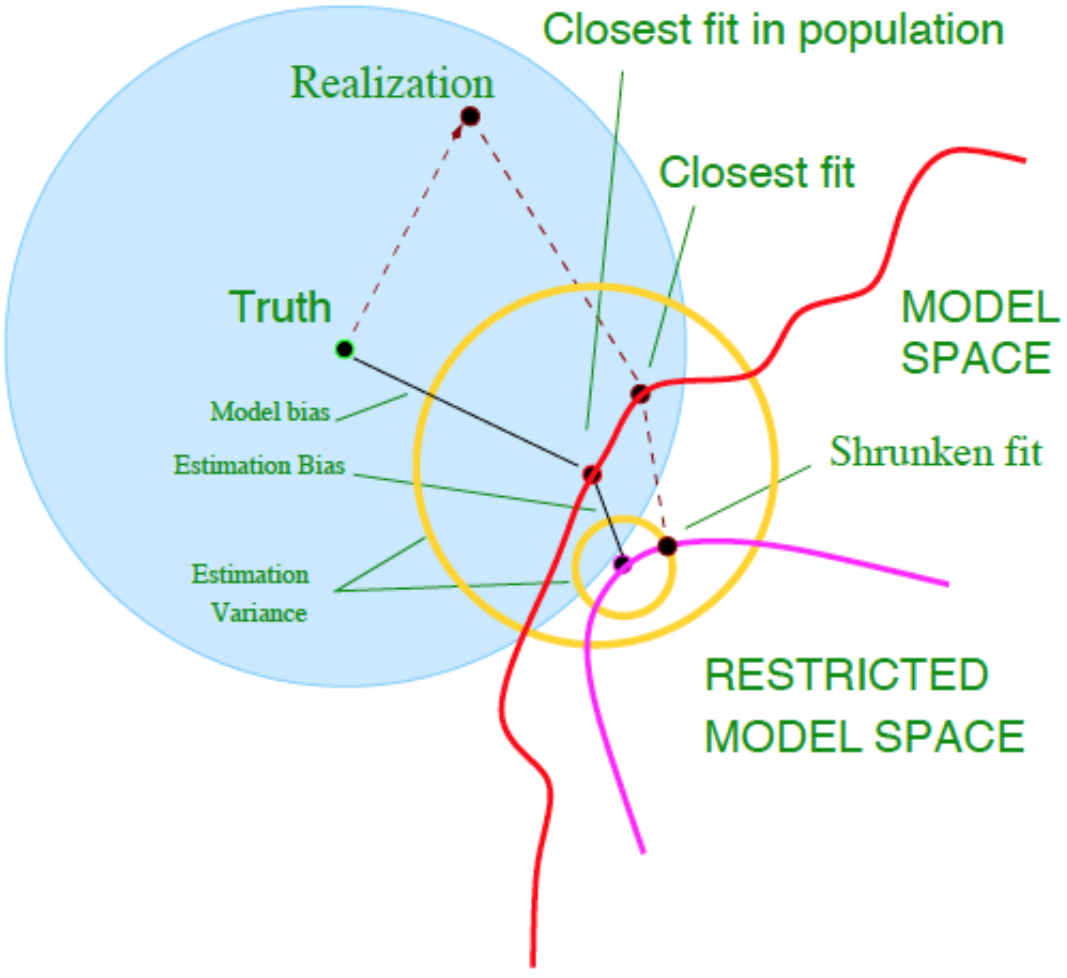
where $h(\boldsymbol{x}; \lambda)$ is the estimated model with regularized linear regression (parameterized with λ).

This term will not be zero anymore!

Thus, bias goes up.

But, as long as this is balanced with a decrease in variance, we are willing to do so.

Visualizing the tradeoff

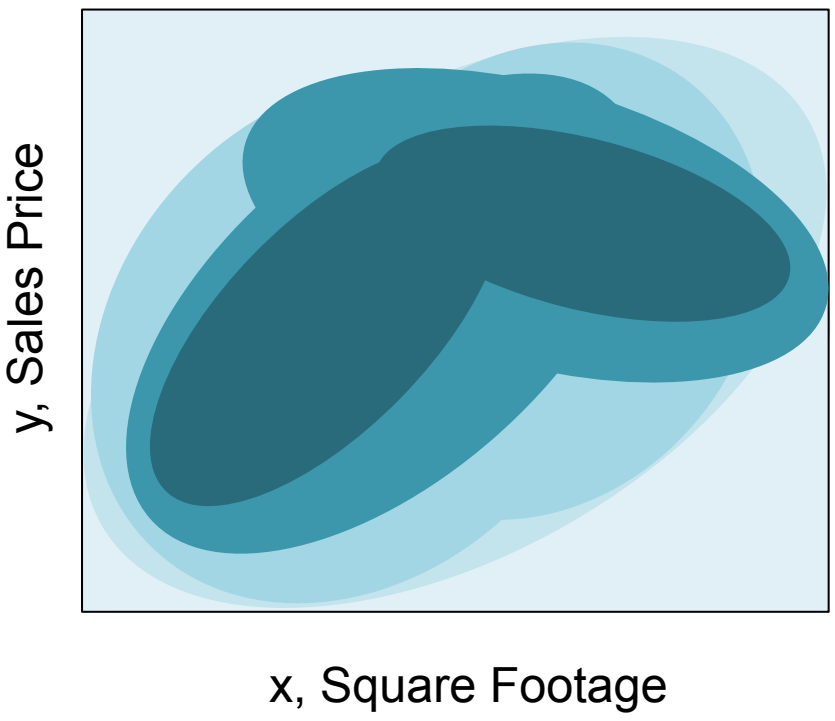


Lecture Summary

- Validation datasets (or cross-validation) are used to determine model hyperparameters.
- Many ML models use empirical risk minimization to find the optimal parameters.
- ERM leads to an error consisting of bias, variance, and noise terms.
  - Variance: Due to only optimizing over an empirical sample of the complete $(\boldsymbol{x}, y)$ distribution.
  - Bias: Due to our choosing a model that does not fit the exact $(\boldsymbol{x}, y)$ relationship.
  - Noise: Due to the output y's randomness with respect to the input $\boldsymbol{x}$.
- Choosing a more complex model improves the bias, but increases the variance (and vice versa for less complex models).
- The noise is independent of the model that we choose.

