COMP 790-125: Goals for today

- Two more algorithms for logistic regression
  - Newton algorithm
  - Hybrid of sequential quadratic approximation and coordinate ascent
- Assessing predictive performance

Fast scan of the homework

Logistic regression and posterior in a mixture of Gaussians

$$
\begin{aligned}
p(\mathbf{x}|y=-1) &= \frac{1}{\sqrt{(2\pi)^k \det \Sigma}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1)\right\} \\
p(\mathbf{x}|y=+1) &= \frac{1}{\sqrt{(2\pi)^k \det \Sigma}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_2)^T\Sigma^{-1}(\mathbf{x}-\mu_2)\right\} \\
p(y=-1) &= q \\
p(y=+1) &= 1-q
\end{aligned}
$$

Posterior is given by $p(y=-1|\mathbf{x}) = \dfrac{1}{1+\underbrace{\dfrac{p(\mathbf{x},y=-1)}{p(\mathbf{x},y=+1)}}_{\exp z}}$

$$
\begin{aligned}
z &= \log(q) - \log(1-q) - \frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1) - \frac{1}{2}(\mathbf{x}-\mu_2)^T\Sigma^{-1}(\mathbf{x}-\mu_2) \\
&= -(\mathbf{x}^T \underbrace{\Sigma^{-1}(\mu_2-\mu_1)\frac{1}{2}}_{\beta} + \underbrace{(\mu_1^T\Sigma^{-1}\mu_1 - \mu_2^T\Sigma^{-1}\mu_2)\frac{1}{2} + \log(1-q) - \log(q))}_{\beta_0}
\end{aligned}
$$

Newton's method

Newton's method finds zeros of functions by using their gradient.

A point where all of the partial derivatives are zero is a critical point (maximum for a concave function).

$$\nabla \mathrm{LL}(\beta_0, \beta) = \mathbf{0}$$

To use Newton's method we need Hessian ($\nabla^2 \mathrm{LL}(\beta_0, \beta)$)

Newton's method

If we have the gradient ($\nabla \mathrm{LL}$) and Hessian ($\nabla^2 \mathrm{LL}$) then we iterate update for $k = 1, \ldots, \mathrm{MAXIT}$

$$\begin{bmatrix} \beta_0^{k+1} \\ \beta_1^{k+1} \\ \ldots \\ \beta_p^{k+1} \end{bmatrix} = \begin{bmatrix} \beta_0^{k} \\ \beta_1^{k} \\ \ldots \\ \beta_p^{k} \end{bmatrix} - s \nabla^2 \mathrm{LL}(\beta_0^k, \beta^k)^{-1} \nabla \mathrm{LL}(\beta_0^k, \beta^k)$$

with β_0^1, β^1 set to some sensible initial values, for example zeros.

The costly part is Hessian matrix inversion (solution of a linear system).

There is still step size to worry about.

Optimization: Hessian of logistic regression likelihood

Gradient

$$\begin{aligned}
\nabla \mathrm{LL}(\beta_0, \beta) &= \sum_i (1 - p(y_i|\mathbf{x}_i, \beta_0, \beta)) y_i \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix} \\
&= \sum_i (1 - \sigma(y_i(\beta_0 + \langle \mathbf{x}_i, \beta \rangle)) y_i \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix} \\
&= \sum_i \sigma(-y_i(\beta_0 + \langle \mathbf{x}_i, \beta \rangle)) y_i \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix} \\
&= \sum_i \sigma(-y_i(\beta_0 + \langle \mathbf{x}_i, \beta \rangle)) y_i \tilde{\mathbf{x}}_i
\end{aligned}$$

We will be taking derivatives of $\sigma(-y_i(\beta_0 + \langle \mathbf{x}_i, \beta \rangle))$

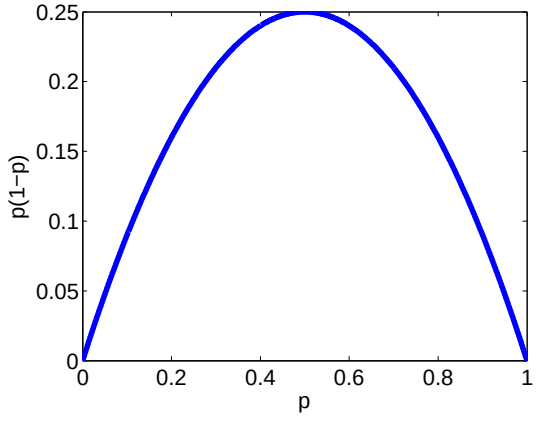
$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)\sigma(-z)dz$$

Optimization: Hessian of logistic regression likelihood

With the gradient (partial derivatives) and the equality for the derivative of the sigmoid we can can compute Hessian ($\nabla^2 \mathrm{LL}$)

