Linear Models and Metrics (Supervised Learning)

David Li

Outline

- Linear Models for Regression
- Linear Models for Classification
- Evaluating Predictors

Applications

- Unsupervised Learning
- **Supervised Learning**
  - Classification
  - Regression

Supervised learning

Build a system that can:

- Predict housing price from:
  - House size, lot size, rooms, neighborhood, ...
- Predict weight from:
  - Gender, height, ethnicity, ...
- Predict life expectancy increase from:
  - Medication, disease state, ...
- Predict crop yield from:
  - Precipitation, fertilizer, temperature, ...
- **Hotdog/Not-hotdog classifier**
  - https://www.youtube.com/watch?v=AJsOA4Zl6Io

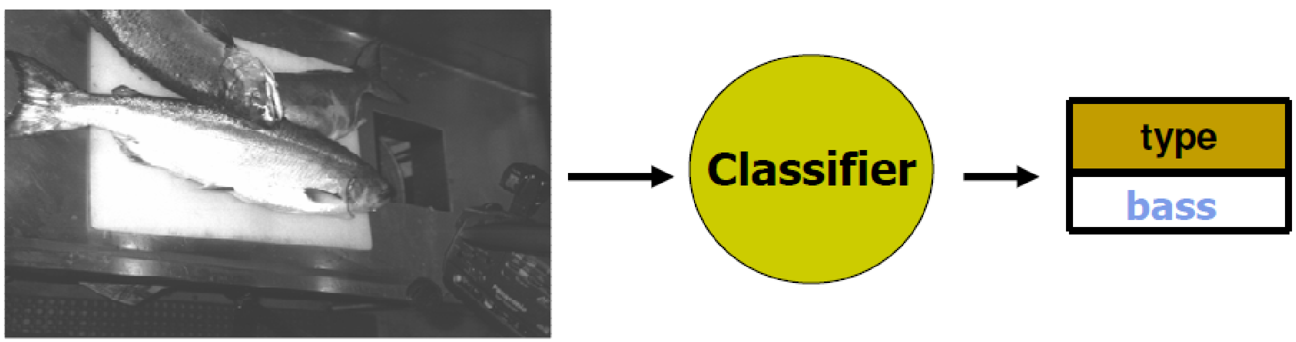
An example: Fish Classifier

Sort Fish

into Species

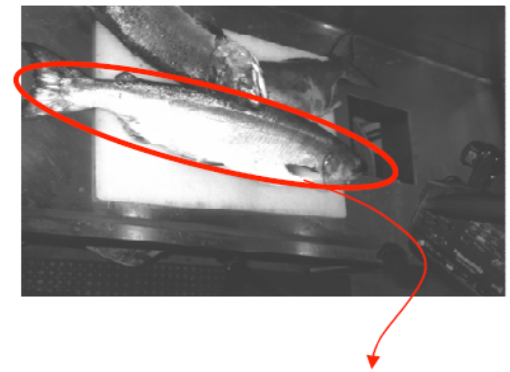
- **Sea bass**
- **Salmon**

using optical sensing

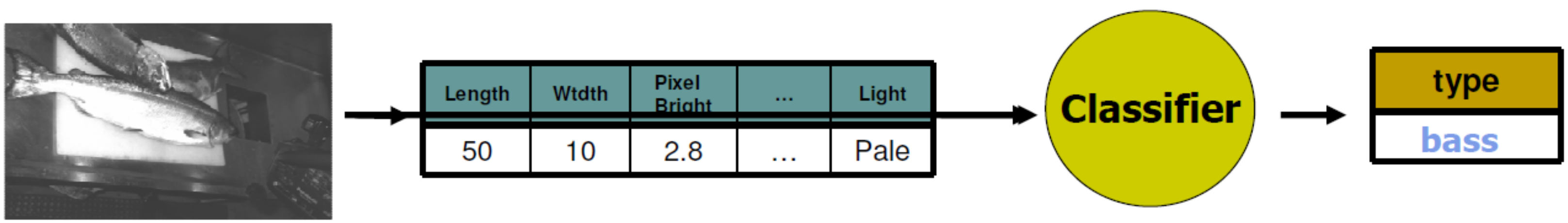


Problem Analysis

- Extract *features* from sample images:
  - Length
  - Width
  - Average pixel brightness
  - Number and shape of fins
  - Position of mouth
  - …



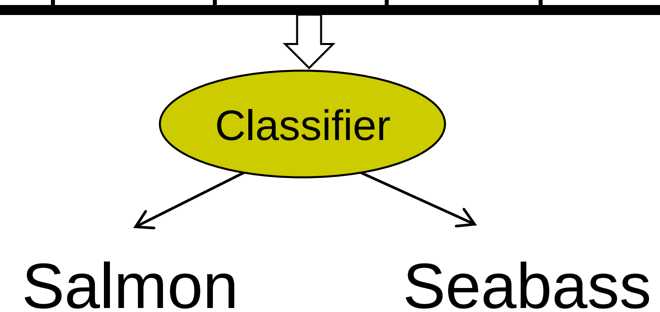
[L=50, W=10, PB=2.8, #fins=4, MP=(5,53), …]



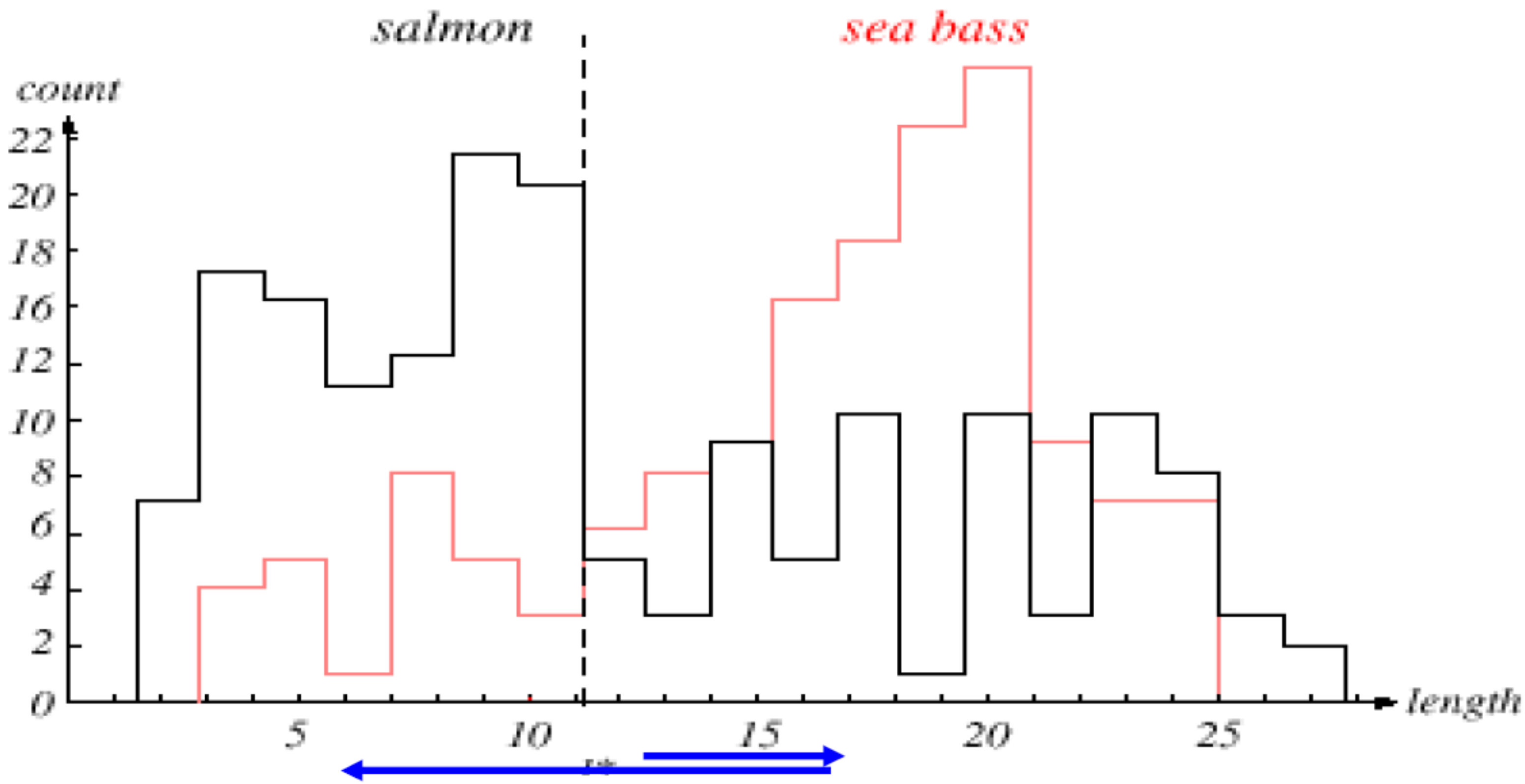
Preprocessing

- Use *segmentation to isolate (remove noise)*
  - fish from background
  - fish from one another
- Send info about each single fish to *feature extractor,*
  - compresses data into small set of features (*feature selection*)
- Classifier sees these features

Length	Wtdth	Pixel Bright	...	Light
50	10	2.8	...	Pale

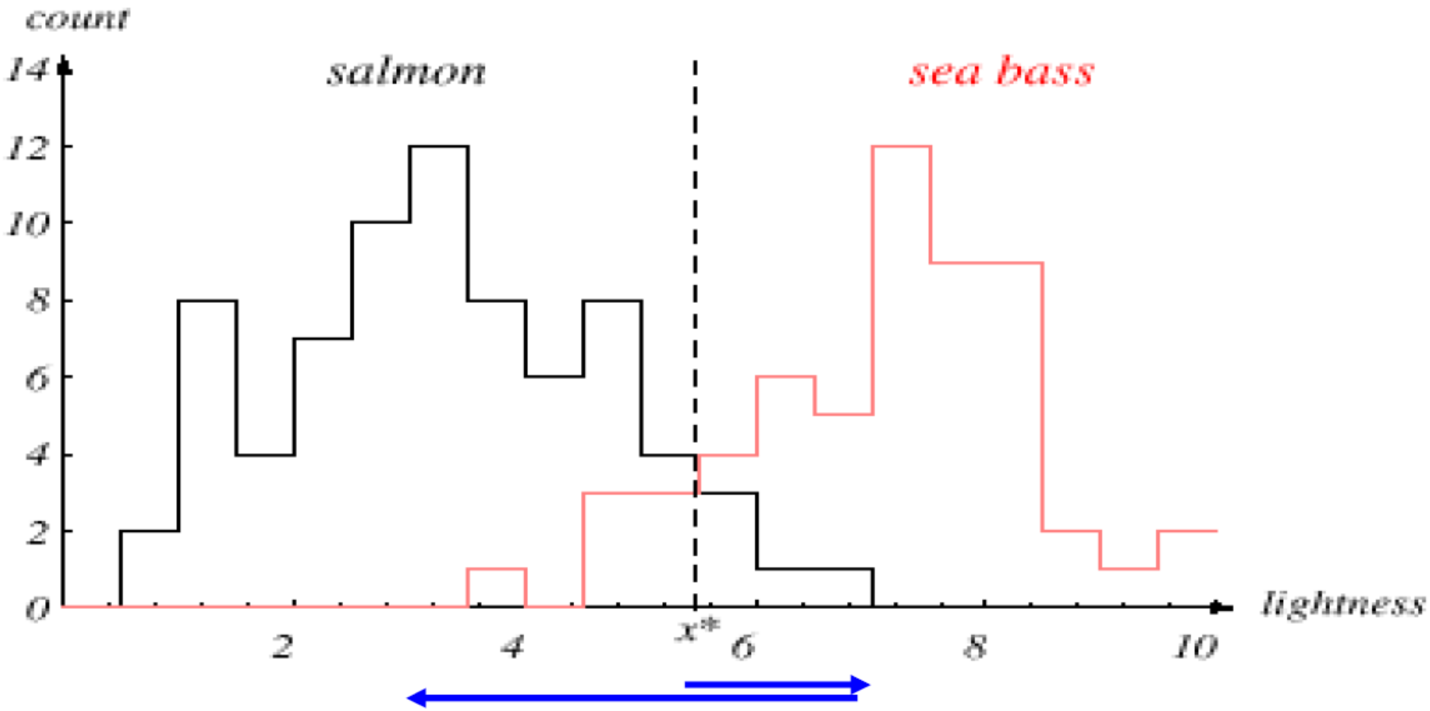


Use “Length”？



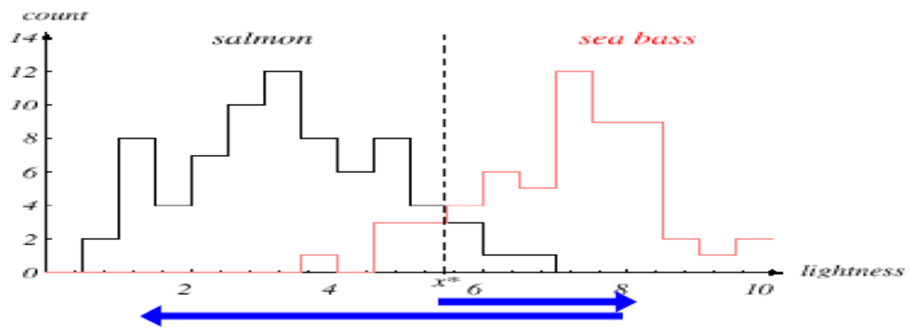
- Problematic… many incorrect classifications

Use "Lightness" ?



- Better… fewer incorrect classifications
- Still not perfect

Where to place boundary?



- *Salmon Region* intersects *SeaBass Region*
  - ⇒ So no "boundary" is perfect
  - *Smaller* boundary ⇒ fewer SeaBass classified as Salmon
  - *Larger* boundary ⇒ fewer Salmon classified as SeaBass
- Which is best… depends on misclassification costs

Task of decision theory

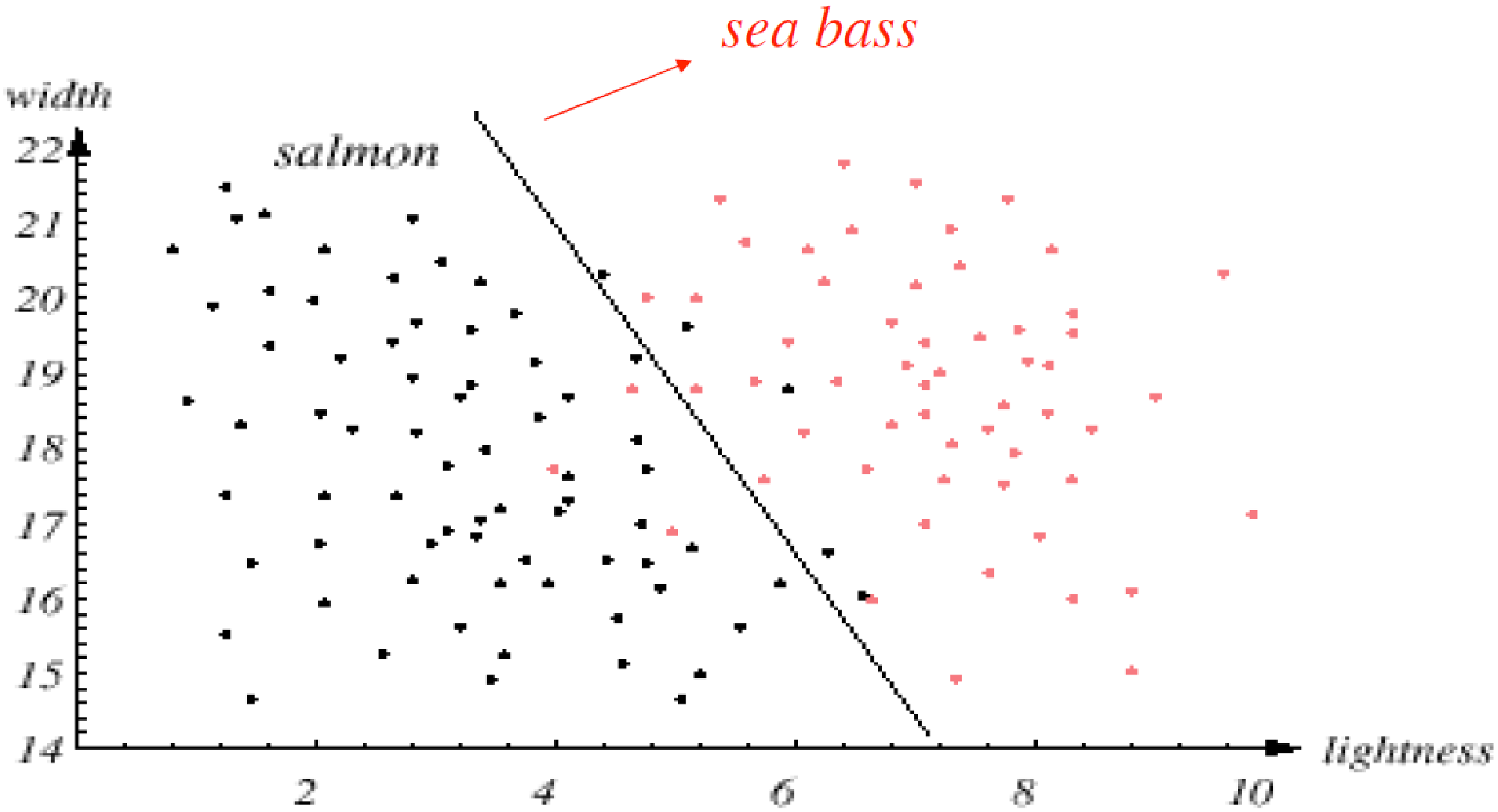
How about 2 features?