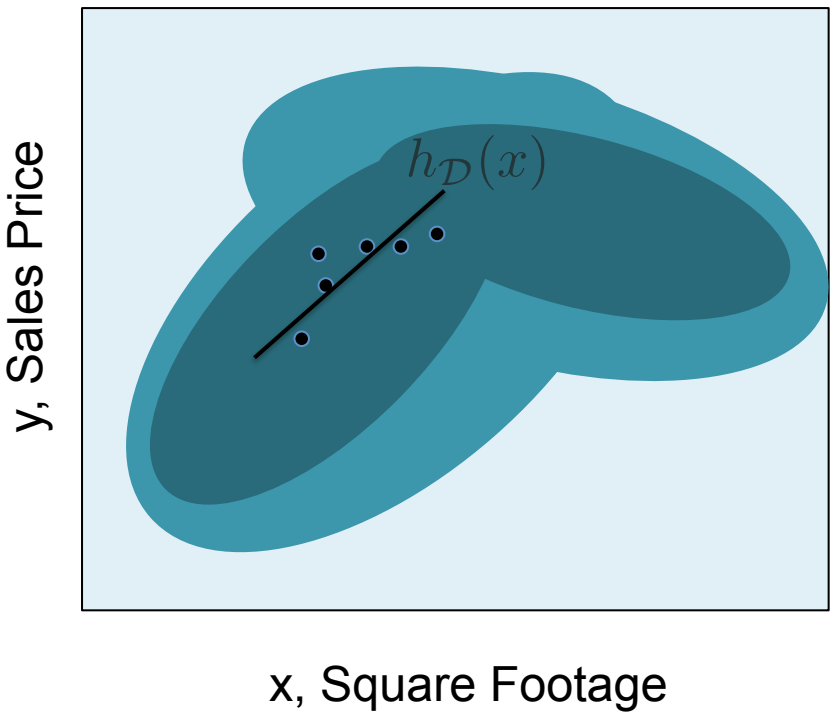
Bias-Variance Trade-off: Illustration

- Joint distribution of square footage x and house sales price y
- Darker color indicates higher probability regions



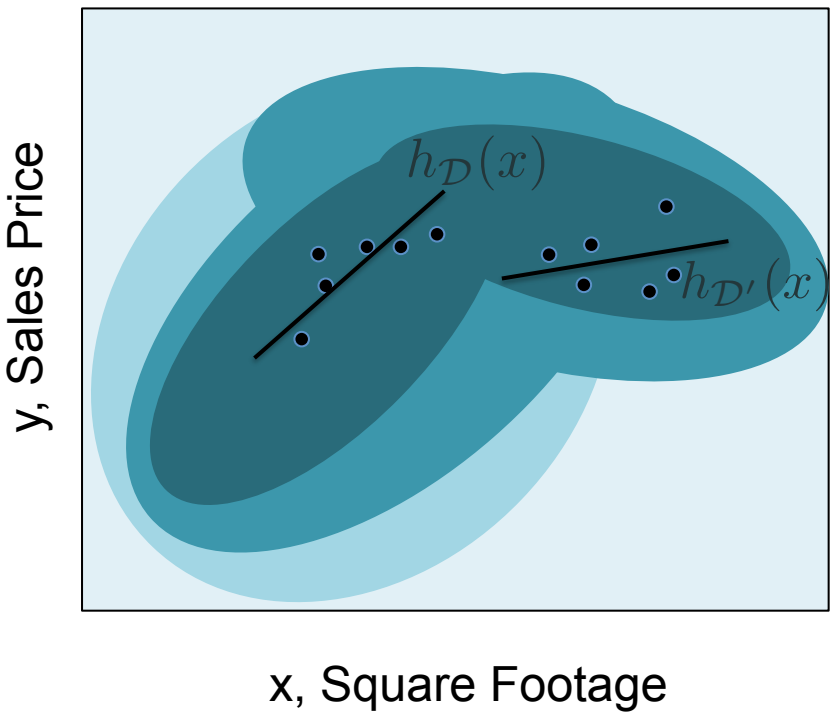
Bias-Variance Trade-off: Illustration

- We have access to dataset $\mathcal{D}$ sampled from the joint distribution
- Learn linear model $h_{\mathcal{D}}(x)$ based on $\mathcal{D}$



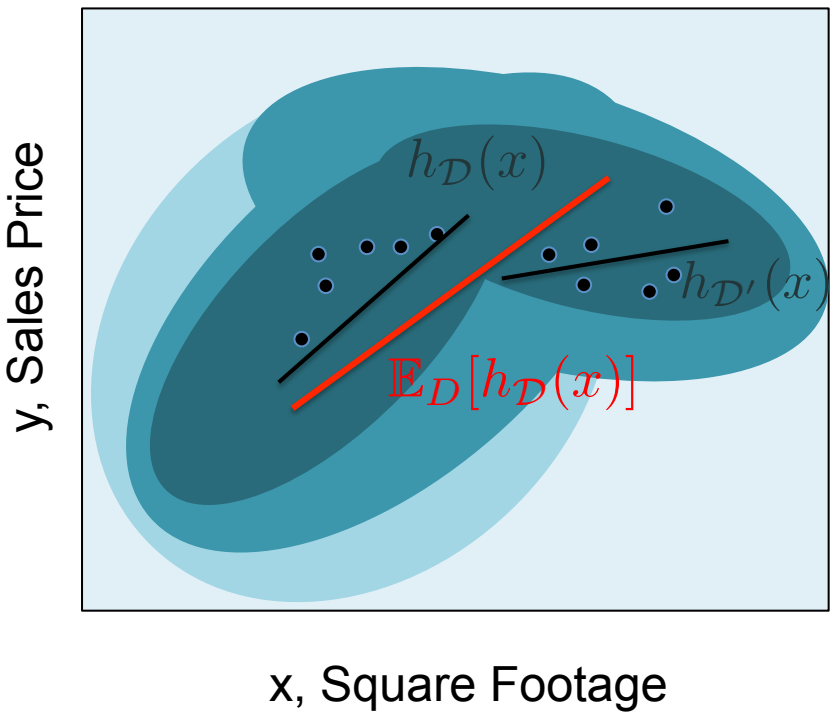
Bias-Variance Trade-off: Illustration

- Repeating this process for a different dataset $\mathcal{D}'$ yields a new linear model $h_{\mathcal{D}}(x)$
- Average of such models over infinitely many datasets sampled from the joint distribution is $\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})$



