Nearest Neighbor

Machine Learning – CSE546
Kevin Jamieson
University of Washington

October 26, 2017
Some data, Bayes Classifier

Training data:
- True label: +1
- True label: -1

Optimal “Bayes” classifier:
\[ P(Y = 1|X = x) = \frac{1}{2} \]

Predicted label: +1
Predicted label: -1

Figures stolen from Hastie et al
Linear Decision Boundary

Training data:
- True label: +1
- True label: -1

Learned:
- Linear Decision boundary
  \[ x^T w + b = 0 \]
- Predicted label: +1
- Predicted label: -1

Figures stolen from Hastie et al
15 Nearest Neighbor Boundary

Training data:
- True label: +1
- True label: -1

Learned:
- 15 nearest neighbor decision boundary (majority vote)

Predicted label: +1
Predicted label: -1
1 Nearest Neighbor Boundary

Training data:
- True label: +1
- True label: -1

Learned:
- 1 nearest neighbor decision boundary (majority vote)
  - Predicted label: +1
  - Predicted label: -1
k-Nearest Neighbor Error

Bias-Variance tradeoff

As $k \rightarrow \infty$?
- Bias: $\infty$
- Variance: $0$

As $k \rightarrow 1$?
- Bias: small
- Variance: large
Notable distance metrics (and their level sets)

$L_2$ norm

$$d(x, y) = \|x - y\|_2$$

\{y : d(x, y) = c\}

$L_1$ norm (taxi-cab)

$$d(x, y) = \max \{ |x_1 - y_1|, |x_2 - y_2|, \ldots, |x_n - y_n| \}$$

Mahalanobis

(here, $\Sigma$ on the previous slide is not necessarily diagonal, but is symmetric)

$L_\infty$ (max) norm
1 nearest neighbor

One can draw the nearest-neighbor regions in input space.

\[ \text{Dist}(x^i, x^j) = (x^i_1 - x^j_1)^2 + (x^i_2 - x^j_2)^2 \]

\[ \text{Dist}(x^i, x^j) = (x^i_1 - x^j_1)^2 + (3x^i_2 - 3x^j_2)^2 \]

The relative scalings in the distance metric affect region shapes.
1 nearest neighbor guarantee

\[ \{(x_i, y_i)\}_{i=1}^{n} \quad x_i \in \mathbb{R}^d, \ y_i \in \{1, \ldots, k\} \]

As \( n \to \infty \), assume the \( x_i \)'s become dense in \( \mathbb{R}^d \)

Note: any \( x_a \in \mathbb{R}^d \) has the same label distribution as \( x_b \) with \( b = 1NN(a) \)

\[
(x_0, y_i) \overset{i.i.d.}{\sim} P_{x,y} \quad P(y = y_i \mid x = x)
\]

[Cover, Hart, 1967]
1 nearest neighbor guarantee

\[ \{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, y_i \in \{1, \ldots, k\} \]

As \( n \to \infty \), assume the \( x_i \)'s become dense in \( \mathbb{R}^d \)

Note: any \( x_a \in \mathbb{R}^d \) has the same label distribution as \( x_b \) with \( b = \text{1NN}(a) \)

If \( p_\ell = \mathbb{P}(Y_a = \ell) = \mathbb{P}(Y_b = \ell) \) and \( \ell^* = \arg \max_{\ell=1,\ldots,k} p_\ell \) then

Bates error = \( 1 - p_{\ell^*} \)
1 nearest neighbor guarantee

\[ \{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, y_i \in \{1, \ldots, k\} \]

As \( n \to \infty \), assume the \( x_i \)'s become dense in \( \mathbb{R}^d \)

Note: any \( x_a \in \mathbb{R}^d \) has the same label distribution as \( x_b \) with \( b = 1NN(a) \)

If \( p_\ell = \mathbb{P}(Y_a = \ell) = \mathbb{P}(Y_b = \ell) \) and \( \ell^* = \arg \max_{\ell=1,\ldots,k} p_\ell \) then

Bates error = 1 − \( p_{\ell^*} \)

1-nearest neighbor error = \( \mathbb{P}(Y_a \neq Y_b) = \sum_{\ell=1}^k \mathbb{P}(Y_a = \ell, Y_b \neq \ell) \)

\[ = \sum_{\ell} \mathbb{P}(Y_a = \ell) \mathbb{P}(Y_b \neq \ell) \]
\[ = \sum_{\ell} P_\ell (1 - P_\ell) \frac{k-1}{2} P_\ast (1 - P_{\ast\ell}) \]
1 nearest neighbor guarantee

\[ \{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, y_i \in \{1, \ldots, k\} \]

