Machine Learning

4771

Instructors:
Adrian Weller and Ilia Vovsha
Lecture 3+4: Parametric Approaches to Statistical Inference

- Bayesian Decision Theory (Duda 2.1-2.4)
- Gaussian Distribution (Duda 2.5)
- Classification with Gaussians (Duda 2.6)
- Regression
- Polynomial Approximation (Bishop 1.1)
- Cornerstones of Parametric Statistics
Regression
Function Approximation

- Start with training dataset

\[ \mathcal{X} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \quad x \in \mathbb{R}^D = \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(D) \end{bmatrix} \quad y \in \mathbb{R}^1 \]

- Have N (input, output) pairs
- Find a function \( f(x) \) to predict \( y \) from \( x \)
  - should fit the training data well

- Example: predict the price of house in dollars \( y \) using \( x = [\#\text{rooms}; \text{latitude}; \text{longitude}; \ldots] \)

- Need:
  a. Method (criterion) to evaluate how good a fit we have
  b. Class (set) of functions from which to select/search \( f(x) \)
Minimizing Training Error

• Idea: minimize ‘loss’ on the training data set
• Training = Empirical: use the training set to find the best fit
• Define a loss function for a single point (example):
  \[ L(y_i, f(x_i)) \]

• Average loss over the dataset (empirical risk):
  \[ R = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) \]

• Simplest loss: squared error from y value
  \[ L(y_i, f(x_i)) = \frac{1}{2} (y_i - f(x_i))^2 \]

• Other possible loss: absolute error
  \[ L(y_i, f(x_i)) = |y_i - f(x_i)| \]
Linear is simplest class of functions to search over:

\[ f(x; w) = w^T x + w_0 = \sum_{d=1}^{D} w_d x(d) + w_0 \]

Start with \( x \) being 1-dimensional (D=1):

\[ f(x; w) = w_1 x + w_0 \]

Plug in the above & minimize empirical risk over \( \omega \)

\[ R(w) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - w_1 x_i - w_0)^2 \]

Note: minimum occurs when \( R(w) \) gets flat (not always!)

Note: when \( R(w) \) is flat, gradient \( \nabla_w R = 0 \)
Minimize Risk \( (\nabla_w R = 0) \)

- Gradient set to 0
  - all partial derivatives set to 0

- Take partials of empirical risk:

\[
R(w) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - w_1 x_i - w_0)^2
\]

\[
\nabla_w R = \begin{bmatrix} \frac{\partial R}{\partial w_0} \\ \frac{\partial R}{\partial w_1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]
Minimize Risk \( (\nabla_w R = 0) \)

- Gradient set to 0
  - all partial derivatives set to 0

- Take partials of empirical risk:

\[
R(w) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - w_1 x_i - w_0)^2 \\
\frac{\partial R}{\partial w_0} = 0 \Rightarrow \frac{1}{N} \sum_{i=1}^{N} (y_i - w_1 x_i - w_0)(-1) = 0
\]
Minimize Risk \( (\nabla_w R = 0) \)

- Gradient set to 0
  - all partial derivatives set to 0

- Take partials of empirical risk:

\[
R(w) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - w_1 x_i - w_0)^2 \\
\frac{\partial R}{\partial w_0} = 0 \Rightarrow \frac{1}{N} \sum_{i=1}^{N} (y_i - w_1 x_i - w_0)(-1) = 0 \\
\frac{\partial R}{\partial w_1} = 0 \Rightarrow \frac{1}{N} \sum_{i=1}^{N} (y_i - w_1 x_i - w_0)(-x_i) = 0
\]
Minimize Risk ( $\nabla_w R = 0$ )

- Gradient set to 0
  - all partial derivatives set to 0

- Take partials of empirical risk:

$$R(w) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - w_1 x_i - w_0)^2$$

$$\frac{\partial R}{\partial w_0} = 0 \Rightarrow \frac{1}{N} \sum_{i=1}^{N} (y_i - w_1 x_i - w_0)(-1) = 0 \quad (1)$$

$$\frac{\partial R}{\partial w_1} = 0 \Rightarrow \frac{1}{N} \sum_{i=1}^{N} (y_i - w_1 x_i - w_0)(-x_i) = 0 \quad (2)$$

$$w_0 = \frac{1}{N} \sum y_i - w_1 \frac{1}{N} \sum x_i \quad (3)$$

$$w_1 \sum x_i^2 = \sum y_i x_i - w_0 \sum x_i \quad (4)$$
Minimize Risk \( (\nabla_w R = 0) \)

