Classificazione bio-molecolare di tessuti e geni come problema di apprendimento automatico e validazione dei risultati

Giorgio Valentini

e-mail: valentini@dsi.unimi.it

DSI – Dip. Scienze dell'Informazione
Università degli Studi di Milano

Classificazione bio-molecolare di tessuti e geni

- Diagnosi a livello bio-molecolare:
  - Identificazione e classificazione di pazienti sani e malati.
  - Classificazione di differenti tipologie patologiche
  - Diagnosi basata sulla integrazione di dati biologici eterogenei
- Predizione di esiti clinici:
  - Predizione delle risposte di pazienti a trattamenti farmacologici
  - Sviluppo di tool prognostici per uso clinico e per la risposta a terapie
- Identificazione di molecole target per lo sviluppo di farmaci
- Classificazione di classi funzionali di geni su base bio-molecolare.
- Analisi di qualità di prodotti agro-alimentari.

Classificazione biomolecolare come problema di apprendimento automatico (1)

I dati generati da bio-tecnologie high-throughput (ad es: DNA microarray) sono rappresentabili come insiemi di coppie (**x**,t):

- $\mathbf{x} \in R^d$, $\mathbf{x} = [x_1, x_2, \ldots, x_d]$ rappresenta i livelli di espressione genica di *d* geni
- t rappresenta un particolare stato funzionale

Es: $t \in C = \{ s, m \}$, s → paziente sano, m→malato

Classificazione biomolecolare come problema di apprendimento automatico (2)

Obiettivo dell' apprendimento automatico:

Apprendere la funzione non nota f :
$f : R^d \rightarrow$ C che mappa i livelli di espressione genica $\mathbf{x} \in R^d$
nella corrispondente classe funzionale $t \in C$ (es: paziente sano o malato)

tramite un algoritmo di apprendimento (learning machine) L che utilizza solo un training set $D = \left\{ (x_i, t_i) \right\}_{i=1}^{n}$ di campioni distribuiti in accordo alla distribuzione di probabilità congiunta P(**x**,t).

Algoritmi di apprendimento supervisionato e learning machine

- L' algoritmo di apprendimento *L* genera un' approssimazione $g : R^d \rightarrow C$ della funzione non nota *f* utilizzando il training set *D*: $L(D) \rightarrow g$.
- Si desidera che tale funzione sia la più "vicina" possibile ad *f*
- A tal fine si usa una funzione di perdita Loss*(f(***x**),g(**x**)) che misuri quanto *g* differisca da *f*.
- Nei problemi di classificazione si usa la funzione di perdita 0/1:

$$Loss(g(x), f(x)) = \begin{cases} 1 & se \quad g(x) \neq f(x) \\ 0 & se \quad g(x) = f(x) \end{cases}$$

Ma f non è nota (se lo fosse avremmo risolto il problema) ...

Addestramento delle learning machine

- Nella realtà si dispone spesso solo di un insieme relativamente limitato di dati (ad es: un insieme *D* di dati di espressione genica) e la learning machine viene addestrata ad approssimare *f* utilizzando tali dati come una serie di esempi da apprendere:

$(\mathbf{x}_1,t_1)$, $(\mathbf{x}_2,t_2)$, ... $(\mathbf{x}_n,t_n)$.

- La learning machine verrà addestrata ad apprendere una funzione *g* tale che $g(\mathbf{x}_1)=t_1$, $g(\mathbf{x}_2)=t_2$, ..., $g(\mathbf{x}_n)=t_n$, in modo da minimizzare il rischio empirico R_{emp} rispetto al training set $D=\left\{\left(x_i,t_i\right)\right\}_{i=1}^{n}$.