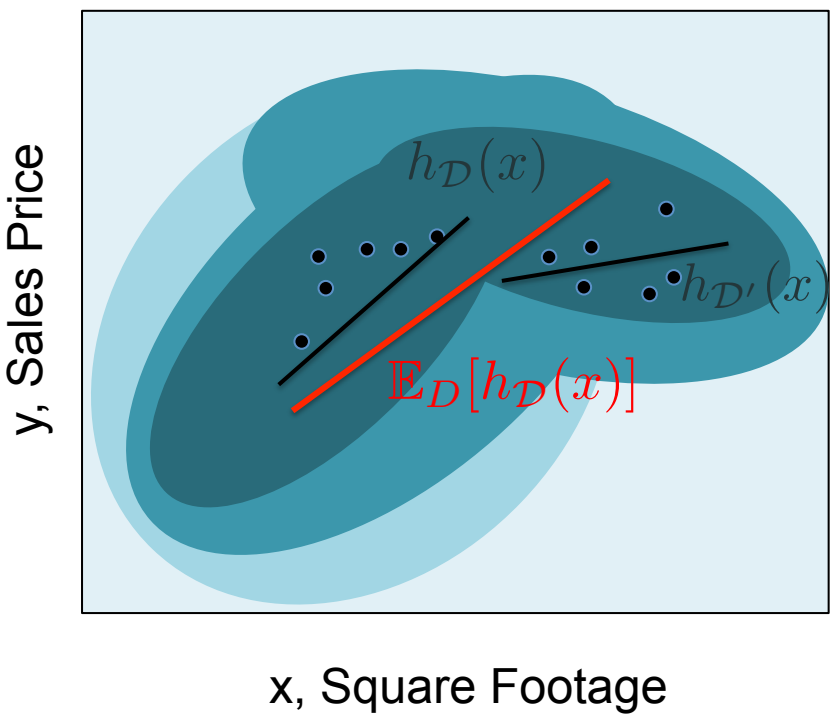
Bias-Variance Trade-off: Illustration

- Repeating this process for a different dataset $\mathcal{D}'$ yields a new linear model $h_{\mathcal{D}}(\boldsymbol{x})$
- Average of such models over infinitely many datasets sampled from the joint distribution is $\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})$



Bias-Variance Trade-off: Illustration

- Average of such models over infinitely many datasets sampled from the joint distribution is $\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})$
- Variance term captures how much individual models differ from the average $\underbrace{\int_{\mathcal{D}} \int_{\boldsymbol{x}} \int_{y} [h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy \; P(\mathcal{D}) d\mathcal{D}}_{\text{VARIANCE: error due to training dataset}}$

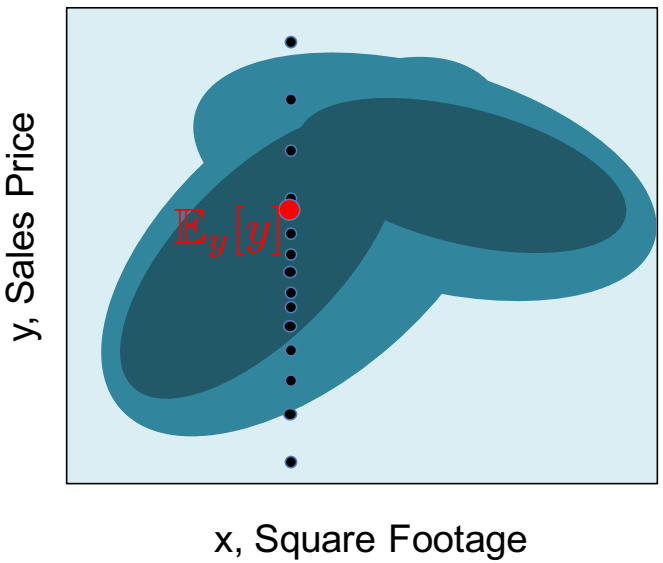


Bias-Variance Trade-off: Illustration

- For a given $\boldsymbol{x}$, we have a conditional distribution $p(y|\boldsymbol{x})$ of sales prices; the Bayesian optimal prediction of the label value for a given $\boldsymbol{x}$ is $\mathbb{E}_y[y|\boldsymbol{x}]$;
- The noise term measures the inherent variance in labels y

$$\underbrace{\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_y[y|\boldsymbol{x}] - y]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy}_{\text{NOISE: error due to randomness of } y}$$

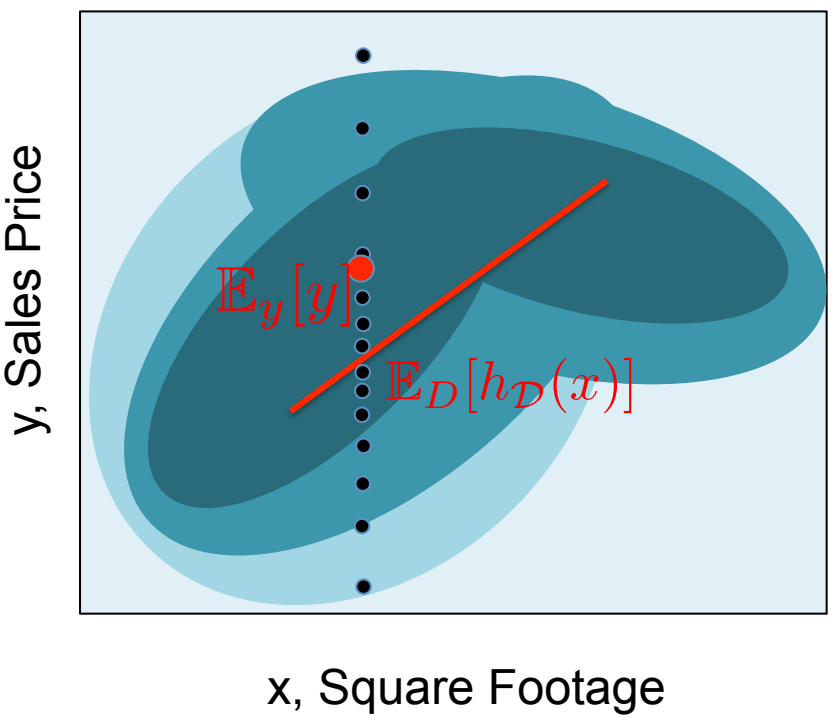
- This term has nothing to do with our prediction $h_{\mathcal{D}}(\boldsymbol{x})$



