Adversarial Examples for k-Nearest Neighbor Classifiers Based on Higher-Order Voronoi Diagrams

Chawin Sitawarin[†] Evgenios M. Kornaropoulos[*] Dawn Song[†]
David Wagner[†]

[†]UC Berkeley
[*]George Mason University

NeurIPS 2021

NEURAL INFORMATION PROCESSING SYSTEMS

Contact: chawins@berkeley.edu

Problem

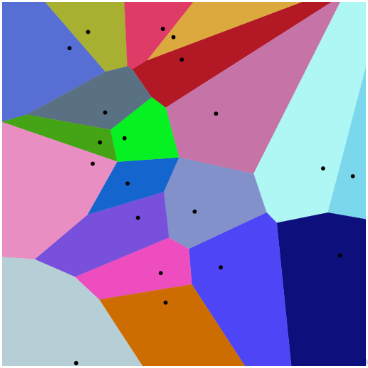
How do we evaluate the robustness of k-NN classifiers under malicious manipulation, e.g., adversarial examples?

Motivation

- Attacks on machine learning models are becoming real concerns.
- Fast and reliable evaluation is the first step.
- k-NN models are widely used in the industry.
- k-NN is conceptually simple and has many nice geometric properties that we can exploit.

Setup: Voronoi Diagram

- k-NN classifiers partition input space ($\mathbb{R}^d$) into a set of polytopes, called a Voronoi diagram.
- Each polytope corresponds to k generators (or training points): $L_i^{(k)} = \{x_{i1}, \dots, x_{ik}\}$.
- For a set of generators X where $|X| = n$, there are at most $\binom{n}{k}$ polytopes/cells.



Setup: Voronoi Diagram

- A polytope corresponding to $L_i^{(k)}$ is called an **order-k Voronoi Cell**.

Setup: Voronoi Diagram

- A polytope corresponding to $L_i^{(k)}$ is called an **order-k Voronoi Cell**.
- We can also describe $V(L_i^{(k)})$ by an **intersection of halfspaces**, $H(a,b) = \{p \mid d(p,a) \leq d(p,b)\}$,

$$V(L_i^{(k)}) = \bigcap_{a \in L_i^{(k)}} \bigcap_{b \in X \setminus L_i^{(k)}} H(a,b)$$

Setup: Voronoi Diagram

- A polytope corresponding to $L_i^{(k)}$ is called an **order-k Voronoi Cell**.
- We can also describe $V(L_i^{(k)})$ by an **intersection of halfspaces**, $H(a,b) = \{p \mid d(p,a) \leq d(p,b)\}$,

$$V(L_i^{(k)}) = \bigcap_{a \in L_i^{(k)}} \bigcap_{b \in X \setminus L_i^{(k)}} H(a,b)$$

- We limit $d(\cdot,\cdot)$ to Euclidean distance.

Setup: Voronoi Diagram

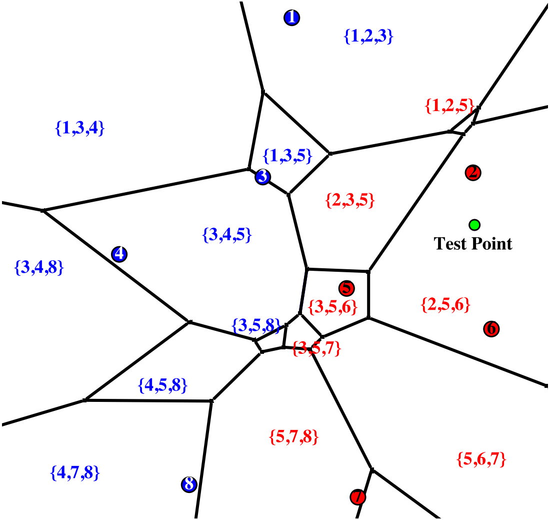
- A polytope corresponding to $L_i^{(k)}$ is called an **order-k Voronoi Cell**.
- We can also describe $V(L_i^{(k)})$ by an **intersection of halfspaces**, $H(a,b) = \{p \mid d(p,a) \le d(p,b)\}$,

$$V(L_i^{(k)}) = \bigcap_{a \in L_i^{(k)}} \bigcap_{b \in X \setminus L_i^{(k)}} H(a,b)$$