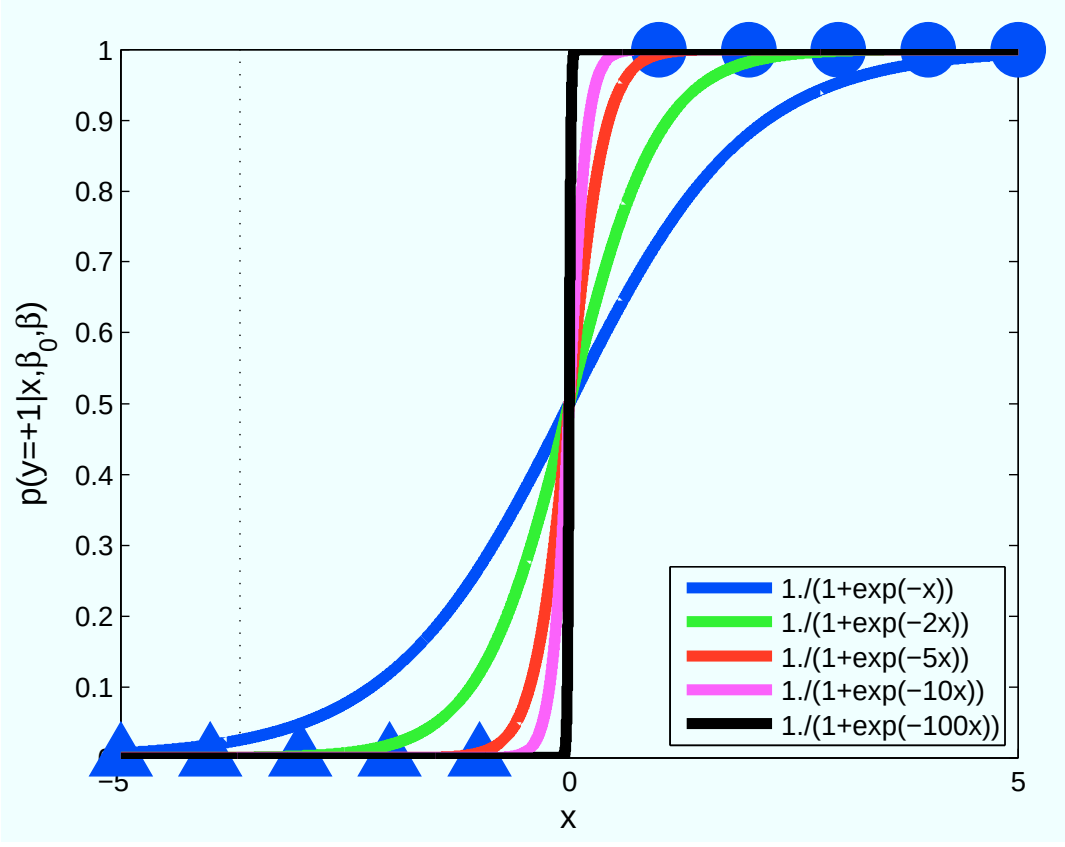
$$\begin{aligned}\frac{\partial^2 \mathrm{LL}(\beta_0, \beta)}{\partial \beta_j \partial \beta_k} &= -\sum_i \tilde{x}_{i,j} \tilde{x}_{i,k} \sigma(y_i(\beta_0 + \langle \mathbf{x}_i, \beta \rangle)) \sigma(-y_i(\beta_0 + \langle \mathbf{x}_i, \beta \rangle)) \\ &= -\sum_i \tilde{x}_{i,j} \tilde{x}_{i,k} p(y_i | \mathbf{x}_i, \beta_0, \beta)(1 - p(y_i | \mathbf{x}_i, \beta_0, \beta))\end{aligned}$$

where $\tilde{x}_{i,0} = 1$ and $\tilde{x}_{i,j} = x_{i,j}$ for $j > 0$.



Logistic regression in a well separated case

There is nothing stopping the fitting procedure from scaling βs up if data is separable



Ridge ... again

$$\mathrm{LL}(\beta_0, \beta) = -\sum_i \log\left\{1 + \exp\left\{-y_i(\beta_0 + \mathbf{x}_i'\beta)\right\}\right\} - \frac{\lambda}{2}\sum_j \beta_j^2$$

which is same as putting a Gaussian prior on the feature weights (β)

$$\begin{aligned} p(y|\mathbf{x}, \beta_0, \beta) &= \frac{1}{1 + \exp\left\{-y(\beta_0 + \langle \mathbf{x}, \beta\rangle)\right\}} \\ p(\beta_j) &= \frac{1}{\sqrt{(2\pi)\psi^2}} \exp\left\{-\frac{\beta_j^2}{2\psi^2}\right\} \qquad (j > 0) \end{aligned}$$