As \( n \to \infty \), assume the \( x_i \)'s become dense in \( \mathbb{R}^d \)

Note: any \( x_a \in \mathbb{R}^d \) has the same label distribution as \( x_b \) with \( b = 1NN(a) \)

If \( p_\ell = \mathbb{P}(Y_a = \ell) = \mathbb{P}(Y_b = \ell) \) and \( \ell^* = \arg \max_{\ell=1,\ldots,k} p_\ell \) then

\[ \text{Bates error} = 1 - p_{\ell^*} \]

1-nearest neighbor error = \( \mathbb{P}(Y_a \neq Y_b) = \sum_{\ell=1}^{k} \mathbb{P}(Y_a = \ell, Y_b \neq \ell) \)
\[ = \sum_{\ell=1}^{k} p_\ell (1 - p_\ell) \leq 2(1 - p_{\ell^*}) - \frac{k}{k-1} (1 - p_{\ell^*})^2 \]

As \( n \to \infty \), then 1-NN rule error is at most twice the Bayes error!

[Cover, Hart, 1967]
Curse of dimensionality Ex. 1

side length $r$

$X$ is uniformly distributed over $[0, 1]^p$. What is $\mathbb{P}(X \in [0, r]^p)$?
Curse of dimensionality Ex. 2

\( \{X_i\}_{i=1}^n \) are uniformly distributed over \([-0.5, 0.5]^p\).

What is the median distance from a point at origin to its 1NN?
Nearest neighbor regression

\[ \{(x_i, y_i)\}_{i=1}^{n} \]

\[ \mathcal{N}_k(x_0) = k\text{-nearest neighbors of } x_0 \]

\[ \hat{f}(x_0) = \sum_{x_i \in \mathcal{N}_k(x_0)} \frac{1}{k} y_i \]
Nearest neighbor regression

\[
\{(x_i, y_i)\}_{i=1}^n
\]

\[N_k(x_0) = k\text{-nearest neighbors of } x_0\]

\[\hat{f}(x_0) = \sum_{x_i \in N_k(x_0)} \frac{1}{k} y_i\]

Why are far-away neighbors weighted same as close neighbors!

Kernel smoothing: \[K(x, y)\]

\[\hat{f}(x_0) = \frac{\sum_{i=1}^{n} K(x_0, x_i) y_i}{\sum_{i=1}^{n} K(x_0, x_i)}\]
Nearest neighbor regression

\[ \left\{ (x_i, y_i) \right\}_{i=1}^n \]

\[ \mathcal{N}_k(x_0) = \text{k-nearest neighbors of } x_0 \]

\[ \hat{f}(x_0) = \sum_{x_i \in \mathcal{N}_k(x_0)} \frac{1}{k} y_i \]

\[ \hat{f}(x_0) = \frac{\sum_{i=1}^n K(x_0, x_i) y_i}{\sum_{i=1}^n K(x_0, x_i)} \]
Nearest neighbor regression

\[ \{(x_i, y_i)\}_{i=1}^n \]

\[ N_k(x_0) = k\text{-nearest neighbors of } x_0 \]

\[ \hat{f}(x_0) = \sum_{x_i \in N_k(x_0)} \frac{1}{k} y_i \]

\[ \hat{f}(x_0) = \frac{\sum_{i=1}^n K(x_0, x_i) y_i}{\sum_{i=1}^n K(x_0, x_i)} \]

Why just average them?
Nearest neighbor regression

\[ \left\{ (x_i, y_i) \right\}_{i=1}^n \]

\[ N_k(x_0) = \text{k-nearest neighbors of } x_0 \]

\[ \hat{f}(x_0) = \sum_{x_i \in N_k(x_0)} \frac{1}{k} y_i \]

\[ \hat{f}(x_0) = \frac{\sum_{i=1}^n K(x_0, x_i)y_i}{\sum_{i=1}^n K(x_0, x_i)} \]

\[ \hat{f}(x_0) = b(x_0) + w(x_0)^T x_0 \]

\[ w(x_0), b(x_0) = \arg \min_{w, b} \sum_{i=1}^n K(x_0, x_i)(y_i - (b + w^T x_i))^2 \]

Local Linear Regression
Nearest Neighbor Overview

- Very simple to explain and implement
- No training! But finding nearest neighbors in large dataset at test can be computationally demanding (kD-trees help)
Nearest Neighbor Overview

- Very simple to explain and implement
- No training! But finding nearest neighbors in large dataset at test can be computationally demanding (kD-trees help)
- You can use other forms of distance (not just Euclidean)
- Smoothing with Kernels and local linear regression can improve performance (at the cost of higher variance)
Nearest Neighbor Overview