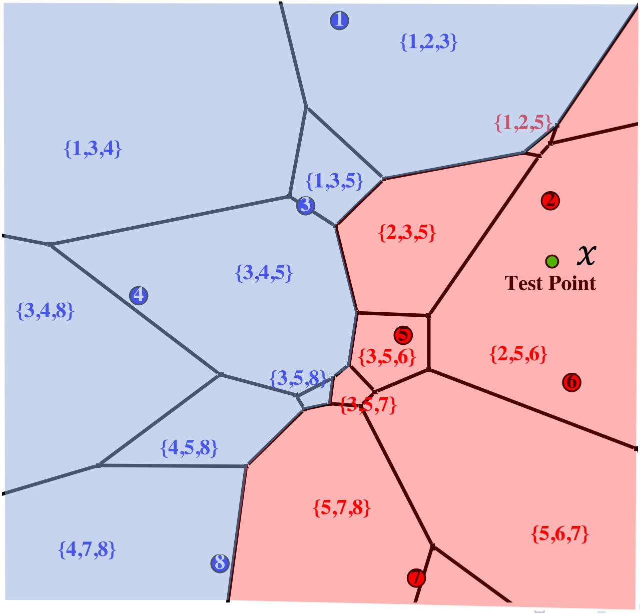
- We limit $d(\cdot,\cdot)$ to Euclidean distance.
- Classification of a test point x:
  - Find $L_i^{(k)}$ such that $x \in V(L_i^{(k)})$.
  - Use majority vote of $\{y_{i1}, \dots, y_{ik}\}$.

Setup: Voronoi Diagram

Order-3 Voronoi diagram with binary labels (red and blue).



Setup: Voronoi Diagram



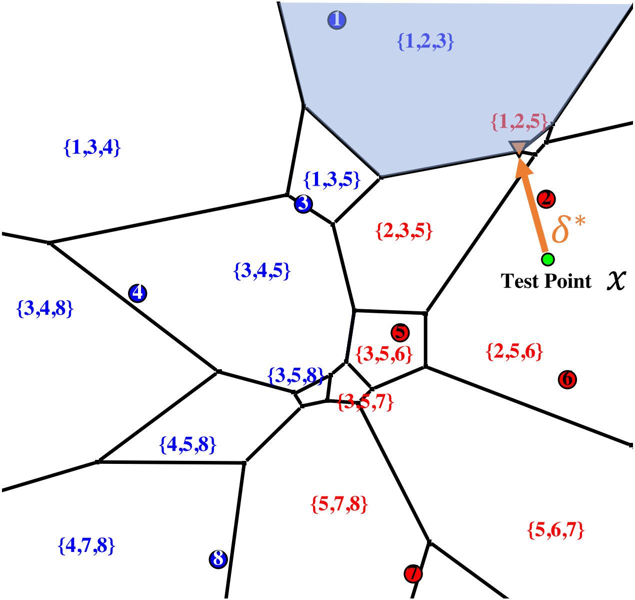
Setup: Adversarial Examples

- Find the smallest perturbation δ^* that moves a test point (x, y) to an **adversarial cell** $A(x)$, i.e. any cell with different class from y.

$$\begin{aligned} \delta^* \;=\; \arg\min_{\delta} \quad & \|\delta\|_2^2 \\ \text{s.t.} \quad & x + \delta \in A(x) \end{aligned} \tag{1}$$

- Call $\epsilon^* := \|\delta^*\|_2$ **optimal adversarial distance**.

Setup: Adversarial Examples



First Attempt

- Brute-force / naive algorithm: Just search every cell!

First Attempt

- Brute-force / naive algorithm: Just search every cell!
- Solving one quadratic program (QP) per adversarial cell.

$$\begin{aligned} \delta^* \quad = \quad \arg\min_{\delta} \quad & \|\delta\|_2^2 \\ \text{s.t.} \quad & x + \delta \in V(L_i^{(k)}) \end{aligned}$$

for $i \in \{1, \dots, t\}$ where $t \in \mathcal{O}(\binom{n}{k})$ is the number of adversarial cells.

First Attempt

- Brute-force / naive algorithm: Just search every cell!
- Solving one quadratic program (QP) per adversarial cell.

$$\begin{aligned} \delta^* \;=\; & \arg\min_{\delta} \quad \|\delta\|_2^2 \\ & \quad \text{s.t.} \quad x + \delta \in V(L_i^{(k)}) \end{aligned}$$

  for $i \in \{1, \ldots, t\}$ where $t \in \mathcal{O}(\binom{n}{k})$ is the number of adversarial cells.
- This is not scalable for $k > 1$ or for large n: Solving t QP's each with $k(n-k)$ constraints is $\mathcal{O}(\binom{n}{k} \cdot poly(n, k, d))$.

GeoAdEx

- We don't need to search every cell, only ones close to x.

