Machine Learning Methods for Gene Expression Data

Day 3

Dennis Wylie, UT Bioinformatics Consulting Group

May 21, 2016

Outline

Linear Models for Univariate Feature Selection

Most common univariate filter is t-test (or F-test if more than 2 groups) originating from model

$$\mathbb{P}(X_g = x \mid Y = y) = \frac{1}{\sqrt{2\pi\sigma_g^2}} \exp\left[\frac{(x-\mu_{yg})^2}{2\sigma_g^2}\right]$$

Note that we are now considering conditional probabilities of the X_g given Y ...

...and that we are considering the various X_g separately ("univariate" analysis; not necessarily a very good approximation to reality, but at least a tractable one).

Linear Models

Alternately, this model may be described by

$$X_g = \mu_{0g} + (\mu_{1g} - \mu_{0g})Y + \sigma_g \epsilon_g$$

with $\epsilon_g \sim \mathcal{N}(0, 1)$, implying

$$\mathbb{E}(X_g \mid Y = y) = \mu_{yg}$$
$$\mathbb{V}(X_g \mid Y = y) = \sigma_g^2$$

Can then rank features based on $|t_g|$ where:

$$t_g = \frac{\hat{\mu}_{0g} - \hat{\mu}_{1g}}{\hat{\sigma}_g \sqrt{\frac{1}{n_0} + \frac{1}{n_1}}}$$

Linear Models

Alternately, this model may be described by

$$X_g = \mu_{0g} + (\mu_{1g} - \mu_{0g})Y + \sigma_g \epsilon_g$$

with $\epsilon_g \sim \mathcal{N}(0,1)$, implying

$$\mathbb{E}(X_g \mid Y = y) = \mu_{yg}$$
$$\mathbb{V}(X_g \mid Y = y) = \sigma_g^2$$

Can then rank features based on $|t_g|$ where:

$$t_g = \frac{\hat{\mu}_{0g} - \hat{\mu}_{1g}}{\hat{\sigma}_g \sqrt{\frac{1}{n_0} + \frac{1}{n_1}}}$$

While X_g is linear in Y here, it is linearity in μ_{yg} that is generally implied by "linear model" in statistics.

Linear Models

Alternately, this model may be described by

$$X_g = \mu_{0g} + (\mu_{1g} - \mu_{0g})Y + \sigma_g \epsilon_g$$

with $\epsilon_g \sim \mathcal{N}(0, 1)$, implying

$$\mathbb{E}(X_g \mid Y = y) = \mu_{yg}$$
$$\mathbb{V}(X_g \mid Y = y) = \sigma_g^2$$

Can then rank features based on $|t_g|$ where:

$$t_g = \frac{\hat{\mu}_{0g} - \hat{\mu}_{1g}}{\hat{\sigma}_g \sqrt{\frac{1}{n_0} + \frac{1}{n_1}}}$$

While X_g is linear in Y here, it is linearity in μ_{yg} that is generally implied by "linear model" in statistics.

We will see later that with a few modifications linear models can be good candidates for modeling Y based on $\mathbf{X}$ too.

Student's t-Test and Pearson's Correlation

We mentioned correlation coefficients when discussing clustering (though there correlations were between samples and across genes).

t_g is directly related to Pearson's correlation between X_g and the binary outcome variable Y:

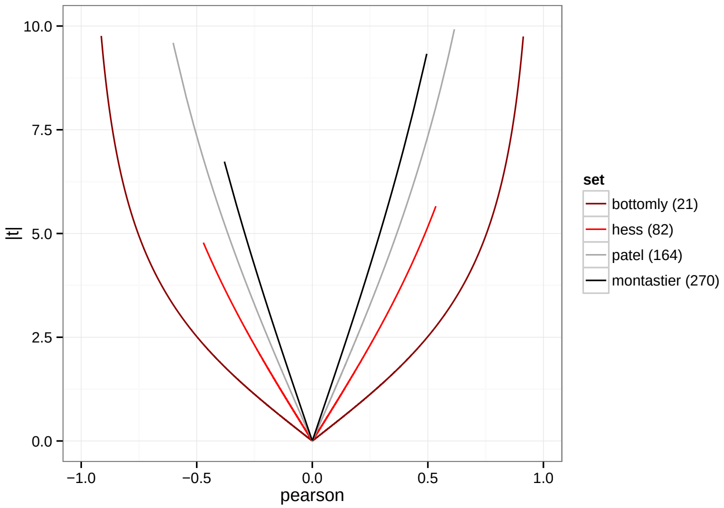
$$|t_g| = \sqrt{n-2}\sqrt{\frac{\hat{\rho}^2(\underline{x}_g, \underline{y})}{1-\hat{\rho}^2(\underline{x}_g, \underline{y})}}$$

