Machine Learning Methods

Dennis Wylie, UT CBRS Bioinformatics Consulting Group

What is Machine Learning?

Perhaps better thought of as "algorithms for learning."

Such algorithms may also be referred to as modeling strategies

$$M$$

which, when provided training data

$$D_{\text{train}}$$

from some particular experiment, "learn" parameters

$$\boldsymbol{\theta}$$

such that the pair

$$(M, \boldsymbol{\theta})$$

can be used to predict likely observations

$$D_{\text{other}}$$

from similar experiments.

Unsupervised Learning

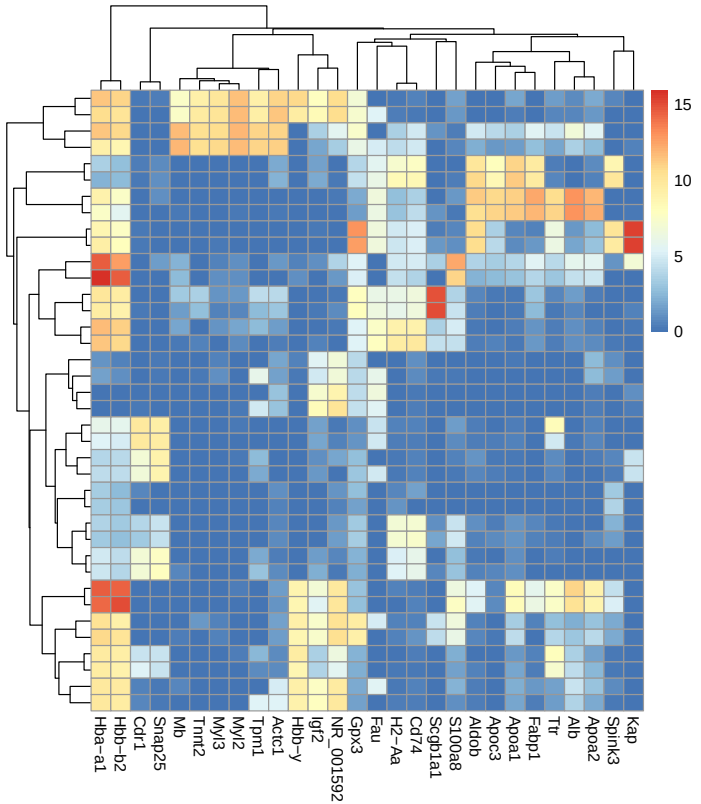
No pre-specified outcome/target to predict.

Identify interesting patterns in attribute vector x.

What "patterns?"

- clusters of "similar" samples or attributes
- relationships between attributes
  - underyling latent factors

Useful for **dimensionality reduction**.

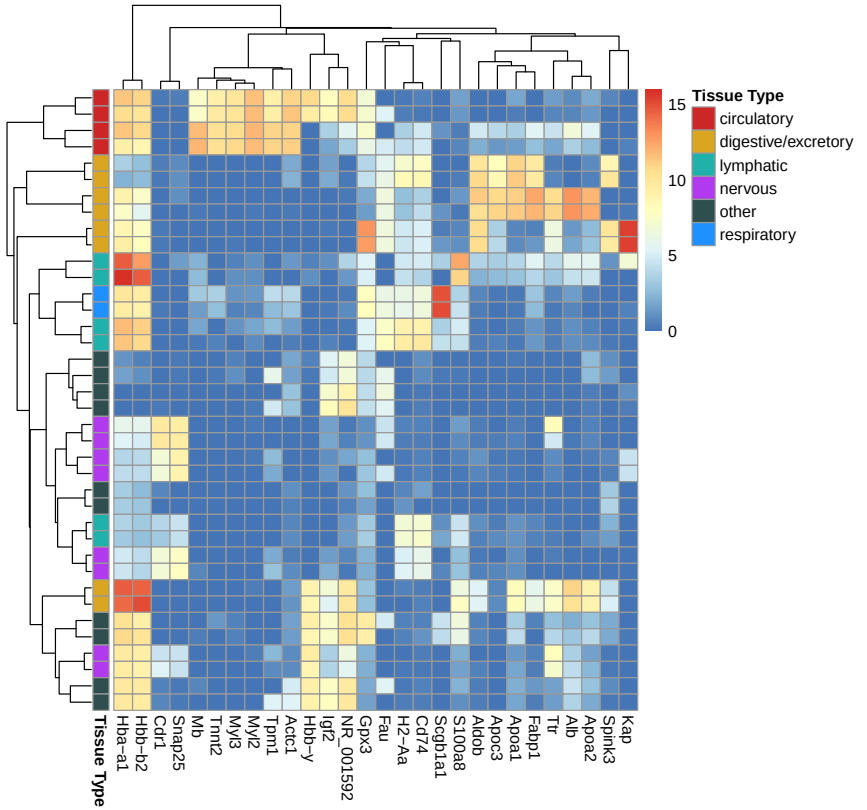


Supervised Learning

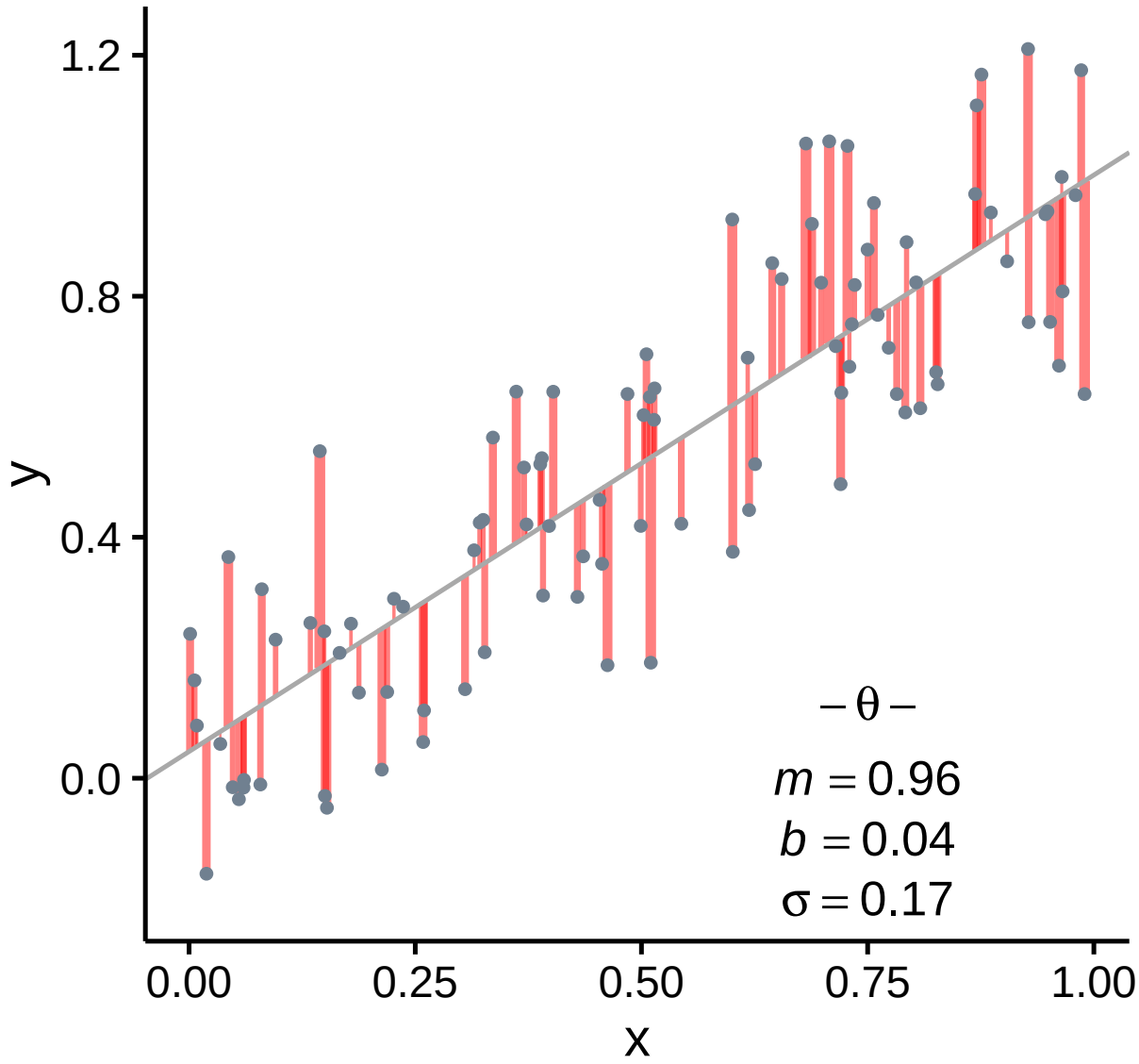
Use attributes x to predict target y.