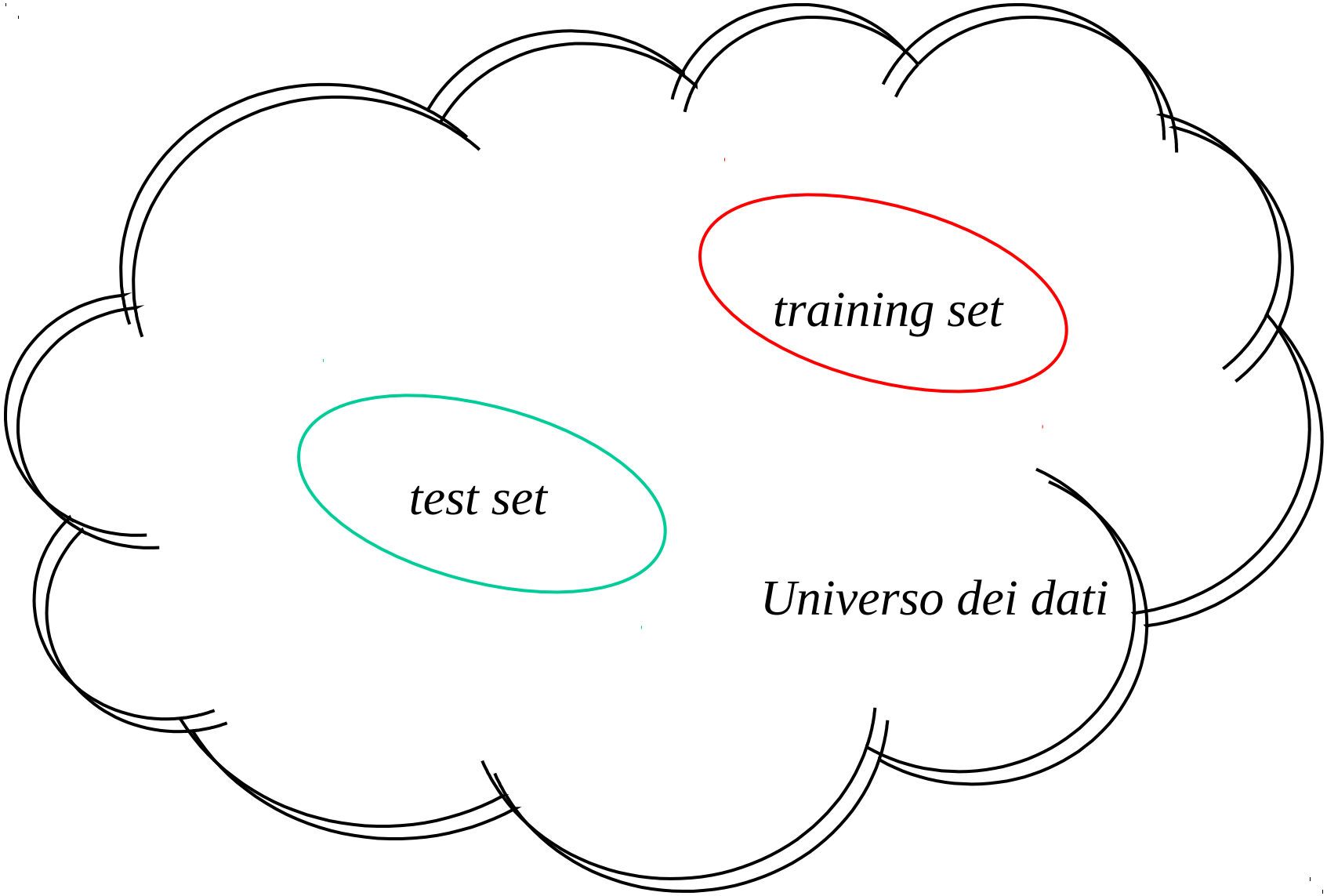
$$R_{emp}=\frac{1}{n}\sum_{i=1}^{n} Loss\left(g\left(x_i\right),t_i\right)$$

Generalizzazione

- L viene addestrata su un *training set* $D \subset U$.
- La learning machine L è utile se può fare delle previsioni sull' Universo non noto U dei dati:

  vogliamo cioè che *generalizzi* correttamente su dati che non conosce.
- A questo fine L deve prevedere correttamente non tanto i dati D su cui è stata addestrata, ma i dati $(\mathbf{x},t) \in U$, $(\mathbf{x},t) \notin D$ che non conosce.
- Siccome di solito non si conosce a priori U o equivalentemente la distribuzione di probabilità congiunta $P_U(\mathbf{x},t)$, le capacità di generalizzazione di L vengono valutate rispetto ad un test set T separato da D, cioè tale che $T \subset U$ e $T \cap D = \emptyset$.