- Use *lightness* and *width* of fish

Fish $\Rightarrow$ $x^T = [x_1, x_2]$

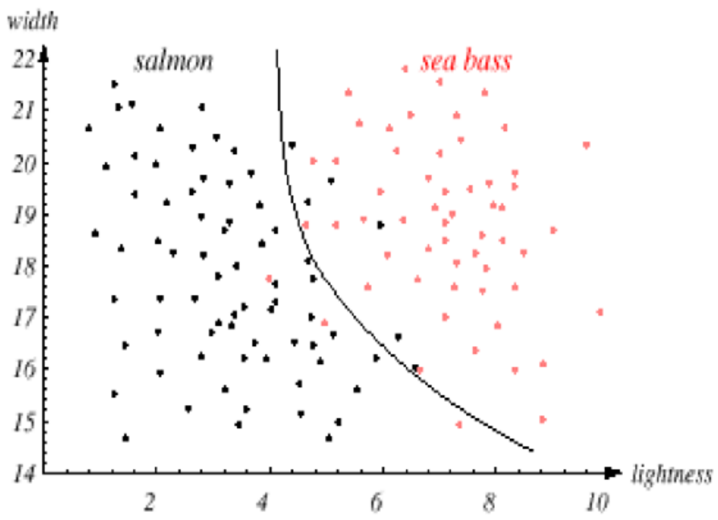
Lightness Width

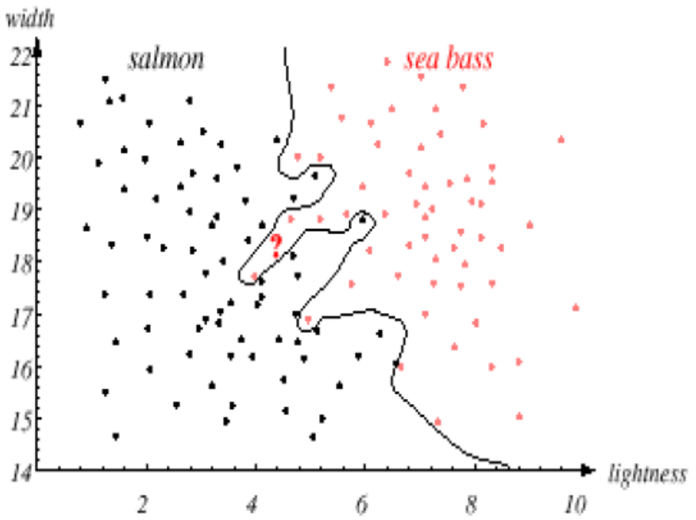
Use Simple Line ?



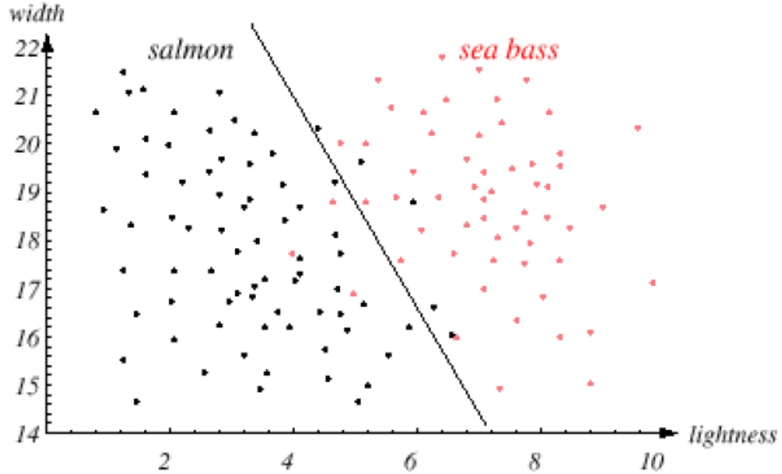
How to produce Better Classifier?

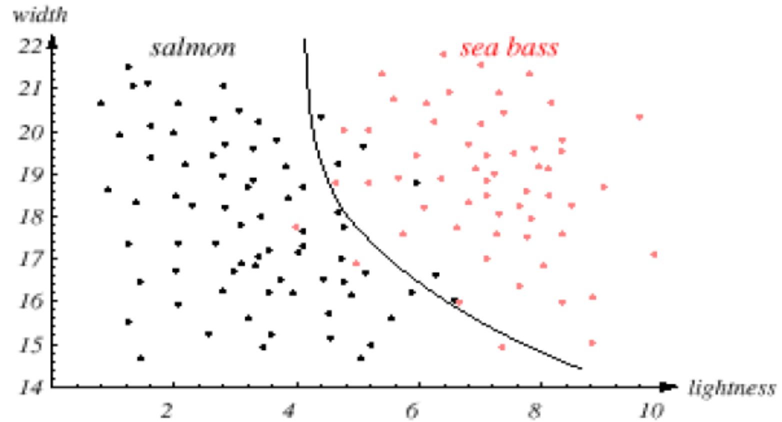
- Perhaps add other features?
  - Best: not correlated with current features
  - Warning: "noisy features" will ***reduce performance***
- Best decision boundary = one that provides optimal performance
  - Not necessarily LINE
  - For example ...

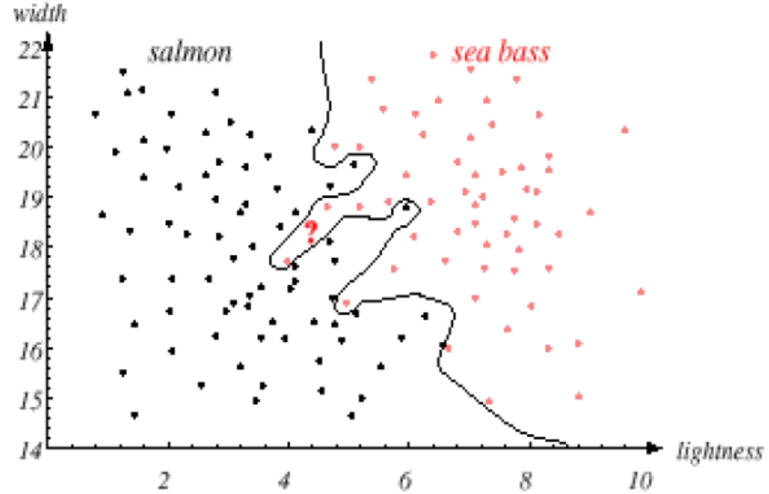




Comparison… wrt NOVEL Fish







Objective: Handle Novel Data

- Goal:
  - Optimal performance on *NOVEL* data
  - Performance on TRAINING DATA

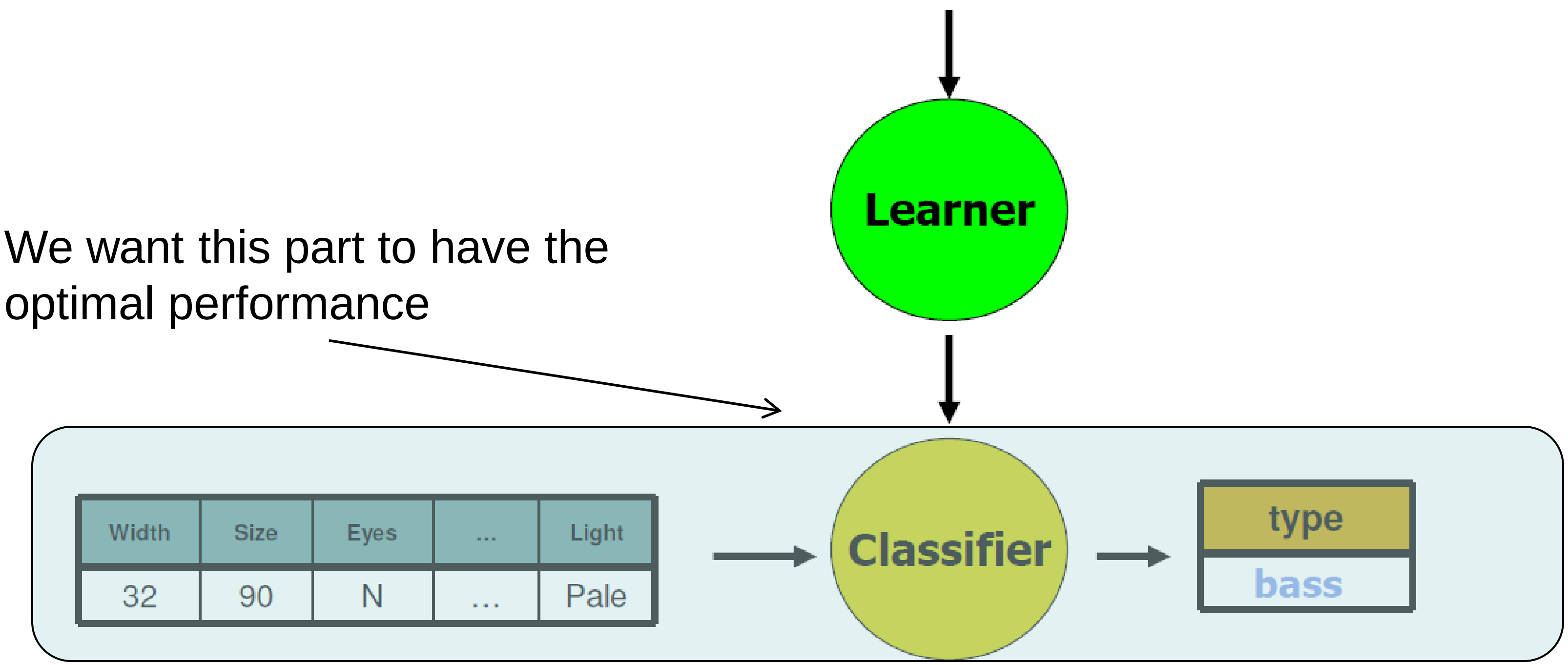
    $\neq$

    Performance on NOVEL data

Issue of generalization!

Training a Classifier

Width	Size.	Eyes	…	Light	type
35	95	Y	…	Pale	bass
22	110	N	…	Clear	salmon
:	:			:	:
10	87	N	…	Pale	bass



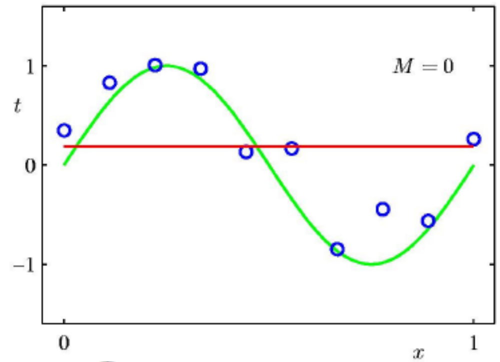
Major Steps of Machine Learning

- Data collection
  - Need set of examples for <u>training</u> and <u>testing</u> the system
  - sufficiently large # of instances
  - Representative
- Feature Choice
  - Depends on characteristics of problem domain
  - Ideally…
    - Simple to extract
    - Invariant to irrelevant
    - Transformation
    - Insensitive to noise
- Model Choice
- Training
- Evaluation

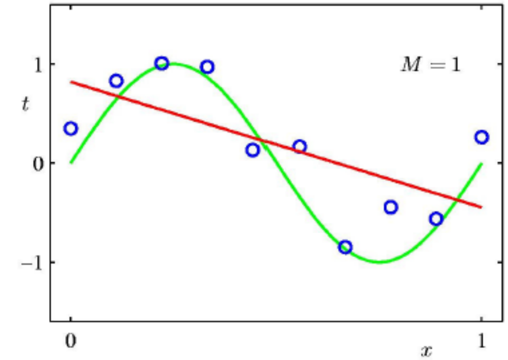
Major Steps of supervised learning

- Data collection
- Feature Choice
- Model Choice
- Training
- Evaluation

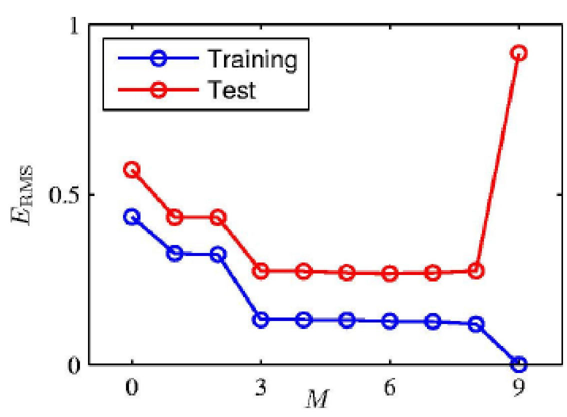
Which model?

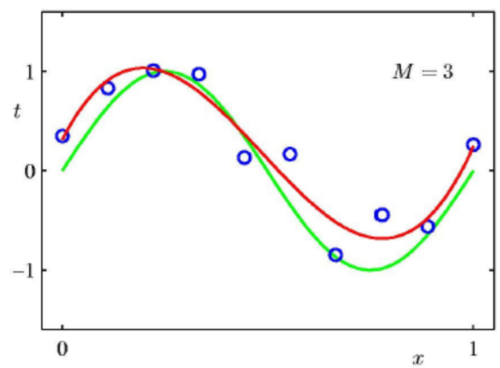


Constant (0)

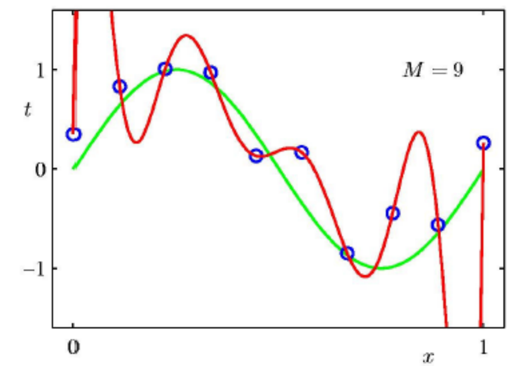


Linear (1)





Cubic (3)



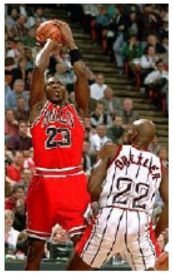
9th degree

Real-world Applications (1)

- to find ideal customers
  - Credit Card approval (AMEX)
    - Humans $\approx$50%; ML is >70% !

- to find best person for job
  - Telephone Technician Dispatch [Danyluk/Provost/Carr 02]
    - BellAtlantic used ML to learn rules to decide which technician to dispatch
    - Saved $10+ million/year

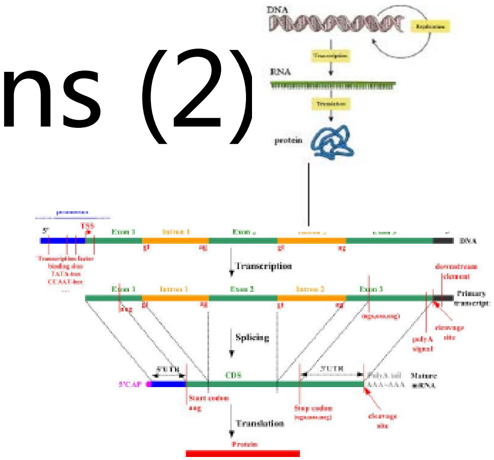
- to predict purchasing patterns
  - Victoria Secret (stocking)



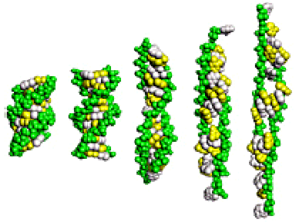
- to help win games
  - NBA (scouting)

- to catalogue celestial objects [Fayyad et al. 93]
  - Discovered 22 new quasars
  - >92% accurate, over tetrabytes

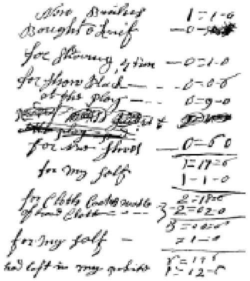
Real-world Applications (2)



- **BioInformatics 1:** identifying genes
  - Glimmer [Delcher et al, 95]
  - identifies 97+% of genes, automatically!
- **BioInformatics 2:** Predicting protein function, …