y could be:

- categorical label (**classification**)
- continuous number (**regression**)



Supervised Learning: A Familiar Example



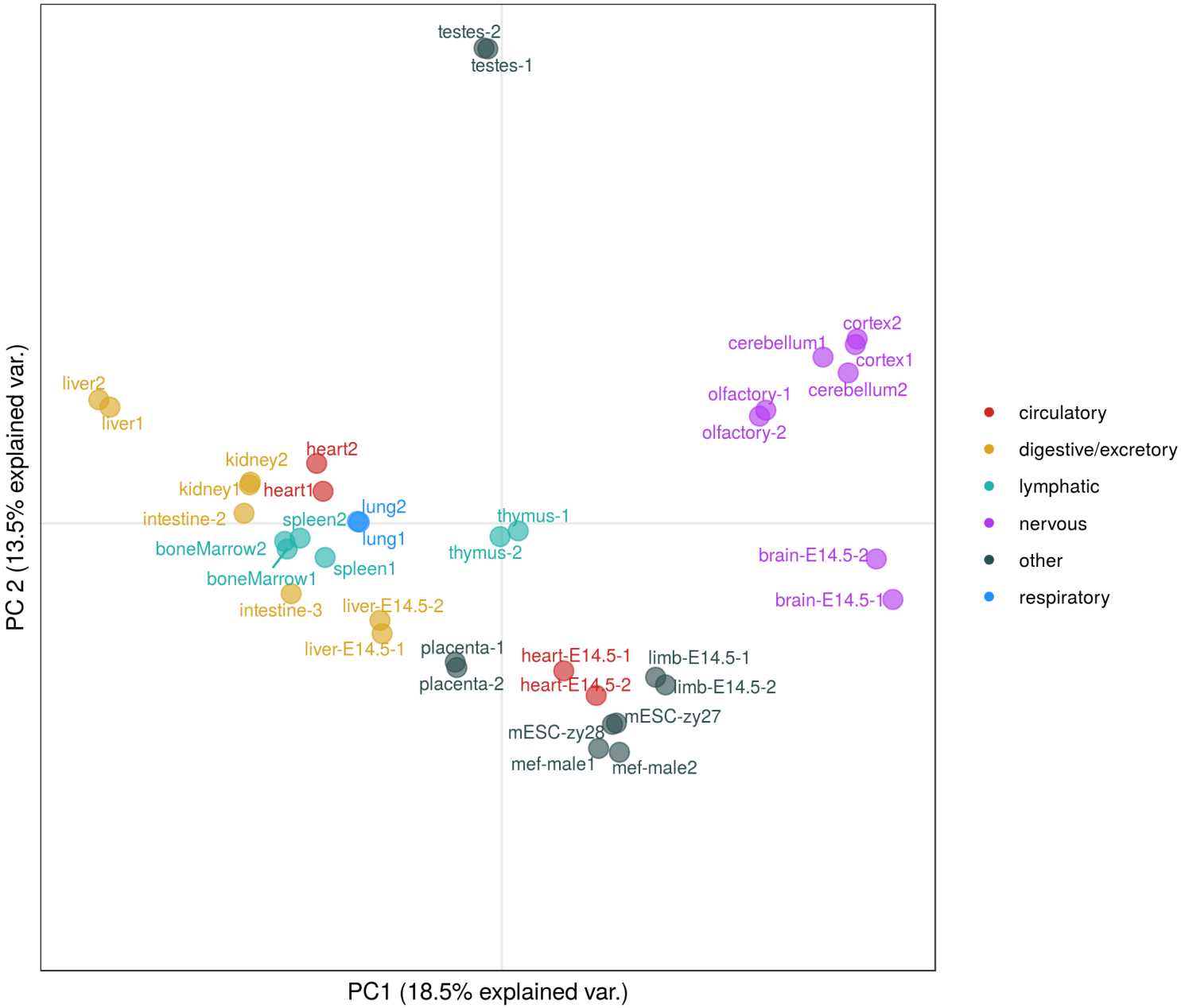
M : OLS regression

$\boldsymbol{\theta} = (m, b, \sigma)$

$y_i = mx_i + b + \sigma\epsilon_i$

$\epsilon_i \sim \mathcal{N}(0, 1)$

PCA

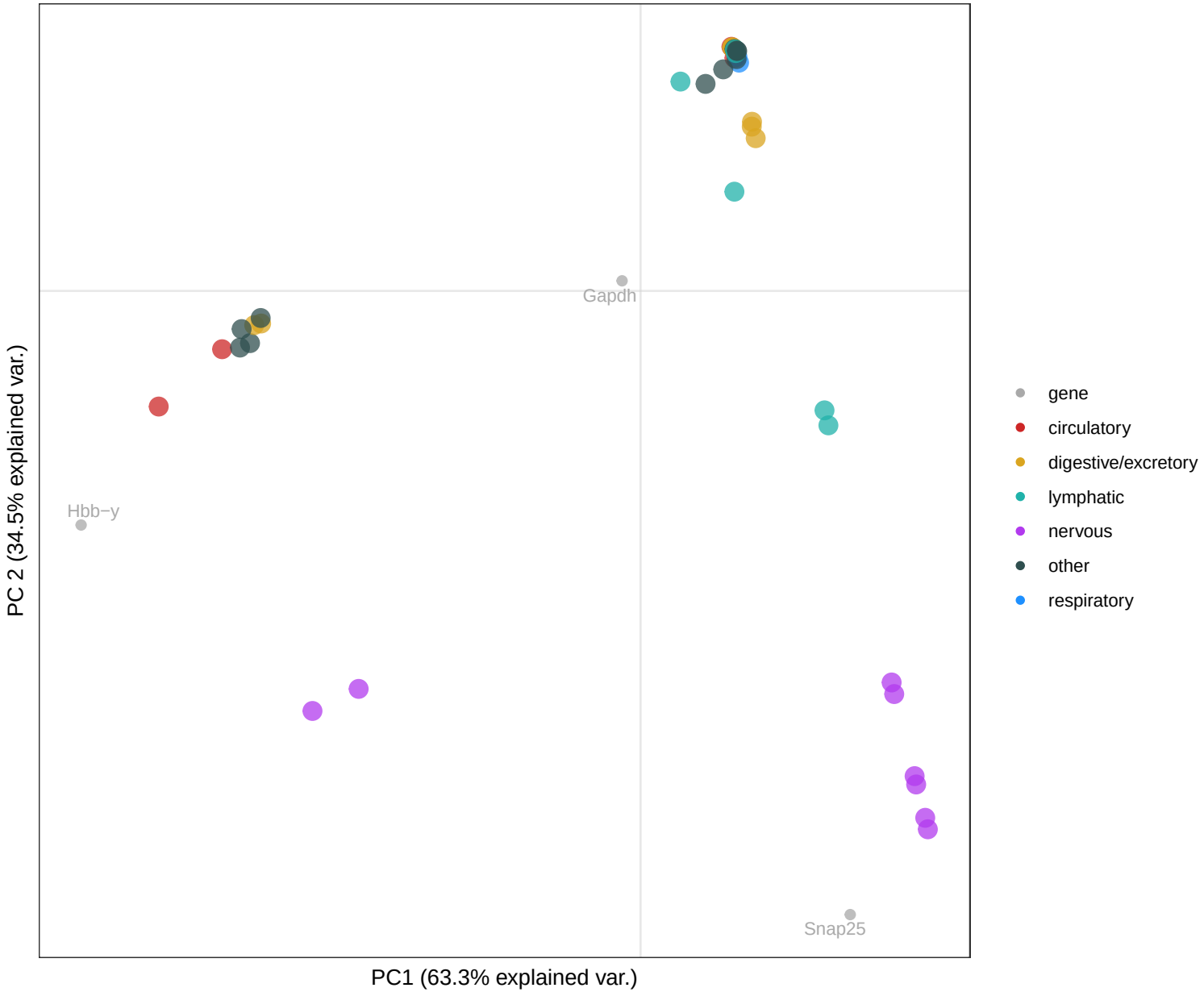


What is PCA?

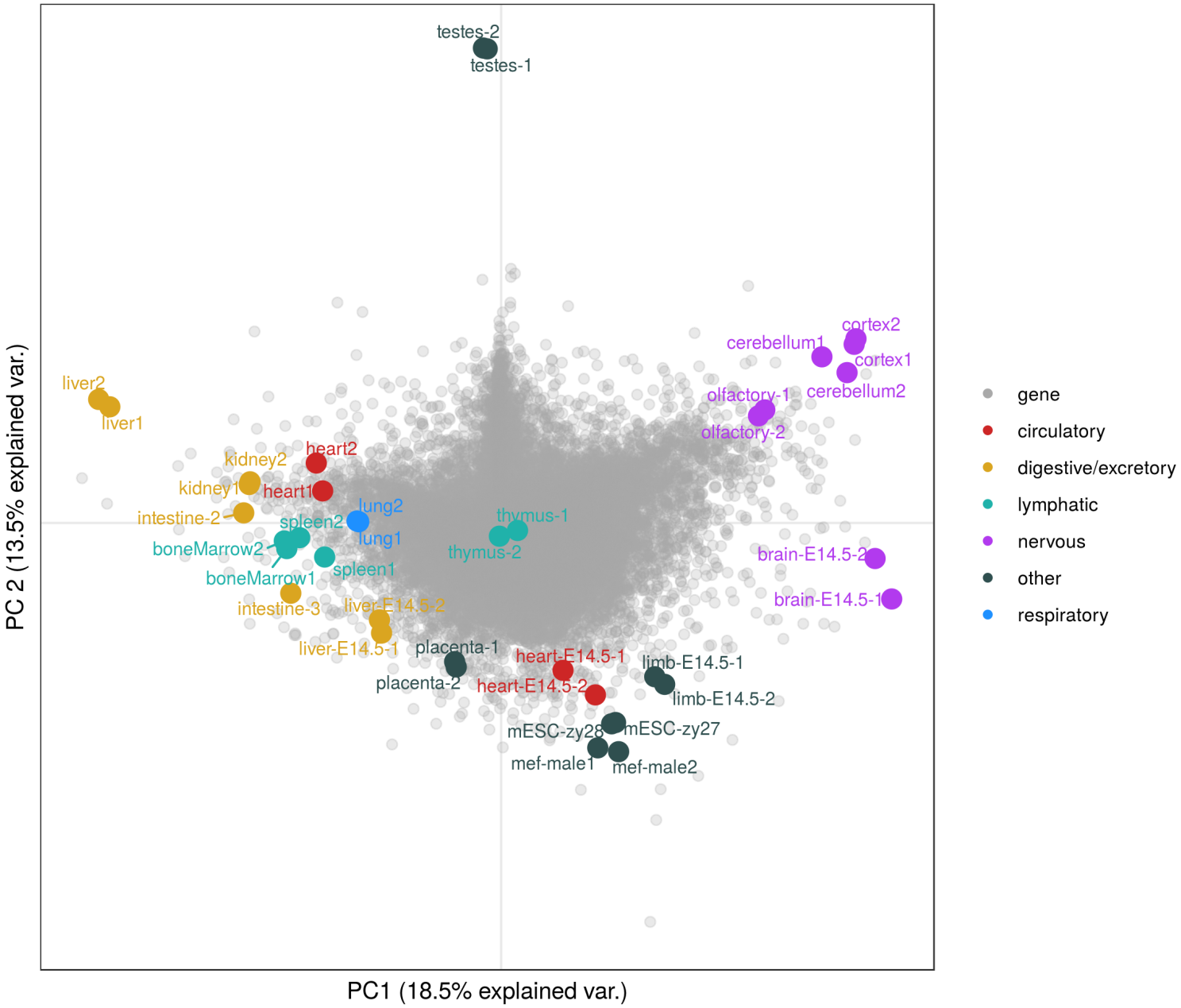
From https://en.wikipedia.org/wiki/Principal_component_analysis:

> . . . a statistical procedure that uses an **orthogonal transformation** to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components . . .
>
> This transformation is defined in such a way that the **first principal component** has the **largest possible variance** (that is, accounts for as much of the variability in the data as possible),
>
> and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components.
>
> The resulting vectors are an uncorrelated orthogonal basis set. The principal components are orthogonal because they are the eigenvectors of the covariance matrix, which is symmetric.
>
> PCA is **sensitive to the relative scaling** of the original variables.