Universo dei dati e campioni



Apprendimento supervisionato

- *Apprendimento da dati "etichettati"*: ciascun campione viene etichettato (ad es: normale o malato)
- *Supervisionato* in quanto un "supervisore" assegna le etichette ai campioni da apprendere: cioè la learning machine è addestrata tramite un insieme di dati etichettati (**x**,*t*)
- La learning machine impara ad associare un determinato campione **x** ad una classe *t*
- L' obiettivo della learning machine consiste nell' *assegnare un' etichetta corretta a campioni la cui classe di appartenenza non è nota a priori* (ad es: deve essere in grado di predire sulla base dei dati di espressione genica se un paziente sia sano o malato)

Obiettivo dell’ apprendimento supervisionato

- Consiste nel predire correttamente la classe si appartenenza di campioni non noti (*generalizzazione*).

La generalizzazione dipende da:

- Accuratezza del classificatore sul training set
- Complessità della funzione computata dal classificatore
- Dimensione training set

In caso i dati siano caratterizzati da ridotta cardinalità e/o elevata dimensionalità, può sorgere il problema di *overfitting* (sovraadattamento)

Esempio: classificazione di tessuti tramite dati di espressione genica

- Campioni di ridotta cardinalità
- Elevata dimensionalità dei dati
- Rumore
- Dati mancanti

Un problema complesso
di apprendimento automatico

Validazione dei risultati e stima della qualità dei classificatori

Rischio atteso e rischio empirico

- L` apprendimento di una funzione non nota $f : R^d \rightarrow$ C avviene tramite un algoritmo L che genera un insieme di funzioni g che approssimano f utilizzando solo un training set $D = \{(x_i, t_i)\}_{i=1}^{n}$ distribuito secondo un distribuzione di probabilità non nota $P(\mathbf{x},t)$:

$$g : \Re^d \times \Omega \rightarrow C$$

Ω rappresenta un insieme di parametri della learning machine (ad es., l' insieme dei pesi delle unità di calcolo di una rete neurale).

• Obiettivo dell' apprendimento non è minimizzare il rischio empirico $R_{emp}(\omega)$:

$$R_{emp}(\omega) = \frac{1}{n} \sum_{i=1}^{n} Loss\left(g(x_i, \omega), t_i\right)$$

bensì il rischio atteso R(ω):

$$R(\omega) = \iint Loss\left(g(x, \omega), t\right) p(x, t) dx \; dt$$

A parte le difficoltà matematiche della minimizzazione del funzionale R(ω), quasi sempre la funzione di densità di probabilità congiunta non è nota ...

Stima del rischio atteso

- Il rischio empirico non sempre converge al rischio atteso.
- La Teoria Statistica dell' Apprendimento di Vapnik ha mostrato che un limite superiore al rischio atteso può essere scomposto in due componenti:

$$R(\omega) \leq R_{emp}(\omega) + \Phi(h/m)$$

dove il primo termine dipende dal rischio empirico, mentre l' intervallo di confidenza Φ dipende principalmente dal rapporto fra la complessità *h* della learning machine e la cardinalità *m* del training set disponibile.

• Per valutare le capacità di generalizzazione delle learning machine è necessario stimare il rischio atteso e non semplicemente il rischio empirico.

• Il problema è: come stimare il rischio atteso ?

Due approcci principali alla stima del rischio atteso

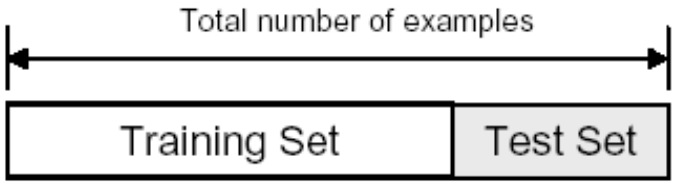
1. Stima teorica dei limiti superiori al rischio atteso (basati sull’ errore empirico e sulla stima della complessità della learning machine)
2. Stima sperimentale (basata sul campionamento dei dati disponibili)