- Gradient set to 0
  - all partial derivatives set to 0

- Take partials of empirical risk:

\[
R(w) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - w_1 x_i - w_0)^2
\]

\[
\frac{\partial R}{\partial w_0} = 0 \Rightarrow \frac{1}{N} \sum_{i=1}^{N} (y_i - w_1 x_i - w_0)(-1) = 0
\] (1)

\[
\frac{\partial R}{\partial w_1} = 0 \Rightarrow \frac{1}{N} \sum_{i=1}^{N} (y_i - w_1 x_i - w_0)(-x_i) = 0
\] (2)

\[
w_0 = \frac{1}{N} \sum y_i - w_1 \frac{1}{N} \sum x_i
\] (3)

\[
w_1 \sum x_i^2 = \sum y_i x_i - w_0 \sum x_i
\] (4)

\[
w_1 \sum x_i^2 = \sum y_i x_i - \left( \frac{1}{N} \sum y_i - w_1 \frac{1}{N} \sum x_i \right) \sum x_i
\]

\[
w_1 (\sum x_i^2 - \frac{1}{N} \sum x_i \sum x_i) = \sum y_i x_i - \frac{1}{N} \sum y_i \sum x_i
\]

\[
w_1 = \frac{\sum y_i x_i - \frac{1}{N} \sum y_i \sum x_i}{\sum x_i^2 - \frac{1}{N} \sum x_i \sum x_i}
\] (5)
Properties of the Solution

• Setting \( w^* \) as before gives least squared error
• Define error on each data point as:
  \[
e_i = y_i - w_1^* x_i - w_0^*
  \]

• Note property #1:
  \[
  \frac{\partial R}{\partial w_0} = \frac{1}{N} \sum_{i=1}^{N} (y_i - w_1 x_i - w_0) = 0
  \]
  ...average error is zero
  \[
  \frac{1}{N} \sum e_i = 0
  \]

• Note property #2:
  \[
  \frac{\partial R}{\partial w_1} = \frac{1}{N} \sum_{i=1}^{N} (y_i - w_1 x_i - w_0) x_i = 0
  \]
  ...error not correlated with data
  \[
  \frac{1}{N} \sum e_i x_i = \frac{1}{N} e^T x = 0
  \]
Multi-Dimensional Regression

- More elegant/general to do \( \nabla_w R = 0 \) with linear algebra
- Rewrite empirical risk in vector-matrix notation:
- Can add more dimensions by adding columns to \( X \) matrix and rows to \( w \) vector:

\[
R(w) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - w_1 x_i - w_0)^2 \\
= \frac{1}{2N} \sum_{i=1}^{N} (y_i - [1, x_i] \begin{bmatrix} w_0 \\ w_1 \end{bmatrix})^2 \\
= \frac{1}{2N} \sum_{i=1}^{N} (y_i - [1, x_i(1) \ldots x_i(d)] \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix})^2 \\
= \frac{1}{2N} \|y - Xw\|^2
\]
Multi-Dimensional Regression

- More realistic dataset: many measurements
- Have N apartments each with D measurements
- Each row of $X$ is [\#rooms; latitude; longitude,...]

$$X = \begin{bmatrix}
1 & x_1(1) & \cdots & x_1(D) \\
\vdots & \vdots & \ddots & \vdots \\
1 & x_N(1) & \cdots & x_N(D)
\end{bmatrix}$$
2D Linear Regression

- Once best $w^*$ is found, we can plug it into the function:

$$f(x; w^*) = w_2^* x(2) + w_1^* x(1) + w_0^*$$
Polynomial Function Classes

• Back to 1-dim x (D=1) BUT Nonlinear Function Classes

• Polynomial: \( f(x; w) = \sum_{p=1}^{P} w_p x^p + w_0 \)

• Writing Risk: \( R(w) = \frac{1}{2N} \| y - Xw \|^2 \), \( X = \begin{bmatrix} 1 & x_1^1 & \cdots & x_1^P \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_N^1 & \cdots & x_N^P \end{bmatrix} \)

• Order-P polynomial regression fitting for 1D variable is the same as P-dimensional linear regression!