Bias-Variance Trade-off: Illustration

- If our model class was rich enough (eg. a high-degree polynomial), then $\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x})$ should perfectly match $\mathbb{E}_y[y|\boldsymbol{x}]$
- Restricting to simpler models (eg. linear) results in a bias

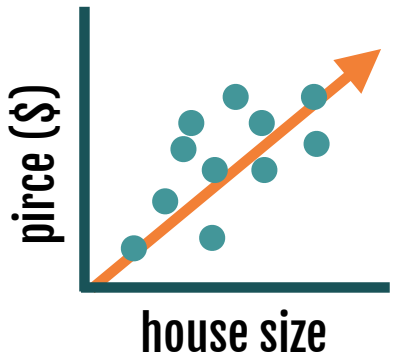
$$\underbrace{\int_{\boldsymbol{x}} \int_{y} [\mathbb{E}_{\mathcal{D}} h_{\mathcal{D}}(\boldsymbol{x}) - \mathbb{E}_y[y|\boldsymbol{x}]]^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy}_{\text{BIAS}^2\text{: error due to the model approximation}}$$



Classification Example: Spam Detection

Task 1: Regression (so far we studied this)

How much should you sell your house for?

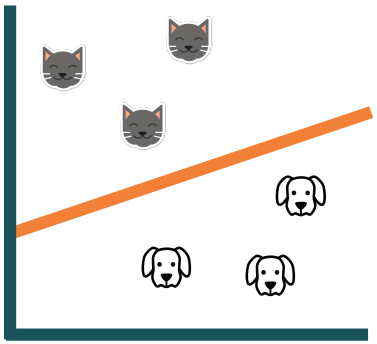


input: houses & features **learn**: $x \rightarrow y$ relationship **predict**: y (*continuous*)

Task 2: Classification (next topic)

Cat or dog?

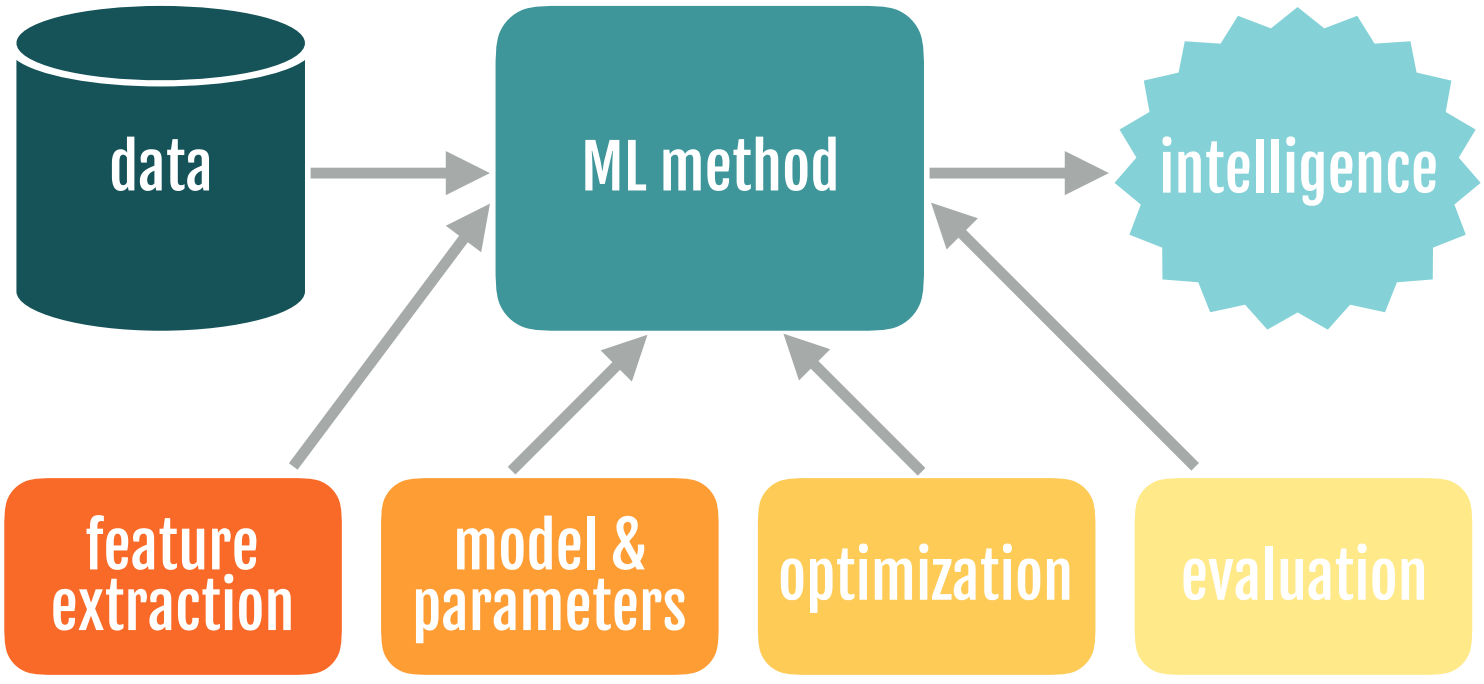
input: cats and dogs



learn: $x \rightarrow y$ relationship

predict: y (*categorical*)

Goal: Learn about ML pipeline



Spam Classification: A daily battle

I'm going to be rich!!

FROM THE DESK OF MR. AMINU SALEH
DIRECTOR, FOREIGN OPERATIONS DEPARTMENT
AFRI BANK PLC
Afribank Plaza,
14th Floormoney344.jpg
51/55 Broad Street,
P.M.B 12021 Lagos-Nigeria

Attention: Honorable Beneficiary,

IMMEDIATE PAYMENT NOTIFICATION VALUED AT **US$10 MILLION**

It is my modest obligation to write you this letter in regards to the authorization of your owed payment through our most respected financial institution (AFRI BANK PLC). I am Mr. Aminu Saleh, The Director, Foreign Operations Department, AFRI Bank Plc, NIGERIA. The British Government, in conjunction with the US GOVERNMENT, WORLD BANK, UNITED NATIONS ORGANIZATION on foreign payment matters, has empowered my bank after much consultation and consideration, to handle all foreign payments and release them to their appropriate beneficiaries with the help of a representative from Federal Reserve Bank.

To facilitate the process of this transaction, please kindly re-confirm the following information below:

1) Your full Name and Address:
2) Phones, Fax and Mobile No. :
3) Profession, Age and Marital Status:
4) Copy of any valid form of your Identification:

How to tell spam from ham?

