Introduction to Machine Learning
236756
Prof. Nir Ailon

Lecture 4:
K-Nearest Neighbor
The Nearest Neighbor Classifier

- Training sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
- Want to predict label of new point $x$
- The Nearest Neighbor Rule:
  - Find the closest training point: $i = \arg\min_i \rho(x, x_i)$
  - Predict label of $x$ as $y_i$
- As learning rule: $NN(S) = h$ where $h(x) = y_{\arg\min_i \rho(x, x_i)}$
- Unlike ERM, not a “training procedure” that outputs $h \in \mathcal{H}$
  - At training time: do nothing
  - Given a point $x$ to classify: search the training set for a NN
Where is the Bias Hiding?

- Find the closest training point: \( i = \arg \min_i \rho(x, x_i) \)
- Predict label of \( x \) as \( y_i \)

- What is the right “distance” between images? Between sound waves? Between sentences?
- Option 1: \( \rho(x, x') = \| \phi(x) - \phi(x') \|_2 \)
  - What representation \( \phi(x) \)?
  - Maybe a different distance? \( \| \phi(x) - \phi(x') \|_1 \)? \( \| \phi(x) - \phi(x) \|_\infty \)? \( \sin(\angle(\phi(x), \phi(x'))) \)? \( KL(\phi(x) \| \phi(x')) \)?

- Option 2: Special-purpose distance measure on \( x \)
  - E.g. edit distance, deformation measure, etc
Nearest Neighbor Learning Guarantee

- Optimal predictor: \( h^* = \arg \min L_D(h) \)

\[ h^*(x) = \begin{cases} +1, & \eta(x) > 0.5 \\ -1, & \eta(x) < 0.5 \end{cases} \]
\( \eta(x) = P_D(y = 1 | x) \)

- For the NN rule with \( \rho(x, x') = \| \phi(x) - \phi(x') \|_2 \), and \( \phi: \mathcal{X} \to [0,1]^d \):

\[
\mathbb{E}_{S \sim \mathcal{D}^m} \left[ L(\text{NN}(S)) \right] \leq 2L(h^*) + 4c_D \frac{\sqrt{d}}{d+1} \frac{1}{\sqrt{m}}
\]

\[ |\eta(x) - \eta(x')| \leq c_D \cdot \rho(x, x') \]
Data Fit / Complexity Tradeoff

\[ \mathbb{E}_{S \sim D^m}[L(NN(S))] \leq 2L(h^*) + 4c_D \frac{\sqrt{d}}{d^{1/2} \sqrt{m}} \]

- \(k\)-Nearest Neighbor: predict according to majority among \(k\) closest point from \(S\).
$k$-Nearest Neighbor: Data Fit / Complexity Tradeoff

$S = \quad h^* =$

$k = 1 \quad k = 5 \quad k = 12$  
$k = 50 \quad k = 100 \quad k = 200$
**k-Nearest Neighbor Guarantee**

- For $k$-NN with $\rho(x, x') = \|\phi(x) - \phi(x')\|_2$, and $\phi: \mathcal{X} \to [0,1]^d$:
  
  $|\eta(x) - \eta(x')| \leq c_D \cdot \rho(x, x')$

  
  $\mathbb{E}_{S \sim \mathcal{D}^m} \left[ L(\text{NN}_k(S)) \right] \leq \left(1 + \sqrt{\frac{8}{k}}\right) L(h^*) + \frac{6c_D \sqrt{d + k}}{\sqrt{m}}$

- Should increase $k$ with sample size $m$
  
  - Above theory suggests $k_m \propto L(h^*)^{2/3} \cdot m^{2/(d+1)}$

- “Universal” Learning: for any “smooth” $\mathcal{D}$ and representation $\phi(\cdot)$ (with continuous $P(y|\phi(x))$), if we increase $k$ slowly enough, we will eventually converge to optimal $L(h^*)$

- Very non-uniform: sample complexity depends not only on $h^*$, but also on $\mathcal{D}$
**k-Nearest Neighbor: Issues**

- **Curse of Dimensionality**
  - Analysis uses sample complexity of at least \( m = \Omega \left( \left( \frac{c_d}{\epsilon} \right)^{d+1} \right) \)
  - Exponential dependence on \( d + 1 \) is not artifact of analysis! Can construct examples where this is required.

- **Need to store entire training set**
  - “replace intelligence with fast hard drives [or lots of RAM]”

- **Computational problem of finding nearest neighbors**
  - \( O(md) \) for each query
  - In low dimensions: efficient geometric structures to allow quickly finding nearest neighbors
  - Alternative: fast algorithms for finding approximate nearest neighbors (often good enough) based on *locality sensitive hashing*

- **Advantages:**
  - Simple (naïve implementation in one line of Python!), intuitive
  - Easily generalized to multi-class (predict most common class among \( k \) neighbors) and real-valued \( y \) (predict average among neighbors)