• Construct a multidim x-vector from x scalar: \( x_i = [x_i^0, x_i^1, x_i^2]^T \)

• More generally any function: \( x_i = [\phi_0(x_i), \phi_1(x_i), \phi_2(x_i)]^T \)
Sinusoidal Basis Functions

- More generally, we don’t just have to deal with polynomials, use any set of basis fn’s:
  \[ f(x; \theta) = \sum_{p=1}^{P} \theta_p \phi_p(x) + \theta_0 \]
- These are generally called **Additive Models**
- Regression adds linear combinations of the basis fn’s

- For example: Fourier (sinusoidal) basis
  \[ \phi_{2k}(x_i) = \sin(kx_i) \quad \phi_{2k+1}(x_i) = \cos(kx_i) \]
- Note, don’t have to be a basis per se, usually subset

\[ \theta_1 \times \quad + \quad \theta_2 \times \quad + \quad \theta_3 \times \]
**Radial Basis Functions**

- Can act as prototypes of the data itself
  \[ f(x; \theta) = \sum_{k=1}^{N} \theta_k \exp\left(-\frac{1}{2\sigma^2} \| x - x_k \|^2 \right) + \theta_0 \]

- Parameter \( \sigma = \text{standard deviation} \)
  \( \sigma^2 = \text{covariance} \)

  controls how wide bumps are
  what happens if too big/small?

- Also works in multi-dimensions
Underfitting/Overfitting

- Try varying $P$. Higher $P$ fits a more complex function class.
- Observe $R(w^*)$ drops with bigger $P$.
Evaluating the Model

- Unfair to use training error to find best order $P$
- High $P$ (vs. $N$) can overfit, even linear case!
- $\min R(w^*)$ not on training but on future data
- Want model to Generalize to future data

Expected (true) loss: $R_{\text{expected}}(w) = \int L(y, f(x; w)) P(x, y) dx dy$

- One approach: split data into training / testing portion

\[
\{(x_1, y_1), \ldots, (x_v, y_v)\} \quad \{ (x_{v+1}, y_{v+1}), \ldots, (x_N, y_N) \} 
\]

- Estimate $\omega^*$ with training (empirical) loss:

\[
R_{\text{train}}(w) = \frac{1}{2v} \sum_{i=1}^{v} (y_i - w^T x_i)^2
\]

- Evaluate $P$ with testing loss:

\[
R_{\text{test}}(w) = \frac{1}{2(N-v)} \sum_{i=v+1}^{N} (y_i - w^T x_i)^2
\]
Validation

- Try fitting with different polynomial order $P$
- Select $P$ which gives lowest $R_{test}(w^*)$

Think of $P$ as a measure of the complexity of the model
Higher order polynomials are more flexible and complex
Cross-Validation

• Better idea: split data into three sets (training / validation / test)
• Even better idea: split data into two sets (training / test) but do \textit{K-fold cross-validation} on the training set
  ➢ $K$ folds, $K-1$ for training, $1$ for testing; repeat process $K$ times; average error
• Best idea (sometimes): \textit{leave-one-out cross-validation} on the training set
  ➢ $N$ examples, $N-1$ for training $1$ for testing; repeat process $N$ times; average error
The Weierstrass Approximation Theorem

• Theorem (1885):

Suppose $f(x)$ is a continuous real-valued function defined on the real interval $[a, b]$. For every $\epsilon > 0$, there exists a polynomial function $p$ over $\mathbb{R}$ such that $\forall x \in [a, b]$, we have $|f(x) - p(x)| < \epsilon$, or equivalently, $\|f(x) - p(x)\|_\infty < \epsilon$.

• Definition of supremum (infinity) norm:

$$\|f(x) - p(x)\|_\infty = \sup\{|f(x) - p(x)|\}$$