Changes for optimization

Very little work involved in changing gradients (can proceed on to gradient ascent)

$$\frac{\partial \mathrm{LL}(\beta_0, \beta)}{\partial \beta_j} = \sum_i (1 - p(y_i|\mathbf{x}_i, \beta_0, \beta)) y_i \tilde{x}_{i,j} - \lambda \beta_j$$

and even less in Hessian – only diagonal changes

$$\begin{aligned}
\frac{\partial^2 \mathrm{LL}(\beta_0, \beta)}{\partial^2 \beta_j} &= -\sum_i \tilde{x}_{i,j}^2 p(y_i|\mathbf{x}_i, \beta_0, \beta)(1 - p(y_i|\mathbf{x}_i, \beta_0, \beta)) - \lambda \\
\frac{\partial^2 \mathrm{LL}(\beta_0, \beta)}{\partial^2 \beta_0} &= -\sum_i \tilde{x}_{i,j}^2 p(y_i|\mathbf{x}_i, \beta_0, \beta)(1 - p(y_i|\mathbf{x}_i, \beta_0, \beta))
\end{aligned}$$

where $\tilde{x}_{i,0} = 1$ and $\tilde{x}_{i,j} = x_{i,j}$ for $j > 0$

Newton method – recap

If objective is smooth, has a gradient gradient ($\nabla \mathrm{LL}(\beta_0, \beta)$) and Hessian ($\nabla^2 \mathrm{LL}(\beta_0, \beta)$) then we can apply Newton's method.

The method iterates updates for $k = 1, \ldots, \mathrm{MAXIT}$

$$\begin{bmatrix} \beta_0^{k+1} \\ \beta_1^{k+1} \\ \ldots \\ \beta_p^{k+1} \end{bmatrix} = \begin{bmatrix} \beta_0^{k} \\ \beta_1^{k} \\ \ldots \\ \beta_p^{k} \end{bmatrix} - s \nabla^2 \mathrm{LL}(\beta_0^k, \beta^k)^{-1} \nabla \mathrm{LL}(\beta_0^k, \beta^k)$$

Using least squares to optimize logistic regression

The log likelihood function for logistic regression

$$\mathrm{LL}(\beta_0, \beta) = -\sum_i \log\left\{1 + \exp\left\{-y_i(\beta_0 + \mathbf{x}_i'\beta)\right\}\right\}$$

Quadratic approximation around $\hat{\beta}_0, \hat{\beta}$

$$q_{\hat{\beta}_0,\hat{\beta}}(\mathbf{d}) = \mathrm{LL}(\hat{\beta}_0, \hat{\beta}) + \mathbf{d}^T\nabla\mathrm{LL}(\hat{\beta}_0, \hat{\beta}) + (1/2)\mathbf{d}^T\nabla^2\mathrm{LL}(\hat{\beta}_0, \hat{\beta})\mathbf{d}$$

where

$$\nabla\mathrm{LL}(\hat{\beta}_0, \hat{\beta})_j = \sum_i (1 - p(y_i|\mathbf{x}_i, \hat{\beta}_0, \hat{\beta}))y_i\tilde{x}_{i,j}$$

and

$$[\nabla^2\mathrm{LL}(\hat{\beta}_0, \hat{\beta})]_{j,k} = -\sum_i \tilde{x}_{i,j}\tilde{x}_{i,k}p(y_i|\mathbf{x}_i, \hat{\beta}_0, \hat{\beta})(1 - p(y_i|\mathbf{x}_i, \hat{\beta}_0, \hat{\beta}))$$

with $\tilde{x}_{i,0} = 1$ and $\tilde{x}_{i,j} = x_{i,j}$

Using least squares to optimize logistic regression

After some massaging[1] we can obtain a problem of familiar form

$$\underset{\beta_0,\beta}{\operatorname{argmax}} -(1/2)\sum_i (1-t_i)t_i\left(z_i - \beta_0 - \mathbf{x}_i'\beta\right)^2$$

where $t_i = p(y_i|\mathbf{x}_i, \hat{\beta}_0, \hat{\beta})$ and $z_i = \frac{y_i}{t_i} + \hat{\beta}_0 + \mathbf{x}_i'\hat{\beta}$.

This is a weighted least squares problem since we have different weights associated with each instance.

Coordinate ascent is an option again. Note that the coordinate ascent is done on an approximate objective (our quadratic)!

[1] complete squares while working with d; do change of variables $d = \beta - \hat{\beta}$

Coordinate ascent updates

Take the derivative, set it to 0, solve for update

$$\begin{aligned}\beta_0^* &= \frac{\sum_i (1-t_i)t_i(z_i - \mathbf{x}_i'\beta)}{\sum_i (1-t_i)t_i} \\ \beta_j^* &= \frac{\sum_i (1-t_i)t_i(z_i - \beta_0 - \sum_{k\neq j} x_{i,k}\beta_k)x_{i,j}}{\sum_i (1-t_i)t_i x_{i,j}^2}\end{aligned}$$

Note that β_0^*, β^* yield an improvement for the quadratic approximation, but not necessarily for the original objective.

