Quantifying Performance

Contents

- nearest neighbor classifiers, histogram classifiers
- performance metrics in machine learning
- loss and risk

Training: estimating a function from data

- 1-d example: learning a function $f(\cdot) : \mathbb{R} \mapsto \mathbb{R}$
- start with a **model** - *limited set of possible functions*

$$\text{example: } f(x) = a + b\cos\left(\frac{2\pi x}{T} + \theta\right)$$

- plot some points, find paramaters (a, b, θ, T) that fit data

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n = \{(\text{April 1 1981}, 3.2), (\text{June 17 1956}, 18.2), \dots\}$$

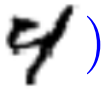
1. $f(x) \approx y$ for (most) of the training data
2. $f(x)$ should work for brand new x

Finding efficient ways to learn functions from data is (one of the) fundamental challenges of machine learning.

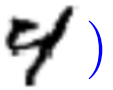
Probabilistic models

$x =$ 

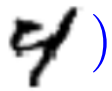
What is the probability of each class given the data?

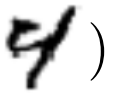
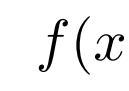
$\mathbb{P}(y = 0 | x =$  $)$

$\vdots$

$\mathbb{P}(y = 4 | x =$  $)$

$\vdots$

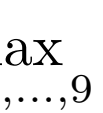
$\mathbb{P}(y = 9 | x =$  $)$

$$f(x = \text{  }) = \arg\max_{i=0,\ldots,9} \mathbb{P}(y = i | x = \text{  })$$

MAP (maximum a posteriori) estimate

Probabilistic models use probability to help design $f(x)$.

Memorization, histogram classifier

$$f(x = \text{  }) = \arg\max_{i=0,\ldots,9} \mathbb{P}(y = i | x = \text{  })$$

for $x =$  , count how many examples in $\mathcal{D}$ correspond to $y = 1$, $y = 2$, ...

$$\mathbb{P}(y = 0 | x = \text{  }) \approx \frac{\text{count of } x = \text{  } \text{ with } y = 0 \text{ in } \mathcal{D}}{\text{count of } x = \text{  } \text{ in } \mathcal{D}}$$

$$\vdots$$

$$\mathbb{P}(y = 9 | x = \text{  })$$

Why won't this work?

k-nearest neighbor classifier (knn)

$$\mathbb{P}(y = 0 | x = \text{  }) \approx \frac{\text{count of examples with } y = 0 \text{ out of } k \textit{ closest} \text{ to } x = \text{  }}{k}$$

$$\vdots$$

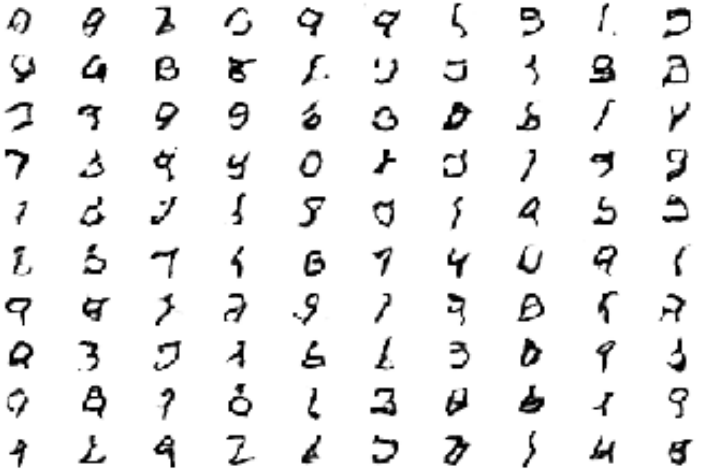
$$\mathbb{P}(y = 9 | x = \text{  })$$

$$f(x) = \arg \max_{i=0,\ldots,9} \mathbb{P}(y = i | x = \text{  })$$

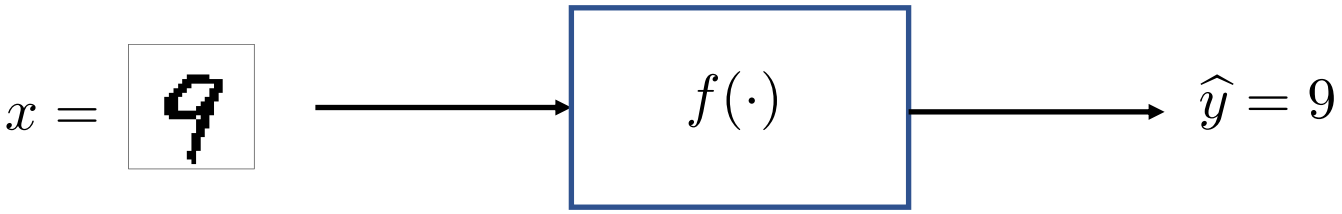
Discriminative and generative classifiers

