Machine Learning

Prof. Sergei Gleyzer

Week 2

PH451 PH551

January 23, 2025

Announcements

- **Read Chapter 3 in textbook**

- **Quiz date: Thu, Feb. 13**

Loss Functions

Loss Functions

$\vec{X}_i = \{x_1, x_2 \dots x_m\}_i$ | y_i

Input | Output

GOAL: Evaluate hypothesis on $\boxed{\text{training}}$ data (how bad?)

↑ Loss (worse) ↓ Loss (better)

Loss = 0 ⟶ Perfect

Examples

0/1 Loss

Count the Mistakes

Usually use normalized 0/1 Loss

⟹ fraction of missclassified samples ("training error")

$$\boxed{L_{0/1}(f) = \frac{1}{n}\sum_{i=1}^{n} \delta f(x_i)}$$

where

$$\delta_{f(x_i)} = \overbrace{\begin{cases} 1 & \underbrace{f(\vec{x}_i) \neq 1}_{\text{miss-classified}} \\ 0 & \end{cases}}^{\text{non-continuous "impractical"}}$$

Absolute Loss

$$\boxed{L_{Abs}(f) = \frac{1}{n}\sum_{i=1}^{n} |f(x_i) - y_i|}$$

$\boxed{\text{MAE, L1}}$

"Manhattan" Norm

⟶ non-negative

⟶ grows linearly w. missclassification

⟶ typically useful for (noisy) regression ← more robust to outliers

Squared Loss

$$\boxed{L_{SQ}(f) = \frac{1}{n}\sum_{i=1}^{n} \left(f(\vec{x}_i) - y_i\right)^2}$$

$\boxed{\text{MSE, L2}}$

RMSE: "Euclidean Norm"

⟶ non-negative

⟶ grows quadratically w. missed predictions

⟶ useful for regression [Ordinary Least Squares]

⟶ estimates mean given x_i

Huber

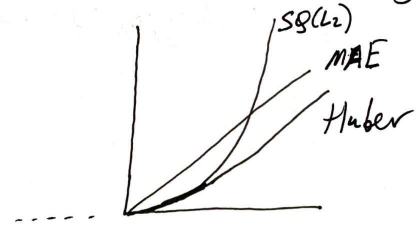
$$\boxed{L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & |a| \leq \delta \\ \delta(|a| - \frac{1}{2}\delta) & \end{cases}}$$

$a = y - f(x)$

"residual"