We still have to do a line search and find $\beta^{\text{new}} \in [\beta, \beta^*]$ and $\beta_0^{\text{new}} \in [\beta_0, \beta_0^*]$ that also yield an improvement in the original objective

$$\begin{aligned}\beta_0^{\text{new}} &= \tau\beta_0 + (1-\tau)\beta_0^* \\ \beta^{\text{new}} &= \tau\beta + (1-\tau)\beta^*\end{aligned}$$

where $\tau \in [0, 1]$

Iterating updates

Note that $t_i = p(y_i|\mathbf{x}_i, \beta_0, \beta)$ and this will change as you update your βs.

A new linear regression problem is constructed on the fly to better approximate the underlying logistic regression problem.

The payoff – after taking all of those derivatives

A week or so ago we labored to get updates for *linear regression* with various penalties/priors.

It turns out that we can convert logistic regression cost into a sequence of weighted linear regression problems[2].

Then we can convert penalized logistic regression into penalized weighted linear regression.

$$
\begin{aligned}
\mathrm{LL}(\beta_0, \beta) &\approx q_{\hat{\beta}_0, \hat{\beta}}(\beta_0 - \hat{\beta}_0, \beta - \hat{\beta}) \\
\mathrm{LL}(\beta_0, \beta) - \lambda \sum_j \beta_j^2 &\approx q_{\hat{\beta}_0, \hat{\beta}}(\beta_0 - \hat{\beta}_0, \beta - \hat{\beta}) - \lambda \sum_j \beta_j^2 \\
\mathrm{LL}(\beta_0, \beta) - \lambda \sum_j |\beta_j| &\approx q_{\hat{\beta}_0, \hat{\beta}}(\beta_0 - \hat{\beta}_0, \beta - \hat{\beta}) - \lambda \sum_j |\beta_j| \\
&\dots
\end{aligned}
$$

[2] Iteratively Reweighted Least Squares (IRLS)

Logistic regression with elastic net

Since elastic net subsumes lasso and ridge we will procede directly to that update using $w_i = (1 - t_i)t_i$

$$\begin{aligned}
\beta_0^* &= \frac{\sum_i w_i(z_i - \mathbf{x}_i'\beta)}{\sum_i w_i} \\
\beta_j^* &= \frac{1}{\sum_i w_i x_{i,j}^2 + \alpha\lambda} S\left(\sum_i w_i(z_i - \beta_0 - \sum_{k \neq j} x_{i,k}\beta_k)x_{i,j}, (1-\alpha)\lambda\right),
\end{aligned}$$

and we can write updates

$$\begin{aligned}
\beta_0^{\text{new}} &= \tau\beta_0 + (1-\tau)\beta_0^* \\
\beta^{\text{new}} &= \tau\beta + (1-\tau)\beta^*,
\end{aligned}$$

that still require a search over τ.

Looking at some code

Prediction error

In the context of classification the prediction error is discrete.

It is a count of test indices ($i \in \mathrm{Test}$) for which
$p(y_i|\mathbf{x}_i, \beta_0, \beta) < 0.5$

So your cross validation scheme should minimize label prediction error.

Think: In some contexts false positives (incorrect predictions of $y = +1$) are more important than false negatives (incorrect prediction of $y = -1$). How would you deal with this?

Types of errors and rates

- False positive – a negative example mispredicted as positive
- False negative – positive example mispredicted as negative
- False positive rate – proportion of negative examples missclasified
- False negative rate – proportion of positive examples missclasified

Rule of thumb for rates: on which class are you making a mistake?

Confusion matrix

	ground truth pos	ground truth neg
predicted pos	TP	FP
predicted neg	FN	TN
	TP+FN	FP+TN

$$
\begin{aligned}
FNR &= FN/(TP + FN) \\
FPR &= FP/(TN + FP) \\
TPR &= TP/(TP + FN) \\
TNR &= TN/(TN + FP)
\end{aligned}
$$

Such tables represent one of many behaviors that your classifier can achieve.

Classifier output

Once we fit a classifier (e.g. logistic regression) on the training set we can evaluate the probability of a positive label for each of the test set instances.