Metodi di stima sperimentale dell' errore di generalizzazione

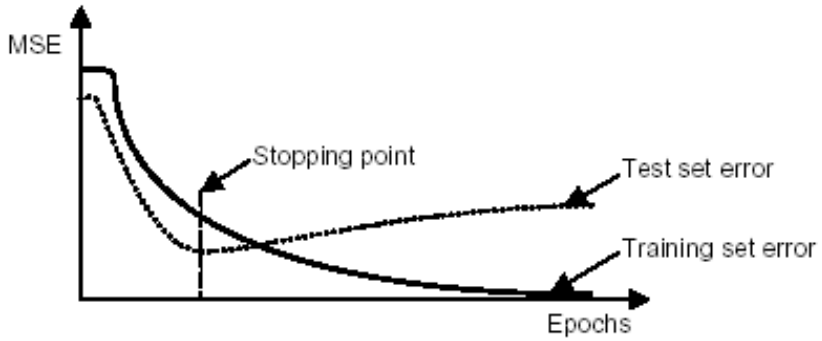
- Holdout
  - Suddivisione dei dati in *training* e *test* set (tipicamente 2/3 ed 1/3)
- Sottocampionamento casuale
  - Holdout ripetuto n volte
- Cross validation
  - Partizione dei dati in k sottoinsiemi disgiunti (fold)
  - k-fold: training con k-1 fold, test sul rimanente; il processo è ripetuto k volte utilizzando ognimvolta come test set un fold differente.
  - Leave-one-out: k = numero dei campioni disponibili
- Bootstrap
  - Campionamento con rimpiazzo
- Metodi out-of-bag
  - Training sui campioni estratti tramite bootstrap e testing sui rimanneti campioni non selezionati. Il proceso è ripetuto n volte.

Metodo di hold-out (1)

- **Split dataset into two groups**
  - Training set: used to train the classifier
  - Test set: used to estimate the error rate of the trained classifier



- **A typical application the holdout method is determining a stopping point for the back propagation error**

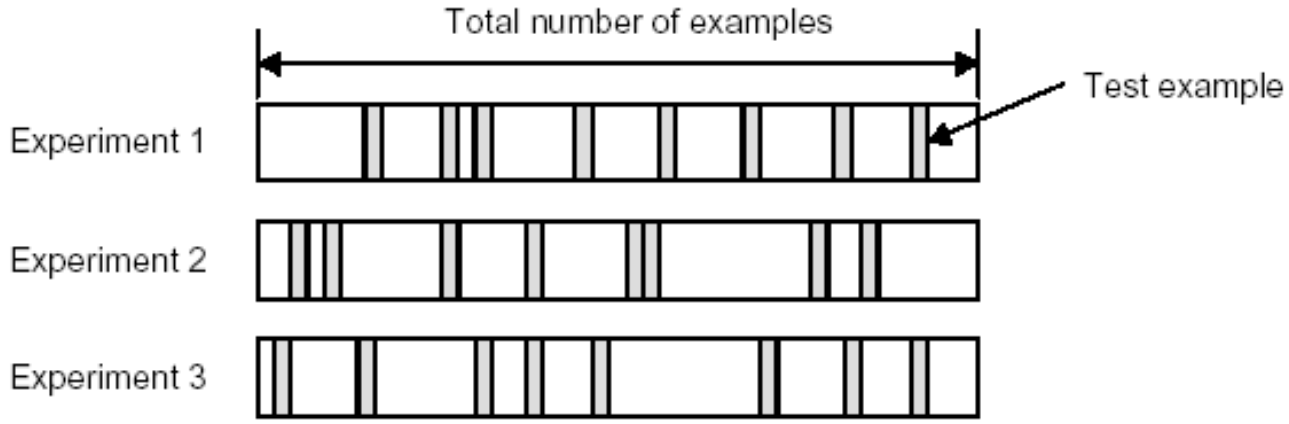


Metodo di hold-out (2)

- **The holdout method has two basic drawbacks**
  - In problems where we have a sparse dataset we may not be able to afford the “luxury” of setting aside a portion of the dataset for testing
  - Since it is a single train-and-test experiment, the holdout estimate of error rate will be misleading if we happen to get an “unfortunate” split
- **The limitations of the holdout can be overcome with a family of resampling methods at the expense of higher computational cost**
  - Cross Validation
    - Random Subsampling
    - K-Fold Cross-Validation
    - Leave-one-out Cross-Validation
  - Bootstrap

Campionamento casuale (holdout ripetuto)

- **Random Subsampling performs K data splits of the entire dataset**
  - Each data split randomly selects a (fixed) number of examples without replacement
  - For each data split we retrain the classifier from scratch with the training examples and then estimate E_i with the test examples