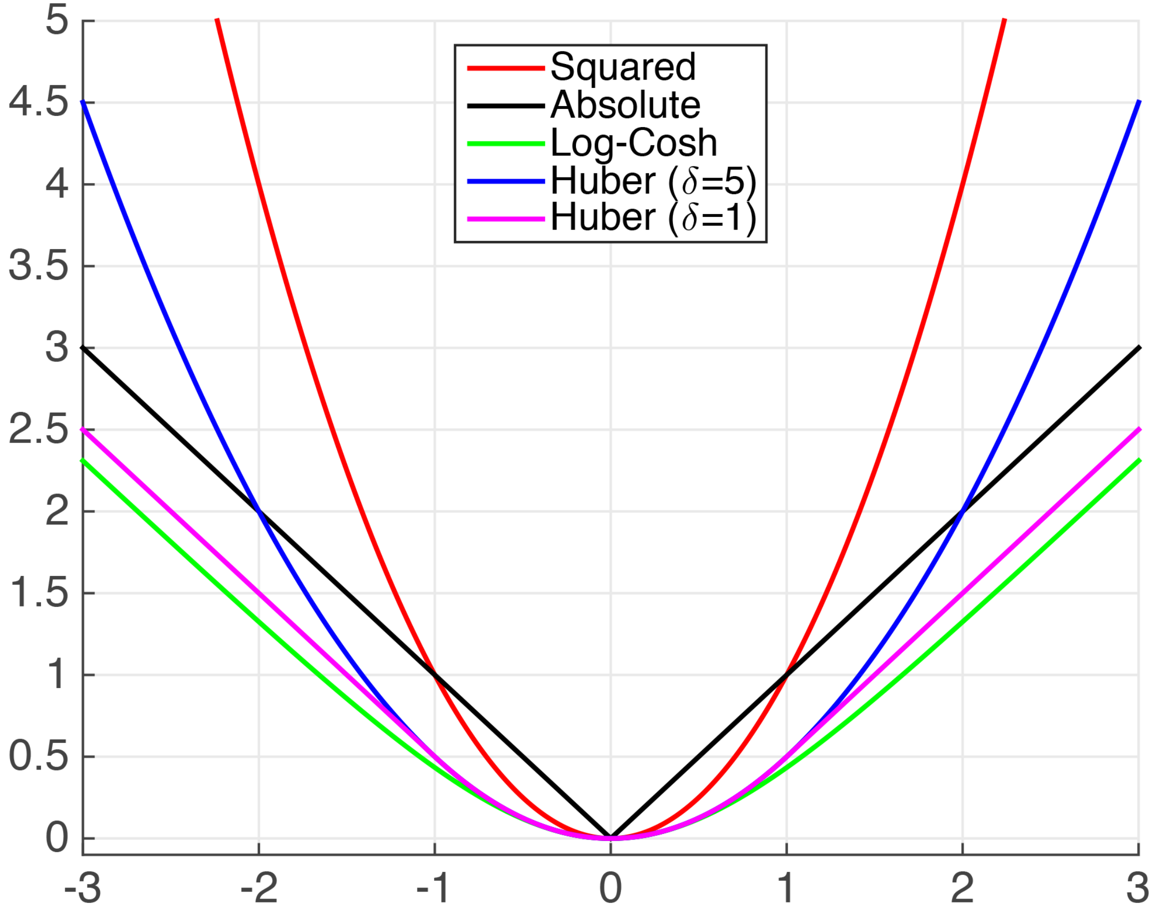
⟶ quadratic for small x

⟶ linear for large x



⟶ "best of both worlds"

Loss Functions



Cross Entropy

$$L_{CE} = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$$

Log loss

binary class.