FROM THE DESK OF MR. AMINU SALEH
DIRECTOR, FOREIGN OPERATIONS DEPARTMENT
AFRI BANK PLC
Afribank Plaza,
14th Floormoney344.jpg
51/55 Broad Street,
P.M.B 12021 Lagos-Nigeria

Attention: Honorable Beneficiary,

IMMEDIATE PAYMENT NOTIFICATION VALUED AT **US$10 MILLION**

Hi Virginia,

Can we meet today at 2pm?

thanks,

Carlee

How might we create features?

Intuition

- Q: How might a human solve this problem?
- A: Simple strategy would be to look for keywords that we often associate with spam

Spam

- We expect to see words like "money", "free", "bank account"

Ham

- Expect to see less spam words, more personalization (e.g., name)

Simple strategy: Count the words

Bag-of-word representation of documents (and textual data)

$$\begin{pmatrix} \text{free} & 100 \\ \text{money} & 2 \\ \vdots & \vdots \\ \text{account} & 2 \\ \vdots & \vdots \end{pmatrix}$$

Just wanted to send a quick reminder about the guest lect
noon. We meet in RTH 105. It has a PC and LCD projector
connection for your laptop if you desire. Maybe we can m
to setup the A/V stuff.

Again, if you would be able to make it around 30 minutes
great.

Thanks so much for your willingness to do this,
Mark

$$\begin{pmatrix} \text{free} & 1 \\ \text{money} & 1 \\ \vdots & \vdots \\ \text{account} & 2 \\ \vdots & \vdots \end{pmatrix}$$

Weighted sum of those telltale words

$$\begin{pmatrix} \text{free} & 100 \\ \text{money} & 2 \\ \vdots & \vdots \\ \text{account} & 2 \\ \vdots & \vdots \end{pmatrix}$$

$$\begin{pmatrix} 100 \times 0.2 \\ 2 \times 0.3 \\ \vdots \\ 2 \times 0.3 \\ \vdots \end{pmatrix}$$

$$\begin{pmatrix} 100 \times 0.01 \\ 2 \times 0.02 \\ \vdots \\ 2 \times 0.01 \\ \vdots \end{pmatrix}$$

different weights for spam and ham: representing how compatible the word pattern is to each category

Weighted sum of those telltale words

$$\begin{pmatrix} \text{free} & 100 \\ \text{money} & 2 \\ \vdots & \vdots \\ \text{account} & 2 \\ \vdots & \vdots \end{pmatrix}$$

$$\begin{pmatrix} 100 \times 0.2 \\ 2 \times 0.3 \\ \vdots \\ 2 \times 0.3 \\ \vdots \end{pmatrix}$$

$$\begin{pmatrix} 100 \times 0.01 \\ 2 \times 0.02 \\ \vdots \\ 2 \times 0.01 \\ \vdots \end{pmatrix}$$

different weights for spam and ham: representing how compatible the word pattern is to each category

= 3.2

= 1.03

Our intuitive model of classification

1. **Assign weight to each word**
   - Let s:= spam weights, h:= ham weights
   - Compute compatibility score of **spam**
     - (# "free" $\times s_{\text{free}}$)+(# "account" $\times s_{\text{account}}$)+(# "money" $\times s_{\text{money}}$)
   - Compute compatibility score of **ham**
     - (# "free" $\times h_{\text{free}}$)+(# "account" $\times h_{\text{account}}$)+(# "money" $\times h_{\text{money}}$)
2. **Make a decision**
   - if spam score $>$ ham score then spam
   - else ham

Your turn

Suppose you see the following email:

```
CONGRATULATIONS!!  Your email address have won you the
lottery sum of US$2,500,000.00 USD to claim your prize,
  contact your office agent (Athur walter) via email
  claims2155@yahoo.com.hk or call +44 704 575 1113
```

And our weights for spam and ham are:
spam: [lottery=0.3, prize=0.3, office=0.01, email=0.01, . . .]
ham: [lottery=0.01, prize=0.01, office=0.1, email=0.05, . . .]

Will we predict that the email is spam or ham?

spam = 0.3*1 + 0.3*1 + 0.01*1 + 0.01*2 =0.63
ham = 0.01*1 + 0.01*1 + 0.1*1 + 0.05*2 = 0.22
so we predict spam!

How do we get the weights?

Learn from experience

- get a lot of spams
- get a lot of hams

But what to optimize?

Naive Bayes Model

Naive Bayes model for identifying spam

- **Class label**: binary
  - $y = \{$ spam, ham $\}$
- **Features**: word counts in the document (bag-of-words)
  - $\mathbf{x} = \{$('free', 100), ('lottery', 5), ('money', 10)$\}$
  - Each pair is in the format of (word$_i$, #word$_i$), namely, a unique word in the dictionary, and the number of times it shows up
- **Model**

$$p(\mathbf{x}|spam) = p(\text{`}free'|spam)^{100} p(\text{`}lottery'|spam)^{5} p(\text{`}money'|spam)^{10} \cdots$$

  - Choose the "most likely" option: $p(\mathbf{x}|spam)p(spam)$ vs. $p(\mathbf{x}|ham)p(ham)$

These conditional probabilities are the parameters we need to estimate

What is naive about this?

- Strong assumption of conditional independence:

$$p(\text{word}_i, \text{word}_j|y) = p(\text{word}_i|y)p(\text{word}_j|y)$$

- Previous example:

$$p(\mathbf{x}|spam) = p(\text{`}free'|spam)^{100}p(\text{`}lottery'|spam)^{5}p(\text{`}money'|spam)^{10}\cdots$$

- Independence across different words as well as multiple occurrences of the same word
- This assumption makes estimation much easier (as we'll see)

Naive Bayes classification rule

For any document $\mathbf{x}$, we want to compare $p(\text{spam}|\mathbf{x})$ and $p(\text{ham}|\mathbf{x})$

Recall that by Bayes rule we have:

$$p(\text{spam}|\mathbf{x}) = \frac{p(\mathbf{x}|\text{spam})p(\text{spam})}{p(\mathbf{x})}$$

$$p(\text{ham}|\mathbf{x}) = \frac{p(\mathbf{x}|\text{ham})p(\text{ham})}{p(\mathbf{x})}$$

Denominators are the same, and easier to compute logarithms, so we compare the spam and ham probabilities:

$$\log[p(\mathbf{x}|\text{spam})p(\text{spam})] \quad \text{versus} \quad \log[p(\mathbf{x}|\text{ham})p(\text{ham})]$$

Classifier in linear form

$$\log[p(\mathbf{x}|\text{spam})p(\text{spam})] = \log\left[\prod_i p(\text{word}_i|\text{spam})^{\#\text{word}_i} p(\text{spam})\right]$$
$$= \sum_i (\#\text{word}_i) \log p(\text{word}_i|\text{spam}) + \log p(\text{spam})$$

Similarly, we have

$$\log[p(\mathbf{x}|\text{ham})p(\text{ham})] = \sum_i (\#\text{word}_i) \log p(\text{word}_i|\text{ham}) + \log p(\text{ham})$$

We're back to the idea of comparing weighted sums of word occurrences!