or

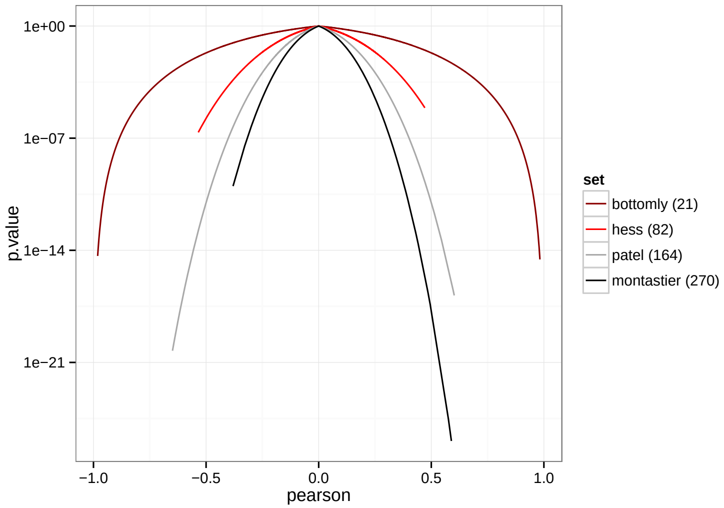
$$\begin{aligned}\hat{\rho}(\underline{x}_g, \underline{y}) &= \frac{\delta\underline{x}_g \cdot \delta\underline{y}}{\|\delta\underline{x}_g\|\|\delta\underline{y}\|} \\ &= \frac{\text{sign}(t_g)}{\sqrt{1+\frac{n-2}{t_g^2}}}\end{aligned}$$

where $\delta\underline{x} = \underline{x} - \sum_{i=1}^{n} x_i$.

Student's *t*-Test and Pearson's Correlation



Student's *t*-Test and Pearson's Correlation



Multivariate Filtration by Linear Algebra

Let's say we know we like gene g. How can we find another gene h that complements g well?

Consider the **projection operator** $\text{proj}_{\underline{u}}$:

$$\text{proj}_{\underline{u}}(\underline{v}) = \frac{\underline{u} \cdot \underline{v}}{\|\underline{u}\|^2}\, \underline{u}$$

and the associated operator for "projecting $\underline{u}$ out"

$$\text{projout}_{\underline{u}}(\underline{v}) = \underline{v} - \frac{\underline{u} \cdot \underline{v}}{\|\underline{u}\|^2}\, \underline{u}$$

The gene h which maximizes R^2 when paired with g in the linear model $Y = \beta_0 + \beta_g X_g + \beta_h X_h$ can be found as

$$h = \arg\max_{h'} \hat{\rho}\left(\text{projout}_{\delta \underline{x}_g}(\delta \underline{x}_{h'}),\ \text{projout}_{\delta \underline{x}_g}(\delta \underline{y})\right)$$

Multivariate Filtration by Linear Algebra

In matrix notation,

$$\text{projout}_{\underline{u}}(\underline{\mathbf{X}}) = \underline{\mathbf{P}}^{(\underline{u})}\underline{\mathbf{X}}$$

$$\text{projout}_{\underline{u}}(\underline{y}) = \underline{\mathbf{P}}^{(\underline{u})}\underline{y}$$

where

$$p_{ij}^{(\underline{u})} = \delta_{ij} - \frac{u_i u_j}{\|\underline{u}\|^2}$$

Now we're back to sorting genes by correlation except that we've replaced $\underline{\mathbf{X}}$ and $\underline{y}$ with $\underline{\mathbf{P}}^{(\delta \underline{x}_g)}\delta\underline{\mathbf{X}}$ and $\underline{\mathbf{P}}^{(\delta \underline{x}_g)}\delta\underline{y}$.

Implemented in `MaclearnUtilities.(R|py)` as `gramSchmidtSelect`.

*$\delta\underline{\mathbf{X}}$ and $\delta\underline{y}$ are again defined by:

$$\delta x_{ig} = x_{ig} - \sum_{i=1}^{n} x_{ig}$$

$$\delta y_i = y_i - \sum_{i=1}^{n} y_i$$

Regularization

Linear models can be made robust in high-dimensional settings using **regularization**.

Unregularized linear regression uses maximum likelihood to select coefficients β_g; fit by ordinary least-squares (OLS) estimator:

$$\hat{\boldsymbol{\beta}}_{\mathrm{OLS}} = \arg\min_{\boldsymbol{\beta}} \sum_i (y_i - \boldsymbol{\beta} \cdot \mathbf{x}_i)^2$$

Bayesian derivation of OLS uses uniform prior on $\boldsymbol{\beta}$.