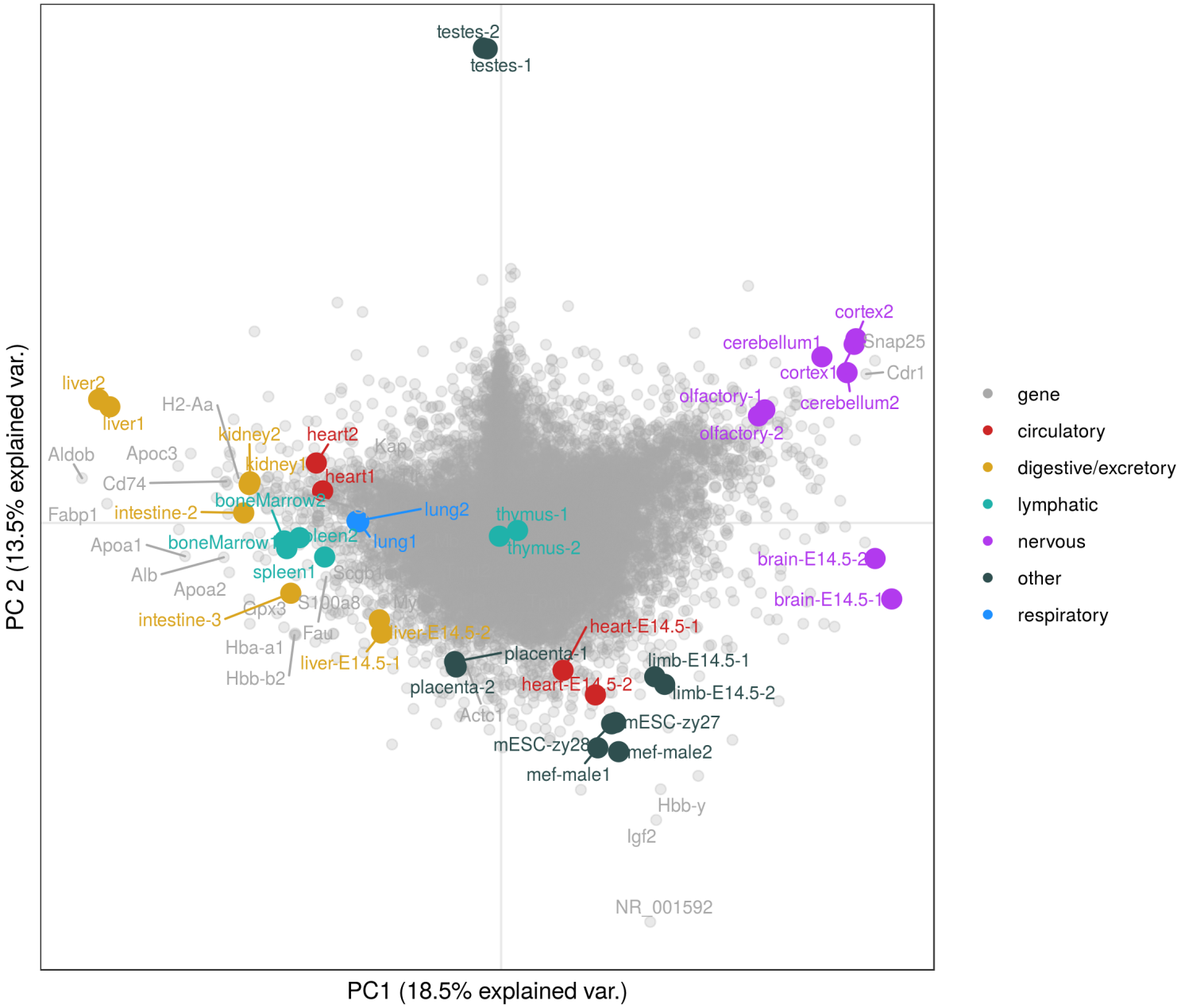
PCA Biplot: 3 Genes



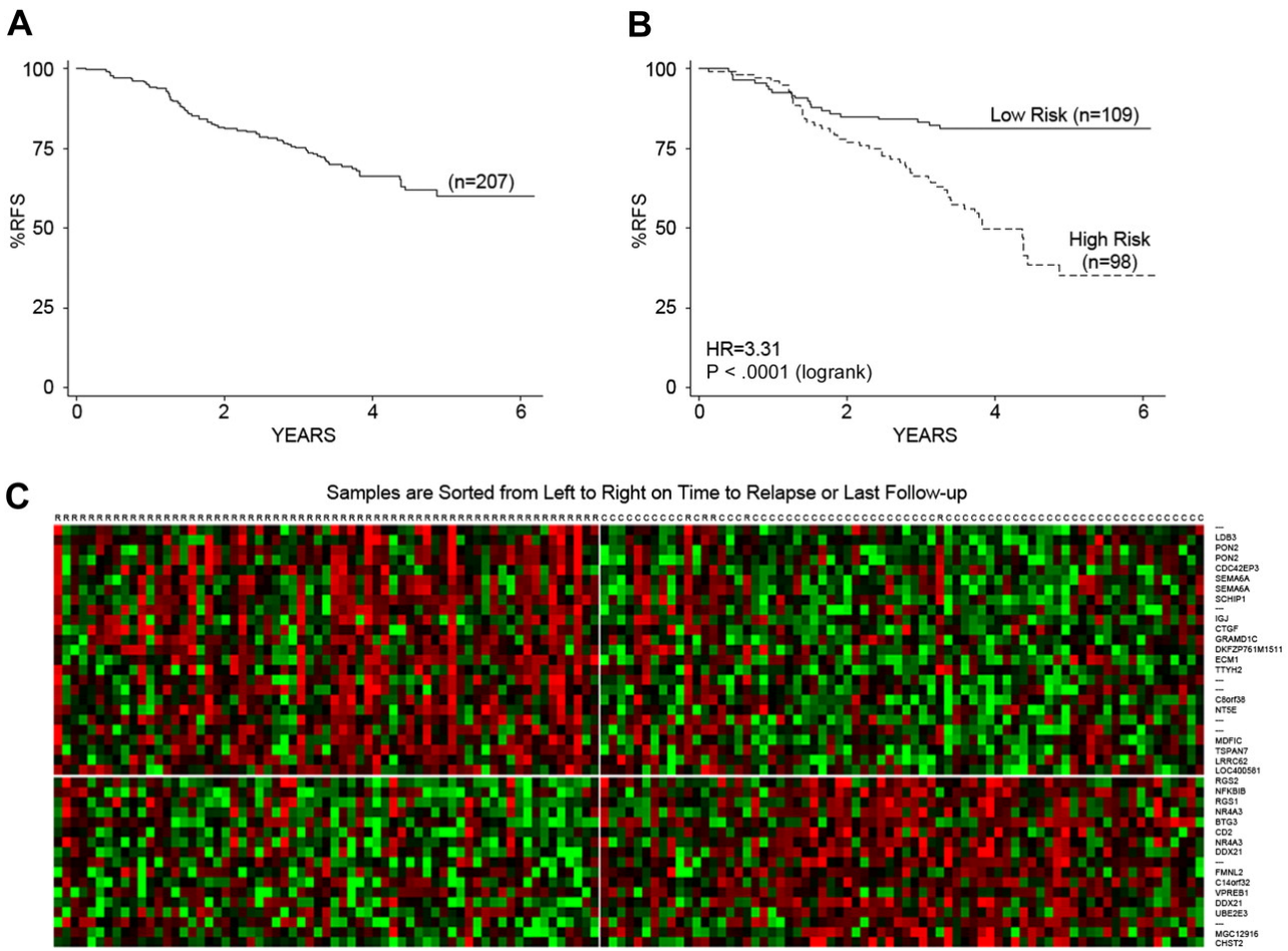
PCA Biplot: 24,827 Genes



PCA Biplot: 24,827 Genes



What is a classifier?