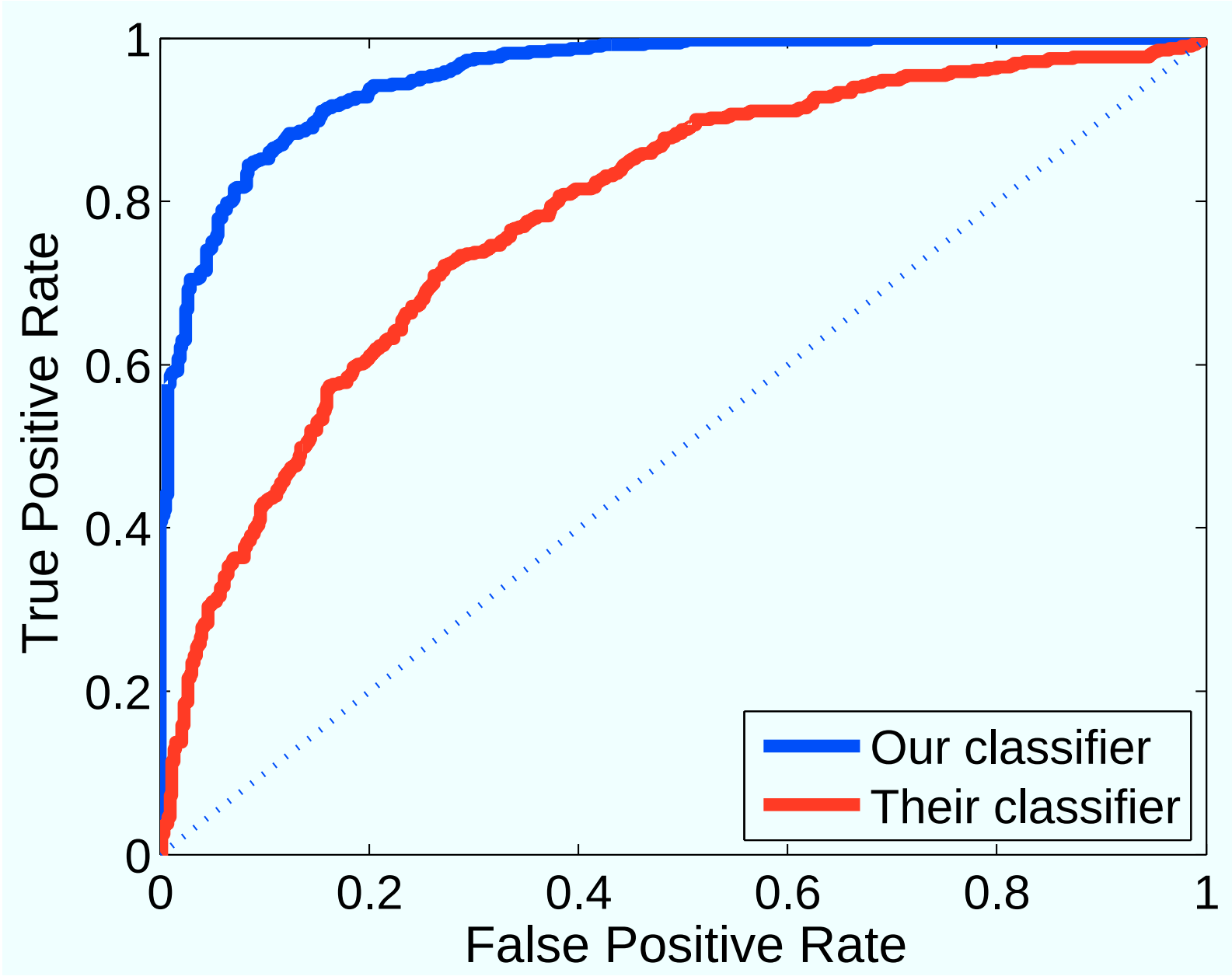
$$q_i = p(y = +1|\mathbf{x}_i), i \in \text{Test}$$

Note that the computation of q_i involves only $\mathbf{x}_i$ and *not* the ground truth labels y_i.

We can use $q_i > 0.5$ to predict label $+1$. But we can use different cutoffs: 0.4,0.7,0.1 ... and these cutoffs produce different error rates.

ROC curve allows us to see them all at once.

