STEVENS
INSTITUTE OF TECHNOLOGY
1870

Lecture 3 Logistic Regression

Jiarui Li

Sept 22, 2025

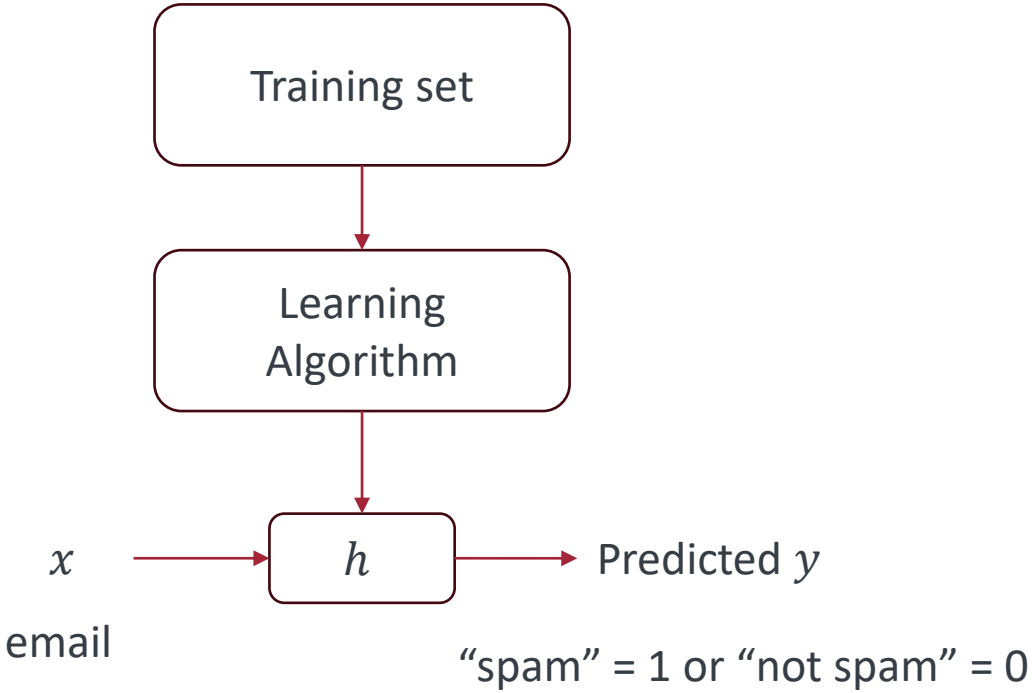
Overview

- Logistic Regression
- Multiclass Regression

Classification

- While in regression, the target variables is continuous, the values y we want to predict take on only a small number of ***discrete values.*** Unlike regression, here the goal is to find a **decision boundary** that separates the classes.

- We will focus on the ***binary classification,*** means the predicted y can take on only 0 and 1.



Logistic Regression – A Probability View

- A hard boundary would help(e.g. step/threshold function), but in reality, data points are often near the boundary.

$\mathbf{Pr}(\boldsymbol{y}|\boldsymbol{x})$, where y is the target binary variable, we define $p = \Pr(y = 1|x)$

e.g. Pr(Raining tomorrow | windy today)

Let $\Pr(y = 1|x;\theta) = h_\theta(x)$

$\Pr(y = 0|x;\theta) = 1 - h_\theta(x)$

Assumption: The probability is modeled with parameter θ; otherwise, optimization problem doesn't work

We can write it in a more compact way:

$$p(y|x;\theta) = h_\theta(x)^{\,y}(1 - h_\theta(x))^{1-y}$$

Logistic Regression

- Maximize the likelihood of the parameters:

$$\prod_{i=1}^{m} h_\theta(x^{(i)})^{\,y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}}$$

- Task: estimate the parameter θ to maximize the likelihood

Can we use **linear regression** to solve this?