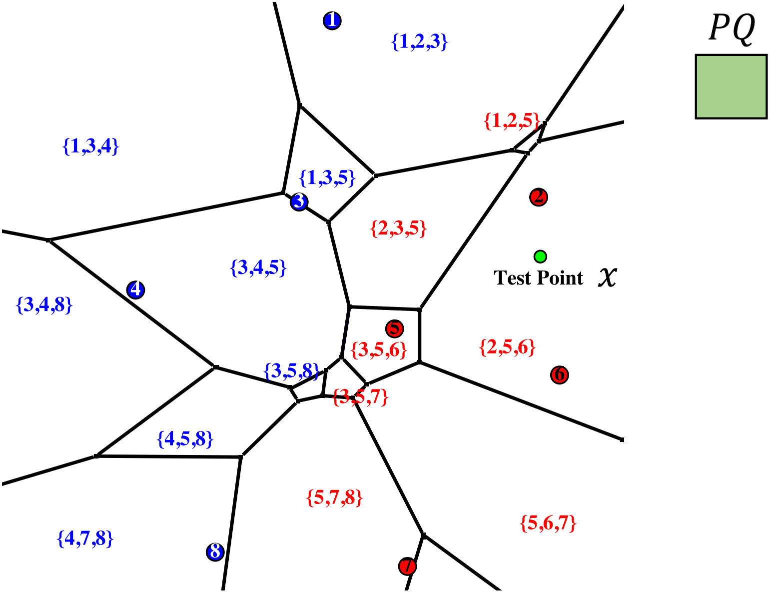
GeoAdEx

- We don't need to search every cell, only ones close to x.
- A local search that starts around x and then keep expanding is very suitable for this problem.

GeoAdEx

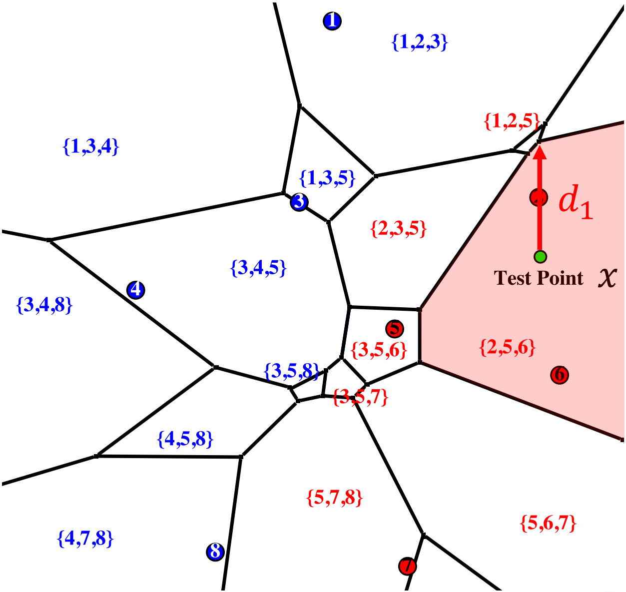
- We don't need to search every cell, only ones close to x.
- A local search that starts around x and then keep expanding is very suitable for this problem.
- We can use the **neighboring relationship** of the Voronoi diagram to find the next cell to search.

GeoAdEx



1
{1,2,3}
{1,2,5}
{1,3,4}
{1,3,5}
3
2
{2,3,5}
{3,4,5}
4
{3,4,8}
Test Point x
5
{3,5,6}
{2,5,6}
{3,5,8}
6
{3,5,7}
{4,5,8}
{5,7,8}
{5,6,7}
{4,7,8}
8
7
PQ