$f(x)$

probability 1

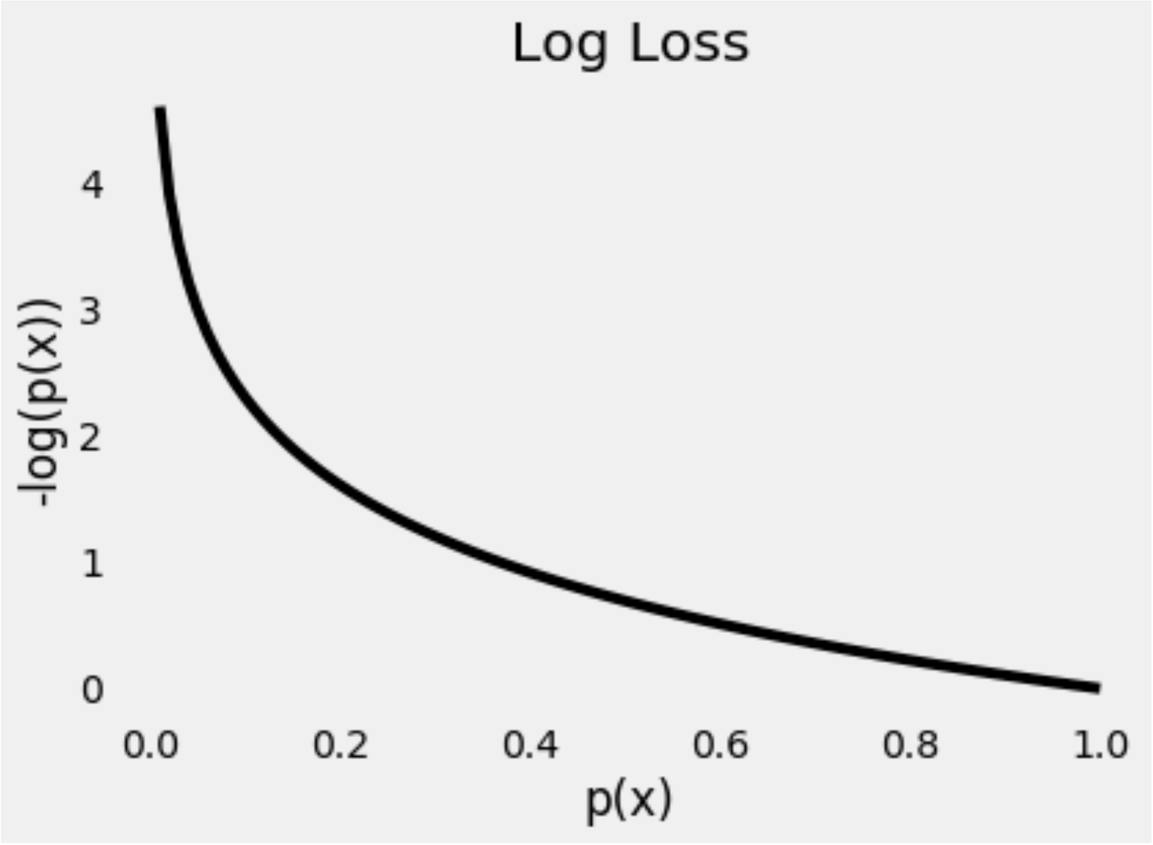
convex

$\uparrow$ loss

$y_i = 0 \quad \log(1-f(\vec{x}))$

$y_i = 1 \quad \log(f(x))$

Cross Entropy



Types of Learning

Typical Learning Tasks

Classification

- Put in categories (classes) based on inputs

Regression

- Estimate a function/predict a numeric value

Learning Types

Human supervision?

Supervised
Unsupervised
Semi-supervised
Reinforcement learning

Offline? Incrementally?

Building a predictive model?

model-based or **instance-based**

[Un]Supervised Learning

How much supervision during training?

- **Supervised** (100% expert labeled)
- **Unsupervised** (unlabeled – learn on your own)
- **Semi-supervised** (partially labeled)

- **Reinforcement Learning**
  - Learning system observes environment and gets rewards based on actions (i.e. training your dog)
  - "Agent" identifies policy that maximizes reward

Online vs. Offline Learning

How much data during training?

- **Offline (batch)** – train on all available data
  - Expensive for large datasets

- **Online (incremental)** – small mini-batches
  - Learn on the fly
  - Good for limited resources

Model vs. Instance Learning

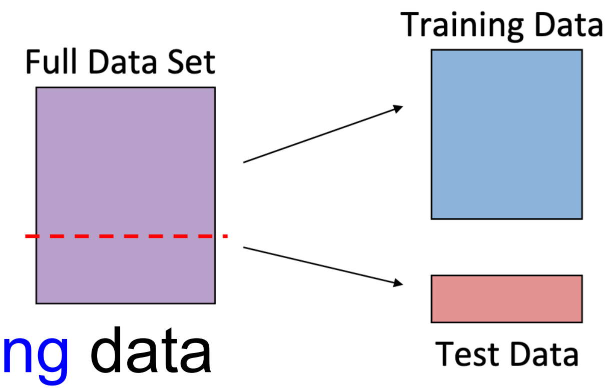
How to Generalize?

- **Instance (based)**
  - similarity measure compared to labeled examples

- **Model (based)**
  - build a predictive model
  - then apply to unknown instances

Train vs. Test Data

How to Generalize:



- **Split the data**
  - Learn on training data
  - Evaluate performance on testing data
  - Easy to overfit the training data
  - **Care more about test accuracy than train accuracy**
    - **i.e. generalization is key**
- **Soon** – we will add another split to optimize the model
  - Validation set

Hyperparameter Tuning

Fine-tuning the model

- Hold out a **"validation"** set
- Evaluate model with **varying hyperparameters**
  - A hyper-parameter is a parameter of the learning algorithm not of the model