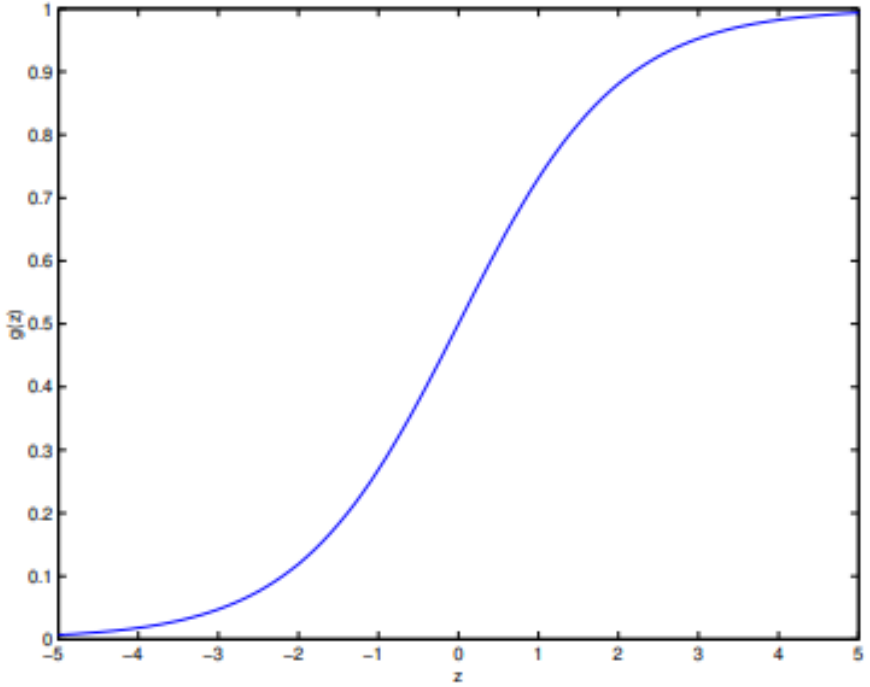
Sigmoid Function

Any solution? “Compress”

- Logistic regression models the **log-odds** as a linear function: $log\frac{p}{1-p} = \theta^T x$

$$\mathrm{h}_\theta(\mathrm{x}) = \mathrm{g}(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

$\mathrm{g}(\mathrm{z}) = \frac{1}{1+e^{-z}}$ is called the sigmoid function



Prediction:

$$y = \begin{cases} 0, & if\ \mathrm{h}_\theta(\mathrm{x}) < 0.5 \\ 1, & if\ \mathrm{h}_\theta(\mathrm{x}) \geq 0.5 \end{cases}$$

Why Sigmoid?

Function	Range	Differentiable?	Pros	Cons
Step (Heaviside)	$\{0, 1\}$	No	Simple interpretation (hard decision)	Not smooth → no gradient, can't optimize
Linear (identity / scaling)	$(-\infty, \infty)$	Yes	Easy to compute	Not bounded in $[0, 1]$, invalid probability
Tanh	$(-1,1)$	Yes	Symmetric, useful in neural networks	Output not in $(0,1)$; needs rescaling
Softmax	$[0, 1]$,sums to 1 across classes	Yes	Best for **multi-class** problems	Overkill for binary classification
Sigmoid (logistic)	$(0,1)$	Yes	Perfect probability mapping; convex loss; interpretable log-odds	Can saturate (slow learning at extremes)

Logistic Regression

- Performance measure: Log loss function

$$J(\theta) = -\log L(\theta) = -\sum_{i=1}^{m} [y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

It is easier to explain the log loss function with one train example case, in which we want to maximize the posterior probability

$$\Pr(y|x) = [h_\theta(x)]^y \left[(1 - h_\theta(x))^{1-y}\right] = \begin{cases} h_\theta(x), & when\ y = 1 \\ 1 - h_\theta(x), & when\ y = 0 \end{cases}$$

$$\log \Pr(y|x) = y \log h_\theta(x) + (1 - y) \log(1 - h_\theta(x))$$

Logistic Regression

- Similarly, next step is to choose the parameters θ that minimize the loss function

$$\theta = argmin\, \mathcal{J}(\theta)$$

Where $J(\theta) = -\sum_{i=1}^{m}[y^{(i)} \log h_\theta(x^{(i)}) + \left(1 - y^{(i)}\right) \log(1 - h_\theta(x^{(i)}))]$