- **Recognizing Handwriting**



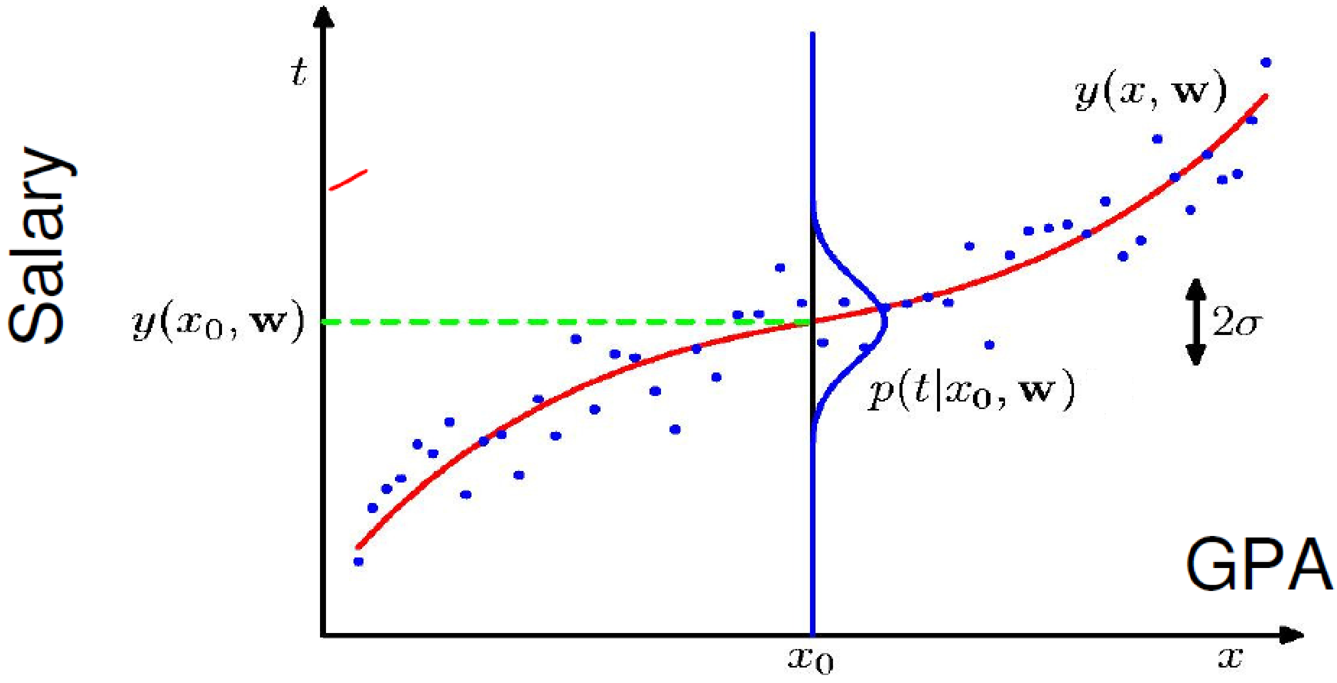
- **Recognizing Spoken Words**
  - **"How to wreck a nice beach"**

Outline

- Linear Models for Regression
- Linear Models for Classification
- Evaluating Predictors

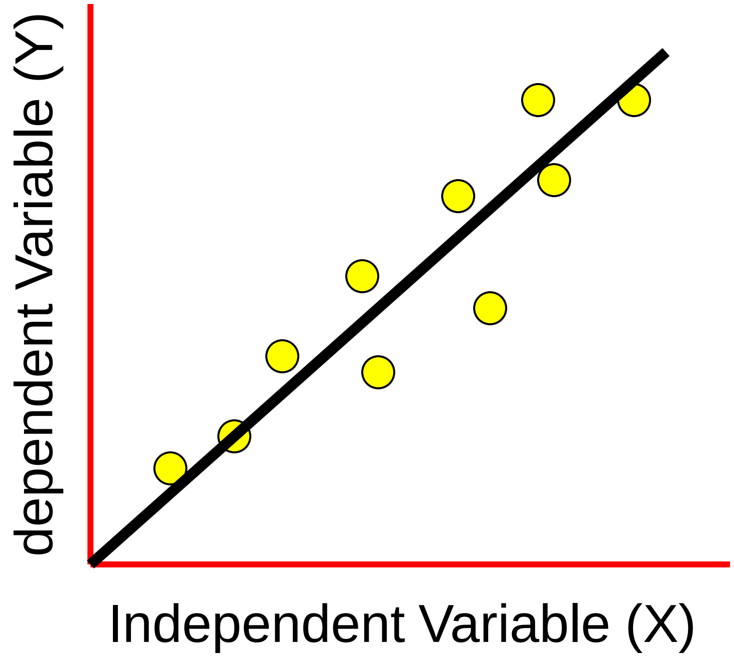
Prediction of Continuous Variables

- Predict a **continuous** variable based on set of inputs:
  - Eg, predict salaries from GPA
  - Regression

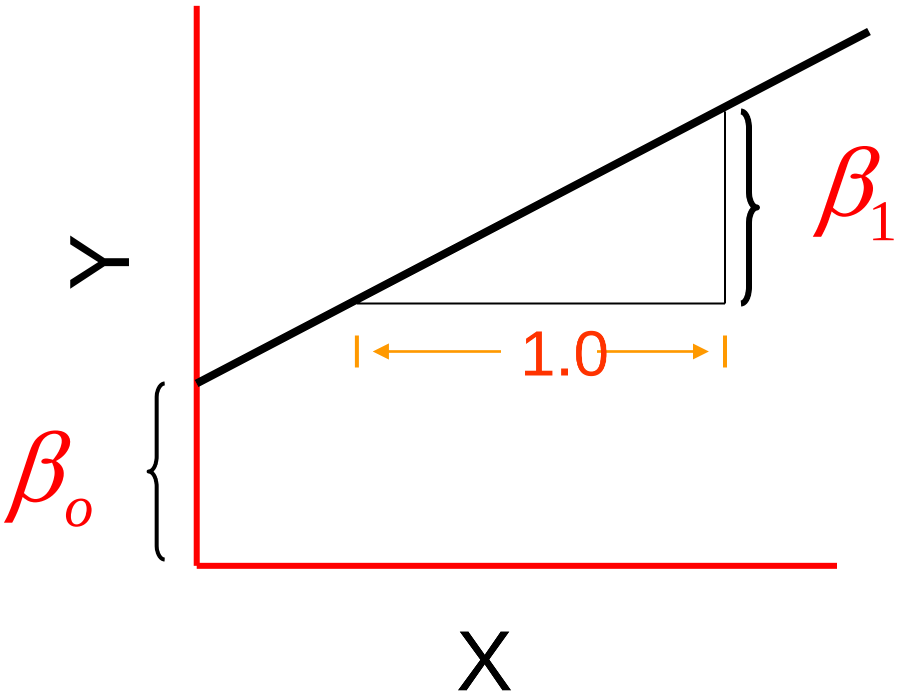


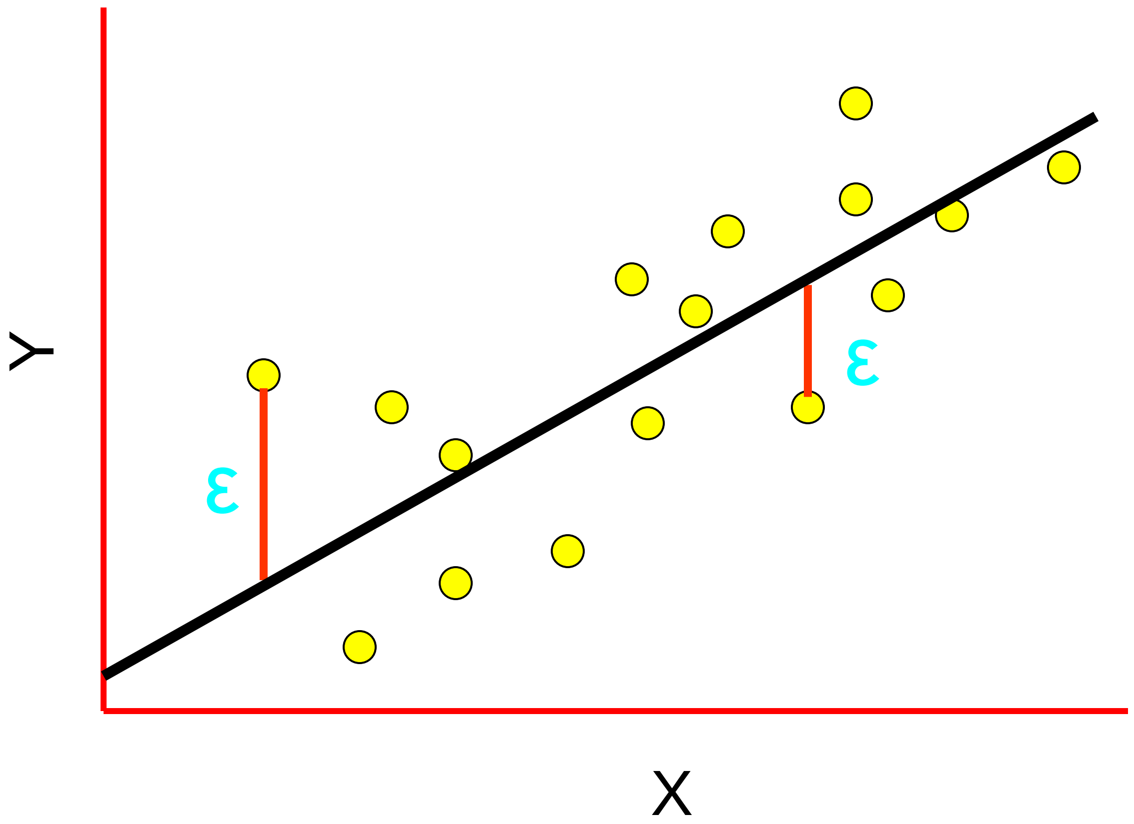
Simple Linear Regression (SLR)

Simple linear regression describes the linear relationship between a predictor variable, plotted on the *x*-axis, and a response variable, plotted on the *y*-axis



$$Y = \beta_o + \beta_1 X$$




ε
ε
Y
X

Fitting data to a linear model

$$Y_i = \beta_o + \beta_1 X_i + \varepsilon_i$$

intercept slope residuals

How to fit data to a linear model?

The Ordinary Least Square Method (OLS)

Least Squares Regression

Model line: $Y' = \beta_0 + \beta_1 X$

Residual (ε) = $Y - Y'$

Sum of squares of residuals = $\sum (Y - Y')^2$

- we must find values of β_o and β_1 that minimise

$$\min \sum (Y - Y')^2$$

Multiple Linear Regression (MLR)

The linear model with a single predictor variable *X* can easily be extended to two or more predictor variables.

$$Y = \beta_o + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \varepsilon$$

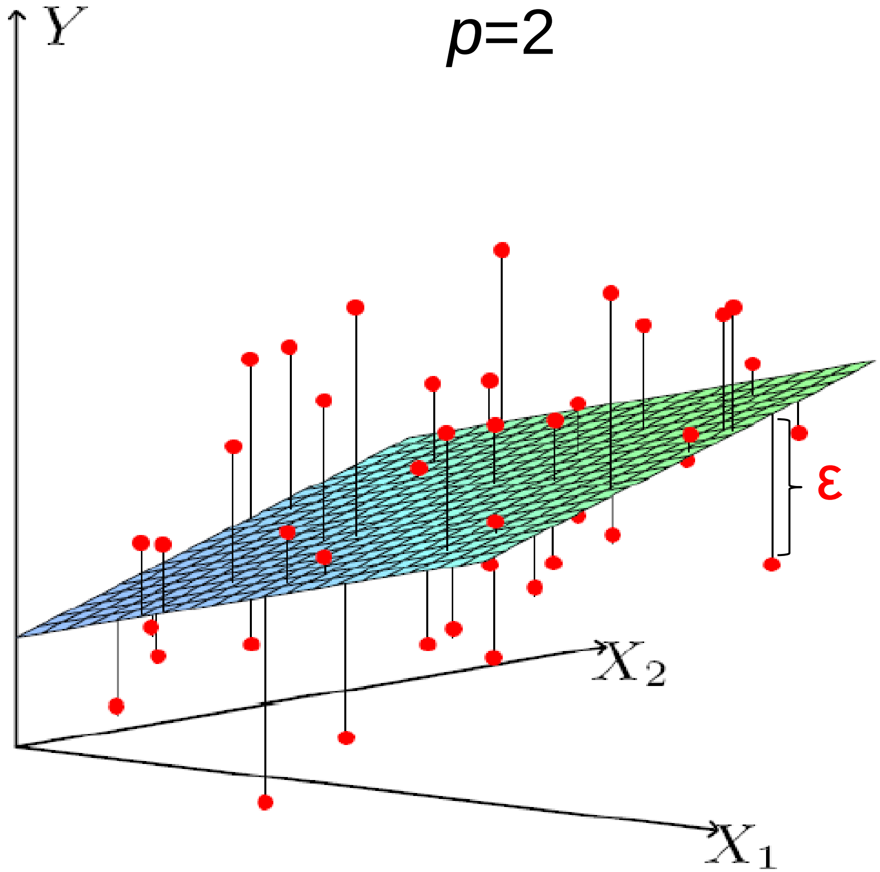
$$Y = \beta_o + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \varepsilon$$

intercept

Regression Coefficients

residuals

p=2

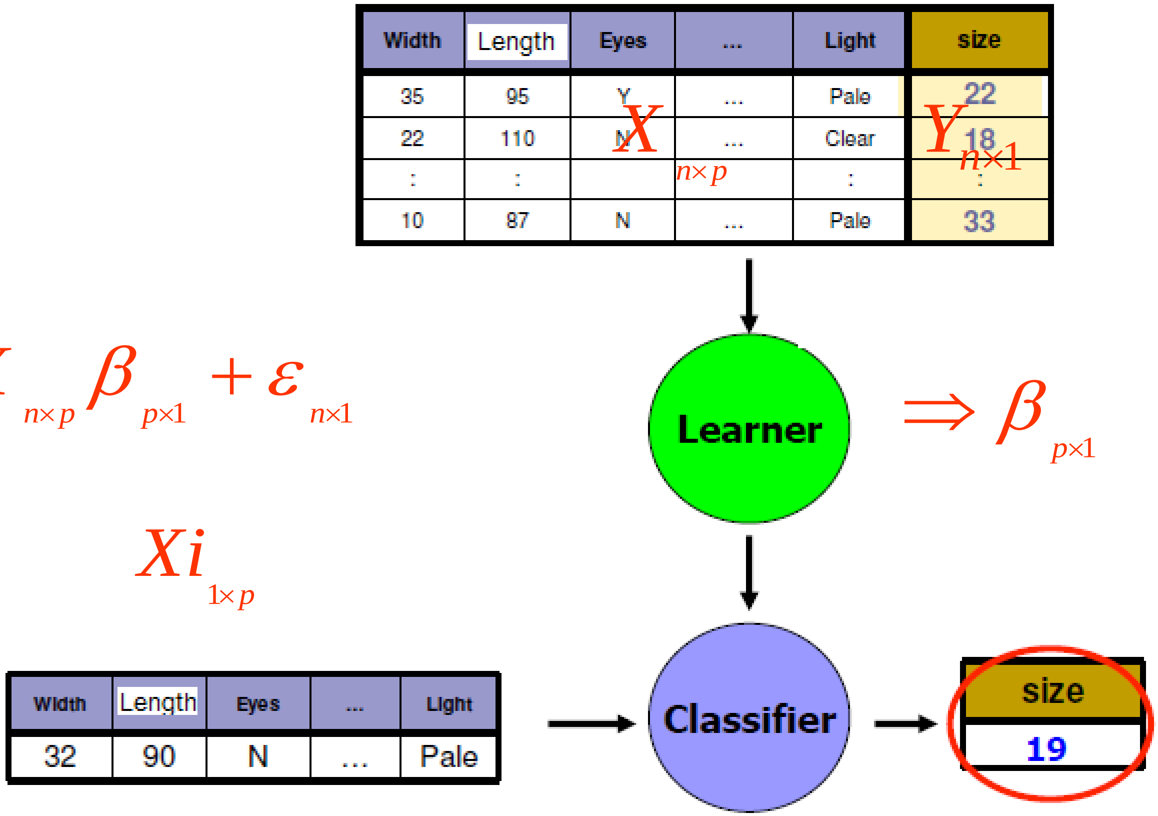


$$Y_{n\times 1} = X_{n\times p}\,\beta_{p\times 1} + \varepsilon_{n\times 1}$$

Training a Regressor

$$Y = \beta_o + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \varepsilon$$

$$Y_{n\times 1} = X_{n\times p}\,\beta_{p\times 1} + \varepsilon_{n\times 1}$$



Best Values of β?