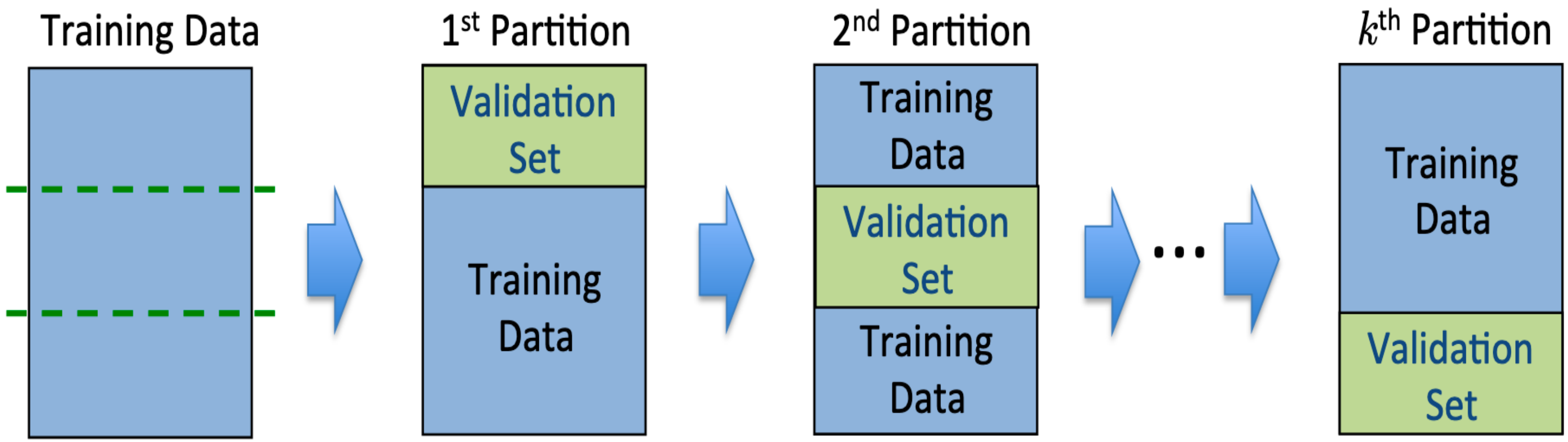
Cross Validation

Generalization of train-test split for more accurate evaluation of **classifier performance**

- Randomly split dataset into **K equal partitions**
- In each fold use **K-1 samples to train**, **leftover to test**

Cross Validation

HPT with Cross Validation



Choose Model Parameter with highest validation performance

Cross Validation

How to tell if a model is

- **too simple or too complex?**

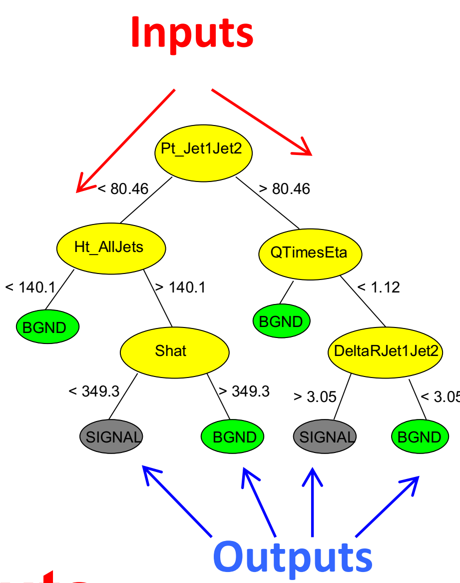
Training Data	Test Data	Model
Bad	Bad	Underfitting
Good	Bad	Overfitting

CONSTRUCTING CLASSIFIERS

Goals

Distinguish f(x), **g(x)** using training set of observations

{**inputs** , **outputs**}



Pass observations to a learning algorithm
neural network, decision tree

that produces **outputs** in response to **inputs**

Use another set of observations to evaluate

Classification

Primary Goal:

Achieve **lowest probability** of error on unseen cases {<$x^{(i)}$, $y^{(i)}$>}

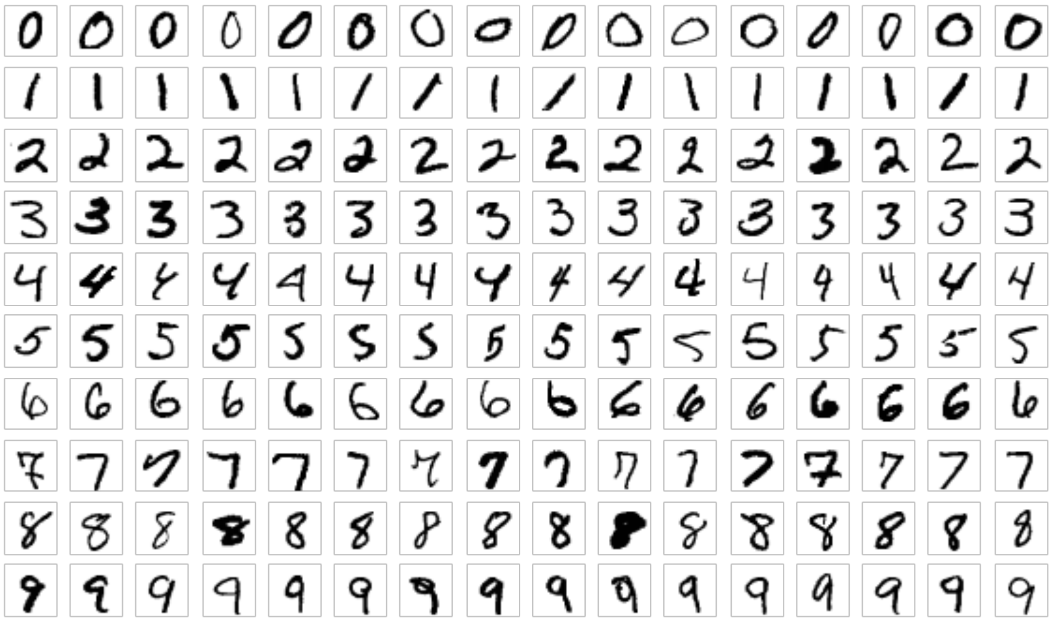
Supervised Approach:

Inductively learn from labeled examples (where classes are known)

MNIST DATASET

70k labeled handwritten digits

- **28 x 28 pixels with intensity [0 – 255]**



Classification Metrics

$$\text{accuracy} = \frac{\#\ \text{correct predictions}}{\#\ \text{test instances}}$$

$$\text{error} = 1 - \text{accuracy}$$

Performance Measures

Accuracy

- limited value if dataset is skewed

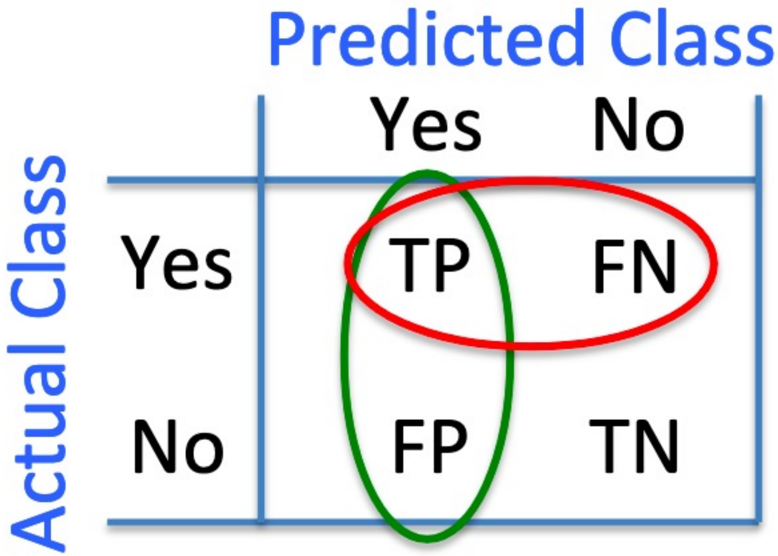
More metrics:

- **MSE** or **RMSE = sqrt(MSE)** for regression
- **Binary cross-entropy (BCE)** for classification

Confusion Matrix

Confusion Matrix

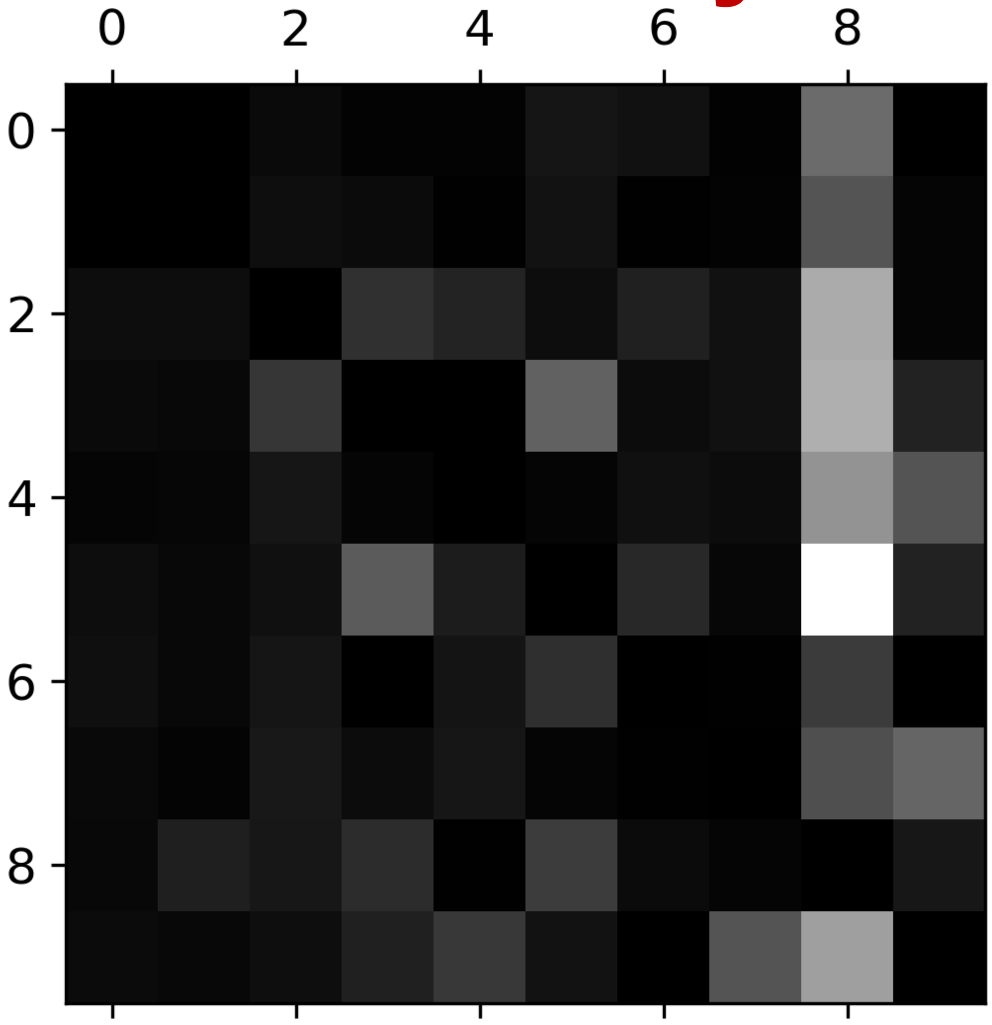
Visualize correct and incorrect classifications