$\log p(spam)$ *and* $\log p(ham)$ *are called "priors" (in our initial example we did not include them but they are important!)*

Mini-summary

What we have shown

By assuming a probabilistic model (i.e., Naive Bayes), we are able to derive a decision rule that is consistent with our intuition

Our next step is to learn the parameters from data

What are the parameters to learn?

Parameter Estimation

Formal definition of Naive Bayes

General case

Given a random vector $\mathbf{X} \in \mathbb{R}^K$ and a dependent variable $Y \in [C]$, the Naive Bayes model defines the joint distribution

$$\begin{aligned} P(\mathbf{X} = \mathbf{x}, Y = c) &= P(Y = c) P(\mathbf{X} = \mathbf{x} | Y = c) \\ &= P(Y = c) \prod_{k=1}^{K} P(\text{word}_k | Y = c)^{x_k} \\ &= \pi_c \prod_{k=1}^{K} \theta_{ck}^{x_k} \end{aligned}$$

where $\pi_c = P(Y = c)$ is the prior probability of class c, x_k is the number of occurences of the kth word, and $\theta_{ck} = P(\text{word}_k | Y = c)$ is the weight of the kth word for the cth class.

Learning problem

Training data

$$\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{\mathrm{N}} \rightarrow \mathcal{D} = \{(\{x_{nk}\}_{k=1}^{\mathrm{K}}, y_n)\}_{n=1}^{\mathrm{N}}$$

Goal

Learn $\pi_c, c = 1, 2, \cdots, \mathrm{C}$, and $\theta_{ck}, \forall c \in [\mathrm{C}], k \in [\mathrm{K}]$ under the constraints:

$$\sum_c \pi_c = 1$$

and

$$\sum_k \theta_{ck} = \sum_k P(\mathrm{word}_k | Y = c) = 1$$

as well as: all π_c, $\theta_{ck} \geq 0$.

Our hammer: Maximum likelihood estimation

Likelihood of the training data

$$\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N} \rightarrow \mathcal{D} = \{(\{x_{nk}\}_{k=1}^{K}, y_n)\}_{n=1}^{N}$$

$$L = P(\mathcal{D}) = \prod_{n=1}^{N} \pi_{y_n} P(\mathbf{x}_n | y_n)$$

Log-Likelihood of the training data

$$\mathcal{L} = \log P(\mathcal{D}) = \log \prod_{n=1}^{N} \pi_{y_n} P(\mathbf{x}_n | y_n)$$

Our hammer: Maximum likelihood estimation

Log-Likelihood of the training data

$$\begin{aligned}
\mathcal{L} = \log P(\mathcal{D}) &= \log \prod_{n=1}^{N} \pi_{y_n} P(\mathbf{x}_n | y_n) \\
&= \log \prod_{n=1}^{N} \left(\pi_{y_n} \prod_k \theta_{y_n k}^{x_{nk}} \right) \\
&= \sum_n \left(\log \pi_{y_n} + \sum_k x_{nk} \log \theta_{y_n k} \right) \\
&= \sum_n \log \pi_{y_n} + \sum_{n,k} x_{nk} \log \theta_{y_n k}
\end{aligned}$$

Optimize it!

$$(\pi_c^*, \theta_{ck}^*) = \arg\max \left(\sum_n \log \pi_{y_n} + \sum_{n,k} x_{nk} \log \theta_{y_n k} \right)$$

Separating the optimization variables

Note the separation of parameters in the likelihood

$$\sum_n \log \pi_{y_n} + \sum_{n,k} x_{nk} \log \theta_{y_n k}$$

this implies that $\{\pi_c\}$ and $\{\theta_{ck}\}$ can be estimated separately

Reorganize terms

$$\sum_n \log \pi_{y_n} = \sum_c \log \pi_c \times (\#\text{of data points labeled as c})$$

and

$$\sum_{n,k} x_{nk} \log \theta_{y_n k} = \sum_c \sum_{n:y_n=c} \sum_k x_{nk} \log \theta_{ck} = \sum_c \sum_{n:y_n=c,k} x_{nk} \log \theta_{ck}$$

The latter implies $\{\theta_{ck}\}$ and $\{\theta_{c'k}\}$ for $c \neq c'$ can be estimated independently!

Estimating $\{\pi_c\}$

We want to maximize

$$\sum_c \log \pi_c \times (\#\text{of data points labeled as c})$$

Intuition

- Similar to roll a dice (or flip a coin): each side of the dice shows up with a probability of π_c (total C sides)
- And we have total N trials of rolling this dice

Solution

$$\pi_c^* = \frac{\#\text{of data points labeled as c}}{\text{N}}$$

Estimating $\{\theta_{ck}, k = 1, 2, \cdots, \mathrm{K}\}$

We want to maximize

$$\sum_{n:y_n=c,k} x_{nk} \log \theta_{ck}$$

Intuition

- Again similar to roll a dice: each side of the dice shows up with a probability of θ_{ck} (total K sides)
- And we have total $\sum_{n:y_n=c,k} x_{nk}$ trials (times a word shows up in class c).