- Again OLS: minimize residual sum-of-squares (RSS)

$$\boldsymbol{\beta} = \arg\min \sum_{i=1}^{n} (Y_i \ - (\beta_0 + \beta_1 X_{i1} + \ldots + \beta_1 X_{ip}))^2$$

$$= \arg\min(\ \mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) = \arg\min\ \mathrm{RSS}(\boldsymbol{\beta})$$

This is a quadratic function in the $p+1$ parameters. Differentiating with respect to β we obtain

$$\begin{aligned} \frac{\partial \mathrm{RSS}}{\partial \beta} &= -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) \\ \frac{\partial^2 \mathrm{RSS}}{\partial \beta \partial \beta^T} &= 2\mathbf{X}^T\mathbf{X}. \end{aligned} \tag{3.4}$$

Assuming (for the moment) that $\mathbf{X}$ has full column rank, and hence $\mathbf{X}^T\mathbf{X}$ is positive definite, we set the first derivative to zero

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) = 0 \tag{3.5}$$

to obtain the unique solution

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{3.6}$$

The matrix algebra of

Ordinary Least Square

Intercept and Slopes:

$$\beta = (X'X)^{-1} X'Y$$

Predicted Values:

$$Y' = X\beta$$

Residuals:

$$Y - Y'$$

$$\beta = (X'X)^{-1}X'Y$$

$$\begin{array}{c} \\ \\ \\ \end{array} X_0$$

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 50 \\ 1 & 8 & 110 \\ 1 & 11 & 120 \\ 1 & 10 & 550 \\ 1 & 8 & 295 \\ 1 & 4 & 200 \\ 1 & 2 & 375 \\ 1 & 2 & 52 \\ 1 & 9 & 100 \\ 1 & 8 & 300 \\ 1 & 4 & 412 \\ 1 & 11 & 400 \\ 1 & 12 & 500 \\ 1 & 2 & 360 \\ 1 & 4 & 205 \\ 1 & 4 & 400 \\ 1 & 20 & 600 \\ 1 & 1 & 585 \\ 1 & 10 & 540 \\ 1 & 15 & 250 \\ 1 & 15 & 290 \\ 1 & 16 & 510 \\ 1 & 17 & 590 \\ 1 & 6 & 100 \\ 1 & 5 & 400 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 9.95 \\ 24.45 \\ 31.75 \\ 35.00 \\ 25.02 \\ 16.86 \\ 14.38 \\ 9.60 \\ 24.35 \\ 27.50 \\ 17.08 \\ 37.00 \\ 41.95 \\ 11.66 \\ 21.65 \\ 17.89 \\ 69.00 \\ 10.30 \\ 34.93 \\ 46.59 \\ 44.88 \\ 54.12 \\ 56.63 \\ 22.13 \\ 21.15 \end{bmatrix}$$

$$\beta = (\underline{X'X})^{-1}\underline{X'Y}$$

The $\mathbf{X}'\mathbf{X}$ matrix is

$$\mathbf{X}'\mathbf{X} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 2 & 8 & \cdots & 5 \\ 50 & 110 & \cdots & 400 \end{bmatrix} \begin{bmatrix} 1 & 2 & 50 \\ 1 & 8 & 110 \\ \vdots & \vdots & \vdots \\ 1 & 5 & 400 \end{bmatrix} = \begin{bmatrix} 25 & 206 & 8{,}294 \\ 206 & 2{,}396 & 77{,}177 \\ 8{,}294 & 77{,}177 & 3{,}531{,}848 \end{bmatrix}$$

and the $\mathbf{X}'\mathbf{y}$ vector is

$$\mathbf{X}'\mathbf{y} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 2 & 8 & \cdots & 5 \\ 50 & 110 & \cdots & 400 \end{bmatrix} \begin{bmatrix} 9.95 \\ 24.45 \\ \vdots \\ 21.15 \end{bmatrix} = \begin{bmatrix} 725.82 \\ 8{,}008.37 \\ 274{,}811.31 \end{bmatrix}$$

The least squares estimates are

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

$$\beta = (X'X)^{-1} X'Y$$

or

$$\begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \end{bmatrix} = \begin{bmatrix} 25 & 206 & 8{,}294 \\ 206 & 2{,}396 & 77{,}177 \\ 8{,}294 & 77{,}177 & 3{,}531{,}848 \end{bmatrix}^{-1} \begin{bmatrix} 725.82 \\ 8{,}008.37 \\ 274{,}811.31 \end{bmatrix}$$

$$= \begin{bmatrix} 0.214653 & -0.007491 & -0.000340 \\ -0.007491 & 0.001671 & -0.000019 \\ -0.000340 & -0.000019 & +0.0000015 \end{bmatrix} \begin{bmatrix} 725.82 \\ 8{,}008.47 \\ 274{,}811.31 \end{bmatrix} = \begin{bmatrix} 2.26379143 \\ 2.74426964 \\ 0.01252781 \end{bmatrix}$$

Therefore, the fitted regression model with the regression coefficients rounded to five decimal places is

$$\hat{y} = 2.26379 + 2.74427x_1 + 0.01253x_2$$

Why minimize OLS?

- Observed value is

$$Y = \sum_{j=1}^{p} \beta_j X_j + \varepsilon$$

- **Model:** assume

$$Y - \sum_{j=1}^{p} \beta_j X_j = \varepsilon \sim N(0, \sigma^2)$$

So,

$$Y \sim N\left(\sum_{j=1}^{p} \beta_j X_j, \sigma^2\right) \Rightarrow$$

$$\Pr(Y \mid X; \boldsymbol{\beta}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(Y - \sum_{j=1}^{p} \beta_j X_j)^2}{2\sigma^2}}$$

Minimizer of OLS is MLE

- Find Most Likely values of β **(MLE)**:

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} \prod_{i=1}^{n} \Pr(Y_i \mid \mathbf{X}_i; \boldsymbol{\beta}) = \arg\max_{\boldsymbol{\beta}} \log \prod_{i=1}^{n} \Pr(Y_i \mid \mathbf{X}_i; \boldsymbol{\beta})$$

$$= \arg\max_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \log \Pr(Y_i \mid \mathbf{X}_i; \boldsymbol{\beta}) \right\}$$

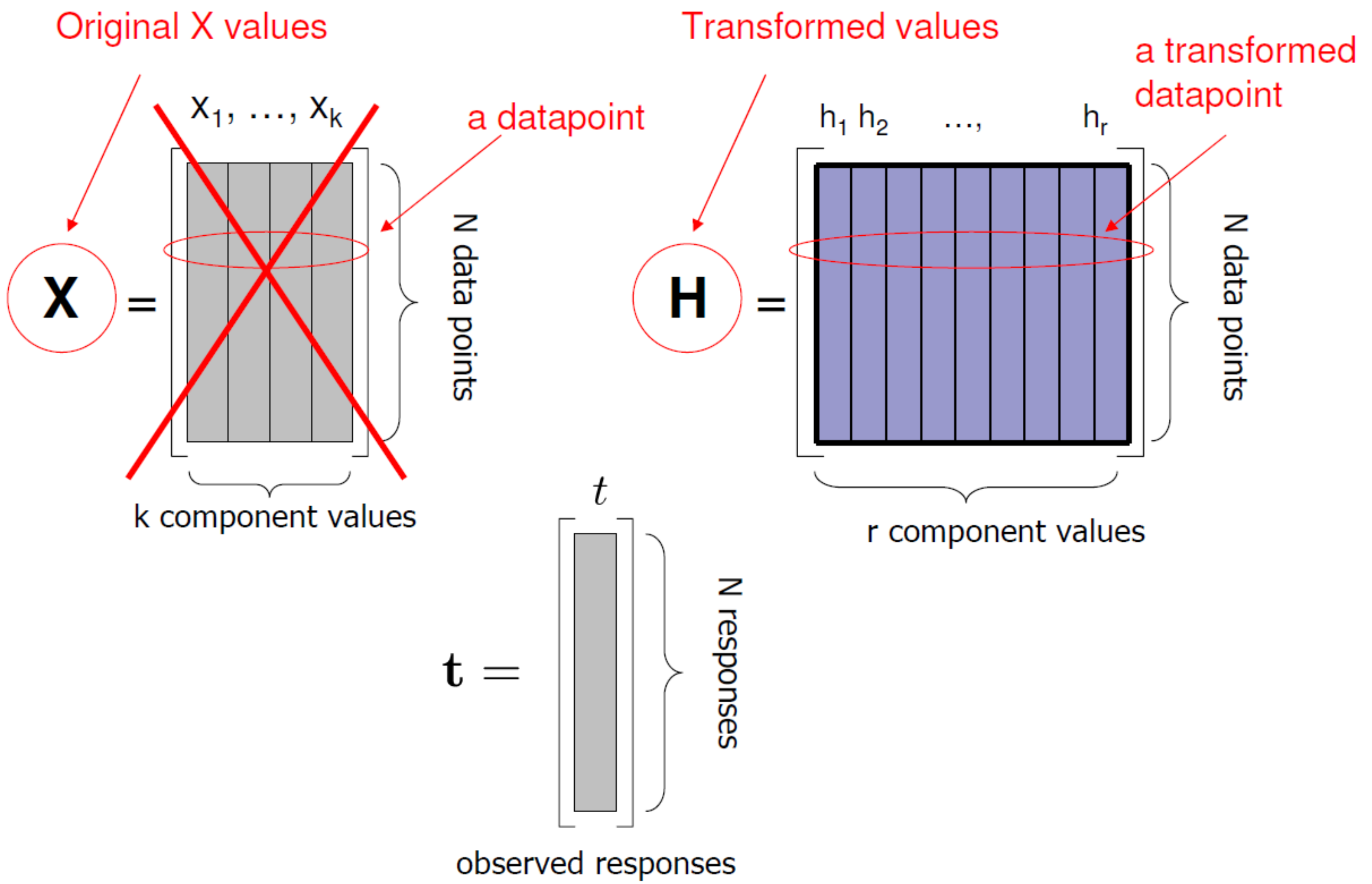
$$= \arg\max_{\boldsymbol{\beta}} \left\{ n \log\left(\frac{1}{\sigma\sqrt{2\pi}} \right) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (Y_i - \sum_{j=1}^{p} \beta_j X_{ij})^2 \right\}$$

$$= \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} (Y_i - \sum_{j=1}^{p} \beta_j X_{ij})^2 \right\}$$

Least-squares Linear Regression

is MLE for Gaussians !!!

General Approach



General Linear Regression Task

- **Data: { [X_i, t_i] }**
- **Learn:** Mapping from **X** to h(**X**)
  - Can use **BASIS** functions: **H = { h_1(x), … h_r(x) }**
    - eg: x_i^2 x_i^3, $x_i\sin(x_i)$, …
  - (Basis) linear mapping

$$t_i = \sum_{j=1}^{r} w_j h_j(X_i)$$

  - Find coeffs **w** = (w_1,…, w_r)
- **Model**: Observed value

$$t_i^* = \sum_{j=1}^{r} w_j h_j(X_i) + \varepsilon$$

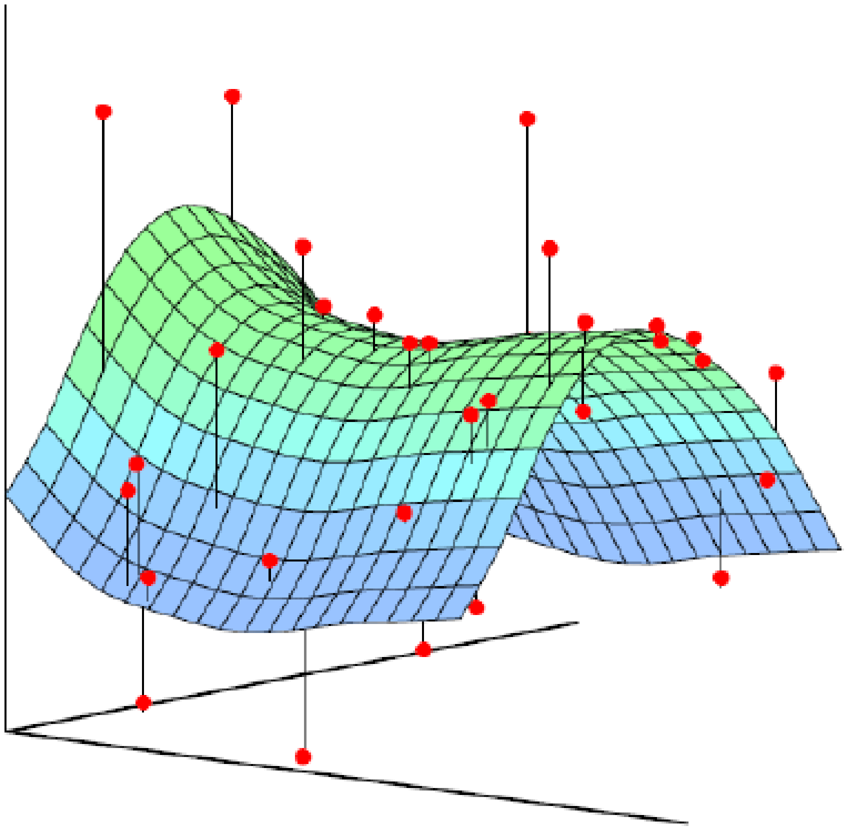
where $\varepsilon \sim N(0, \sigma^2)$

Features/Basis Functions

- Polynomials
  - $1, x, x^2, x^3, x^4, ..$
- Indicators
- Gaussian densities
- Step functions or sigmoids
- Sinusoids (Fourier basis)
- Wavelets
- Anything you can imagine…

Fitting Parameterized Function

- Linear regression over (complex) extended features



- *Least squares fitting of a function of two inputs*
- *Find parameters of $f_\theta(x)$ that minimize the sum-of-squared vertical errors*

Applications

- Predict stock value over time from
  - past values
  - other relevant vars
    - e.g., weather, demands, etc.

Outline

- Linear Models for Regression
- Linear Models for Classification
  - Two class problem (hotdog/not-hotdog classification)
  - Multiple class problem (food classification)
- Evaluating Predictors

The Linear Probability v.s. Logit Model

In the OLS regression:

$$Y = \alpha + \beta X + e; \text{ where } Y = (0, 1)$$

- The error terms are heteroskedastic
- e is not normally distributed because Y takes on only two values
- The predicted probabilities can be greater than 1 or less than 0

The "logit" model solves these problems:

$$\log[p/(1-p)] = \alpha + \beta X + e$$

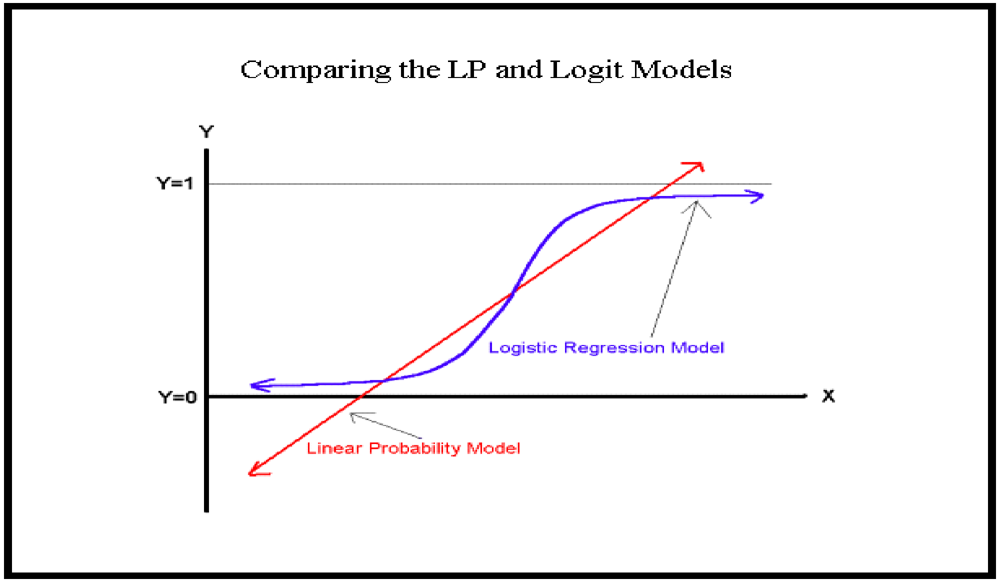
- p is the probability that the event Y occurs given X, $p(Y=1|X)$
- $p/(1-p)$ is the "odds ratio"
- $\log[p/(1-p)]$ is the log odds ratio, or "logit"

More:

- The logistic distribution constrains the estimated probabilities to lie between 0 and 1.
- The estimated probability is:

$$p = 1/[1 + \exp(-\alpha - \beta X)]$$

- if you let $\alpha + \beta X = 0$, then p = .50
- as $\alpha + \beta X$ gets really big, p approaches 1
- as $\alpha + \beta X$ gets really small, p approaches 0



Fitting Logistic Regression Model

- Logistic regression models are usually fit by maximum likelihood, using the **conditional** likelihood of Y given X, Pr(Y|X).
- The log-likelihood can be written as

$$
\begin{aligned}
\ell(\beta) &= \sum_{i=1}^{N} \Big\{ y_i \log p(x_i;\beta) + (1-y_i)\log(1-p(x_i;\beta)) \Big\} \\
&= \sum_{i=1}^{N} \Big\{ y_i \beta^T x_i - \log(1+e^{\beta^T x_i}) \Big\}.
\end{aligned}
$$

- Find MLE of β by Newton-Raphson algorithm

Interpreting Logistic Regression Models

- Since:

  $p = 1/[1 + \exp(-\alpha - \beta X)]$

  The slope coefficient (β) is interpreted as the rate of change in the "Probability" as X changes … not very clear.
- Since:

  $\log[p/(1-p)] = \alpha + \beta X$

  The slope coefficient (β) is interpreted as the rate of change in the "log odds ratio" as X changes

Logistic Regression for K classes

- The model is specified in terms of K-1 log-odds or logit transformations

(reflecting the constraint that the probabilities sum to one).

$$
\begin{aligned}
\log \frac{\Pr(G=1|X=x)}{\Pr(G=K|X=x)} &= \beta_{10} + \beta_1^T x \\
\log \frac{\Pr(G=2|X=x)}{\Pr(G=K|X=x)} &= \beta_{20} + \beta_2^T x \\
&\vdots \\
\log \frac{\Pr(G=K-1|X=x)}{\Pr(G=K|X=x)} &= \beta_{(K-1)0} + \beta_{K-1}^T x.
\end{aligned}
$$