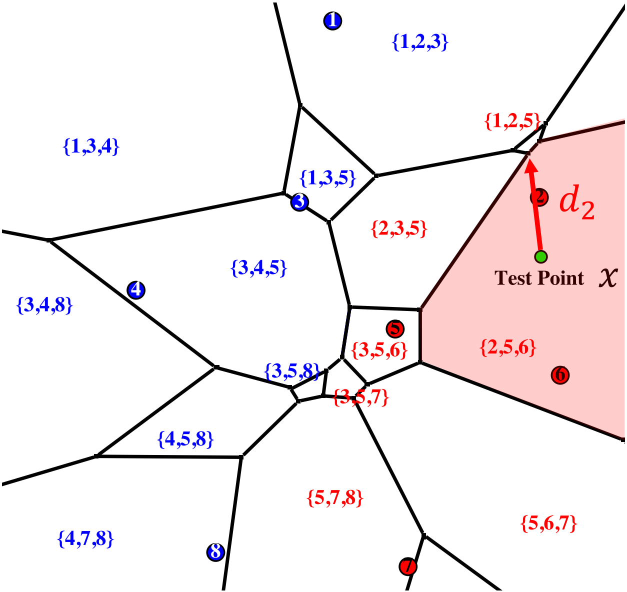
GeoAdEx



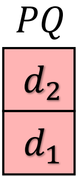
PQ

$\boxed{d_1}$

GeoAdEx

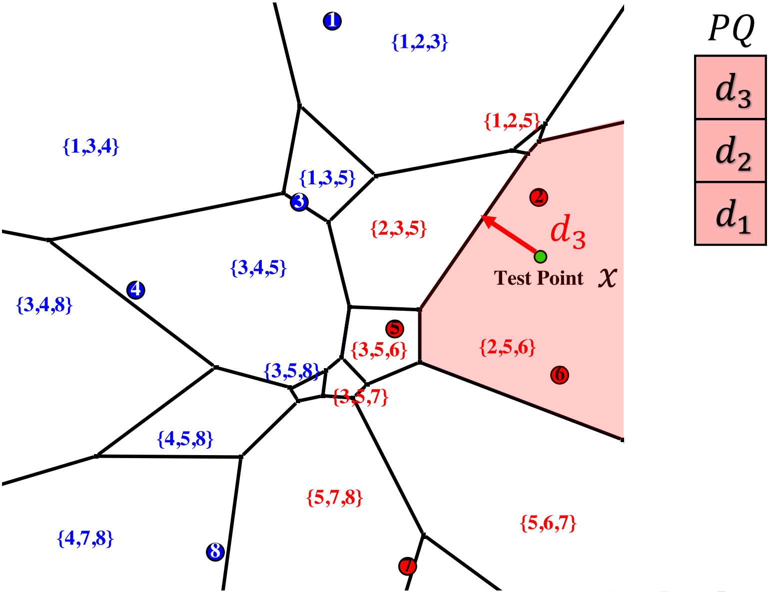


1
{1,2,3}
{1,2,5}
{1,3,4}
{1,3,5}
3
2
d_2
{2,3,5}
{3,4,5}
Test Point x
{3,4,8}
4
5
{3,5,6}
{2,5,6}
{3,5,8}
6
{3,5,7}
{4,5,8}
{5,7,8}
{5,6,7}
{4,7,8}
8
7



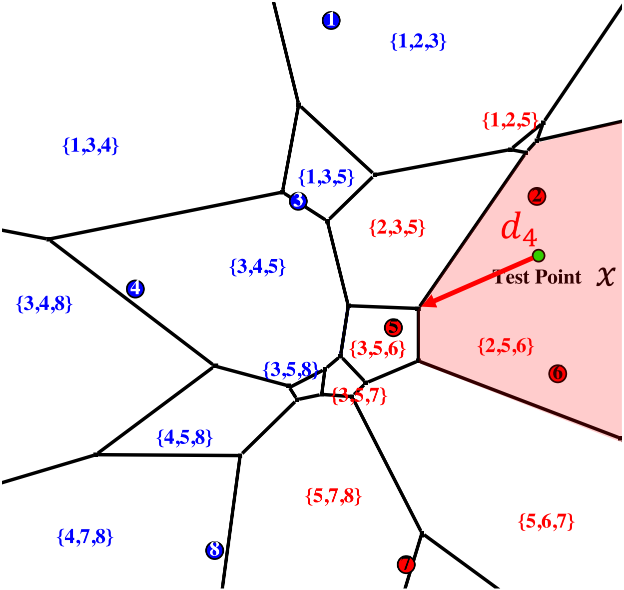
PQ
d_2
d_1

GeoAdEx



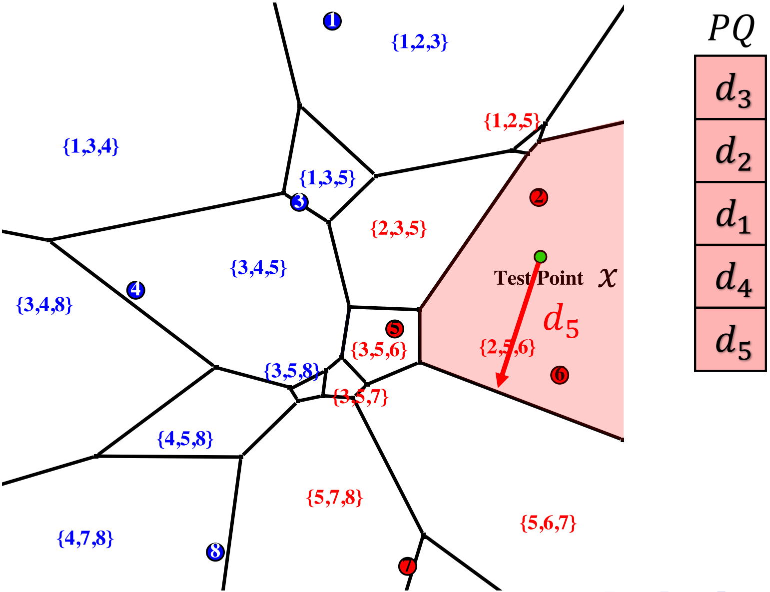
{1,2,3}
{1,2,5}
{1,3,4}
{1,3,5}
{2,3,5}
{3,4,5}
{3,4,8}
{3,5,6}
{2,5,6}
{3,5,8}
{3,5,7}
{4,5,8}
{5,7,8}
{5,6,7}
{4,7,8}
d_3
Test Point x
PQ
d_3
d_2
d_1