Regularization

Linear models can be made robust in high-dimensional settings using **regularization**.

Unregularized linear regression uses maximum likelihood to select coefficients β_g; fit by ordinary least-squares (OLS) estimator:

$$\hat{\boldsymbol{\beta}}_{\mathrm{OLS}} = \arg\min_{\boldsymbol{\beta}} \sum_i (y_i - \boldsymbol{\beta} \cdot \mathbf{x}_i)^2$$

Bayesian derivation of OLS uses uniform prior on $\boldsymbol{\beta}$.

If instead Gaussian prior (**L2 regression**) imposed on $\boldsymbol{\beta}$, maximum a posteriori (MAP) estimator is (Park & Casella (2008)):

$$\hat{\boldsymbol{\beta}}_{\mathrm{L2}} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_i (y_i - \boldsymbol{\beta} \cdot \mathbf{x}_i)^2 + \phi_2 \sum_g \beta_g^2 \right\}$$

where the regularization parameter ϕ_2 determined by variance of the Gaussian prior.

Lasso regression

Alternatively, use of Laplace prior for $\boldsymbol{\beta}$ yields MAP estimator (Park & Casella (2008)):

$$\hat{\boldsymbol{\beta}}_{\mathsf{L1}} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_i (y_i - \boldsymbol{\beta} \cdot \mathbf{x}_i)^2 + \phi_1 \sum_g |\beta_g| \right\}$$

where now ϕ_1 is determined by width of the Laplace prior.

Lasso regression

Alternatively, use of Laplace prior for $\boldsymbol{\beta}$ yields MAP estimator (Park & Casella (2008)):

$$\hat{\boldsymbol{\beta}}_{\mathrm{L1}} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_i (y_i - \boldsymbol{\beta} \cdot \mathbf{x}_i)^2 + \phi_1 \sum_g |\beta_g| \right\}$$

where now ϕ_1 is determined by width of the Laplace prior.

As ϕ_1 is increased, progressively more β_g set to zero, de-selecting the corresponding features (Tibshirani (1996))— **L1, or LASSO, regression** is an embedded feature selection method.

Lasso regression

Alternatively, use of Laplace prior for $\boldsymbol{\beta}$ yields MAP estimator (Park & Casella (2008)):

$$\hat{\boldsymbol{\beta}}_{\mathsf{L1}} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_i (y_i - \boldsymbol{\beta} \cdot \mathbf{x}_i)^2 + \phi_1 \sum_g |\beta_g| \right\}$$

where now ϕ_1 is determined by width of the Laplace prior.

As ϕ_1 is increased, progressively more β_g set to zero, de-selecting the corresponding features (Tibshirani (1996))— **L1, or LASSO, regression** is an embedded feature selection method.

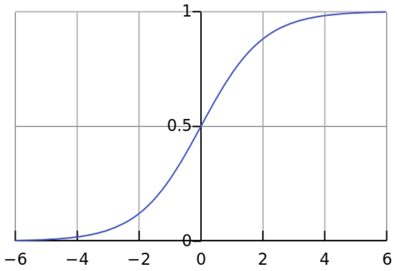
Both L1/LASSO and L2/ridge logistic regression are implemented in the `R` package `glmnet` function `glmnet` using argument `family="binomial"`.

What's Linear?

In the context of classification, "linear model" often means

$$\mathbb{P}(Y = 1 \mid \mathbf{X} = \mathbf{x}) = \text{expit}(\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x})$$

where expit: $\mathbb{R} \to (0, 1)$ defined by $\text{expit}(u) = \frac{\exp(u)}{1+\exp(u)}$ is the logistic, or inverse-logit, function.



Linear Classifiers

In the context of classification, "linear model" often means

$$\mathbb{P}(Y = 1 \mid \mathbf{X} = \mathsf{x}) = \text{expit}(\beta_0 + \boldsymbol{\beta} \cdot \mathsf{x})$$

where expit: $\mathbb{R} \to (0, 1)$ defined by $\text{expit}(u) = \frac{\exp(u)}{1+\exp(u)}$ is the logistic, or inverse-logit, function.

Two main classes of such linear classification models:

1. **linear discriminant analysis (LDA)** (generative): adds assumption $\mathbb{P}(\mathbf{X} = \mathsf{x} | Y = y) \sim \mathcal{N}(\boldsymbol{\mu}_y, \Sigma)$, fit by maximizing joint likelihood $\mathbb{P}(\mathbf{X}, y)$.
2. **logistic regression** (discriminative): makes no explicit distributional assumptions about $\mathbf{X}$, instead fit by maximizing conditional $\mathbb{P}(Y \mid \mathbf{X})$ over training set.