In Kaggle rental competition, K=3, {high, medium, low}

Outline

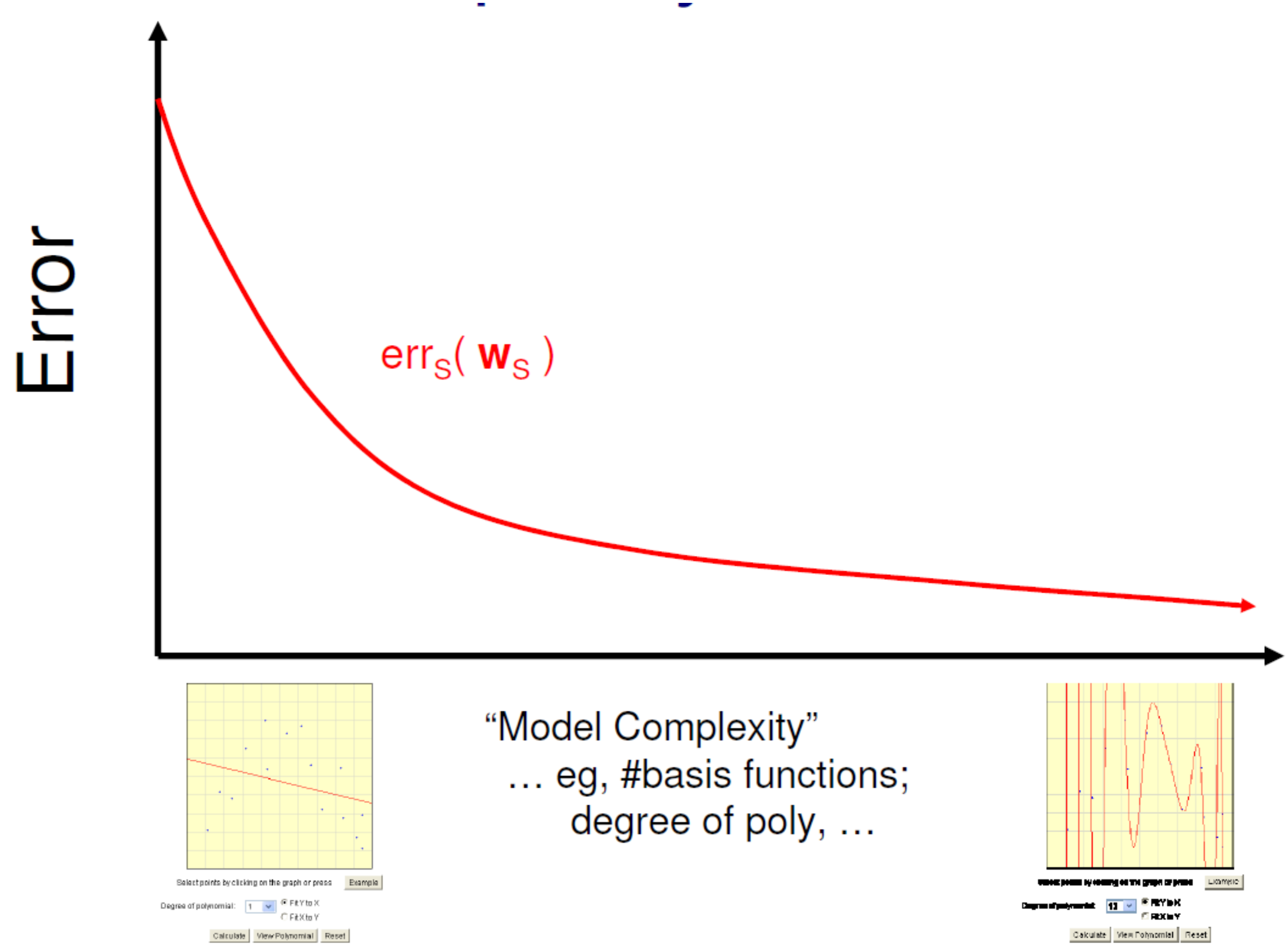
- Linear Models for Regression
- Linear Models for Classification
- Evaluating Predictors
  - Training error
  - Testing error
  - Cross-validation
  - ROC

Training Set Error

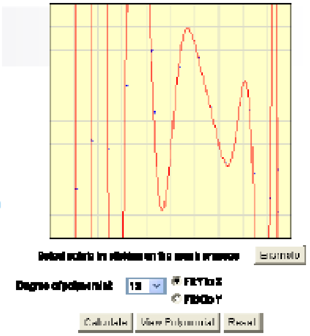
- Choose a loss function
  - e.g. squared error (L_2) for regression (y-y')^2
  - e.g. misclassification error for classification I(y <>y')
- Given a labeled training dataset **S**, learn optimal predictor $\mathbf{w_S}$ that minimize the loss function
- **Training set error: $\mathbf{err_S(w_S)}$**

S = "Training data"

Training Set Error as a function of Model Complexity



True Prediction Error

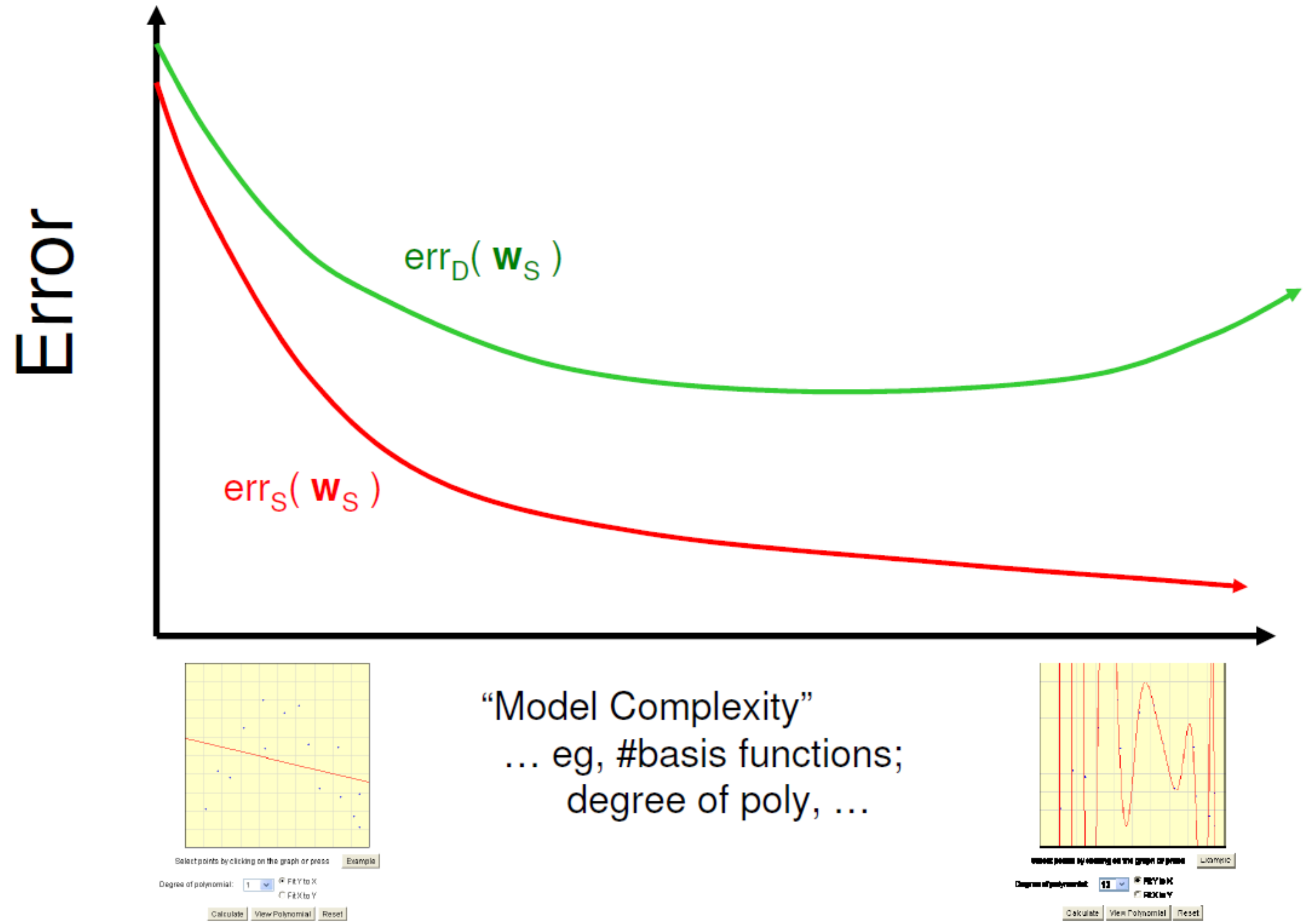


- WARNING: Training set error can be poor measure of "quality" of solution
- **Want:** *error over all possible input points*, **not just training data**:
  - **Prediction error:**

$$
\begin{aligned}
err_D(\mathbf{w}) &= E_{x,t}\left[\left(t(\mathbf{x}) - \sum_i w_i h_i(\mathbf{x})\right)^2\right] \\
&= \int_{x,t} \left(t - \sum_i w_i h_i(\mathbf{x})\right)^2 D(\mathbf{x},\mathbf{t})\; dx\, dt
\end{aligned}
$$

Requires D(**x**,t) – unknown!

Prediction Error as a function of



Computing Prediction Error

- Computing prediction

$$err_D(\mathbf{w}) = \int_{x,t} \left(t - \sum_i w_i h_i(\mathbf{x})\right)^2 D(\mathbf{x}, \mathrm{t})\, dx\, dt$$

  - Depends on D($\mathbf{x}$, t) for every $\mathbf{x}$ – typically not known
  - Hard integral

- New sample: a set of i.i.d. points

  $S' = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_M, t_M)\}$ from D($\mathbf{x}$,t)

$$\mathrm{err}_D(\mathbf{w}_S) \approx \mathrm{err}_{S'}(\mathbf{w}_S) = \frac{1}{|S'|} \sum_{(\mathbf{x},t)\in S'} \left(t - \sum_i w_i h_i(\mathbf{x})\right)^2$$

Training Error ≠ Prediction Error

- Sampling approximation of prediction error:

$$err_{S'}(\mathbf{w}_S) \approx err_D(\mathbf{w}_S)$$

- Training error :

$$err_S(\mathbf{w}_S) \not\approx err_D(\mathbf{w}_S)$$

- Very similar equations!!!
  - Why is *training error* a bad measure of *prediction error*?

Training Error ≠ Prediction Error

- **Because you cheated!!!**

  Training error is good estimate for a single **w,**
  But you optimized **w** with respect to the training error,
  and found **w** that is good for *this set of instances*

  Training error is a (optimistically) biased estimate of prediction error

- Very similar equations!!!
  - Why is *training error* a bad measure of *prediction error*?

Test Set

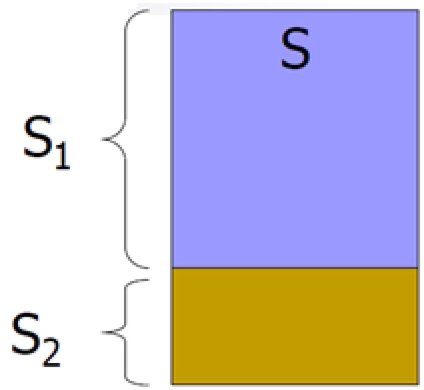
$$\mathbf{w}_S = \mathbf{w}^*(S) = \arg\min_{\mathbf{w}} \sum_{(\mathbf{x},t)\in S} \left(t - \sum_i w_i h_i(\mathbf{x}) \right)^2$$

- **Randomly** split dataset into two parts:
  - Training data – $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_{Ntrain}\}$
  - Test data – $S' = \{\mathbf{x}_{Ntrain+1}, \ldots, \mathbf{x}_{Ntrain+Ntest}\}$
- Use *training data* to optimize $\mathbf{w} = \mathbf{w}_S$
- **Test set error:**
  Given $\mathbf{w}_S$, estimate error using:

$$\mathrm{err}_{S'}(\mathbf{w}_S) = \frac{1}{|S'|} \sum_{(\mathbf{x},t)\in S'} \left(t - \sum_i w_i h_i(\mathbf{x}) \right)^2$$

Estimating Error: Hold-Out Set

- Run learner $L(.)$ on S
  – produce regressor $w_S = L(S)$
  What is true error $err_D(w_S)$?
- Want to return $[w_S, err_D(w_S)]$
  … or at least $[w_S, e]$ where $e \approx err_D(w_S)$



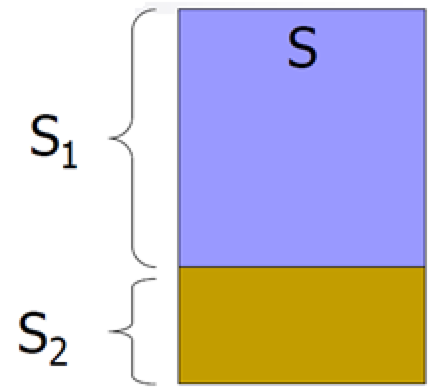
- Divide S into disjoint S_1, S_2
  - *Train* on S_1: computing $w_{S_1} := L(S_1)$
  - *Test* on S_2: $\underline{err}_{S_2}(w_{S_1})$

  Return $[w_S, \underline{err}_{S_2}(w_{S_1})]$

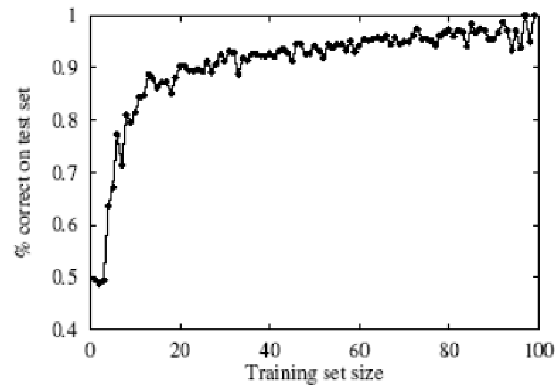
- Why is $\underline{err}_{S_2}(w_{S_1}) \approx err_D(w_S)$?
  - As $S_1 \approx S$, $w_{S_1} = L(S_1) \approx L(S) = w_S$
  - $\underline{err}_{S_2}(w_{S_1})$ is estimate of $err_D(w_{S1}) \approx err_D(w_S)$

Challenge wrt Hold-Out Set

- How to divide S into disjoint S_1, S_2
- As $|S_1| < |S|$,
  $L(S_1)$ not as good as $L(S)$
  Learning curve: $L(S)$ improves as $|S|$ increases)
  $\Rightarrow$ want S_1 to be large



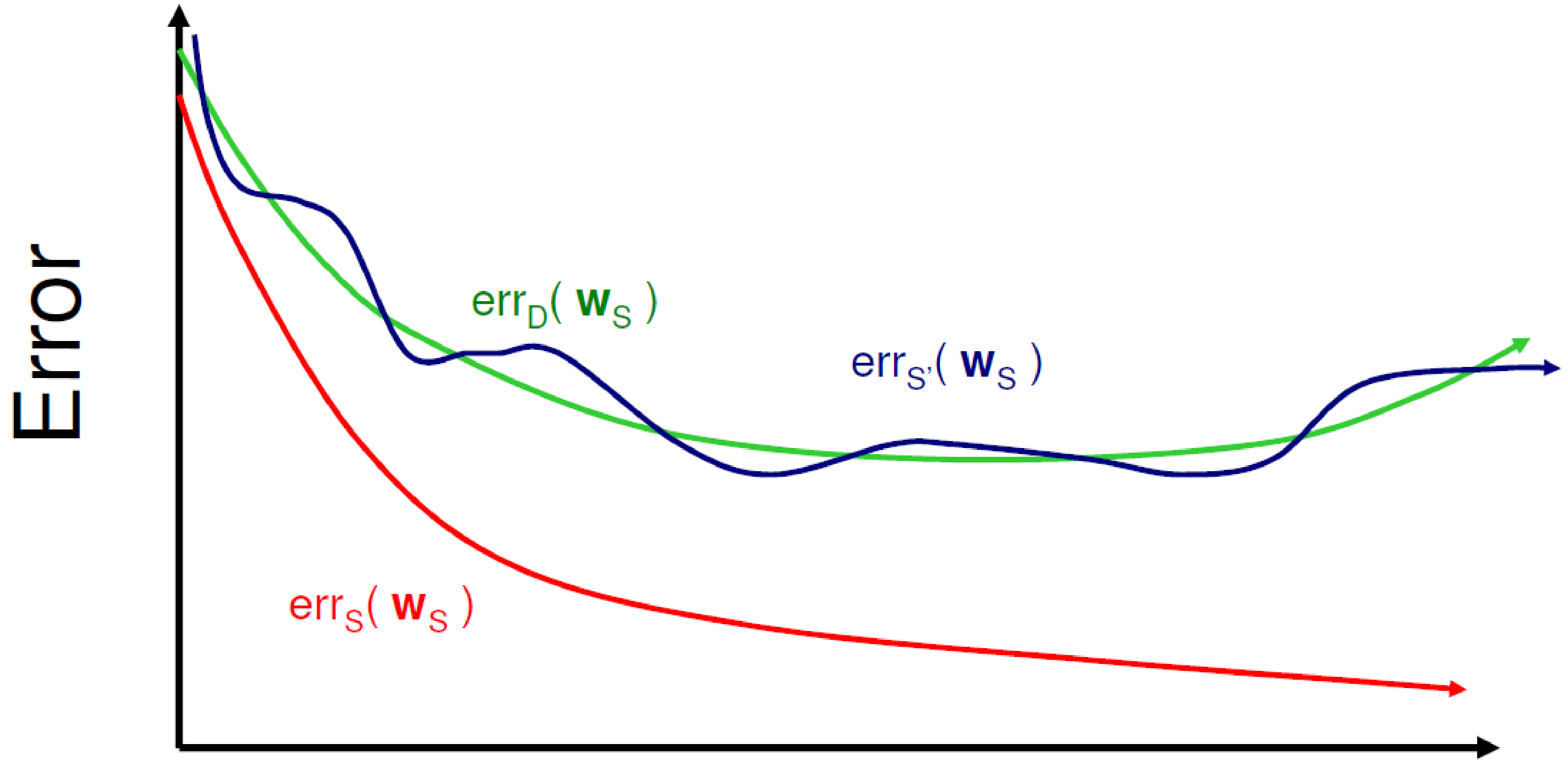
- $\underline{err}_{S_2}(w_{S_1})$ is estimate of $err_D(w_S)$
  Estimate improves as S_2 gets larger
  $\Rightarrow$ want S_2 to be as large as possible