GeoAdEx

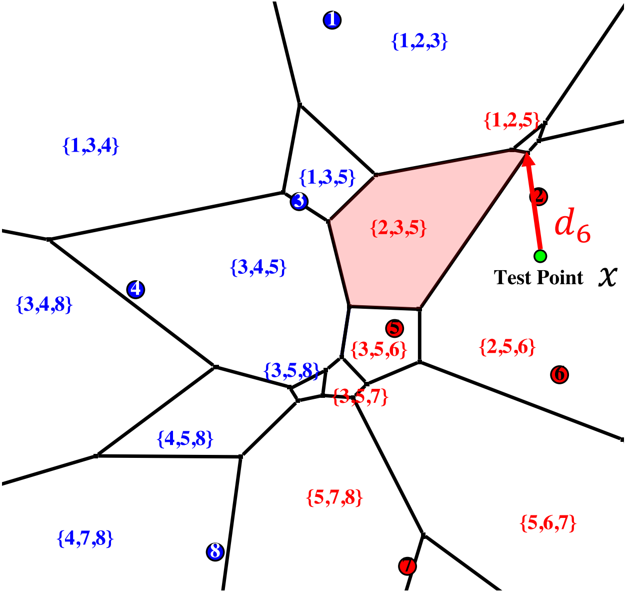


PQ
d_3
d_2
d_1
d_4

GeoAdEx


{1,2,3}
{1,2,5}
{1,3,4}
{1,3,5}
{2,3,5}
{3,4,5}
{3,4,8}
Test Point x
d_5
{3,5,6}
{2,5,6}
{3,5,8}
{3,5,7}
{4,5,8}
{5,7,8}
{5,6,7}
{4,7,8}
PQ
d_3
d_2
d_1
d_4
d_5