Logistic Regression

Logistic regression is widely used because:

1. it's familiar,
2. it has often worked well in the past,
3. $\beta_0 + \boldsymbol{\beta} \cdot \mathbf{x} = \operatorname{logit}(\mathbb{P}(y \mid \mathbf{x})) = \log\left(\frac{\mathbb{P}}{1-\mathbb{P}}\right)$ may be interpreted as a "log-odds;" such quantities are often used in many areas of statistics,
4. there are very efficient techniques for fitting logistic regression models, and
5. logistic regression is a type of **generalized linear model (GLM)**; GLMs have many useful statistical properties that make them easy to work with and to extend (e.g., by adding regularization).

Note that (4) above may be seen as largely a consequence of (5).

LDA

Linear discriminant analysis assumes that the class densities $P(\mathbf{X} = \mathbf{x} \mid Y = y)$ are Gaussian with

- common covariance matrix Σ
- but distinct, class-dependent, means $\boldsymbol{\mu}_y$:

$$\mathbf{X} \mid_{Y=y} \sim \mathcal{N}(\boldsymbol{\mu}_y, \Sigma)$$

LDA

Linear discriminant analysis assumes that the class densities $P(\mathbf{X} = \mathbf{x} \mid Y = y)$ are Gaussian with

- common covariance matrix Σ
- but distinct, class-dependent, means $\boldsymbol{\mu}_y$:

$$\mathbf{X} \mid_{Y=y} \sim \mathcal{N}(\boldsymbol{\mu}_y, \Sigma)$$

or

$$\mathbb{P}(\mathbf{X} = \mathbf{x} \mid Y = y) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_y)^\mathsf{T}\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_y)\right]$$

LDA

Linear discriminant analysis assumes that the class densities $P(\mathbf{X} = \mathbf{x} \mid Y = y)$ are Gaussian with

- common covariance matrix Σ
- but distinct, class-dependent, means $\boldsymbol{\mu}_y$:

$$\mathbf{X} \mid_{Y=y} \sim \mathcal{N}(\boldsymbol{\mu}_y, \Sigma)$$

or

$$\mathbb{P}(\mathbf{X} = \mathbf{x} \mid Y = y) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_y)^{\mathsf{T}}\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_y)\right]$$

Then, applying Bayes' theorem

$$\mathbb{P}(Y = y \mid \mathbf{X} = \mathbf{x}) = \frac{\pi_y\, \mathbb{P}(\mathbf{X} = \mathbf{x} \mid Y = y)}{\sum_{y'} \pi_{y'}\, \mathbb{P}(\mathbf{X} = \mathbf{x} \mid Y = y')}$$

where $\pi_y = \mathbb{P}(Y = y)$ is the prior probability of class y.

LDA Compared to Logistic Regression

Shrinkage and Diagonal LDA

LDA can also be regularized:
Instead of using the maximum likelihood estimator for the covariance matrix Σ, off-diagonal entries Σ_{gh} are shrunk by a regularization parameter towards 0 (`R` package `sda`).

Shrinkage and Diagonal LDA

LDA can also be regularized:
Instead of using the maximum likelihood estimator for the covariance matrix Σ, off-diagonal entries Σ_{gh} are shrunk by a regularization parameter towards 0 (`R` package `sda`).

In most extreme form, shrinkage LDA sets all $\Sigma_{gh} = 0$ $(g \neq h)$.

Since Σ is now a diagonal matrix, this is referred to as diagonal LDA or **DLDA**. DLDA has been been found to be particularly useful for gene expression data (Dudoit *et al.* (2002)).

A nice implementation of DLDA can be found in the `dlda` function in the `R` package `sparsediscrim`.

Naive Bayes

"Naive Bayes" describes a family of classification methods sharing a common assumption:

$$\mathbb{P}(\mathbf{X} = \mathbf{x} \mid Y = y) = \prod_g \mathbb{P}(X_g = x_g \mid Y = y)$$

which can be substituted into Bayes' formula to yield:

$$\mathbb{P}(Y = y \mid \mathbf{X} = \mathbf{x}) = \frac{\pi_y \prod_g \mathbb{P}(X_g = x_g \mid Y = y)}{\sum_{y'} \pi_{y'} \prod_g \mathbb{P}(X_g = x_g \mid Y = y')}$$

Naive Bayes

"Naive Bayes" describes a family of classification methods sharing a common assumption:

$$\mathbb{P}(\mathbf{X} = \mathbf{x} \mid Y = y) = \prod_g \mathbb{P}(X_g = x_g \mid Y = y)$$

which can be substituted into Bayes' formula to yield:

$$\mathbb{P}(Y = y \mid \mathbf{X} = \mathbf{x}) = \frac{\pi_y \prod_g \mathbb{P}(X_g = x_g \mid Y = y)}{\sum_{y'} \pi_{y'} \prod_g \mathbb{P}(X_g = x_g \mid Y = y')}$$

DLDA is actually a form of naive Bayes classification in which the additional assumption of linearity is posed.