- No normal equation (i.e., closed-form solution) for θ
- The cost function is convex and derivable. Gradient Descent is guaranteed to find global maximum

Training Logistic Regression

- The gradient of the log loss function is

$$\frac{\partial}{\partial \theta_j} \mathcal{J}(\theta) = \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

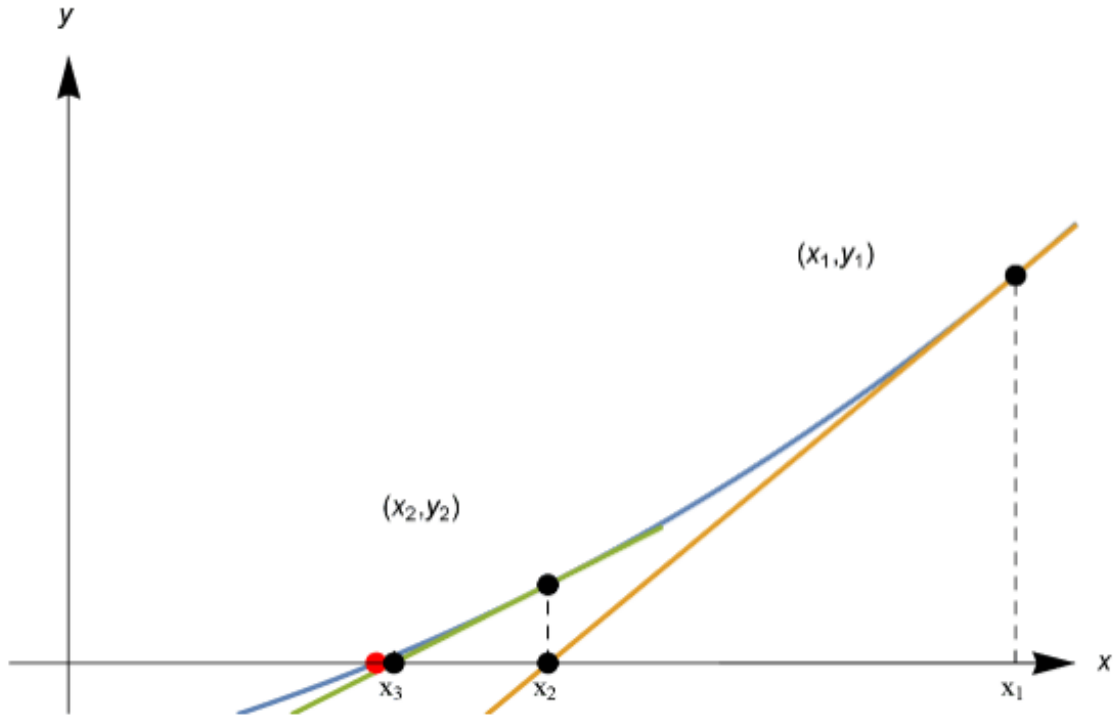
- At each round of gradient descent, θ is updated as following

$$\theta_j = \theta_j - \frac{1}{m} \sum_{i=1}^{m} \alpha \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

Similar to the linear regression, you can choose different mode of gradient descent and add regularization to the loss function

Newton's Method

- Idea: using tangent lines
- Why: efficient, move a much bigger step.
- Process:
  1. Find the tangent.
  2. Find the intersects of tangent line and x-axis
  3. Repeat
- Example:



In (x_1, y_1), the tangent line:

$$\frac{y - y_1}{x - x_1} = f'(x_1)$$

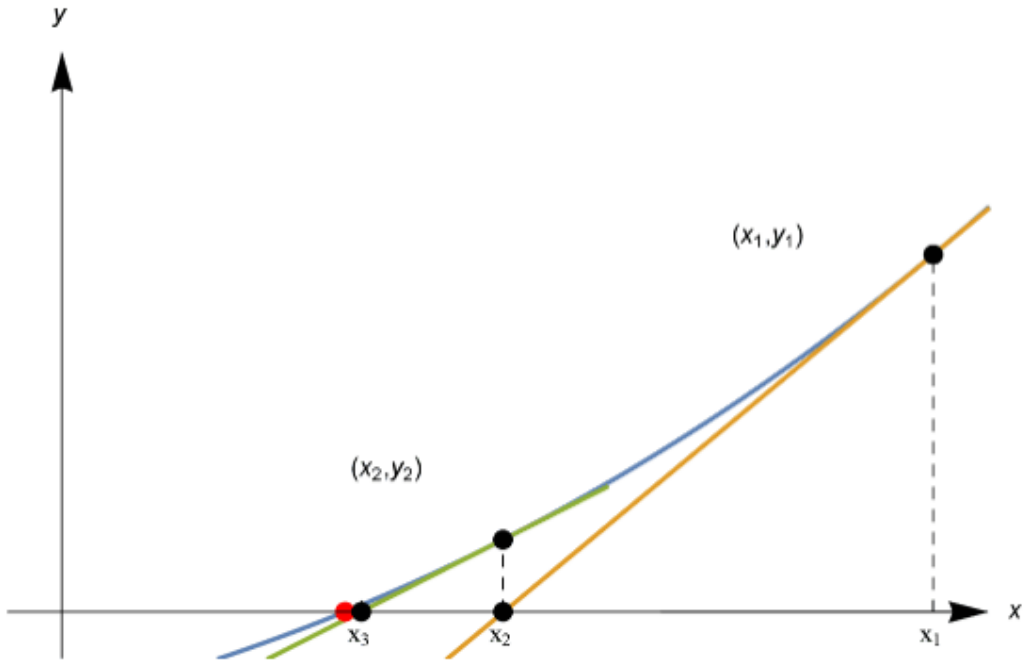
When we set y=0, we have $x_2 = x_1 - \frac{y_1}{f'(x_1)}$ ⟹ $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$

Newton's Method in LR

For LR, we are looking for minimizing the cost function, so actually we set the $f(x) = \mathcal{J}'(\theta)$

Thus, the updating rule:

$$\theta := \theta - \frac{\mathcal{J}'(\theta)}{\mathcal{J}''(\theta)}$$



Imagine x is the parameter axis and y is the corresponding derivatives of cost function

Notice here, θ is a real number which means it is one dimension. It can be generalized to the vector (Newton-Raphson Method):

$$\theta := \theta - H^{-1} \cdot \mathcal{J}'(\theta)$$

Where H is Hessian matrix $H_{ij} = \frac{\partial^2 \mathcal{J}(\theta)}{\partial \theta_i \partial \theta_j}, \mathrm{i,j} \in [0, n]$

Newton's Method in LR

- Advantages:
  - Not sensitive to the initialization
  - Quadratic convergence rate
- Disadvantages:
  - Much more expensive when we have a ``high-dimensional'' dataset
  - Require the cost function to be twice differentiable
  - The method only works for convergent functions and will be not effective in multiple curvature optimization (neural network)

Multi-class Classification

- "One -vs-Rest" method:

We can use **binary classifier** for multi-class classification with so-called the One-Vs-Rest (OvR) method. Specifically, it uses multiple rounds of binary classification for multi-class classification.

For example, to determine if an object X is a dog, cat or fish, we call a binary classifier $f_i()$ as follows:

if f_1(X) outputs dog
    return dog;
else if f_2(X) outputs cat
    return cat;
else return fish

Multi-class Classification

One-Vs-One Method:

The One-Vs-One (OvO) method constructs a **binary classifier** for each pair of classes. Therefore, with ***K*** classes, we need to construct ***K(K-1)/2*** binary classifiers.

The decision at prediction time can be made by **counting the votes** from individual binary classifiers. In case of a tie, it compares the aggregated classification confidence (i.e., the output probability) of individual binary classifiers of each class and the higher one is selected.

The OvO method is **slower** than OvR. But for some algorithms (e.g., Kernel algorithms) which cannot scale with many training examples, this algorithm can be helpful.