Solution

$$\theta^*_{ck} = \frac{\#\text{of times word } k \text{ shows up in data points labeled as } c}{\#\text{total trials for data points labeled as } c}$$

Translating back to our problem of detecting spam emails

- Collect a lot of ham and spam emails as training examples
- Estimate the "prior"

$$p(\text{ham}) = \frac{\#\text{of ham emails}}{\#\text{of emails}}, \quad p(\text{spam}) = \frac{\#\text{of spam emails}}{\#\text{of emails}}$$

- Estimate the weights, e.g., $p(\text{funny\_word}|\text{ham})$

$$p(\text{funny\_word}|\text{ham}) = \frac{\#\text{of funny\_word in ham emails}}{\#\text{of words in ham emails}}$$

$$p(\text{funny\_word}|\text{spam}) = \frac{\#\text{of funny\_word in spam emails}}{\#\text{of words in spam emails}}$$

Example: Spam Classification

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	1	2	3	Ham
Email 4	1	2	3	2	Ham

Find ML estimates of parameters π_c and θ_{ck}

$$\theta_{spam,free} = Pr(free|spam)$$
$$\theta_{spam,bank} = Pr(bank|spam)$$
$$\theta_{spam,meet} = Pr(meet|spam)$$
$$\theta_{spam,time} = Pr(time|spam)$$
$$\pi_{spam} = Pr(spam)$$

Example: Spam Classification

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	1	2	3	Ham
Email 4	1	2	3	2	Ham

Find ML estimates of parameters π_c and θ_{ck}

$$\theta_{spam,free} = Pr(free|spam) = (5+4)/(5+3+1+1+4+2+1+1) = 9/18$$
$$\theta_{spam,bank} = Pr(bank|spam) = (3+2)/18 = 5/18$$
$$\theta_{spam,meet} = Pr(meet|spam) = 2/18$$
$$\theta_{spam,time} = Pr(time|spam) = 2/18$$
$$\pi_{spam} = Pr(spam) = 2/4$$

Example: Spam Classification

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	1	2	3	Ham
Email 4	1	2	3	2	Ham

Find ML estimates of parameters π_c and θ_{ck}

$$\theta_{ham,free} = Pr(free|ham)$$
$$\theta_{ham,bank} = Pr(bank|ham)$$
$$\theta_{ham,meet} = Pr(meet|ham)$$
$$\theta_{ham,time} = Pr(time|ham)$$
$$\pi_{ham} = Pr(ham)$$

Example: Spam Classification

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	1	2	3	Ham
Email 4	1	2	3	2	Ham

Find ML estimates of parameters π_c and θ_{ck}

$$\theta_{ham,free} = Pr(free|ham) = 3/16$$
$$\theta_{ham,bank} = Pr(bank|ham) = 3/16$$
$$\theta_{ham,meet} = Pr(meet|ham) = 5/16$$
$$\theta_{ham,time} = Pr(time|ham) = 5/16$$
$$\pi_{ham} = Pr(ham) = 2/4$$

Classification rule

Given an unlabeled point $\mathbf{x} = \{x_k, k = 1, 2, \cdots, \mathrm{K}\}$, how to label it?

$$
\begin{aligned}
y^* &= \arg\max_{c \in [\mathrm{C}]} P(y = c | \mathbf{x}) \\
&= \arg\max_{c \in [\mathrm{C}]} P(y = c) P(\mathbf{x} | y = c) \\
&= \arg\max_c [\log \pi_c + \sum_k x_k \log \theta_{ck}]
\end{aligned}
$$

Choose class c that maximizes the log-likelihood of an observed email

Example: Spam Classification

$$\theta_{spam,free} = Pr(free|spam) = 9/18 \qquad \theta_{ham,free} = Pr(free|ham) = 3/16$$
$$\theta_{spam,bank} = Pr(bank|spam) = 5/18 \qquad \theta_{ham,bank} = Pr(bank|ham) = 3/16$$
$$\theta_{spam,meet} = Pr(meet|spam) = 2/18 \qquad \theta_{ham,meet} = Pr(meet|ham) = 5/16$$
$$\theta_{spam,time} = Pr(time|spam) = 2/18 \qquad \theta_{ham,time} = Pr(time|ham) = 5/16$$
$$\pi_{spam} = Pr(spam) = 2/4 \qquad \pi_{ham} = Pr(ham) = 2/4$$

We observe a new email with the word counts (free, bank, meet, time) = (1,3,4,2). Should it be classified as spam or ham?

$$\begin{aligned}\log \Pr(spam|\mathbf{x}) &\propto \log\left(\Pr(spam)\cdot \Pr(\mathbf{x}|spam)\right)\\ &= \log\left(\frac{2}{4}\cdot\left(\frac{9}{18}\right)\left(\frac{5}{18}\right)^3\left(\frac{2}{18}\right)^4\left(\frac{2}{18}\right)^2\right)\\ &= -7.99\end{aligned}$$

Example: Spam Classification

$$\theta_{spam,free} = Pr(free|spam) = 9/18 \qquad \theta_{ham,free} = Pr(free|ham) = 3/16$$
$$\theta_{spam,bank} = Pr(bank|spam) = 5/18 \qquad \theta_{ham,bank} = Pr(bank|ham) = 3/16$$
$$\theta_{spam,meet} = Pr(meet|spam) = 2/18 \qquad \theta_{ham,meet} = Pr(meet|ham) = 5/16$$
$$\theta_{spam,time} = Pr(time|spam) = 2/18 \qquad \theta_{ham,time} = Pr(time|ham) = 5/16$$
$$\pi_{spam} = Pr(spam) = 2/4 \qquad \pi_{ham} = Pr(ham) = 2/4$$

We observe a new email with the word counts (free, bank, meet, time) = (1,3,4,2). Should it be classified as spam or ham?

$$\begin{aligned}\log \Pr(ham|\mathbf{x}) &\propto \log \Pr(ham) \cdot \Pr(\mathbf{x}|ham) \\ &= \log\left(\frac{2}{4} \cdot \left(\frac{3}{16}\right)\left(\frac{3}{16}\right)^3\left(\frac{5}{16}\right)^4\left(\frac{5}{16}\right)^2\right) \\ &= -6.2399\end{aligned}$$

Example: Spam Classification

We observe a new email with the word counts (free, bank, meet, time) = (1,3,4,2). Should it be classified as spam or ham?

$$\begin{aligned}\log \Pr(spam|\mathbf{x}) &\propto \log \Pr(spam) \cdot \Pr(\mathbf{x}|spam) \\ &= \log\left(\frac{2}{4} \cdot \left(\frac{9}{18}\right)\left(\frac{5}{18}\right)^3\left(\frac{2}{18}\right)^4\left(\frac{2}{18}\right)^2\right) \\ &= -7.99\end{aligned}$$

$$\begin{aligned}\log \Pr(ham|\mathbf{x}) &\propto \log \Pr(ham) \cdot \Pr(\mathbf{x}|ham) \\ &= \log\left(\frac{2}{4} \cdot \left(\frac{3}{16}\right)\left(\frac{3}{16}\right)^3\left(\frac{5}{16}\right)^4\left(\frac{5}{16}\right)^2\right) \\ &= -6.2399\end{aligned}$$