- As $S = S_1 \cup S_2$, must trade off
  quality of classifier $w_{S_1} = L(S_1)$
  with
  accuracy of estimate $\underline{err}_{S_2}(w_{S_1})$

$$\boxed{|err_S(h) - err_D(h)| \approx \frac{\alpha}{\sqrt{|S|}}}$$

Test Set Error as a function of Model Complexity



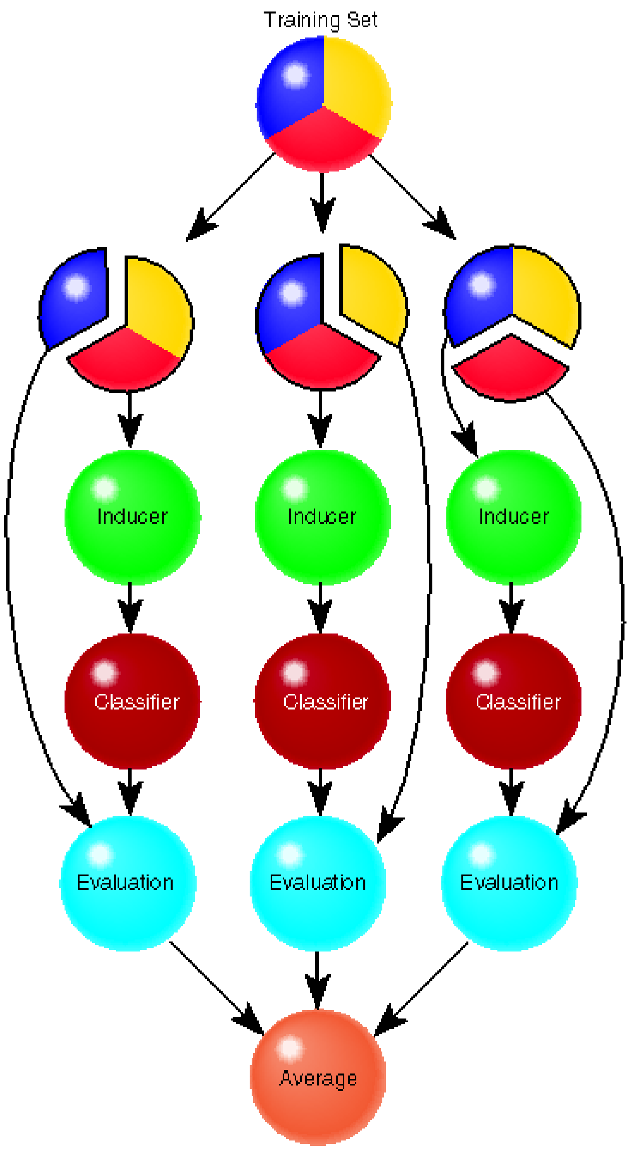
Estimating Error: Cross Validation

- "Cross-Validation"

CV(data S, alg L, int k)
  Divide S into k disjoint sets $\{ S_1, S_2, \ldots, S_k \}$
  For i = 1..k do
    Run L on $S_{-1} = S - S_i$
     obtain $h_i := L(S_{-i})$
    Evaluate h_i on S_i
      $err_{S_i}(h_i) = 1/|S_i| \sum_{\langle x,t \rangle \in S_i} [h_i(x) - t]^2$
  Return Average $1/k \sum_i err_{S_i}(h_i)$

$\Rightarrow$ Less Pessimistic

as train on $(k - 1)/k\ |S|$ of the data



Comments on Cross-Validation

- Every point used as
  Test 1 time, Training k – 1 times
- Computational cost for *k*-fold Cross-validation … linear in *k*
- Should use "balanced CV"
  If class c_i appears in m_i instances,
  insist each S_k include $\approx \frac{1}{k} \frac{m_i}{|S|}$ such instances
- Use **CV(S, L, k)** as ESTIMATE of true error of L(S)
  Return L(S) and CV(S, L, k)
- Leave-One-Out-Cross-Validation k = m
- Notice different folds are correlated
  as training sets overlap: (k-2)/k unless k=2

To Form k *Balanced Folds*

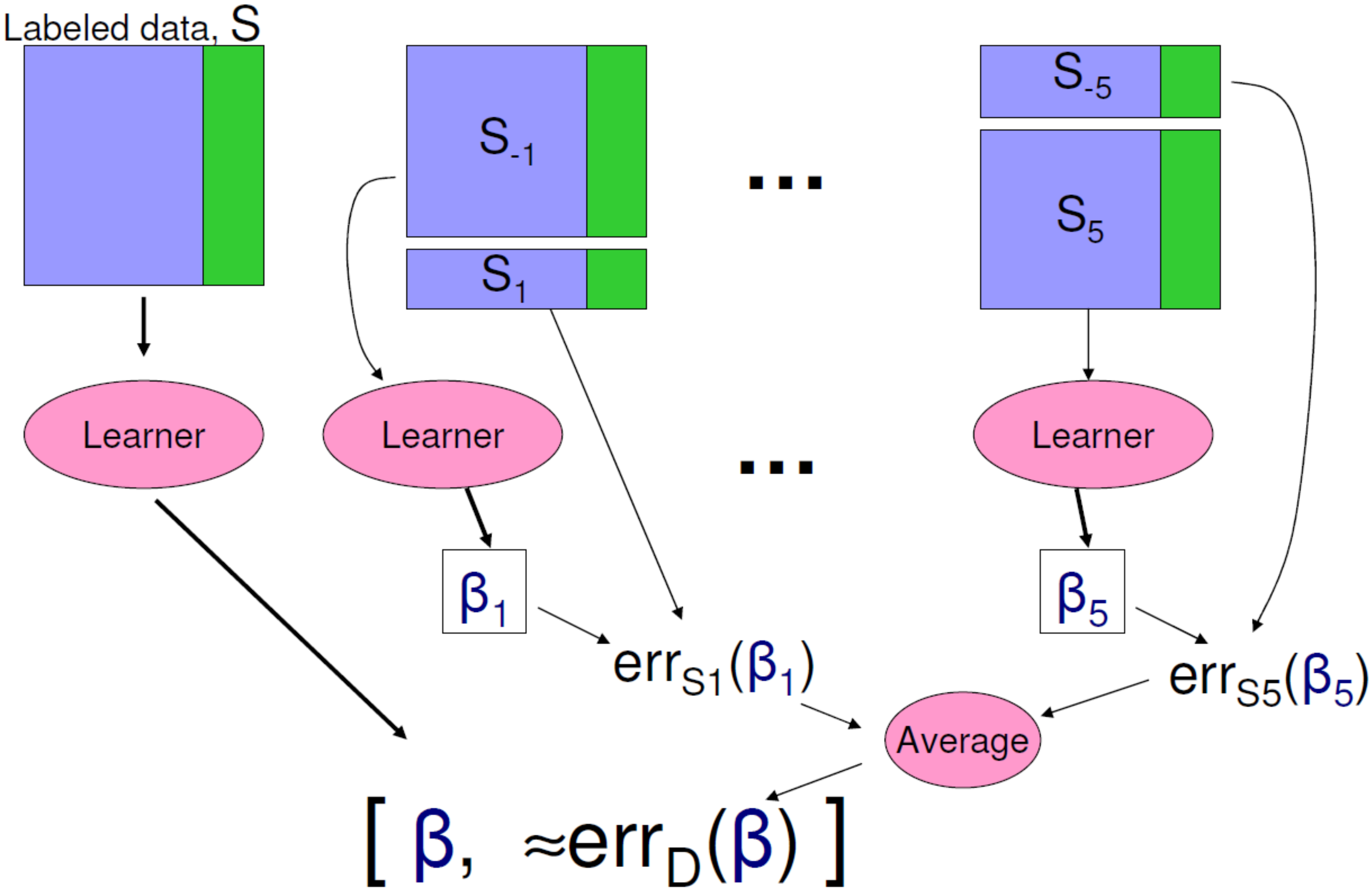
1. Partition the data S based on the class:
   - subset S_+ has all the positive instances,
   - subset S_- has all the negative instances.

2. Randomly partition each subset into k folds:

   $S_+ = \cup \{ S_{+1}, \ldots, S_{+k} \}$

3. $S_j = S_{+j} \cup S_{-j}$ for j=1..k

Return: [Predictor + Est Quality]



How many points needed for training/testing?

- Very hard question to answer!
  - Too few training points, learned **w is bad**
  - Too few test points, you never know if you reached a good solution
- Bounds, such as Hoeffding's inequality can help:

$$P(|\,\widehat{\theta} - \theta^*\,| \geq \epsilon) \;\leq\; 2e^{-2N\epsilon^2}$$

- Typically:
  - if you have a reasonable amount of data, pick test set "large enough" for a "reasonable" estimate of error, and use the rest for learning
  - if you have little data, then you need to pull out the big guns…
    - e.g., bootstrapping

Error Estimators

Be careful!!!

Test set only unbiased if you *never never never never* do *any any any* learning/adjustment/… on the test data

Eg,
if you use the test set to select the degree of the polynomial… no longer unbiased!!!
(We will address this problem later in the semester)

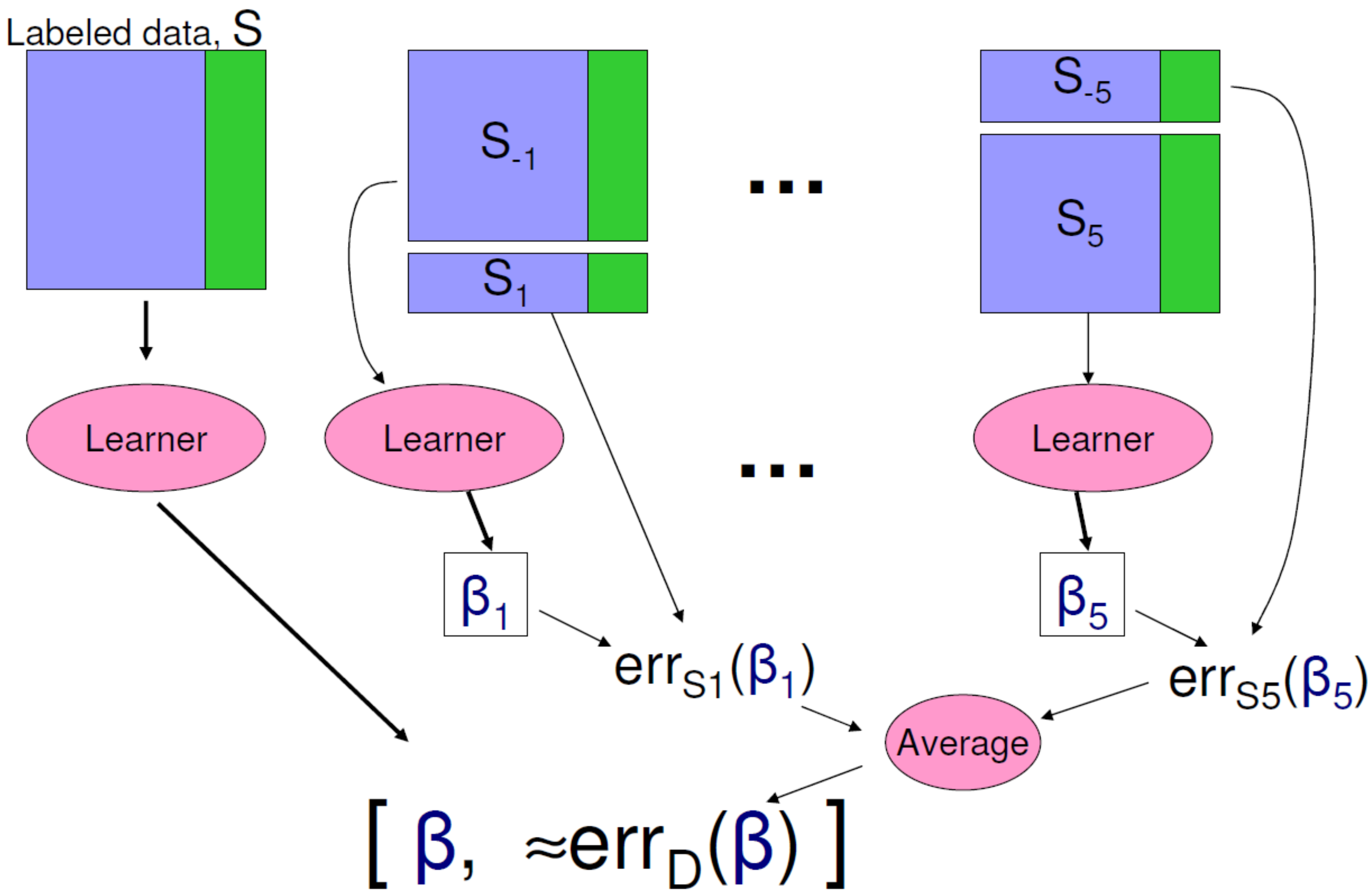
… if you are careful

$$\mathrm{err}_{S'}(\mathbf{w_S}) = \frac{1}{|S'|}\sum_{(\mathbf{x},t)\in S'}\left(t-\sum_i w_i h_i(\mathbf{x})\right)^2$$

Why learn $err_D(\mathbf{w})$: Finding Best Parameters

- Want to learn what "parameters" work best?
  - Best model (RBF vs linear? degree of polynomial)? Feature selection? Trade-off parameter?…
  - $\text{argmin}_v \{ \; err_D(\; L(S, v)) \; \}$
- #1?: Try each value on entire dataset. Report which has smallest TRAINING SET error?
  - $\text{argmin}_v \{ \; err_S(\; L(S, v)) \; \}$
- #2?: For each value, run 5-fold C-V (wrt entire dataset)
  - $v^* = \text{argmax}_v \{ E[\, err_{Si} (L(S_{-I}, v)])$
- Run cross-validation on "best-value" algorithm

Return: [Predictor + Est Quality]



ROC for evaluating 2-class classifiers

- Imagine you have 2 different probabilistic classification models
  - e.g. logistic regression vs. neural network
- How do you know which one is better?
- How do you communicate your belief?
- Can you provide quantitative evidence beyond a gut feeling and subjective interpretation?

Recall Basics: Contingencies

		MODEL PREDICTED	
		It's NOT a Heart Attack	*Heart Attack!!!*
GOLD STANDARD TRUTH	*Was NOT a Heart Attack*	A	B
	Was a Heart Attack	C	D

Some Terms

		MODEL PREDICTED	
		NO EVENT	*EVENT*
GOLD STANDARD TRUTH	*NO EVENT*	TRUE NEGATIVE	B
	EVENT	C	TRUE POSITIVE

Some More Terms

		MODEL PREDICTED	
		NO EVENT	*EVENT*
GOLD STANDARD TRUTH	*NO EVENT*	A	FALSE POSITIVE (Type 1 Error)
	EVENT	FALSE NEGATIVE (Type 2 Error)	D

Accuracy

- What does this mean?
- What is the difference between "accuracy" and an "accurate prediction"?
- Contingency Table Interpretation

$$\frac{\text{(True Positives)} + \text{(True Negatives)}}{\text{(True Positives)} + \text{(True Negatives)} + \text{(False Positives)} + \text{(False Negatives)}}$$

- Is this a good measure? (Why or Why Not?)

Note on Discrete Classes

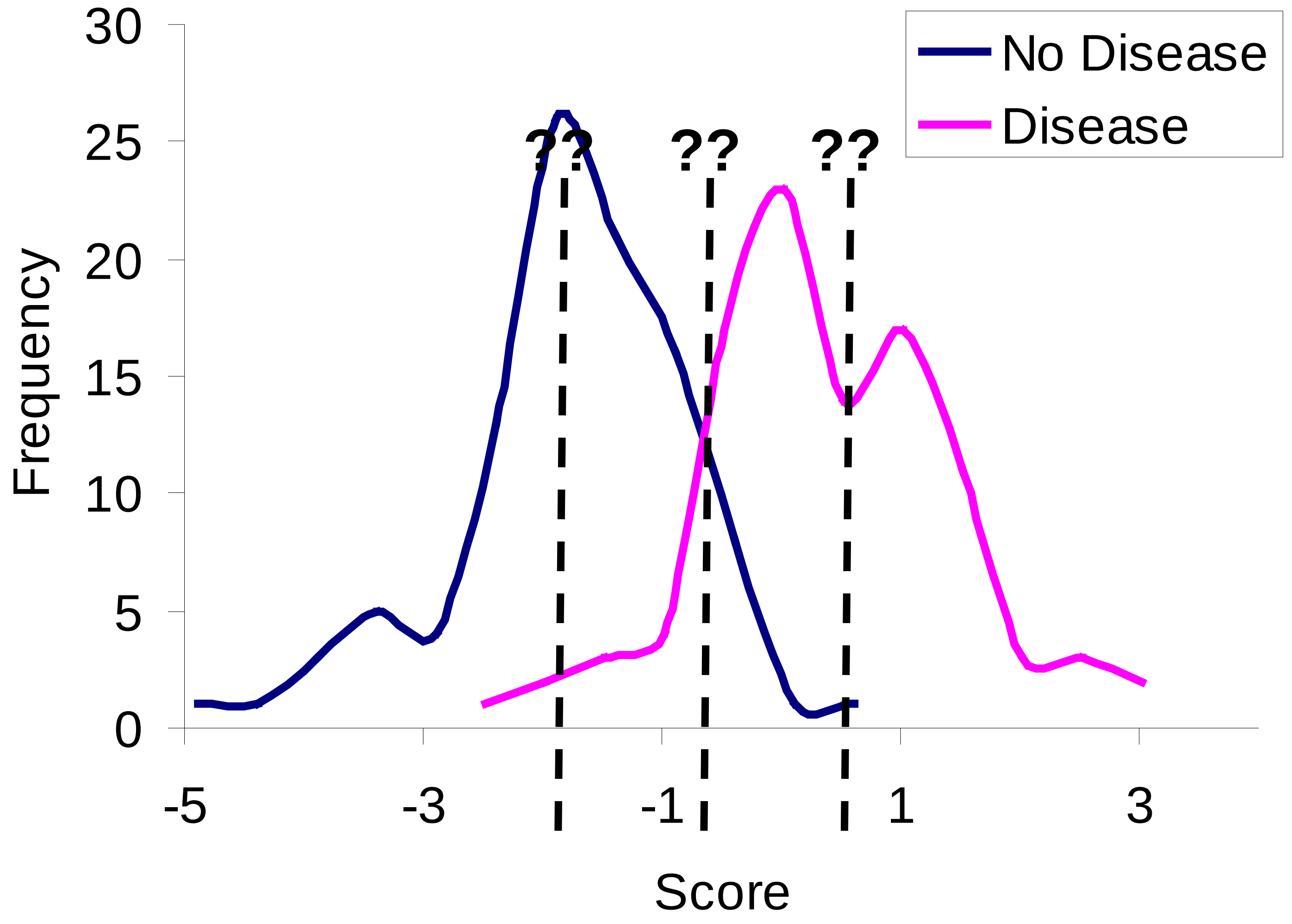
- TRADITION … Show contingency table when reporting predictions of model.