Multi-class Classification

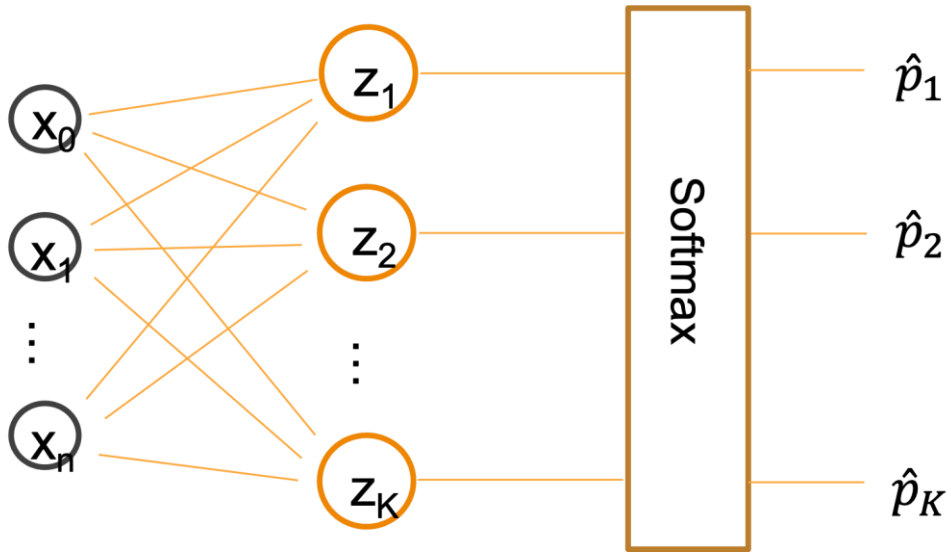
Another approach for multi-class classification is to use the ***multinomial logistic regression***. For each class $1 \leq k \leq K$,

1) first compute $z_k(x) = {\theta_k}^T \cdot x$

2) then compute Softmax function:

$$h_k(x) = g(z_k(x))_k = \frac{e^{z_k(x)}}{\sum_{j=1}^{K} e^{z_j(x)}}$$

where θ_k is the vector of parameters of input features for z_k.



Multi-class Classification

- The performance measure is the cross-entropy cost function

$$\mathcal{J}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_k^{(i)} \log(h_k(x^{(i)}))$$

where $y_k^{(i)} = \begin{cases} 1, \text{ if } x^{(i)} \text{ is an example of class } k \\ 0, \text{ if } x^{(i)} \text{ is not an example of class } k \end{cases}$

- The gradient descent methods can be used to train the multinomial logistic regression model. And the gradients can be computed as:

$$\frac{\partial}{\partial \boldsymbol{\theta}_k}\mathcal{J}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(h_k\left(x^{(i)}\right) - y_k^{(i)}\right)x^{(i)}$$

Multi-class Classification

Other Approaches: **Error-Correcting Output Codes**

The Error-Correcting Output Codes (ECOC) method encodes ***K* classes** into ***N* bit vectors**. Each class is represented as a bit in each bit vector. ECOC trains *N* **binary classifiers**, each splitting one group of classes from another (using the column bit vectors below). At prediction time, the N binary classifiers are called, the outputs of them yielding an N-bit vector. A class with the **closest Euclidean distance** to the **N-bit vector** is selected. To reduce the classification error, error correcting codes are used when generating the "code book"

Class	Code Word														
	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
0	1	1	0	0	0	0	1	0	1	0	0	1	1	0	1
1	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0
2	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1
3	0	0	1	1	0	1	1	1	0	0	0	0	1	0	1
4	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1
5	0	1	0	0	1	1	0	1	1	1	0	0	0	0	1
6	1	0	1	1	1	0	0	0	0	1	0	1	0	0	1
7	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1
8	1	1	0	1	0	1	1	0	0	1	0	0	0	1	1
9	0	1	1	1	0	0	0	0	1	0	1	0	0	1	1

Classifier Evaluation

Classification measures

- Accuracy is only one measure (error = 1-accuracy).