ANSWER: Ham

Missing features: Some words never occur in ham emails

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	0	2	3	Ham
Email 4	1	0	3	2	Ham

Find ML estimates of parameters π_c and θ_{ck}

In this training phase, we can just use all available values and ignore missing values

$$\theta_{spam,free} = Pr(free|spam) = 9/18 \quad \theta_{ham,free} = Pr(free|ham) = 3/13$$
$$\theta_{spam,bank} = Pr(bank|spam) = 5/18 \quad \theta_{ham,bank} = Pr(bank|ham) = 0/13$$
$$\theta_{spam,meet} = Pr(meet|spam) = 2/18 \quad \theta_{ham,meet} = Pr(meet|ham) = 5/13$$
$$\theta_{spam,time} = Pr(time|spam) = 2/18 \quad \theta_{ham,time} = Pr(time|ham) = 5/13$$
$$\pi_{spam} = Pr(spam) = 2/4 \quad \pi_{ham} = Pr(ham) = 2/4$$

Missing features: Test phase

$$\theta_{spam,free} = Pr(free|spam) = 9/18 \qquad \theta_{ham,free} = Pr(free|ham) = 3/13$$
$$\theta_{spam,bank} = Pr(bank|spam) = 5/18 \qquad \theta_{ham,bank} = Pr(bank|ham) = 0/13$$
$$\theta_{spam,meet} = Pr(meet|spam) = 2/18 \qquad \theta_{ham,meet} = Pr(meet|ham) = 5/13$$
$$\theta_{spam,time} = Pr(time|spam) = 2/18 \qquad \theta_{ham,time} = Pr(time|ham) = 5/13$$
$$\pi_{spam} = Pr(spam) = 2/4 \qquad \pi_{ham} = Pr(ham) = 2/4$$

New email with the word counts (free, bank, meet, time) = (1,3,4,2),

$$\begin{aligned}\log \Pr(ham|\mathbf{x}) &\propto \log \Pr(ham) \cdot \Pr(\mathbf{x}|ham) \\ &= \log\left(\frac{2}{4} \cdot \left(\frac{3}{13}\right)\left(\frac{0}{13}\right)^3\left(\frac{5}{13}\right)^4\left(\frac{5}{13}\right)^2\right) \\ &= -\infty\end{aligned}$$

Problem: The email is ALWAYS classified as spam. Just because an event has not happened yet, doesn't mean that it won't ever happen.

Dealing with missing features: Solution 1

Remove the features that take zero values for one or more classes

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	0	2	3	Ham
Email 4	1	0	3	2	Ham

Remove the 'bank' column

Dealing with missing features: Solution 1

Remove the features that take zero values for one or more classes

	free	meet	time	y
Email 1	5	1	1	Spam
Email 2	4	1	1	Spam
Email 3	2	2	3	Ham
Email 4	1	3	2	Ham

We can then use the same procedure as before to learn the parameters, and then use these parameters to classify new emails

But then we are wasting a lot of useful data..

Dealing with missing features: Solution 2

Use Laplacian smoothing: Pretend you've seen each word 1 extra time for each class. This is called a 'pseudo-count'.

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	0	2	3	Ham
Email 4	1	0	3	2	Ham

$$\theta_{ham,free} = Pr(free|ham) = 3/13$$
$$\theta_{ham,bank} = Pr(bank|ham) = 0/13$$
$$\theta_{ham,meet} = Pr(meet|ham) = 5/13$$
$$\theta_{ham,time} = Pr(time|ham) = 5/13$$
$$\pi_{ham} = Pr(ham) = 2/4$$

Dealing with missing features: Solution 2

Use Laplacian smoothing: Pretend you've seen each word 1 extra time for each class (spam and ham). This is called a 'pseudo-count'.

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	0	2	3	Ham
Email 4	1	0	3	2	Ham

$$\theta_{ham,free} = Pr(free|ham) = (3+1)/(13+4)$$
$$\theta_{ham,bank} = Pr(bank|ham) = (0+1)/(13+4)$$
$$\theta_{ham,meet} = Pr(meet|ham) = (5+1)/(13+4)$$
$$\theta_{ham,time} = Pr(time|ham) = (5+1)/(13+4)$$
$$\pi_{ham} = Pr(ham) = 2/4$$

Dealing with missing features: Solution 2

More generally, pretend you've seen each word $\alpha \geq 1$ extra times.

	free	bank	meet	time	y
Email 1	5	3	1	1	Spam
Email 2	4	2	1	1	Spam
Email 3	2	0	2	3	Ham
Email 4	1	0	3	2	Ham

$$p(\text{funny\_word}|\text{spam}) = \frac{\#\text{of funny\_word in spam emails} + \alpha}{\#\text{of words in spam emails} + \alpha \times \#\text{of unique words}}$$

Laplacian Smoothing: History and effect on ML estimate

History: What is the prob. that the sun will rise tomorrow?

- Given a large sample of days with the rising sun, we still can not be completely sure that the sun will still rise tomorrow

$$\Pr(\text{sun rising tomorrow} | \text{it rose } t \text{ times}) = \frac{t+1}{t+2}$$

Effect on the ML estimate

- Laplace smoothing biases the ML estimate
- Equivalent to performing MAP estimation with a Dirichlet (multi-variate Beta) prior
- As training data size grows, the effect of Laplacian smoothing disappears

Summary

You should know

- what a Naive Bayes model is
- why it is 'naive'
- how this model can be used to classify spam vs ham emails
- Handle missing features via Laplace smoothing

Moving forward

Examine the classification rule for naive Bayes

$$y^* = \arg\max_c \left(\log \pi_c + \sum_k x_k \log \theta_{ck} \right)$$

For binary classification, we thus determine the label based on the sign of

$$\log \pi_1 + \sum_k x_k \log \theta_{1k} - \left(\log \pi_2 + \sum_k x_k \log \theta_{2k} \right)$$

This is just a linear function of the features (word-counts) $\{x_k\}$

$$w_0 + \sum_k x_k w_k$$

where we "absorb" $w_0 = \log \pi_1 - \log \pi_2$ and $w_k = \log \theta_{1k} - \log \theta_{2k}$.

Naive Bayes is a linear classifier

Fundamentally, what really matters is the decision boundary

$$w_0 + \sum_k x_k w_k$$

This motivates many new methods. One of them is logistic regression, to be discussed in next lecture.