A 38-gene expression classifier predictive of relapse-free survival (RFS) could distinguish 2 groups with differing relapse risks: low (4-year RFS, 81%, n = 109) versus high (4-year RFS, 50%, n = 98; P < .001).

Taken from Kang *et al.* (2010).

Classification by gene expression

Goal:
Given sample i, use measured gene expression levels $x_{ig} \in \mathbb{R}$ for $g \in \{1, \ldots, p\}$ to assign class label y_i.

x_i represents vector of all gene measurements x_{ig} for sample i.

For two-class problems, $y_i \in \{0, 1\}$.

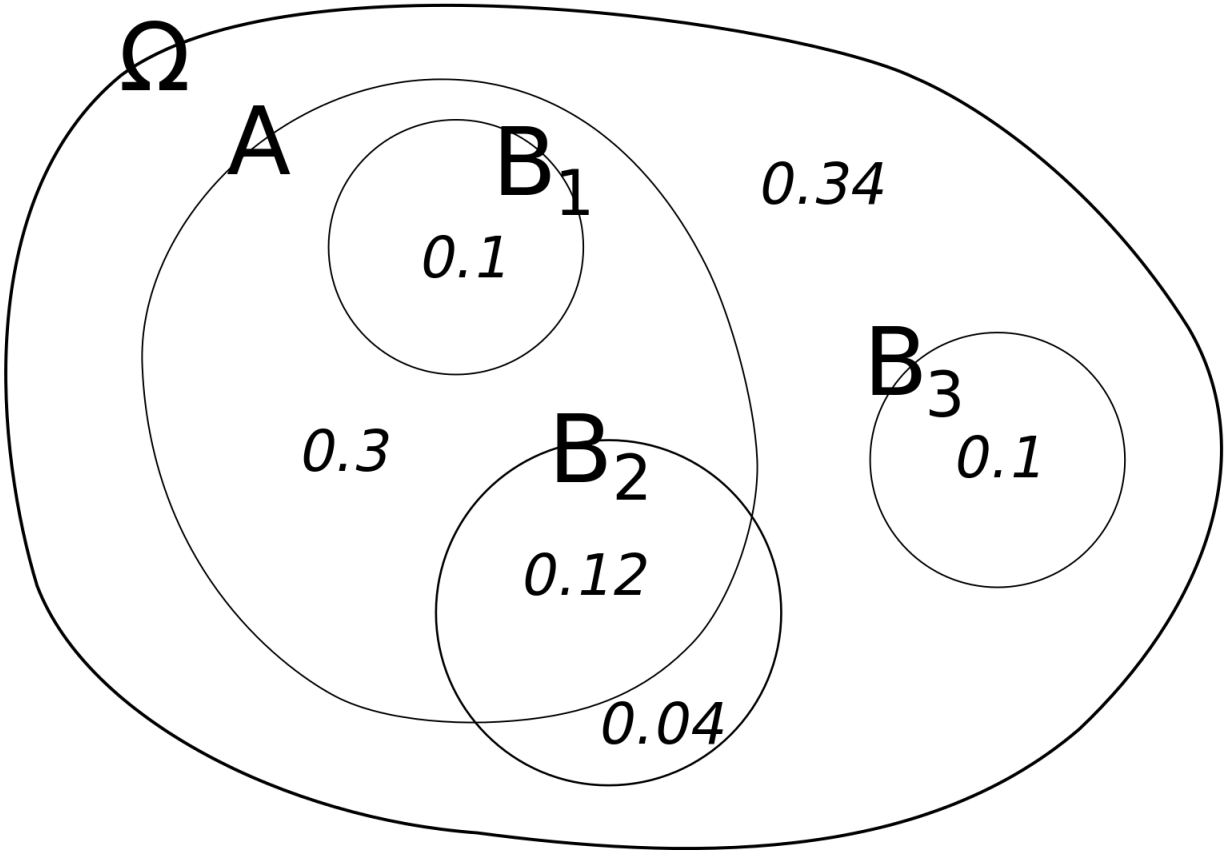
Define random variables X and Y

- x_i and y_i are specific data realizing X and Y.

Model predictions: $\mathbb{P}(Y = y \mid \mathrm{X} = \mathrm{x})$

Probabilities

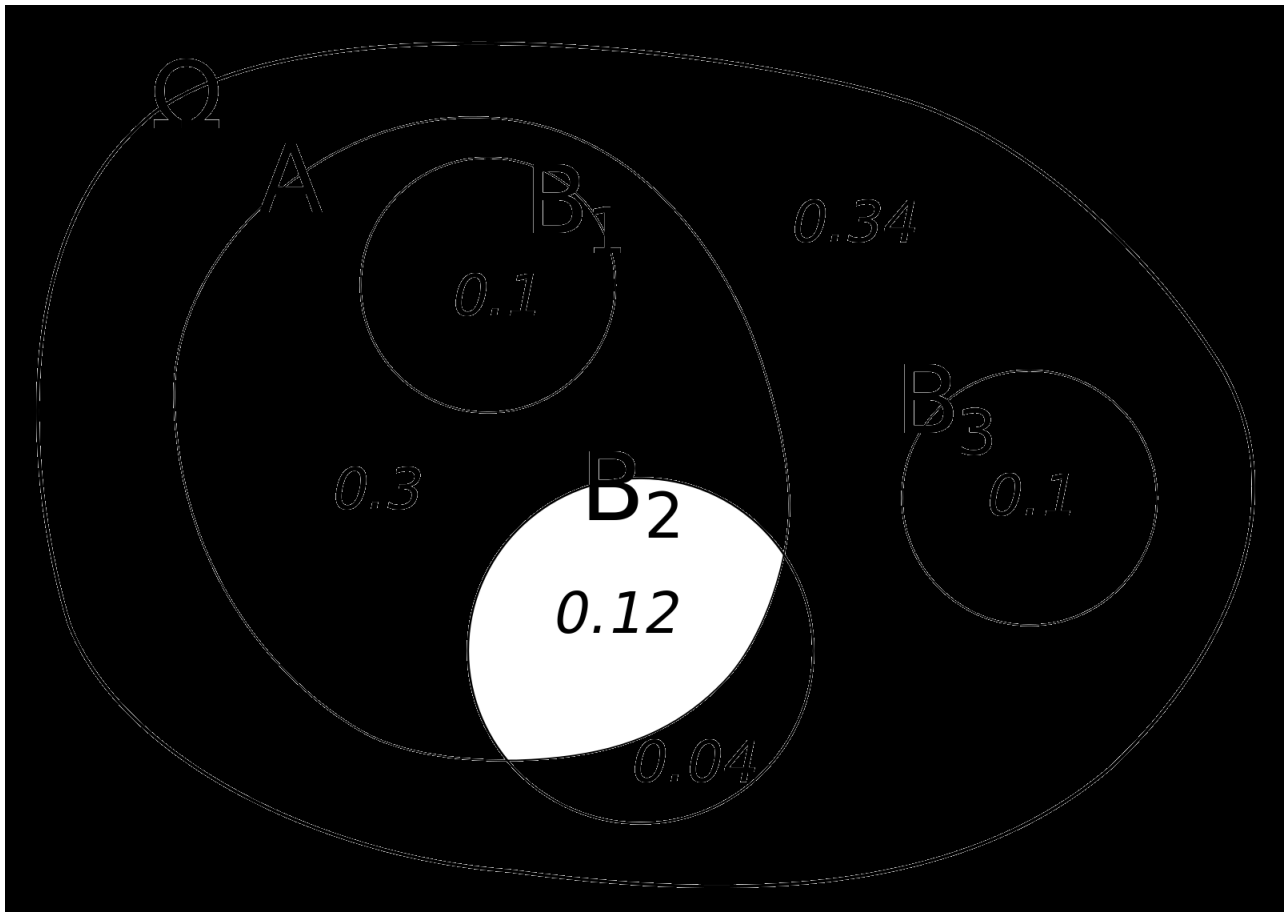
https://en.wikipedia.org/wiki/Conditional_probability#/media/File:Conditional_probability.svg