  Accuracy is not always suitable

  - In text mining, we may only be interested in the documents of a particular topic, which are only a small portion of a big document collection.
  - In classification involving skewed or highly imbalanced data, e.g., network intrusion and financial fraud detections, we are interested only in the minority class.
    - High accuracy does not mean any intrusion is detected.
    - E.g., 1% intrusion. Achieve 99% accuracy by doing nothing.

- The class of interest is commonly called the **positive class**, and the rest **negative classes**.

Performance Measures of Classifiers

Sometimes we need both a high accuracy rate and a low error rate. ***Confusion matrix*** is an important tool to measure both accuracy rates and error rates.

	Classified Positive	Classified Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

True Positive (TP): the number of true positive examples
True Negative (TN): the number of true negative examples
False Positive (FP): the number of false positive examples (type I error)
False Negative (FN): the number of false negative examples (type II error)

Performance Measures of Classifiers

We are often interested in two measures on relevance:

- ***precision*** $p = \frac{TP}{TP+FP}$, is the number of true positive divided by all classified as positive.
- ***recall*** *(a.k.a.,* ***sensitivity****)* $r = \frac{TP}{TP+FN}$, is the number of true positive divided by all positive.
- $F_1 = \frac{2pr}{p+r}$, is a measure of both p and r. F_1-score is large only if both p and r are large.

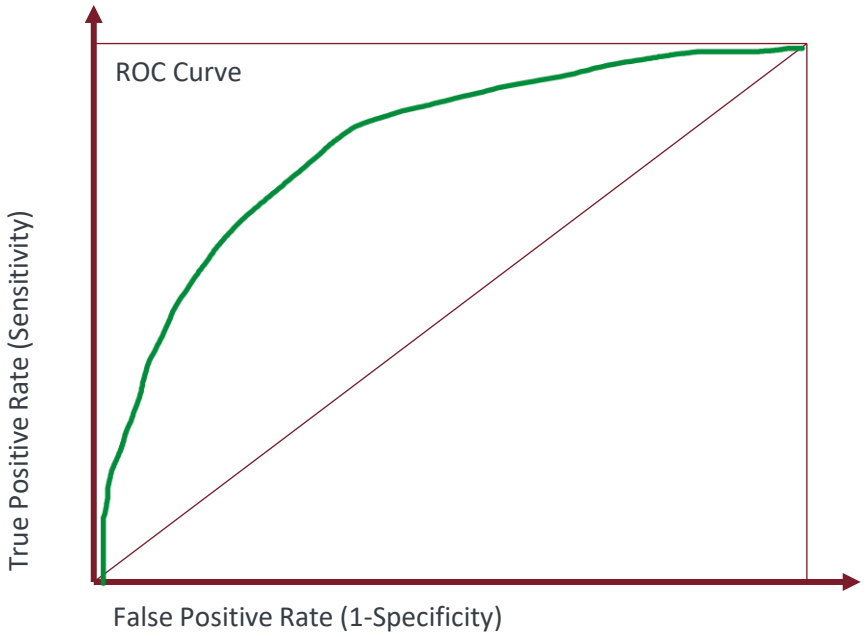
	Classified Positive	Classified Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

"**Precision** can be seen as a measure of quality and **recall** as a measure of quantity.

- Higher precision means that an algorithm returns *more relevant results* than irrelevant ones,
- high recall means that an algorithm returns *most of the relevant results*." (Wikipedia)

Performance Measures of Classifiers

Another common measure is the ***Receiver Operating Characteristic*** (ROC) curve, which shows the tradeoff between **True Positive Rate** (i.e., ***sensitivity*** = $\frac{TP}{TP+FN}$) and **False Positive Rate** (i.e., ***1 – specificity*** = $\frac{FP}{FP+TN}$).



- The closer to the top-left corner of the ROC space, the better the classifier is.
- The closer to the 45-degree diagonal of the ROC space, the worse the classifier is.
- The area under the curve (AUC) is usually used to measure above properties.
- ROC/AUC are usually used to measure ***how well a binary classifier distinguish the two classes***.

STEVENS
INSTITUTE OF TECHNOLOGY
1870

THANK **YOU**

Stevens Institute of Technology
1 Castle Point Terrace, Hoboken, NJ 07030