An example of ROC curves

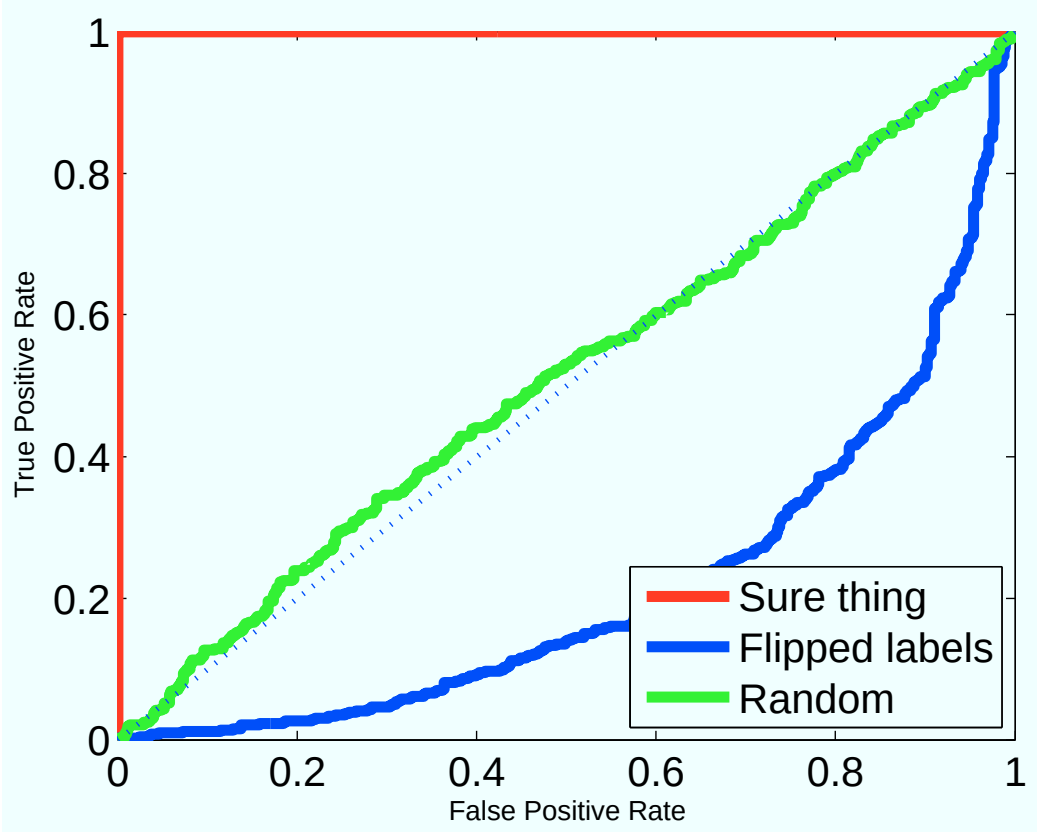


How to compute ROC curve - compact version

Very easy, and if not, you are doing it slowly or wrong.

```
function [tprs,fprs] = roc(q,y)
tprs = zeros(2+length(y),1); % Preallocate space for tps
fprs = zeros(2+length(y),1); % and fps.
[~,ii] = sort(q,'descend');  % Descending ordering for q.
y = y(ii);                   % Reorder y.
totPos = sum(y>0);           % Count positives.
totNeg = sum(y<=0);          % Count negatives.
for i=1:length(y)
    tprs(i+1) = tprs(i) + (y(i)>0);  % count another TP
    fprs(i+1) = fprs(i) + (y(i)<=0); % count another FP
end
tprs(end) = totPos;          % At the end of curve we say
fprs(end) = totNeg;          % everything is positive.
tprs = tprs./totPos;         % Get TPRs out of the counts.
fprs = fprs./totNeg;         % Get FPRs out of the counts.
```

Several ROC curves



Either your problem is super easy or you have a bug.

You've discovered a new random number generator.

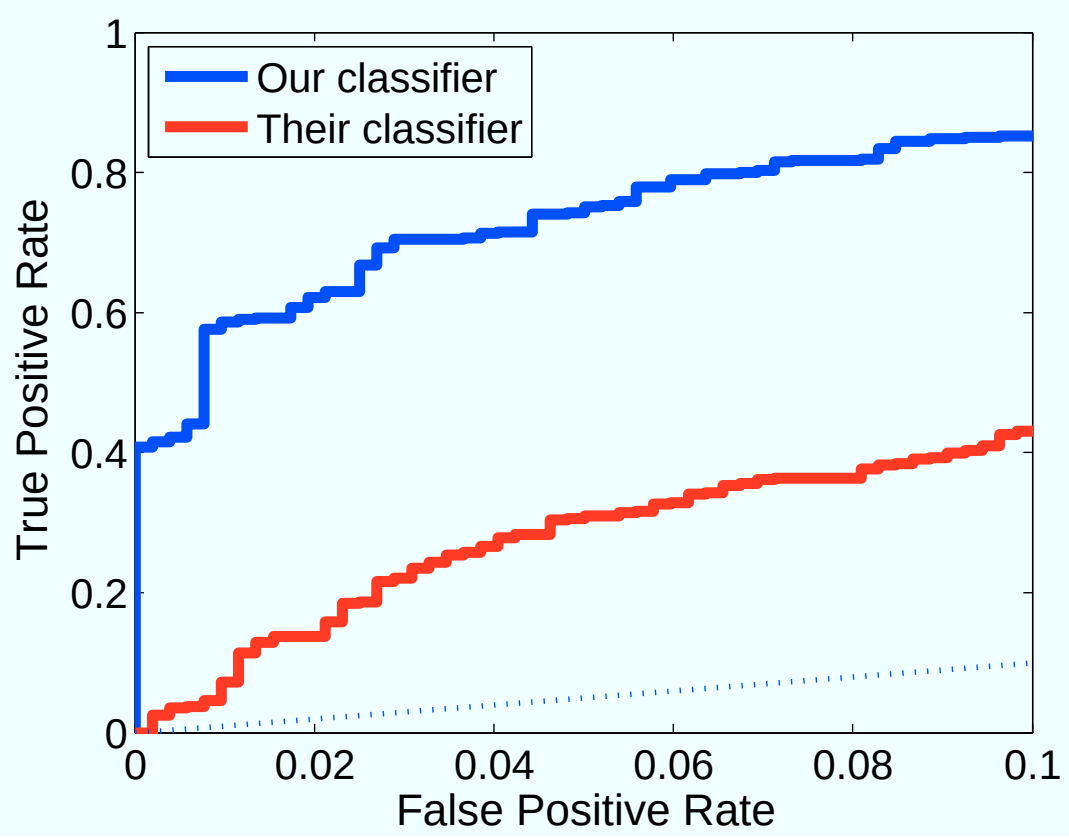
Your classifier is consistently wrong (you flipped the labels, or FPS and TPS values, or sorted q in ascend order.)