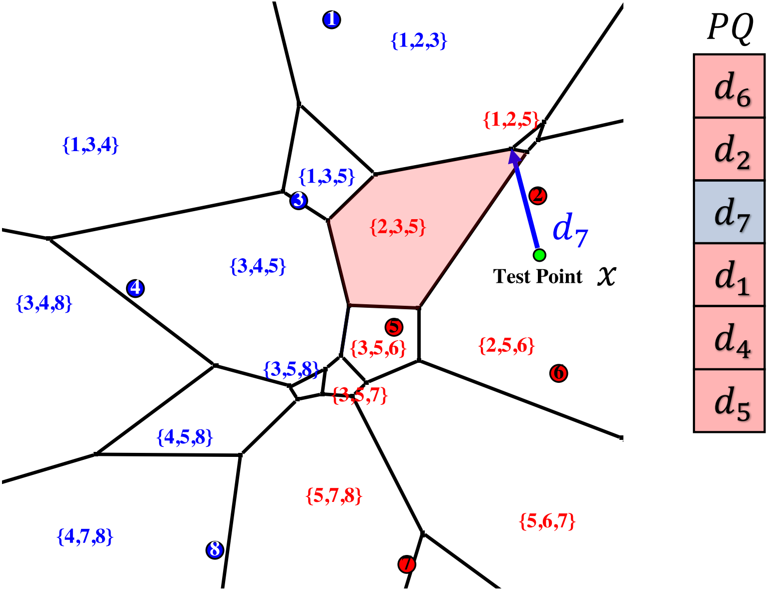
GeoAdEx



PQ

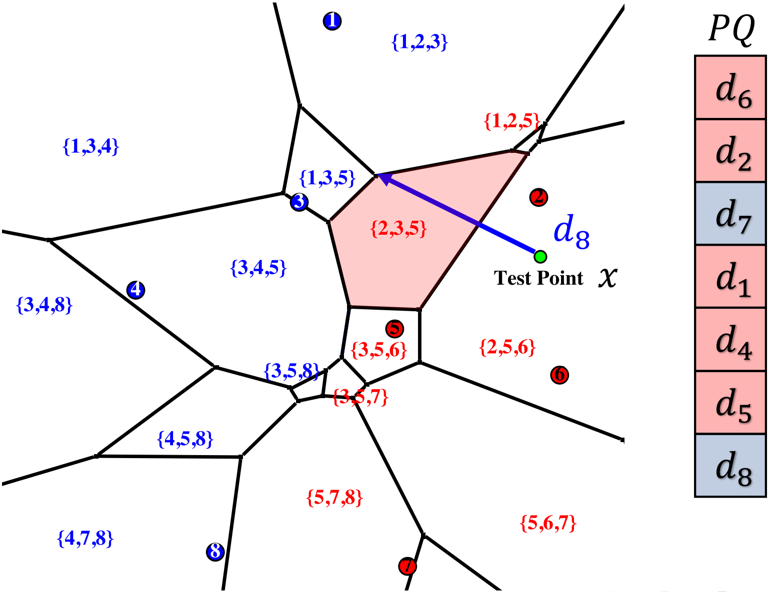
d_6
d_2
d_1
d_4
d_5

GeoAdEx



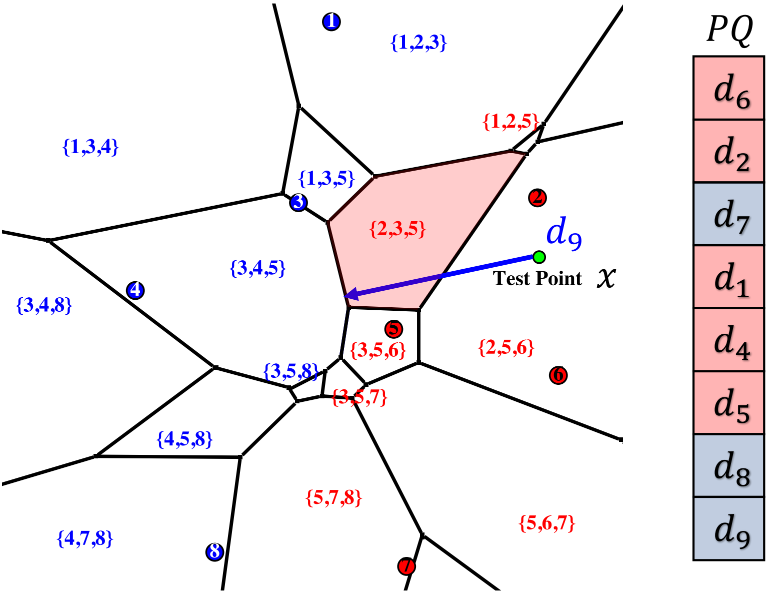
1
{1,2,3}
{1,2,5}
{1,3,4}
{1,3,5}
3
2
{2,3,5}
d_7
Test Point x
{3,4,5}
4
{3,4,8}
5
{3,5,6}
{2,5,6}
6
{3,5,8}
{3,5,7}
{4,5,8}
{5,7,8}
{5,6,7}
{4,7,8}
8
7
PQ
d_6
d_2
d_7
d_1
d_4
d_5

GeoAdEx



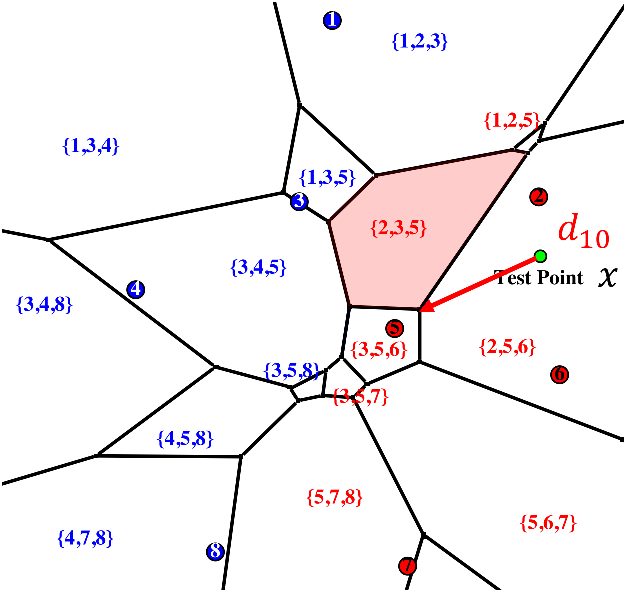
{1,2,3}
{1,2,5}
{1,3,4}
{1,3,5}
{2,3,5}
d_8
Test Point x
{3,4,5}
{3,4,8}
{3,5,6}
{2,5,6}
{3,5,8}
{3,5,7}
{4,5,8}
{5,7,8}
{5,6,7}
{4,7,8}
PQ
d_6
d_2
d_7
d_1
d_4
d_5
d_8

GeoAdEx

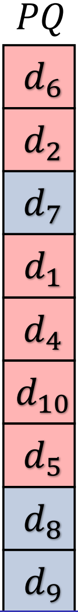


{1,2,3}
{1,2,5}
{1,3,4}
{1,3,5}
{2,3,5}
d_9
Test Point x
{3,4,5}
{3,4,8}
{3,5,6}
{2,5,6}
{3,5,8}
{3,5,7}
{4,5,8}
{5,7,8}
{5,6,7}
{4,7,8}
PQ
d_6
d_2
d_7
d_1
d_4
d_5
d_8
d_9

GeoAdEx



1
{1,2,3}
{1,2,5}
{1,3,4}
{1,3,5}
3
2
{2,3,5}
d_{10}
Test Point x
{3,4,5}
4
{3,4,8}
5
{3,5,6}
{2,5,6}
{3,5,8}
6
{3,5,7}
{4,5,8}
{5,7,8}
{5,6,7}
{4,7,8}
8
7