$$\text{accuracy} = \frac{TP + TN}{P + N}$$

P positive, N negative cases

Error Analysis



Precision and Recall

Precision = TP / (TP + FP) (Eqn. 3.1)

Recall = TP/ (TP + FN) (Eqn. 3.2)

TP = True Positive

FP = False Positive

TN = True Negative

FN = False Negative

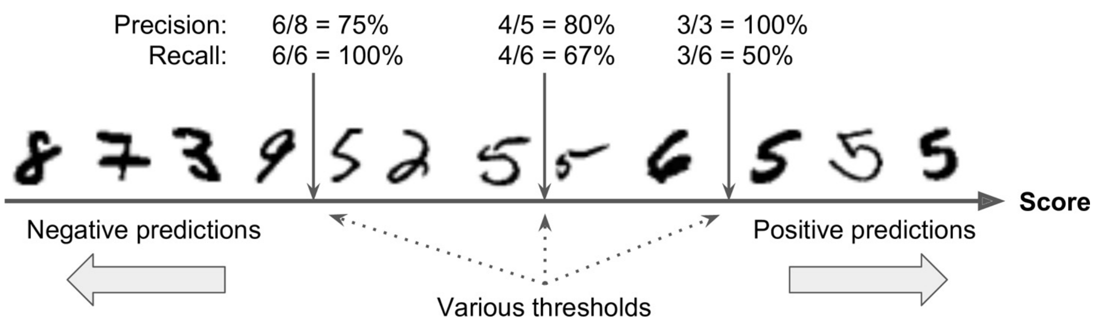
$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