Restricting ranges

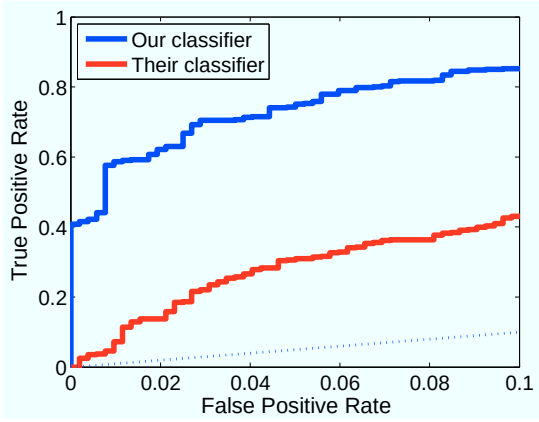
ROC is agnostic to proportion of positives vs. negatives

Some datasets are highly asymmetrical – significantly more negatives than positives.

Restricting range to FPR between 0 and say 0.1 is fine.



Using ROC to make spectacular statements



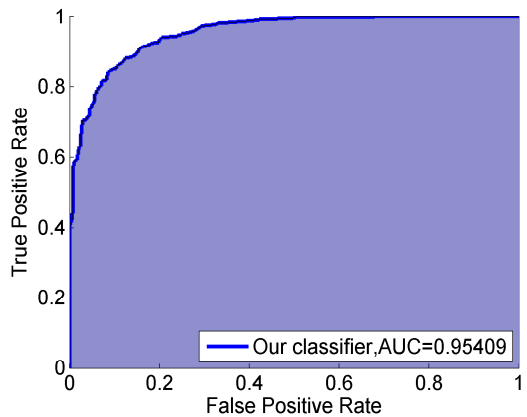
Our classifier yields nearly 100% improvement in TPR over their classifier at 10% FPR. With 0 FP we can achieve 40% TPR; the competitor achieves modest TPR of 2% in a comparable circumstance.

But you should be able to attribute this difference to something (better model, prior, features, ...)

Area under curve

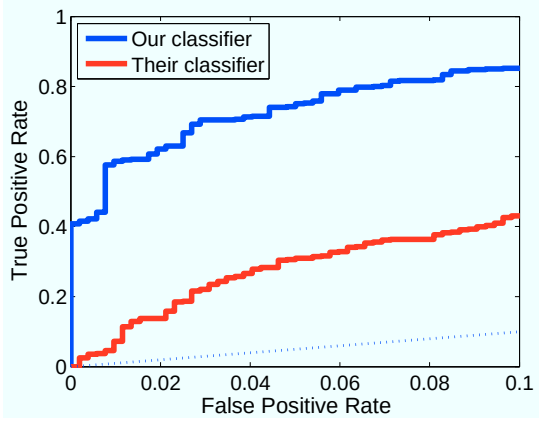
Summarizing ROC curves seems counterintuitive but if you have to spit out a number AUC is it.
Assuming $\mathrm{FPR}(1) = \mathrm{TPR}(1) = 0$ and
$\mathrm{FPR}(n+2) = \mathrm{TPR}(n+2) = 1$ and using trapezoid rule

$$\mathrm{AUC} = (1/2) \sum_{i=2}^{n+1} \left(\mathrm{FPR}_i - \mathrm{FPR}_{i-1}\right) \left(\mathrm{TPR}_i + \mathrm{TPR}_{i-1}\right)$$



One interpretation for AUC: probability of randomly chosen $+1$ example occuring before randomly chosen -1 example in ordering based on q

$\mathrm{AUC}_{R\%}$



If you are focusing on a particular range of FPR $[0, R/100]$

$$\mathrm{AUC}_{C\%} = (1/2) \sum_{i:FPR(i) \leq R/100} (\mathrm{FPR}_i - \mathrm{FPR}_{i-1})\,(\mathrm{TPR}_i + \mathrm{TPR}_{i-1})$$

so for $R = 10$ you get $AUC_{10\%}$.