$$f(x) = \arg\max_{i=0,\ldots,9} \mathbb{P}(y = i | x = \text{})$$

- learning $f(x)$ can require learning $\mathbb{P}(y|x)$
- learning $\mathbb{P}(y|x)$ can require learning $\mathbb{P}(x, y)$
  - *generative learning* requires learning $\mathbb{P}(x, y)$
  - *discriminative learning* does not
- if we know $\mathbb{P}(x, y)$, we can often sample from it



Loss and error



- how well does $f(\cdot)$ work?
- useful to define a *loss* function

$$\ell(\widehat{y}, y)$$

- misclassification loss (0/1 loss):

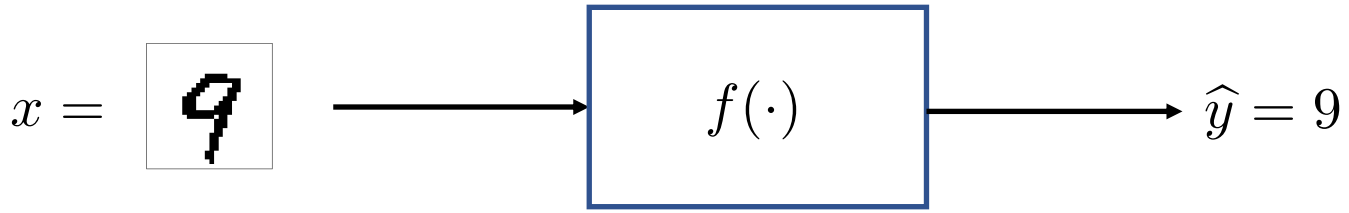
$$\ell(\widehat{y}, y) = \begin{cases} \widehat{y} \neq y & 1 \\ \text{else} & 0 \end{cases}$$

$$\ell(\widehat{y}, y) = \mathbb{1}_{\widehat{y} \neq y}$$

- indicator function:

$$\mathbb{1}\{\text{event}\} = \begin{cases} 1 & \text{if event happens} \\ 0 & \text{if not} \end{cases}$$

Other Loss functions

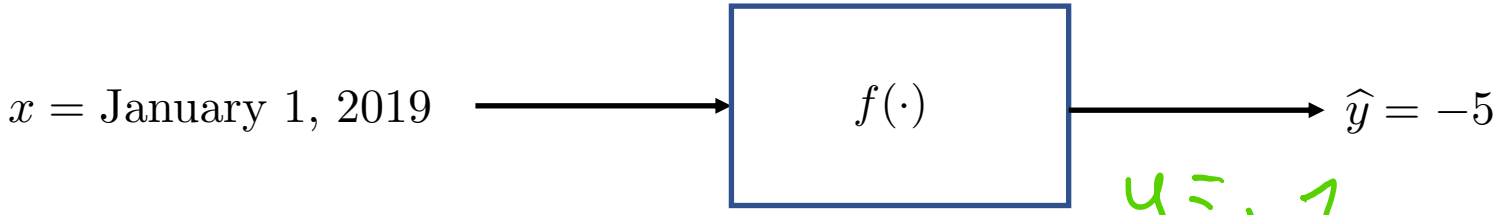


- other loss functions:
  - absolute error: $|\widehat{y} - y|$
  - squared error: $(\widehat{y} - y)^2$
  - hinge loss
  - logistic loss

SVM

logistic regression

Example



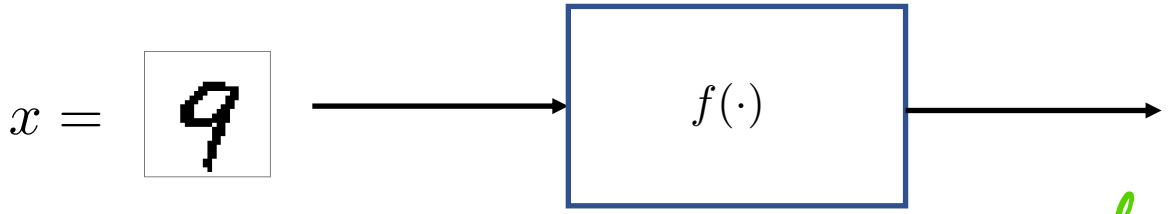
$y = +7$

- predicting high temperature in Madison (on Jan. 1st)