$$0.3 + 0.1 + 0.12 + 0.04 + 0.34 + 0.1 = 1$$

Joint probability $\mathbb{P}(A, B2)$ of event A and event B2

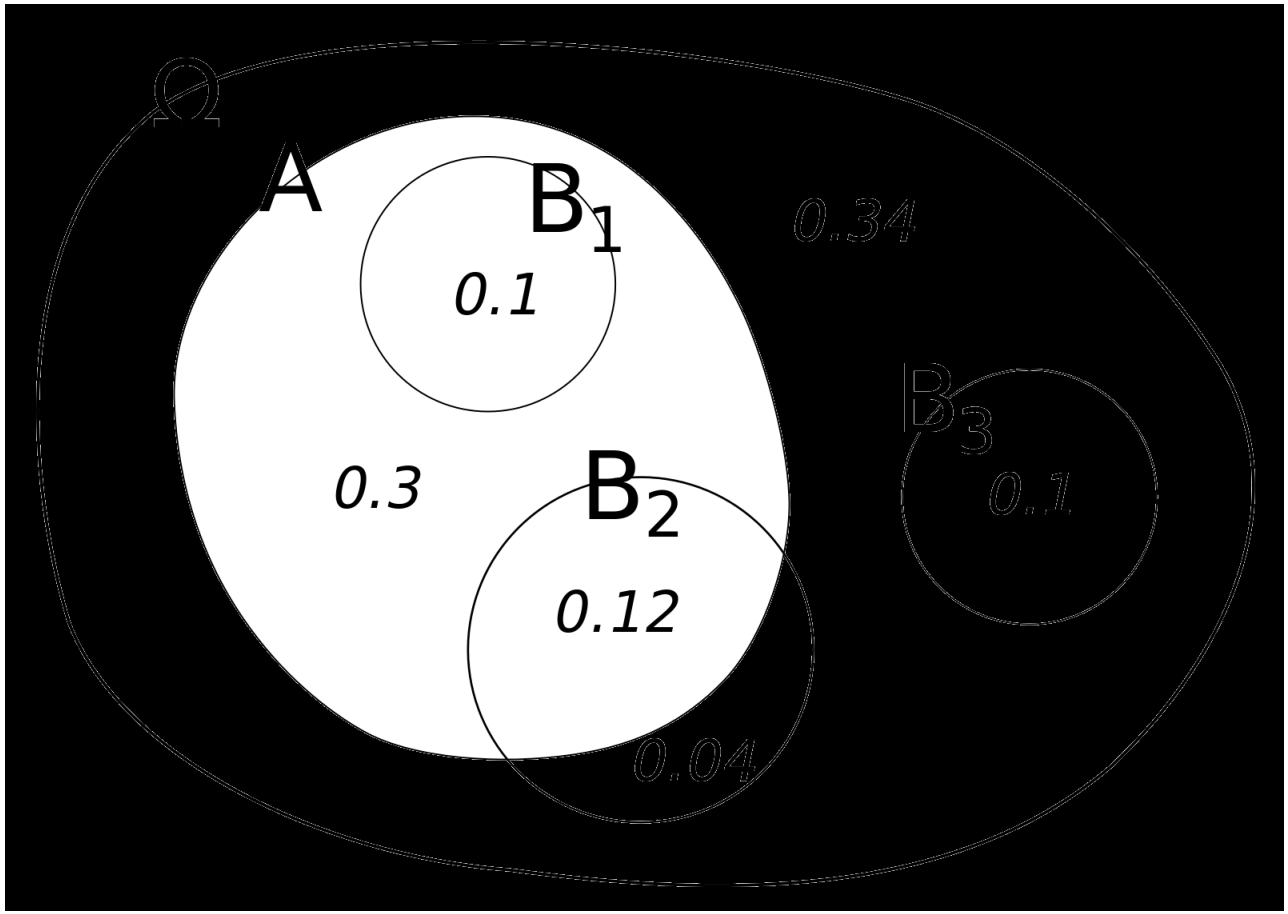
https://en.wikipedia.org/wiki/Conditional_probability#/media/File:Conditional_probability.svg