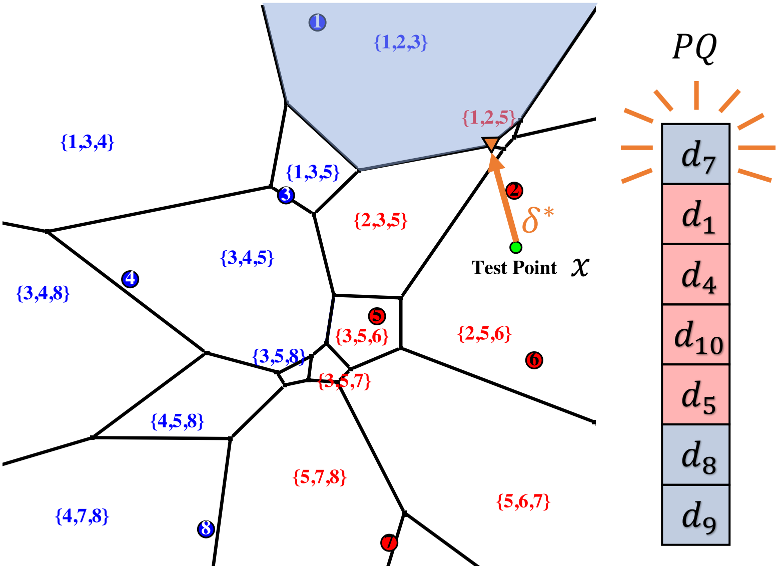


PQ
d_6
d_2
d_7
d_1
d_4
d_{10}
d_5
d_8
d_9

GeoAdEx

Keep going until an adversarial cell is popped from PQ...

GeoAdEx



1
{1,2,3}
{1,2,5}
{1,3,4}
{1,3,5}
3
2
{2,3,5}
δ^*
Test Point x
{3,4,5}
4
{3,4,8}
5
{3,5,6}
{2,5,6}
6
{3,5,8}
{3,5,7}
{4,5,8}
{5,7,8}
{5,6,7}
{4,7,8}
8
7
PQ
d_7
d_1
d_4
d_{10}
d_5
d_8
d_9

GeoAdEx

- A similar algorithm has been applied to piecewise-linear neural network in Jordan et al. [2019].
- We can apply Lemma C.2 and C.1 from Jordan et al. [2019] to show two guarantees of GeoAdEx.
- Details are in the paper.

Lemma 1: Correctness of GeoAdEx

Provided no time limit, GeoAdEx terminates when it finds the optimal adversarial examples or equivalently, one of the solutions of Eqn. (1).

Lemma 2: Lower bound guarantee

If GeoAdEx terminates early, the distance from test point x to the last deleted facet from PQ is a lower bound to ϵ^*.

GeoAdEx: How to Find Neighbors

- Voronoi diagram is expensive to construct for $k > 1$.

GeoAdEx: How to Find Neighbors

- Voronoi diagram is expensive to construct for $k > 1$.
- Given $V(\{x_1, \ldots, x_{k-1}, x_k\})$, we can swap x_k for any other x_l to get its neighbor $V(\{x_1, \ldots, x_{k-1}, x_l\})$. Repeat this for all choices of x_l.

GeoAdEx: How to Find Neighbors

- Voronoi diagram is expensive to construct for $k > 1$.
- Given $V(\{x_1, \ldots, x_{k-1}, x_k\})$, we can swap x_k for any other x_l to get its neighbor $V(\{x_1, \ldots, x_{k-1}, x_l\})$. Repeat this for all choices of x_l.
- But we can also narrow it down: From Theorem 1, we only have to consider x_l s.t. $V(\{x_l\})$ *neighbors with one of the order-1 cells* $V(\{x_1\}), V(\{x_2\}), \ldots, V(\{x_k\})$.

Theorem 1: Order-1 neighbors

Let $S = \{x_1, \ldots, x_{k-1}\} \subset X$ be a set of $k-1$ generators. Let $x_k, x_l \in X$ be two generators such that $x_k, x_l \notin S$. If $V(S \cup \{x_k\})$ and $V(S \cup \{x_l\})$ are two neighboring order-k Voronoi cells, then the order-1 Voronoi cell $V(\{x_l\})$ is neighboring with at least one of the $V(\{x_1\}), \ldots, V(\{x_{k-1}\}), V(\{x_k\})$.

GeoAdEx: Approximate Version

- Theorem 1 reduces the number of potential neighbors to search as well as constraints from $k(n-k)$ to $k(\sum_{j=1}^{k} s_j)$ where s_j is the number of order-1 neighbors of x_j.

GeoAdEx: Approximate Version

- Theorem 1 reduces the number of potential neighbors to search as well as constraints from $k(n-k)$ to $k(\sum_{j=1}^{k} s_j)$ where s_j is the number of order-1 neighbors of x_j.
- This helps significantly when $s_j \ll n$.

GeoAdEx: Approximate Version

- Theorem 1 reduces the number of potential neighbors to search as well as constraints from $k(n-k)$ to $k(\sum_{j=1}^{k} s_j)$ where s_j is the number of order-1 neighbors of x_j.
- This helps significantly when $s_j \ll n$.
- However, it still relies on the order-1 Voronoi diagram which is expensive to construct in high dimension: $\mathcal{O}(n \log n + n^{\lceil d/2 \rceil})$ [Aurenhammer et al., 2013]

GeoAdEx: Approximate Version

- Theorem 1 reduces the number of potential neighbors to search as well as constraints from $k(n-k)$ to $k(\sum_{j=1}^{k} s_j)$ where s_j is the number of order-1 neighbors of x_j.
- This helps significantly when $s_j \ll n$.
- However, it still relies on the order-1 Voronoi diagram which is expensive to construct in high dimension: $\mathcal{O}(n \log n + n^{\lceil d/2 \rceil})$ [Aurenhammer et al., 2013]
- Instead of building a Voronoi diagram, we choose to **approximate** the order-1 neighbors with m nearest neighbors of x_i.