- Very simple to explain and implement
- No training! But finding nearest neighbors in large dataset at test can be computationally demanding (kD-trees help)
- You can use other forms of distance (not just Euclidean)
- Smoothing with Kernels and local linear regression can improve performance (at the cost of higher variance)
- With a lot of data, “local methods” have strong, simple theoretical guarantees. With not a lot of data, neighborhoods aren’t “local” and methods suffer.
Kernels

Machine Learning – CSE546
Kevin Jamieson
University of Washington

October 26, 2017
Machine Learning Problems

- Have a bunch of iid data of the form:
  \[ \{(x_i, y_i)\}_{i=1}^{n} \quad x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R} \]

- Learning a model’s parameters:
  Each \( \ell_i(w) \) is convex.

  \[
  \sum_{i=1}^{n} \ell_i(w)
  \]

  Hinge Loss: \( \ell_i(w) = \max\{0, 1 - y_i x_i^T w\} \)

  Logistic Loss: \( \ell_i(w) = \log(1 + \exp(-y_i x_i^T w)) \)

  Squared error Loss: \( \ell_i(w) = (y_i - x_i^T w)^2 \)

All in terms of inner products! Even nearest neighbor can use inner products!

\[
\|x-y\|_2^2 = x^T x - 2x^T y + y^T y
\]
What if the data is not linearly separable?

Use features of features of features of features....

$\phi(x) : \mathbb{R}^d \rightarrow \mathbb{R}^p$

Feature space can get really large really quickly!
Dot-product of polynomials

\[ \Phi(u) \cdot \Phi(v) = \text{polynomials of degree exactly } d \]

\[ d = 1 : \phi(u) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \langle \phi(u), \phi(v) \rangle = u_1 v_1 + u_2 v_2 \]
Dot-product of polynomials

\( \Phi(u) \cdot \Phi(v) = \text{polynomials of degree exactly } d \)

\[ d = 1 : \phi(u) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \langle \phi(u), \phi(v) \rangle = u_1 v_1 + u_2 v_2 \]

\[ d = 2 : \phi(u) = \begin{bmatrix} u_1^2 \\ u_2^2 \\ u_1 u_2 \\ u_2 u_1 \end{bmatrix} \quad \langle \phi(u), \phi(v) \rangle = u_1^2 v_1^2 + u_2^2 v_2^2 + 2u_1 u_2 v_1 v_2 = \left( \langle u, v \rangle \right)^2 \]
Dot-product of polynomials

\[ \Phi(u) \cdot \Phi(v) = \text{polynomials of degree exactly } d \]

\[ d = 1 : \phi(u) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \langle \phi(u), \phi(v) \rangle = u_1 v_1 + u_2 v_2 \]

\[ d = 2 : \phi(u) = \begin{bmatrix} u_1^2 \\ u_2^2 \\ u_1 u_2 \\ u_2 u_1 \end{bmatrix} \quad \langle \phi(u), \phi(v) \rangle = u_1^2 v_1^2 + u_2^2 v_2^2 + 2u_1 u_2 v_1 v_2 \]

General \( d \):

Dimension of \( \phi(u) \) is roughly \( p^d \) if \( u \in \mathbb{R}^p \)
Observation

There exists an \( \alpha \in \mathbb{R}^n \): 

\[
\hat{w} = \arg \min_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 + \lambda \|w\|^2
\]

Why?

\( \hat{w} \) is the span of \( (x_1, \ldots, x_n) \) if \( \exists \alpha \): 

\[
\hat{w} = \sum_{i=1}^{n} \alpha_i x_i
\]

\( \hat{w} = \hat{w}_{\text{in}} + \hat{w}_{\text{out}}, \quad \hat{w}_{\text{out}}^T (\sum \alpha_i x_i) = 0 \)

\[
\arg \min_{\alpha} \|XX^T \alpha - y\|_2^2 + \lambda \alpha^T XX^T \alpha
\]

\[
K = XX^T \in \mathbb{R}^{n \times n}
\]

\[
\nabla_{\alpha} = \|K \alpha - y\|_2^2 + \lambda \alpha^T K \alpha \rightarrow 2K K \alpha - 2K y + 2\lambda K \alpha = 0
\]

\[
K \begin{bmatrix} (K + \lambda I) \alpha - y \end{bmatrix} = 0 \quad \alpha = (K + \lambda I)^{-1} y
\]
Observation

\[ K = X X^T \in \mathbb{R}^{n \times n} \]

\[
\text{arg min}_{\alpha} \| K\alpha - y \|_2^2 + \lambda \alpha^T K\alpha
\]