Application of logistic regression in GWAS studies

The goal of genome wide association studies is to discover associations between genotypic and phenotypic variability.

An example of phenotype: development of type 2 diabetes

An example of associated genotype: SNPs in a gene SLC30A8 changes coding for 325aa (Arg to Trp)

GWAS studies data

The number of SNPs analyzed in GWAS studies varies ($\sim$ 500k Illumina,$\sim$ 200k ImmunoChip). This can result in upwards of 10^6 features, and this is p number of features

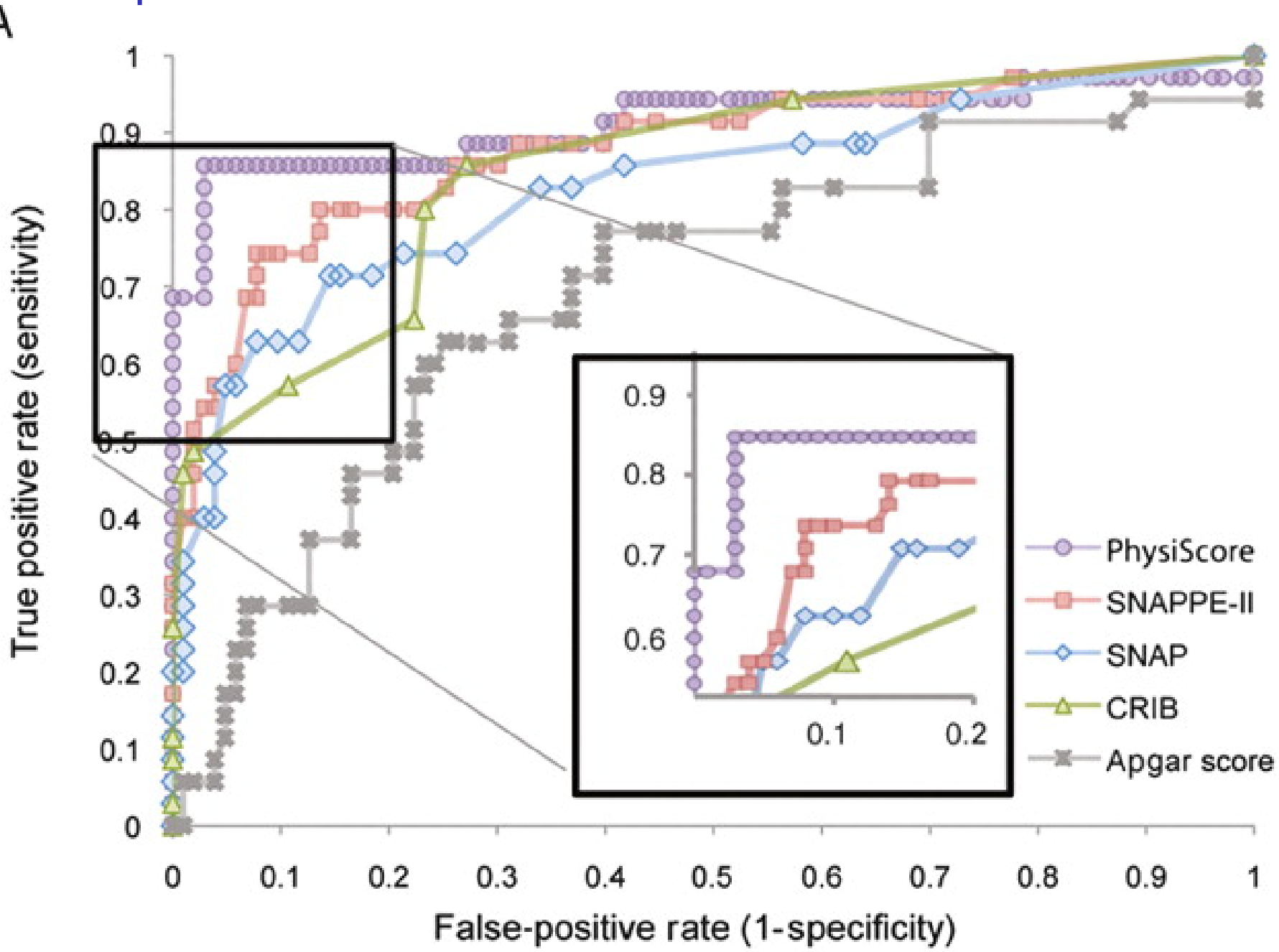
The number of study participants n is much smaller, at best thousands, and this is n number of samples.

The target variable is binary: presence or absence of phenotype.

Clearly this is the case of $p \gg n$ so we need to perform some sort of feature selection.

You have learned about a method that can be used to model binary variables but also enforces sparsity in predictors – logistic regression with Lasso.

ROC comparison



- Two more algorithms for logistic regression
  - Newton algorithm
  - Hybrid of sequential quadratic approximation and coordinate ascent
- Assessing predictive performance