$$f(x) = 10 - 15\cos(\tfrac{2\pi x}{365}) = -5$$

$$\ell(\hat{y}, y) = |7 - -5| = 12$$

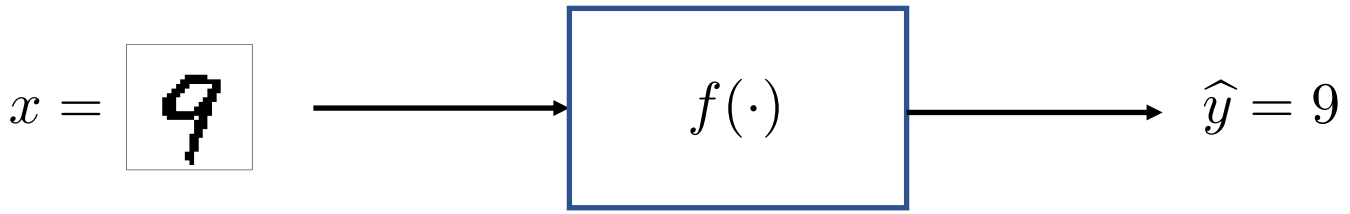
- predicting class of MNIST digits



misclassification loss

$$\ell(\hat{y}, y) = \mathbb{1}$$

Risk and Empirical Risk



- *risk* is the average *loss*
- *empirical risk* is the average *loss* on some data (a quantity that you measure)

$$\frac{1}{n}\sum_i \ell(\widehat{y}_i, y_i)$$

error rate

average error or *accuracy*

- *true risk* is the average *loss* if you keep measuring ...

$$\lim_{n\to\infty} \frac{1}{n}\sum_i \ell(\widehat{y}_i, y_i) \;= E\left[\ell(\widehat{y}_i, y_i)\right]$$

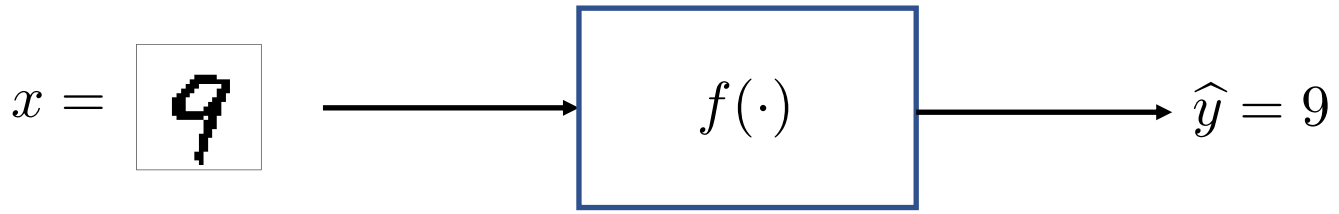
Training/Test split, cross validation

- you're tasked with building a classifier $f(\cdot)$
- you have a dataset: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$

$$\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{m} \qquad \mathcal{D}_{\text{test}} = \{(x_i, y_i)\}_{i=m+1}^{n}$$

- cross validation: 'try $f(\cdot)$ on something new to make sure it works'

Training Error, Test Error



- $f(\cdot)$ was created using training data: $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{m}$
- *training error* is the average *loss* on the training dataset

in-sample results that is error

$$\frac{1}{m}\sum_{i=1}^{m} \ell(\widehat{y}, y_i)$$

you can often get *training error* of zero!

- *test error* is the average *loss* on test data (i.e, not training data)

out of sample error

$$\frac{1}{(n-m)}\sum_{i=m+1}^{n} \ell(\widehat{y}, y_i)$$

Generalization Error and Overfitting

- *test error* is the average *loss* on test data
- *generalization error* is the average *loss* on test data if you keep getting more data

$$\lim_{n\to\infty} \frac{1}{(n-m)} \sum_{i=m+1}^{n} \ell(\widehat{y}, y_i)$$

- *overfitting* - driving the training error to zero, but increasing the generalization error

Bayes Error

- *Bayes error* is . . .

  the best error rate an *oracle* with knowledge of the data generating distribution could achieve

- Bayes error is the error rate of the MAP classifier with oracle knowledge of $\mathbb{P}(y|x)$

- example:

  $x = \{\text{cars.com}, \text{espn.com}, \text{hiking.com}\}$

  $\mathbb{P}(y = \text{female}|x)$

  $\mathbb{P}(y = \text{male}|x)$