0.12

Marginal probability $\mathbb{P}(A)$ of event A

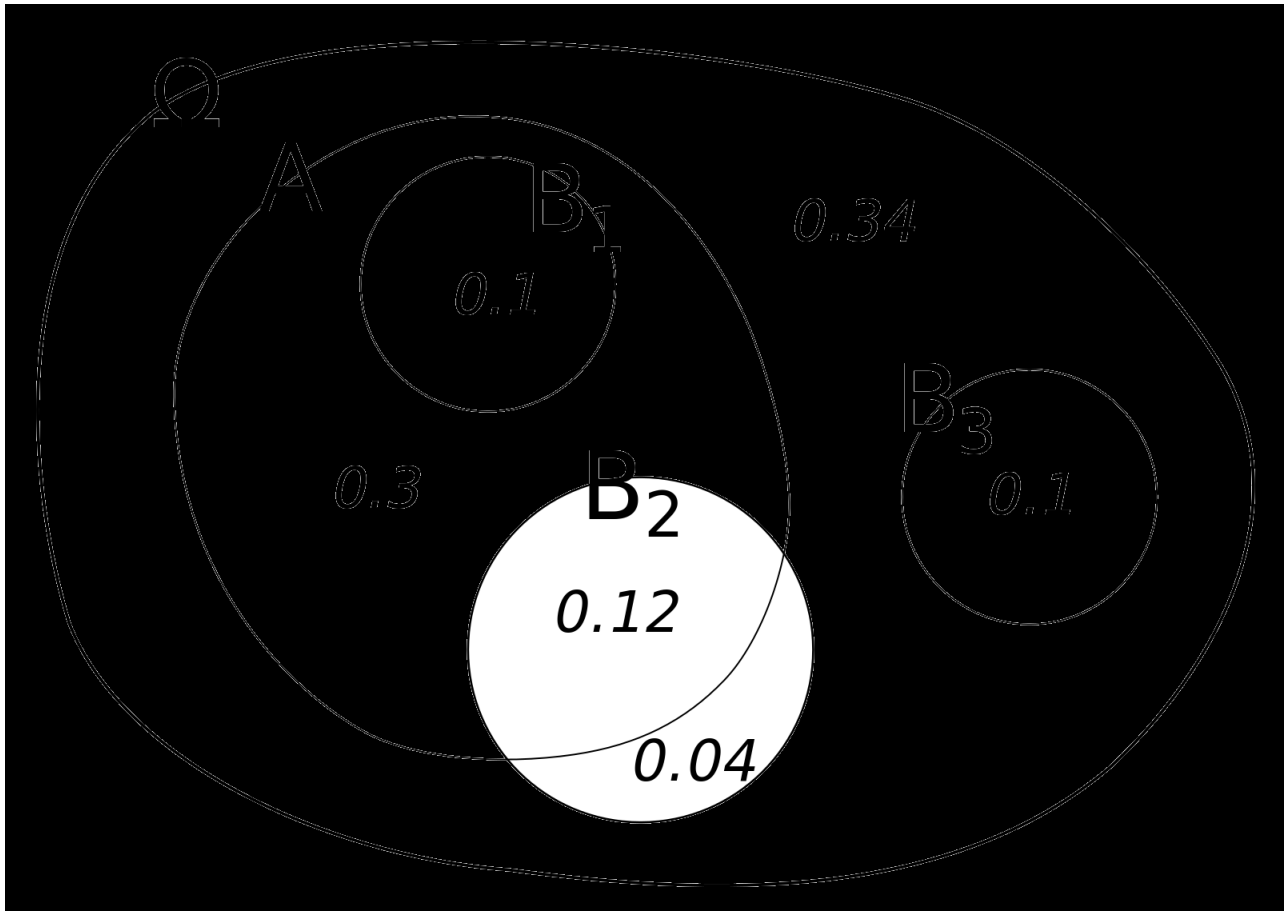
https://en.wikipedia.org/wiki/Conditional_probability#/media/File:Conditional_probability.svg



$$0.3 + 0.1 + 0.12 = 0.52$$

Marginal probability $\mathbb{P}(B2)$ of event B2

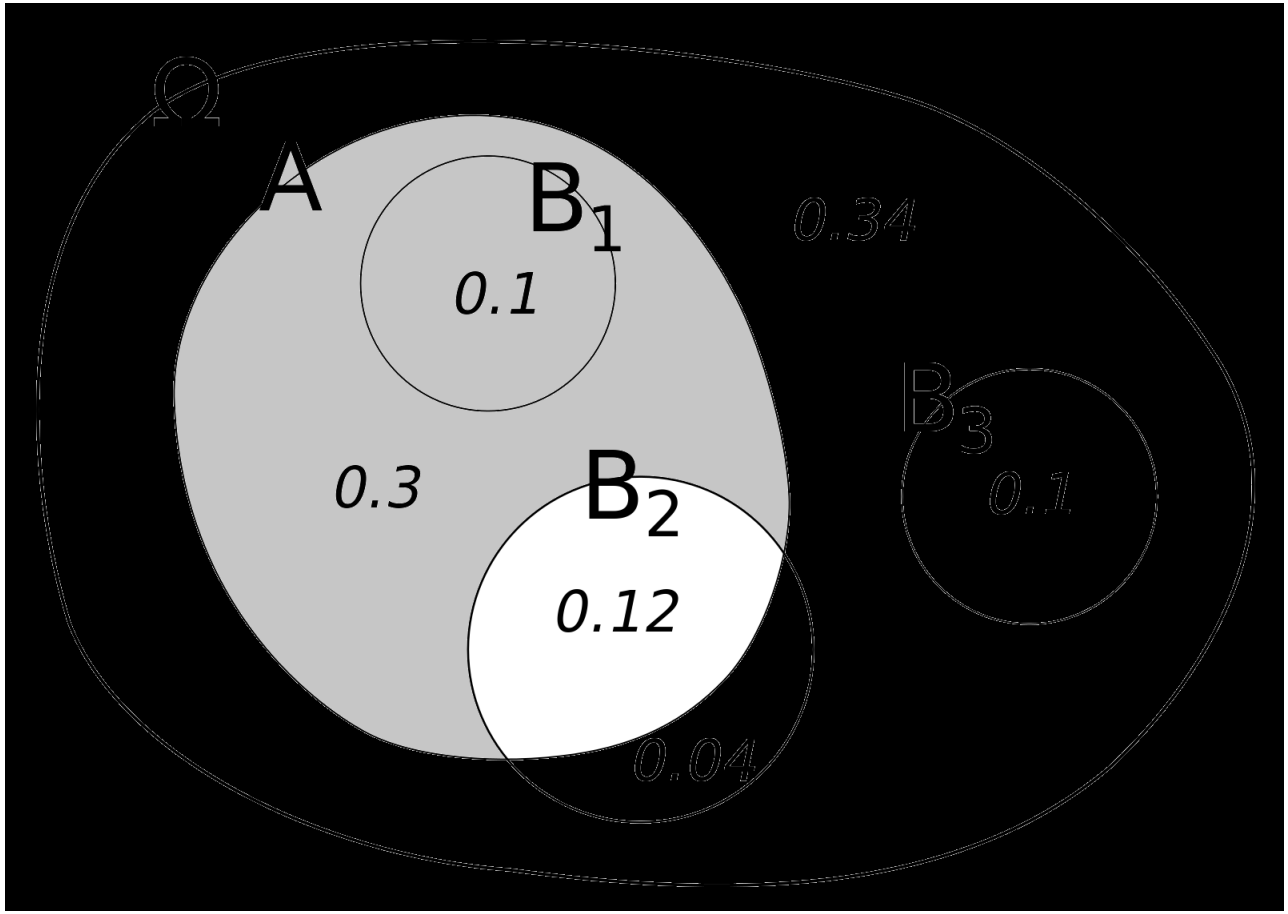
https://en.wikipedia.org/wiki/Conditional_probability#/media/File:Conditional_probability.svg



$$0.12 + 0.04 = 0.16$$

Conditional probability $\mathbb{P}(B2 \mid A)$ of B2 given A

https://en.wikipedia.org/wiki/Conditional_probability#/media/File:Conditional_probability.svg



$$\frac{0.12}{0.12 + 0.3 + 0.1} = 0.23077$$

Training and test sets

Apply M to fit parameters $\boldsymbol{\theta}$ using sample set S_{train} such that

$$\mathbb{P}_{M,\boldsymbol{\theta}}(Y = y_i \mid \mathrm{X} = \mathrm{x}_i)$$

has high probability for the observed class labels y_i for $i \in S_{\text{train}}$.