\[
K_{i,j} = \langle x_i, x_j \rangle
\]

\[
K_{i,j} = \langle \phi(x_i), \phi(x_j) \rangle = K(x_i, x_j)
\]

\[
f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x) \equiv \sum_{j=1}^{d} w_j \phi(x)
\]

\[
W = \sum_{i=1}^{n} \phi(x_i) \alpha_i
\]
Common kernels

- Polynomials of degree exactly $d$
  \[ K(u, v) = (u \cdot v)^d \]
- Polynomials of degree up to $d$
  \[ K(u, v) = (u \cdot v + 1)^d \]
- Gaussian (squared exponential) kernel
  \[ K(u, v) = \exp \left( -\frac{||u - v||^2}{2\sigma^2} \right) \]
- Sigmoid
  \[ K(u, v) = \tanh(\eta u \cdot v + \nu) \]
Mercer’s Theorem

- When do we have a valid Kernel $K(x,x')$?
- Definition 1: when it is an inner product
  
  $\exists \phi: \mathbb{R}^d \rightarrow \mathcal{H} \quad K(z,z') = \langle \phi(z), \phi(z') \rangle$

- Mercer’s Theorem:
  
  - $K(x,x')$ is a valid kernel if and only if $K$ is a positive semi-definite.
  
  - PSD in the following sense:

$$\forall f \quad \iint f(x) K(x,y) f(y) \, dx \, dy \geq 0$$

$x \in \mathbb{R}^d$, $K_{\mathbb{R}^d}$ is PSD iff $x^T K x \geq 0 \ \forall x$
RBF Kernel

\[ K(u, v) = \exp \left( -\frac{||u - v||^2}{2\sigma^2} \right) \]

- Note that this is like weighting “bumps” on each point like kernel smoothing but now we learn the weights.

- Is there an inner product representation of \( K(x, y) \)?
Classification

\[
\hat{w} = \sum_{i=1}^{n} \max\{0, 1 - y_i(b + x_i^T w)\} + \lambda \|w\|^2_2
\]

\[
\min_{\alpha, b} \sum_{i=1}^{n} \max\{0, 1 - y_i(b + \sum_{j=1}^{n} \alpha_j \langle x_i, x_j \rangle)\} + \lambda \sum_{i,j=1}^{n} \alpha_i \alpha_j \langle x_i, x_j \rangle
\]
RBF kernel Secretly random features

\[ 2 \cos(\alpha) \cos(\beta) = \cos(\alpha + \beta) + \cos(\alpha - \beta) \]

\[ b \sim \text{uniform}(0, \pi) \quad w \sim \mathcal{N}(0, 2\gamma) \]

\[ \phi(x) = \sqrt{2} \cos(w^T x + b) \]

\[ \mathbb{E}_{w,b}[\phi(x)^T \phi(y)] = \]
RBF kernel Secretly random features

\[ 2 \cos(\alpha) \cos(\beta) = \cos(\alpha + \beta) + \cos(\alpha - \beta) \]

\[ b \sim \text{uniform}(0, \pi) \quad w \sim \mathcal{N}(0, 2\gamma) \]

\[ \phi(x) = \sqrt{2} \cos(w^T x + b) \]

\[ \mathbb{E}_{w,b}[\phi(x)^T \phi(y)] = e^{-\gamma \|x-y\|_2^2} \]

[ Rahimi, Recht 2007 ]

Hint: use Euler’s formula \( e^{jz} = \cos(z) + j \sin(z) \)
Wait, infinite dimensions?

- Isn’t everything separable there? How are we not overfitting?

- Regularization! Fat shattering \((R/margin)^2\)

- What about sparsity?
String Kernels

Example from Efron and Hastie, 2016

Amino acid sequences of different lengths:

```
x1
IPTSALVKETLALLSTHRTLIANETLRLIPVPHKHNHLCTEEOIFQIGTILESQTVQGTVE
ERLFKNLSSLKGYIDQKCLKCGEERRRQFLDILQEEFLGVMNTEWI

x2
PHRRDLCRSISWLARKIRSDLALTESYVKHQLWSELTEAERLQENLQAYRTFHVLLA
RRLEDQQVHFPTPGDFHQAIHTLLQVAAFAYQIEELMILEYKIPRNEADGMLFEKK
LWGLKVQLQELSQWTVSRIHDLRFISSHQTGIP
```

All subsequences of length 3 (of possible 20 amino acids) $20^3 = 8,000$

$h_{LQE}^3(x_1) = 1$ and $h_{LQE}^3(x_2) = 2$. 
Least squares, tradeoffs