GeoAdEx: Approximate Version

- #neighbors to search and #constraints reduced to at most $k^2 m$.

GeoAdEx: Approximate Version

- #neighbors to search and #constraints reduced to at most $k^2 m$.
- Drawback: the approximate GeoAdEx can miss some cells.

GeoAdEx: Approximate Version

- #neighbors to search and #constraints reduced to at most k^2m.
- Drawback: the approximate GeoAdEx can miss some cells.
- An additional step has to be put in place to ensure that it returns valid adversarial examples.

GeoAdEx: Approximate Version

- #neighbors to search and #constraints reduced to at most $k^2 m$.
- Drawback: the approximate GeoAdEx can miss some cells.
- An additional step has to be put in place to ensure that it returns valid adversarial examples.
- As a result, the optimality guarantee (lower bound) no longer applies.

GeoAdEx: Approximate Version

- #neighbors to search and #constraints reduced to at most k^2m.
- Drawback: the approximate GeoAdEx can miss some cells.
- An additional step has to be put in place to ensure that it returns valid adversarial examples.
- As a result, the optimality guarantee (lower bound) no longer applies.
- We also introduce several performance speed-up's which are explained in the paper.

Results

Table: Mean norm of the adversarial perturbations on 100 random test points on 5-NN classifiers across datasets (lower is better).

Attacks	Australian	Covtype	Diabetes	Fourclass	Gaussian
S&W [2020]	.4748	.2281	.1215	.1087	.0463
Yang et al. [2020]	.5524	.3047	.1824	.1309	.1776
Wang et al. [2018]	.5110	.2613	.1382	.1127	.1195
GeoAdEx	.4608	.1856	.1021	.1066	.0401
	(.9705)	(.8137)	(.8403)	(.9981)	(.8661)

- Results on the remaining datasets and for $k = 3, 7$ have similar trends and can be found in the paper.
- Results for GeoAdEx and the other baselines without approximation on $k = 1$ are also included in the paper.

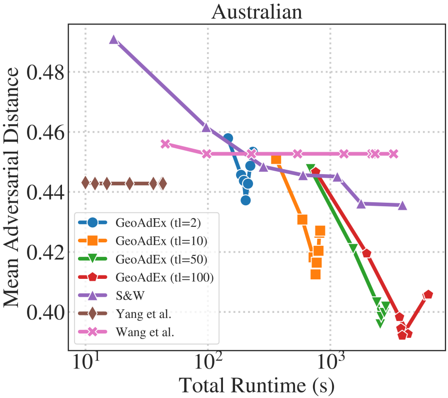
Runtimes

Table: Total runtimes in seconds of the same experiments.

Attacks	Australian	Covtype	Diabetes	Fourclass	Gaussian
S&W [2020]	662	2683	333	334	807
Yang et al. [2020]	11	3443	12	7	4927
Wang et al. [2018]	521	564	213	155	378
GeoAdEx	7798	3370	4470	4208	3830

Runtimes: Attack Hyperparameters

- It is hard to control runtime of each algorithm so we vary their hyperparameters and plot the runtime vs. mean perturbation norm.



GeoAdEx can take advantage of the increased runtime unlike the baselines which plateau quickly.

Summary & Open Problems

GeoAdEx outperforms the baselines in discovering adversarial examples for k-NN classifiers with $k \geq 1$. It finds considerably smaller adversarial perturbation on most of the datasets.

Future improvements:

- Better heuristics to approximate order-1 neighbors.
- Approximate multiple neighboring non-adversarial cells as a single large cell to remove unnecessary computation.
- GeoAdEx can also be extended to other space-partitioning classifiers such as decision trees and random forests.

Thank You!

References I

F. Aurenhammer, R. Klein, and D.-T. Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific Publishing Co., Inc., USA, 1st edition, 2013. ISBN 978-981-4447-63-8.

M. Jordan, J. Lewis, and A. G. Dimakis. Provable certificates for adversarial examples: Fitting a ball in the union of polytopes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

C. Sitawarin and D. Wagner. Minimum-norm adversarial examples on KNN and KNN based models. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 34–40, Los Alamitos, CA, USA, May 2020. IEEE Computer Society. doi: 10.1109/SPW50608.2020.00023.

Y. Wang, S. Jha, and K. Chaudhuri. Analyzing the robustness of nearest neighbors to adversarial examples. Technical report, 2018.

References II

Y.-Y. Yang, C. Rashtchian, Y. Wang, and K. Chaudhuri. Robustness for non-parametric classification: A generic attack and defense. In S. Chiappa and R. Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 941–951. PMLR, Aug. 2020.