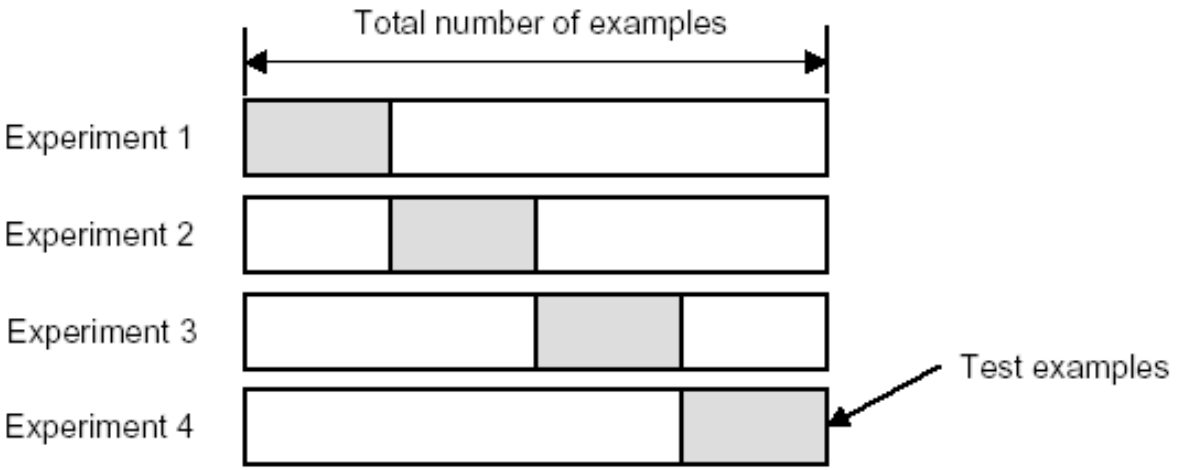


- **The true error estimate is obtained as the average of the separate estimates E_i**
  - This estimate is significantly better than the holdout estimate

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

K-fold cross validation

- **Create a K-fold partition of the the dataset**
  - For each of K experiments, use K-1 folds for training and a different fold for testing
  - This procedure is illustrated in the following diagram for K=4

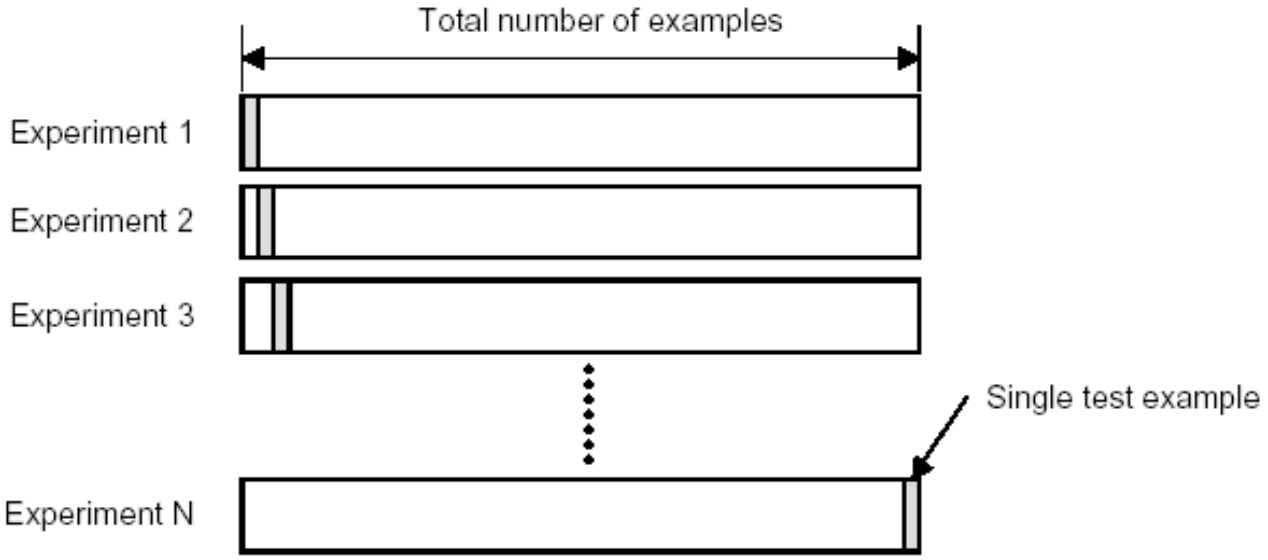


- **K-Fold Cross validation is similar to Random Subsampling**
  - The advantage of K-Fold Cross validation is that all the examples in the dataset are eventually used for both training and testing
- **As before, the true error is estimated as the average error rate on test examples**