- *BUT* … probabilistic models do not provide discrete calculations of the matrix cells!!!

- *IN OTHER WORDS* … Regression does not report the number of individuals predicted positive (*e.g. has a heart attack*) … ***well, not really***

- *INSTEAD* … report probability the output will be certain variable (e.g. 1 or 0)

Visual Perspective



ROC Curves

- Originated from signal detection theory
  - Binary signal corrupted by Guassian noise
  - What is the optimal threshold (i.e. operating point)?

- Dependence on 3 factors
  - Signal Strength
  - Noise Variance
  - Personal tolerance in Hit / False Alarm Rate

ROC Curves

- Receiver operator characteristic
- Summarize & present performance of any binary classification model
- Models ability to distinguish between false & true positives
- Based solely on ranks

Use Multiple Contingency Tables

- Sample contingency tables from range of threshold/probability.
- *tpr --TRUE POSITIVE RATE* (also called *SENSITIVITY*)

$$\frac{\text{True Positives}}{(\text{True Positives}) + (\text{False Negatives})}$$

- *fpr -- FALSE POSITIVE RATE* (also called 1 - *SPECIFICITY*)

$$\frac{\text{False Positives}}{(\text{False Positives}) + (\text{True Negatives})}$$

- Plot Sensitivity vs. (1 – Specificity) for sampling and you are done

- acc – Accuracy

$$\frac{\text{True Positives} + \text{True Negatives}}{\text{Total examples}}$$

Accuracy Lines

- N the # of examples,
- NEG - # of negative examples, and POS - # of positive examples
- neg - fraction of negative examples, pos - fraction of positive examples

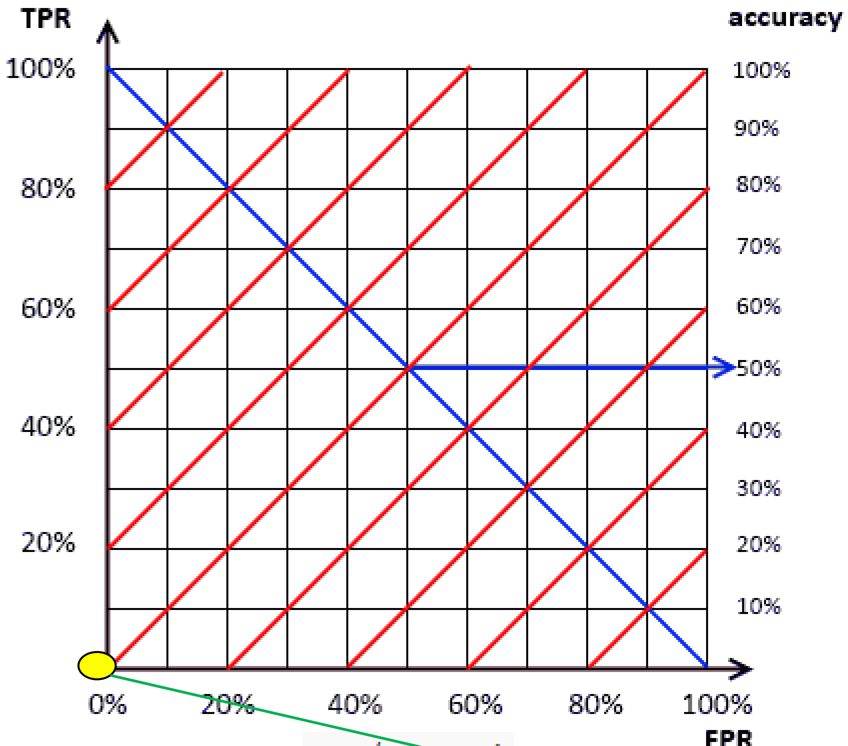
$$\mathrm{acc} = \frac{\mathrm{TP} + \mathrm{TN}}{N} = \frac{\mathrm{TP}}{N} + \frac{\mathrm{TN}}{N} = \frac{\mathrm{TP}}{\mathrm{POS}} \cdot \frac{\mathrm{POS}}{N} + \frac{\mathrm{NEG} - \mathrm{FP}}{N}$$

$$= \frac{\mathrm{TP}}{\mathrm{POS}} \cdot \frac{\mathrm{POS}}{N} + \frac{\mathrm{NEG}}{N} - \frac{\mathrm{FP}}{\mathrm{NEG}} \cdot \frac{\mathrm{NEG}}{N}$$

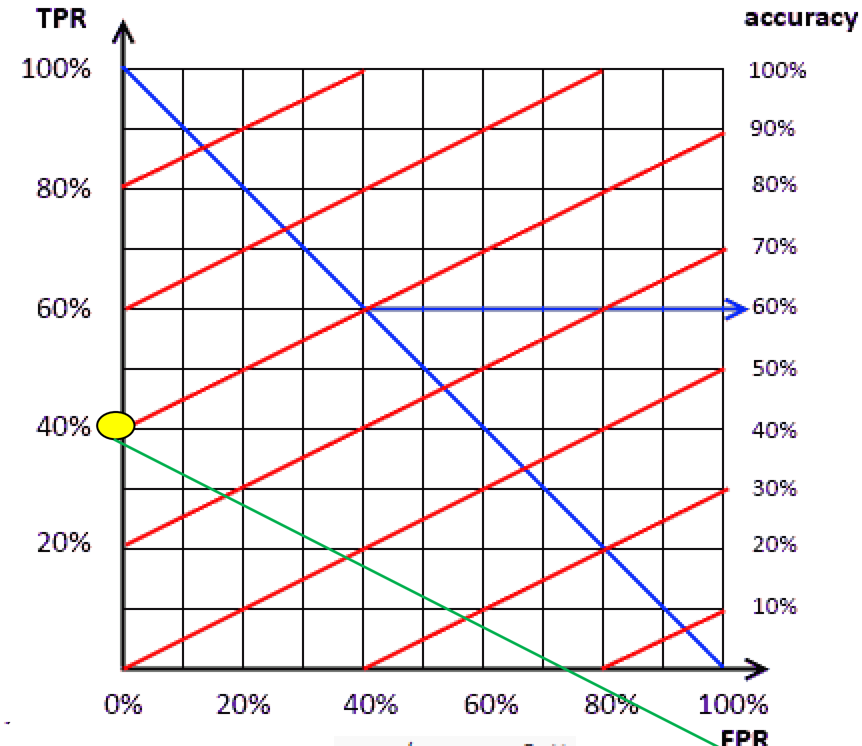
$$= \mathrm{trp} \cdot \mathrm{pos} + \mathrm{neg} - \mathrm{neg} \cdot \mathrm{fpr}$$

- so can rewrite this and get
  - $\mathrm{tpr} = \frac{\mathrm{acc} - \mathrm{neg}}{\mathrm{pos}} + \frac{\mathrm{neg}}{\mathrm{pos}} \cdot \mathrm{fpr}$
- it's a line: $y = ax + b$
  - $y = \mathrm{tpr}, x = \mathrm{fpr}, a = \frac{\mathrm{neg}}{\mathrm{pos}}, b = \frac{\mathrm{acc} - \mathrm{neg}}{\mathrm{pos}}$

Accuracy Lines



$\mathrm{neg/pos} = 1$



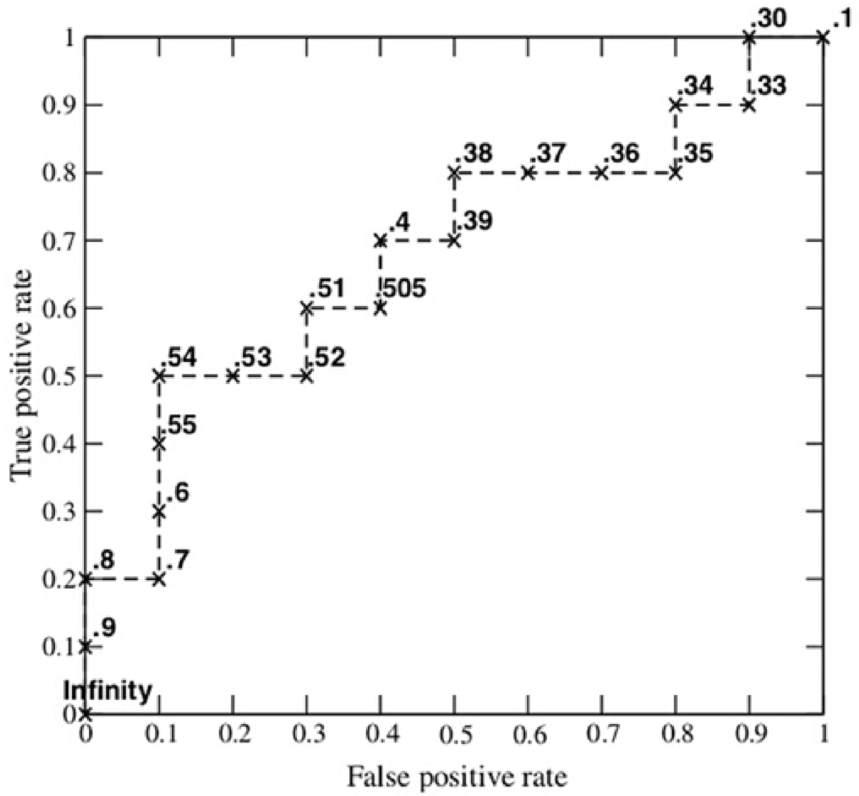
$\mathrm{neg/pos} = 0.5$

neg=pos=0.5, b=(0.5-0.5)/0.5=0

neg=1/3, pos=2/3, b=(3/5-1/3)/(2/3)=0.4

ROC Plot

Inst#	Class	Score	Inst#	Class	Score
1	p	.9	11	p	.4
2	p	.8	12	n	.39
3	n	.7	13	p	.38
4	p	.6	14	n	.37
5	p	.55	15	n	.36
6	p	.54	16	n	.35
7	n	.53	17	p	.34
8	n	.52	18	n	.33
9	p	.51	19	p	.30
10	n	.505	20	n	.1



$$\text{tpr} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad \text{fpr} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

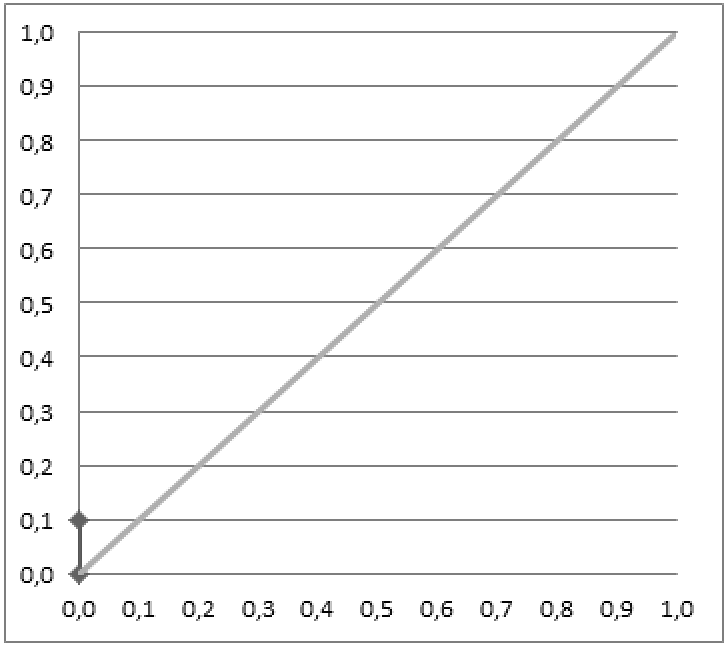
ROC Plot notes

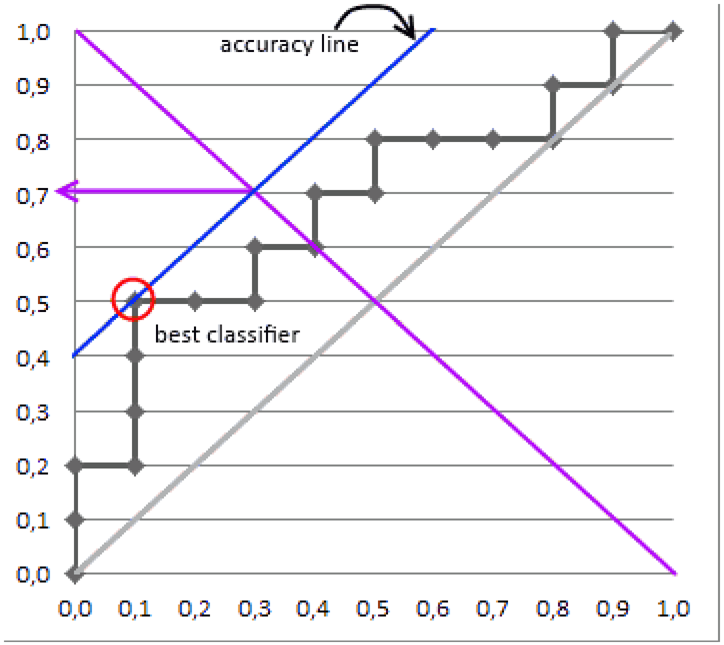
Let's check the point at 0.52 on the curve, it has tpr 0.5 and fpr 0.3.
How do we get them?