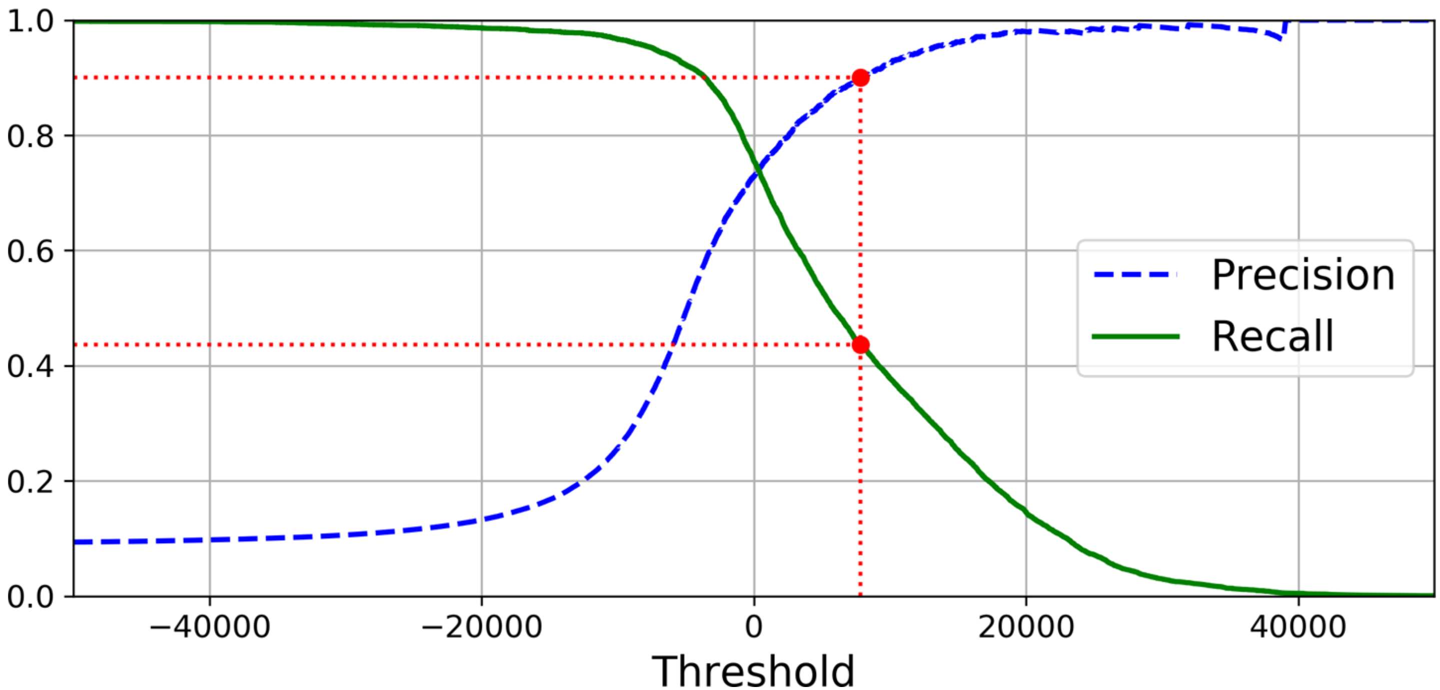
Precision = True Positive / Predicted Positive

Recall = True Positive / Real Positive

Precision and Recall Trade-off



Raise Threshold:
Decrease recall



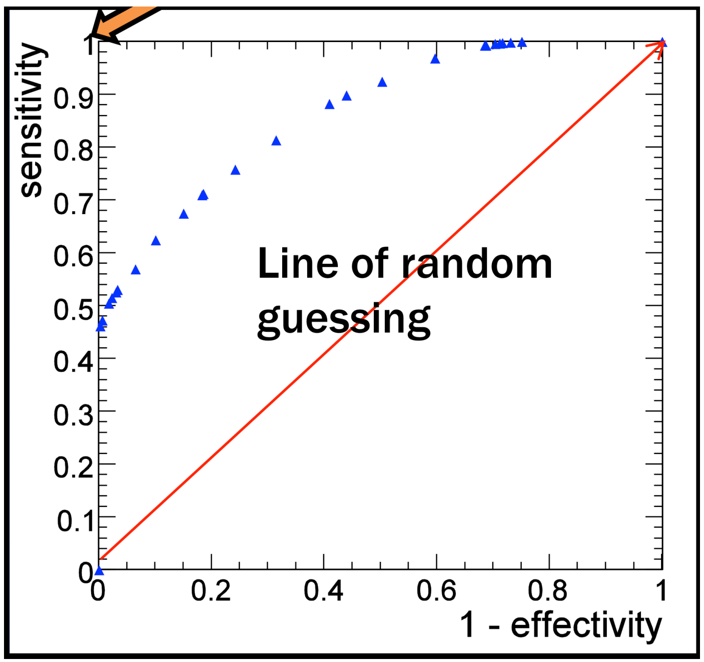
ROC Curve

Receiver Operating Characteristic (ROC)

Commonly used metric

Shows the **relationship** between correctly classified positive cases **TPR (sensitivity)** and incorrectly classified negative cases **FPR (1-effectivity)**

Perfect Classifier



ROC Curve

Machine Learning

Algorithm choice sets hypothesis Class H

- Goal: find the best function within H
  - eg. one that makes the fewest “mistakes”
  - Optimization problem via a learning process
- Evaluate?
  - **Loss (Risk) Function** on training data
  - Many possible loss functions:
    - Squared
    - Absolute - choice depends on the problem!
    - Cross-entropy

Hands-on Activity