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

Leave-one-out

- **Leave-one-out is the degenerate case of K-Fold Cross Validation, where K is chosen as the total number of examples**
  - For a dataset with N examples, perform N experiments
  - For each experiment use N-1 examples for training and the remaining example for testing



- **As usual, the true error is estimated as the average error rate on test examples**

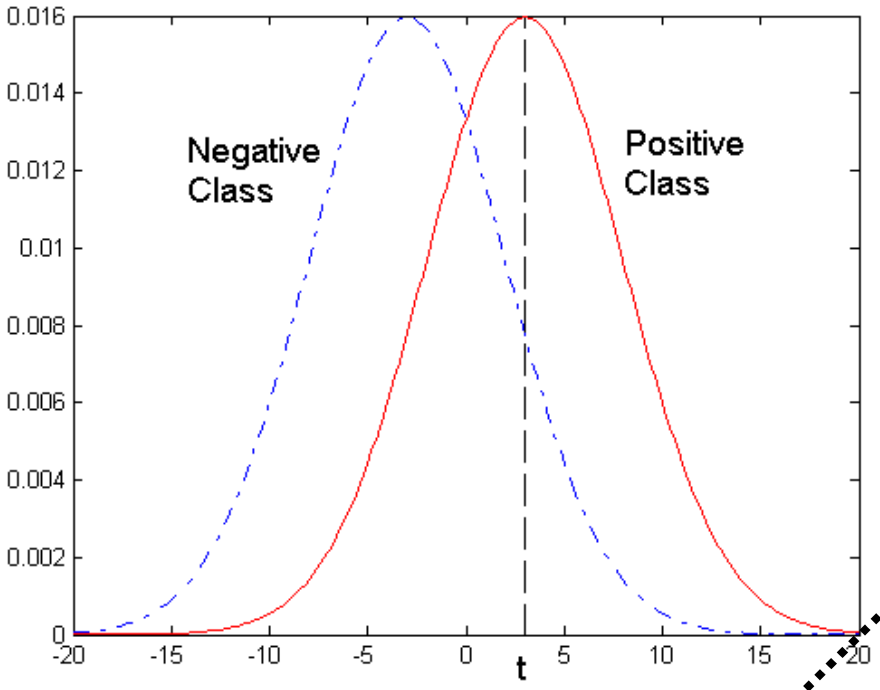
$$E = \frac{1}{N}\sum_{i=1}^{N} E_i$$

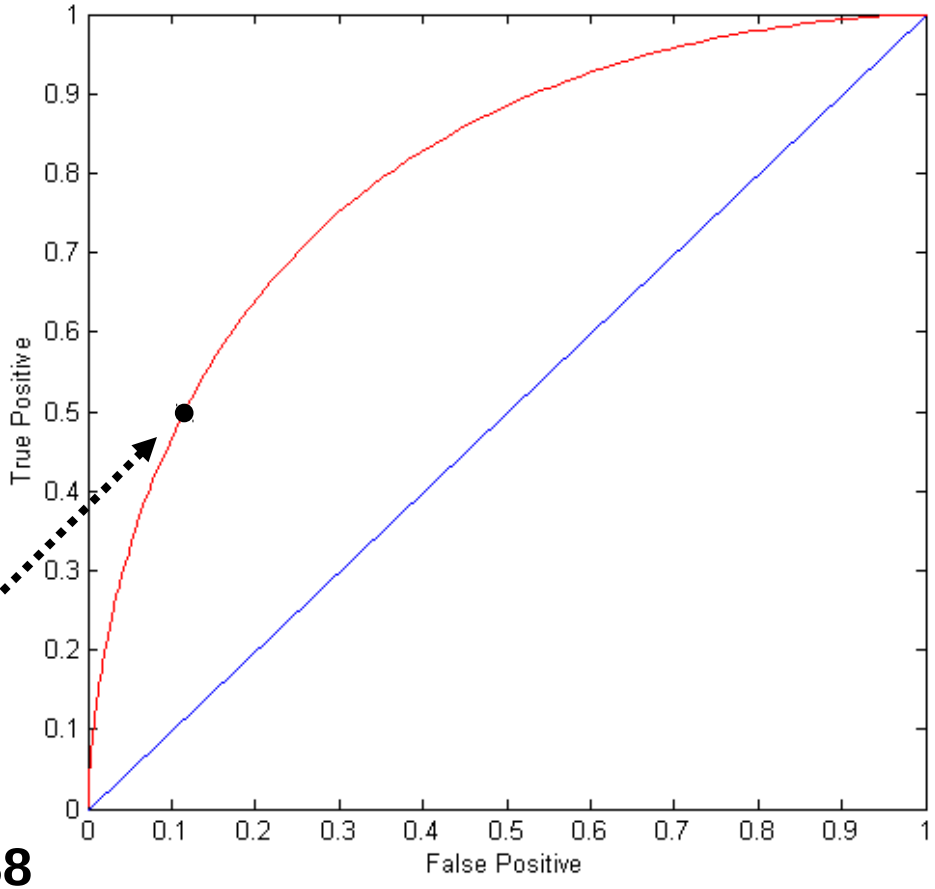
ROC (Receiver Operating Characteristic)

- Developed in 1950s for signal detection theory to analyze noisy signals
  - Characterize the trade-off between positive hits and false alarms
- ROC curve plots TP (on the y-axis) against FP (on the x-axis)
- Performance of each classifier represented as a point on the ROC curve
  - changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point

ROC Curve

- 1-dimensional data set containing 2 classes (positive and negative)
- any points located at x > t is classified as positive





At threshold t:

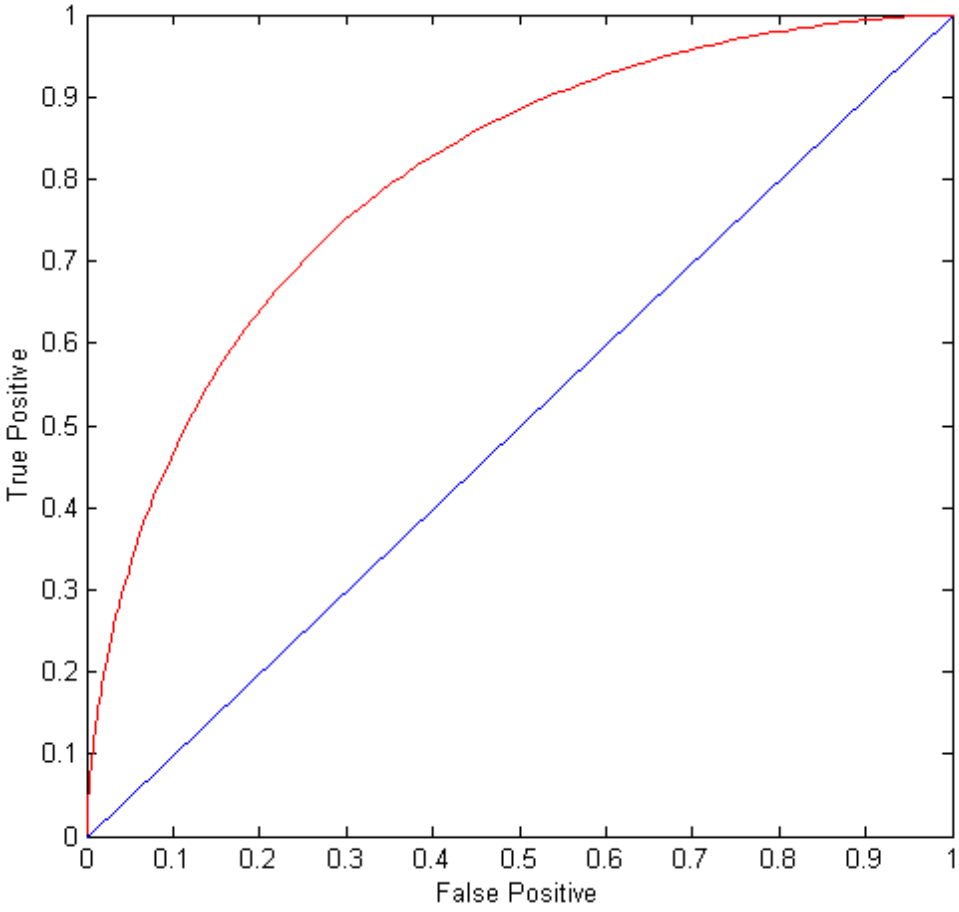
TP=0.5, FN=0.5, FP=0.12, FN=0.88

ROC Curve

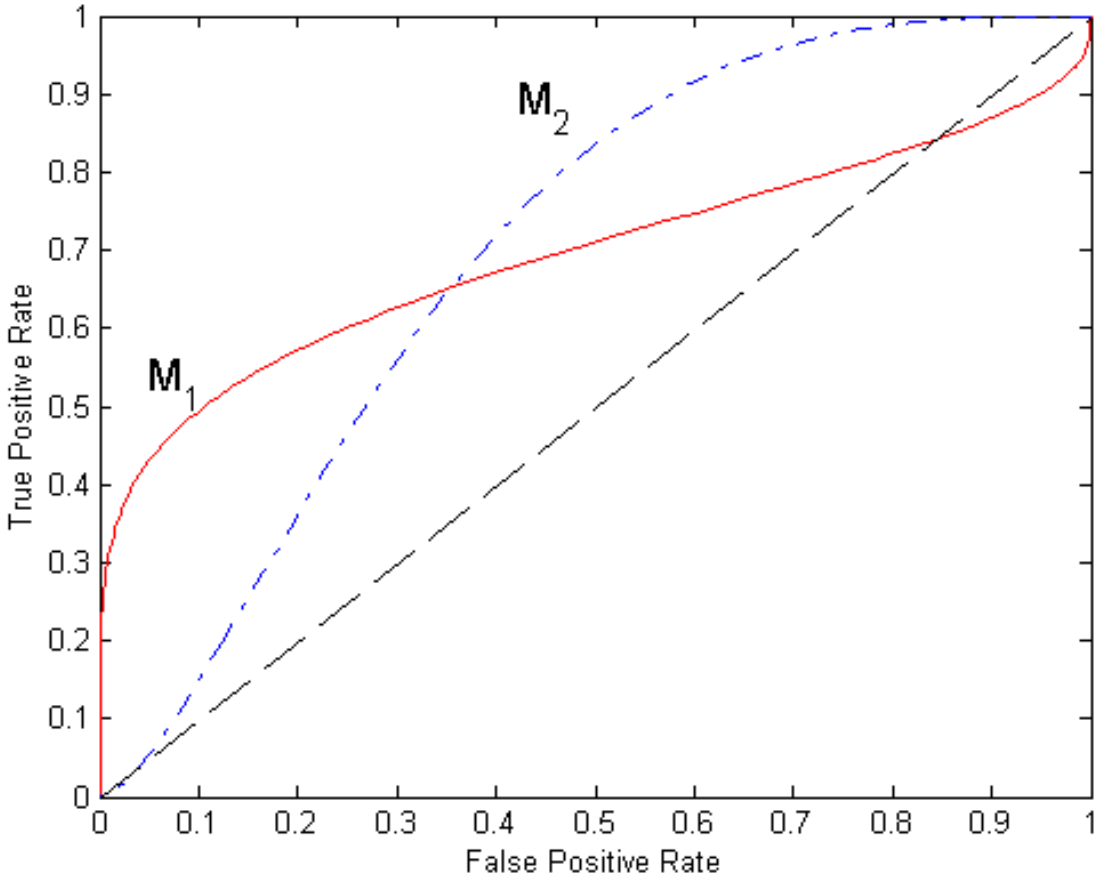
(TP,FP):

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal

- Diagonal line:
  - Random guessing
  - Below diagonal line:
    - prediction is opposite of the true class



Using ROC for Model Comparison



- No model consistently outperform the other
  - M_1 is better for small FPR
  - M_2 is better for large FPR
- Area Under the ROC curve
  - Ideal:
    - Area = 1
  - Random guess:
    - Area = 0.5