1. Assume we use logistic model and the scores are the predicted probability of positive for each sample.
2. We now use 0.52 as threshold so that score >=0.52 will be positive from model. So, from the model, we predict 1-8 be positive, 9-20 negative.
3. Among 1-8 predicted positives, only 1,2,4,5,6 are true positive, so TP=5
4. Among 9-20 predicted negatives, false negatives are 9, 11, 13, 17, 19, so FN=5
5. so tpr=5/(5+5)=0.5
6. Among 1-8 predicted positives, 3,7,8 are false positive, so FP=3
7. Among 9-20 predicted negatives, 10, 12, 14, 15, 16, 18, 20 are true negative, so TN=7
8. so fpr=3/(3+7)=0.3
9. so we get (0.3, 0.5) at the threshold 0.52

ROC Plot

#	C	Score
1	P	0,9
2	P	0,8
3	N	0,7
4	P	0,6
5	P	0,55
6	P	0,54
7	N	0,53
8	N	0,52
9	P	0,51
10	N	0,505
11	P	0,4
12	N	0,39
13	P	0,38
14	N	0,37
15	N	0,36
16	N	0,35
17	P	0,34
18	N	0,33
19	P	0,3
20	N	0,1





ROC Plot

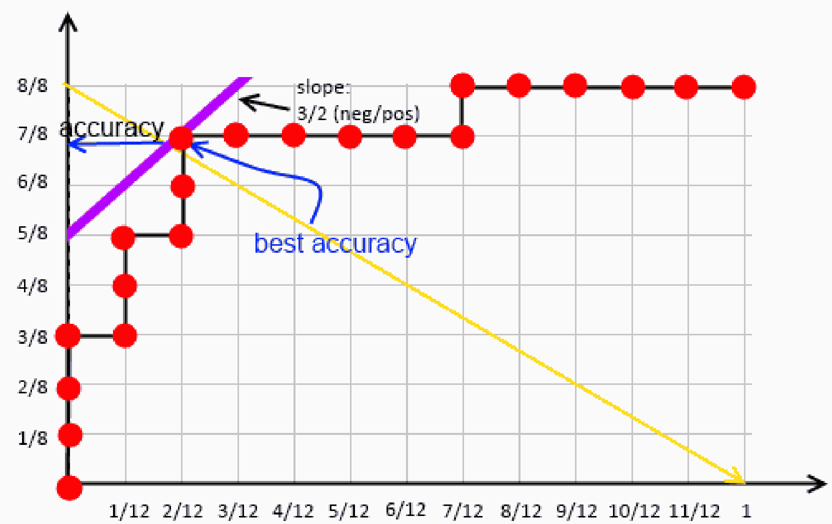
- 20 training examples, 12 negative and 8 positive

#	Cls	Score
1	N	0.18
2	N	0.24
3	N	0.32
4	N	0.33
5	N	0.4
6	N	0.53
7	N	0.58
8	N	0.59
9	N	0.6
10	N	0.7
11	N	0.75
12	N	0.85
13	P	0.52
14	P	0.72
15	P	0.73
16	P	0.79
17	P	0.82
18	P	0.88
19	P	0.9
20	P	0.92

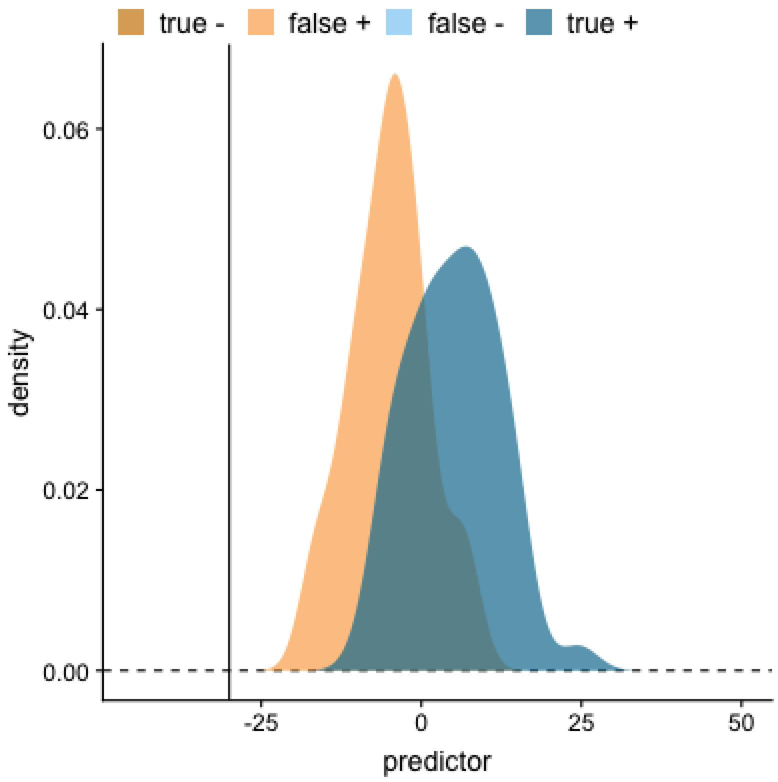
⇒ sort by score

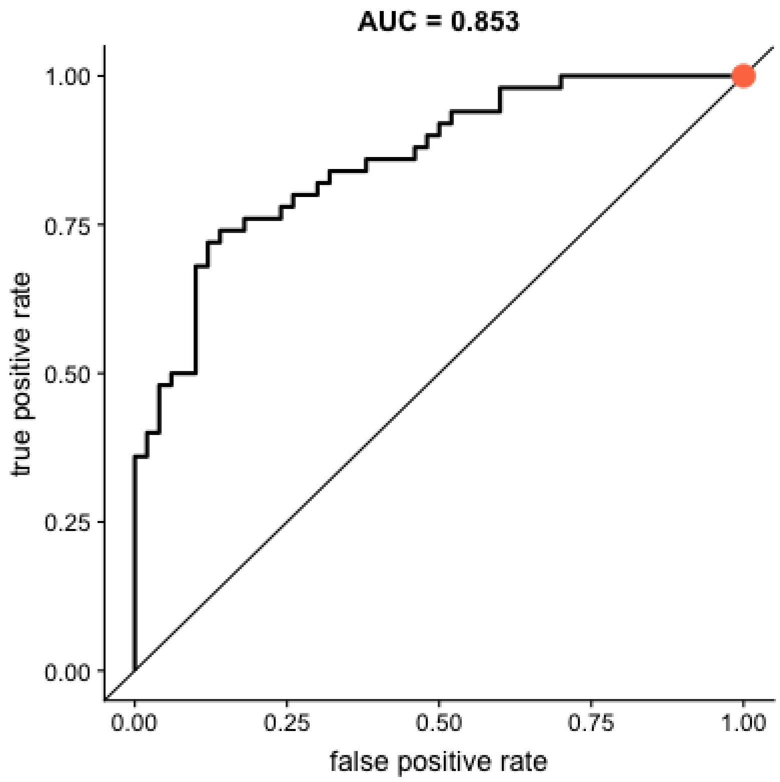
#	Cls	Score
20	P	0.92
19	P	0.9
18	P	0.88
12	N	0.85
17	P	0.82
16	P	0.79
11	N	0.75
15	P	0.73
14	P	0.72
10	N	0.7
9	N	0.6
8	N	0.59
7	N	0.58
6	N	0.53
13	P	0.52
5	N	0.4
4	N	0.33
3	N	0.32
2	N	0.24
1	N	0.18

Now draw the curves:

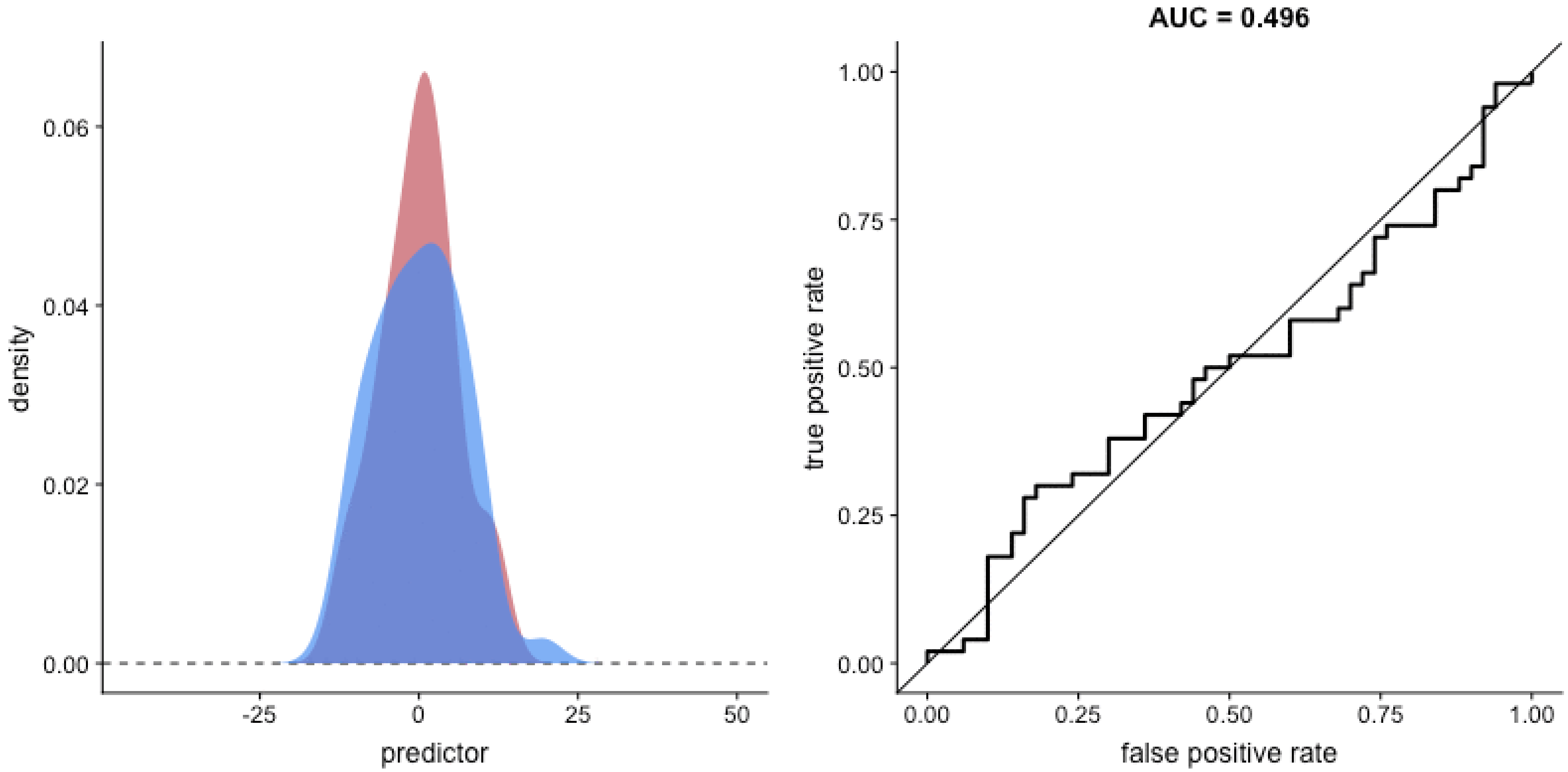


ROC Plot


true -
false +
false -
true +
density
0.06
0.04
0.02
0.00
-25
0
25
50
predictor

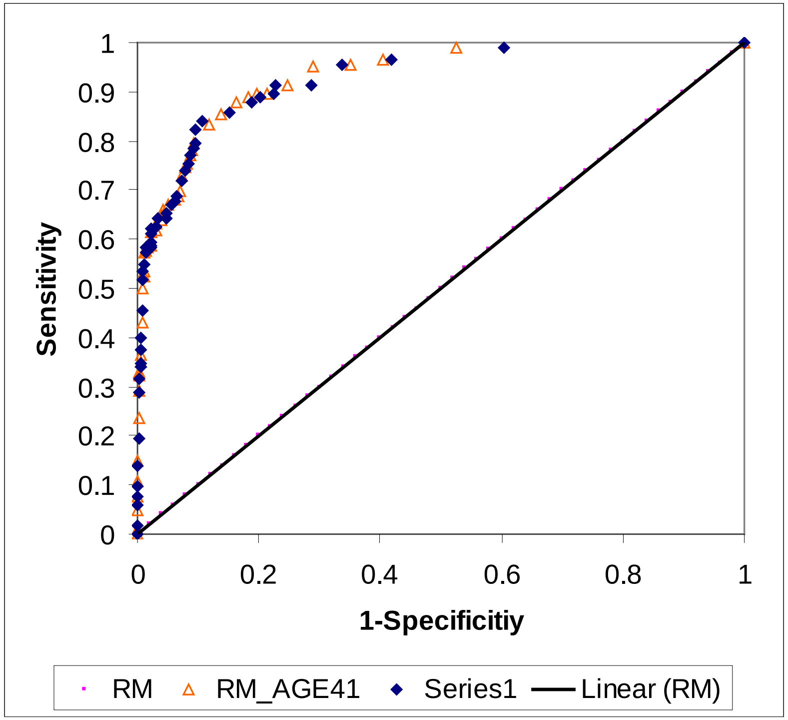

AUC = 0.853
true positive rate
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
false positive rate

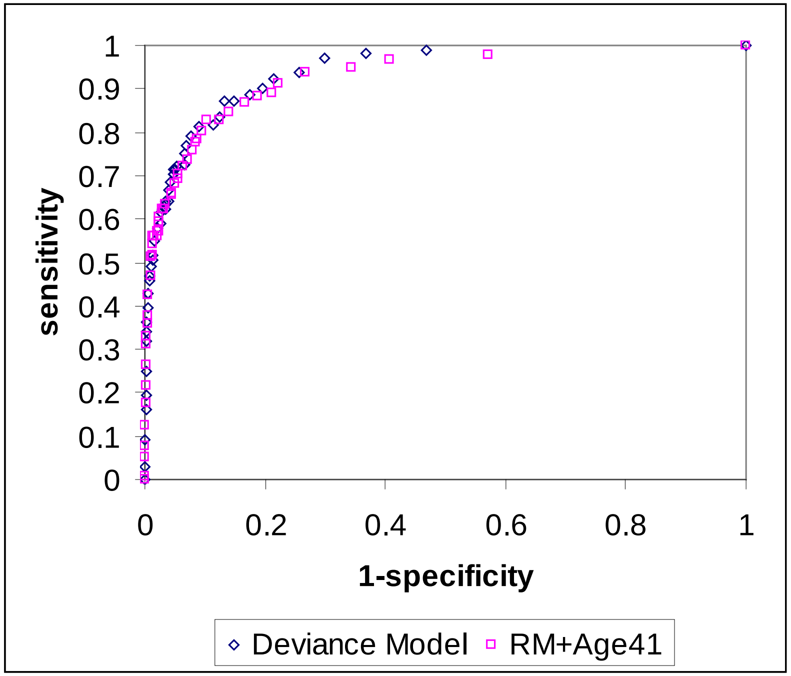
ROC Plot



density
0.06
0.04
0.02
0.00
-25
0
25
50
predictor
AUC = 0.496
true positive rate
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
false positive rate

Sidebar: Use More Samples

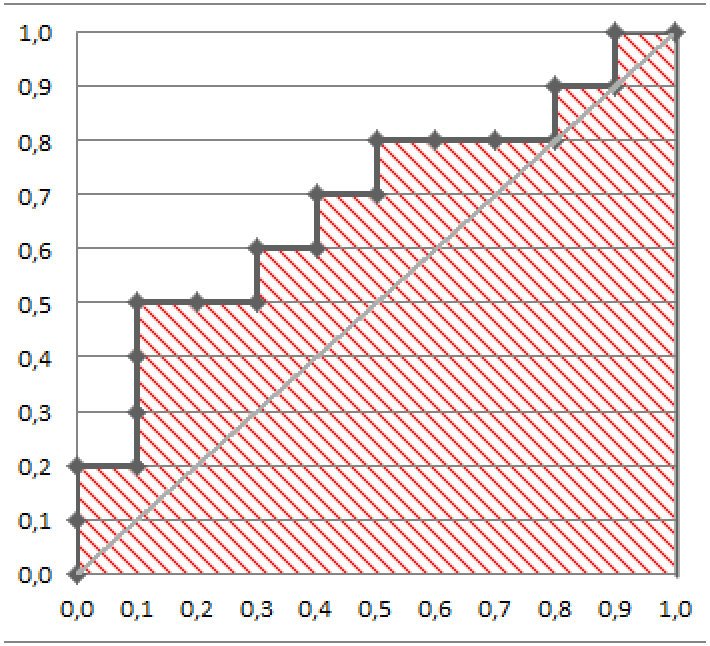




(These are plots from a much larger dataset)

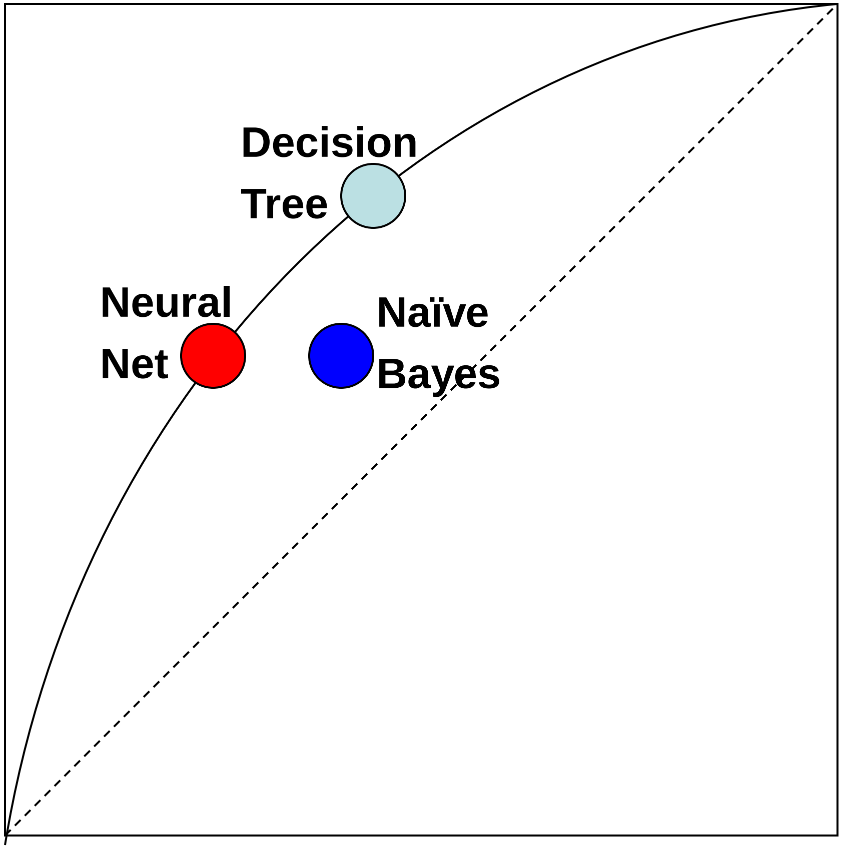
AUC: ROC Quantification

- Area Under ROC Curve (AUC)



Theory: Model Optimality

- Classifiers on convex hull are always "optimal"
  - e.g. Net & Tree
- Classifiers below convex hull are always "suboptimal"
  - e.g. Naïve Bayes



Some Statistical Insight

- Curve Area:
  - Take random healthy patient → score of X
  - Take random heart attack patient → score of Y
  - Area estimate of $P[Y > X]$

- Empirical estimate of AUC

  1……………0 ranked by Prob(Y=Positive)

  {PPNPPNNN}

  Randomly selection one P and one N, what is the chance that P ranks before N?

R Packages/functions

- glm
  - Generalized linear models
- multinomial logistic regression
  - multinom function from the nnet package
- ROCR
  - A package for Visualizing the Performance of Scoring Classifiers
- An R example for multinomial logistic regression
  - http://stats.idre.ucla.edu/r/dae/multinomial-logistic-regression/