Naive Bayes: does it work?

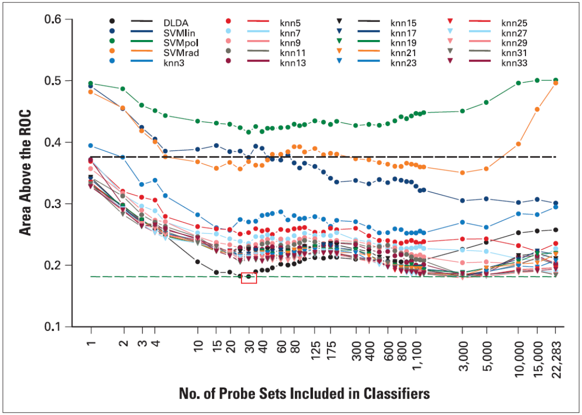


Fig 1. Mean area above the receiver operating characteristic (ROC) curves plotted against the number of top genes included in the classifiers. Complete 5-fold cross validation results (means over the 100 iterations) for 20 classifier algorithms including different numbers of probe sets (39 gene sets) are shown. Green and black horizontal dotted lines indicate the mean +/− 2SD for the nominally best Diagonal Linear Discriminant Analysis (DLDA) classifier with 30 probe sets that was selected for independent validation. polynomial kernels (SVM), and K-nearest neighbor

Taken from Hess *et al.* (2006).

Naive Bayes: does it work?

The conditional independence assumption is basically never true, but:

1. frequently not enough data to accurately assess true inter-feature covariance, so that attempts to do so just lead to overfitting, and

Naive Bayes: does it work?

The conditional independence assumption is basically never true, but:

1. frequently not enough data to accurately assess true inter-feature covariance, so that attempts to do so just lead to overfitting, and
2. while this assumption tends to lead to **overconfident** classifiers—probability scores very near 0 or 1 even when wrong—it still often leads to **accurate** classifiers—most calls aren't wrong.

Naive Bayes: does it work?

The conditional independence assumption is basically never true, but:

1. frequently not enough data to accurately assess true inter-feature covariance, so that attempts to do so just lead to overfitting, and
2. while this assumption tends to lead to **overconfident** classifiers—probability scores very near 0 or 1 even when wrong—it still often leads to **accurate** classifiers—most calls aren't wrong.
3. Naive Bayes methods work well when either:
   - features truly are independent within each class *or*
   - features are very tightly correlated (may actually be more relevant in gene expression context) (Rish *et al.* (2001)).

Bias-Variance Tradeoff

From Wikipedia (http://en.wikipedia.org/wiki/Bias-variance_tradeoff):

> The bias–variance tradeoff (or dilemma) is the problem of simultaneously minimizing two sources of error that prevent supervised learning algorithms from generalizing beyond their training set:
>
> **bias** error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (**underfitting**).
>
> **variance** error from sensitivity to small fluctuations in the training set. High variance can cause **overfitting**: modeling the random noise in the training data, rather than the intended outputs.

References I

Dudoit, Sandrine, Fridlyand, Jane, & Speed, Terence P. 2002. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, **97**(457), 77–87.

Hess, Kenneth R, Anderson, Keith, Symmans, W Fraser, Valero, Vicente, Ibrahim, Nuhad, Mejia, Jaime A, Booser, Daniel, Theriault, Richard L, Buzdar, Aman U, Dempsey, Peter J, *et al.* . 2006. Pharmacogenomic predictor of sensitivity to preoperative chemotherapy with paclitaxel and fluorouracil, doxorubicin, and cyclophosphamide in breast cancer. *Journal of Clinical Oncology*, **24**(26), 4236–4244.

Park, T., & Casella, G. 2008. The Bayesian lasso. *Journal of the American Statistical Association*, **103**(482), 681–686.

Rish, Irina, Hellerstein, Joseph, & Thathachar, Jayram. 2001. An analysis of data characteristics that affect naive Bayes performance. *IBM TJ Watson Research Center*, **30**.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.