Training and test sets

Apply M to fit parameters $\boldsymbol{\theta}$ using sample set S_{train} such that

$$\mathbb{P}_{M,\boldsymbol{\theta}}(Y = y_i \mid \mathsf{X} = \mathsf{x}_i)$$

has high probability for the observed class labels y_i for $i \in S_{\text{train}}$.

However:

- really want $(M, \boldsymbol{\theta})$ to accurately classify samples $j \notin S_{\text{train}}$
- whose true classifications y_j may not already be known.

Generally $(M, \boldsymbol{\theta})$ worse for samples $j \notin S_{\text{train}}$ than for $i \in S_{\text{train}}$.

Training and test sets

Apply M to fit parameters $\boldsymbol{\theta}$ using sample set S_{train} such that

$$\mathbb{P}_{M,\boldsymbol{\theta}}(Y = y_i \mid \mathsf{X} = \mathsf{x}_i)$$

has high probability for the observed class labels y_i for $i \in S_{\text{train}}$.

However:

- really want $(M, \boldsymbol{\theta})$ to accurately classify samples $j \notin S_{\text{train}}$
- whose true classifications y_j may not already be known.

Generally $(M, \boldsymbol{\theta})$ worse for samples $j \notin S_{\text{train}}$ than for $i \in S_{\text{train}}$.

Thus useful to apply $(M, \boldsymbol{\theta})$ to $j \in S_{\text{test}}$

- where $S_{\text{test}} \cap S_{\text{train}} = \emptyset$
- but where the $\{y_j \mid j \in S_{\text{test}}\}$ are still known.

k-nearest-neighbors (knn)

Perhaps simplest approach to classification:

k-nearest neighbors
Given feature vector x with k nearest training vectors:

$$\{\mathrm{x}_j \mid j \in \mathrm{NN}_k\}$$

(so that $\|\mathrm{x}_j - \mathrm{x}\| \leq \|\mathrm{x}_i - \mathrm{x}\|$ if $j \in \mathrm{NN}_k$ and $i \notin \mathrm{NN}_k$):

$$\mathbb{P}(Y = 1 \mid X = x) = \frac{1}{k} \sum_{j \in \mathrm{NN}_k} y_j$$

k-nearest-neighbors (knn)

Perhaps simplest approach to classification:

k-nearest neighbors
Given feature vector x with k nearest training vectors:

$$\{\mathrm{x}_j \mid j \in \mathrm{NN}_k\}$$

(so that $\|\mathrm{x}_j - \mathrm{x}\| \leq \|\mathrm{x}_i - \mathrm{x}\|$ if $j \in \mathrm{NN}_k$ and $i \notin \mathrm{NN}_k$):

$$\mathbb{P}(Y = 1 \mid X = x) = \frac{1}{k} \sum_{j \in \mathrm{NN}_k} y_j$$

knn tends to work well:

- in low-dimensional settings
- when there is natural metric on feature space.

k-nearest-neighbors (knn)

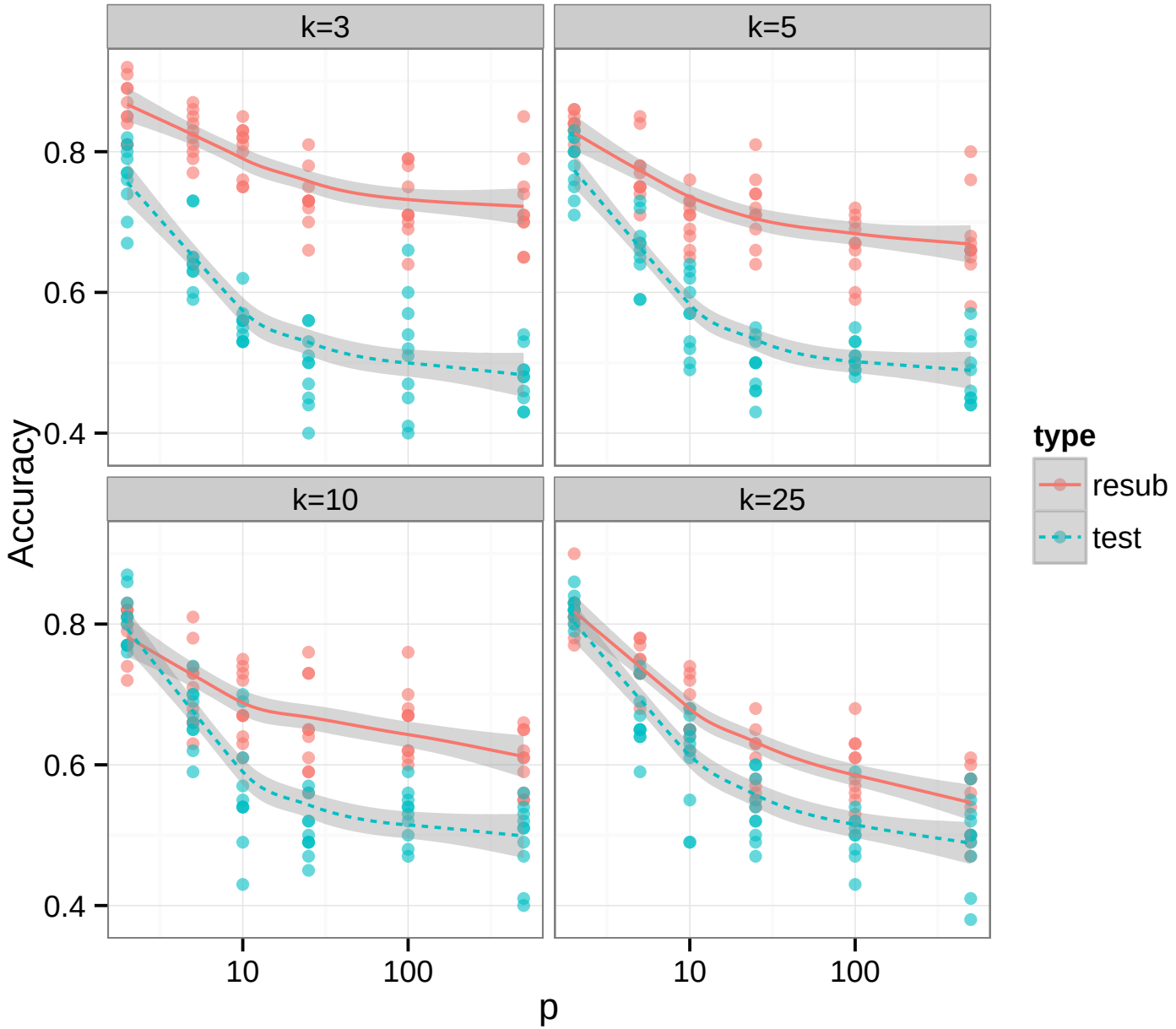
R:

```
knnTest = knn(
    train = xtrain,
    test = xtest,
    cl = ytrain,
    k = 3
)
nCorrect = sum(diag(table(knnTest, ytest)))
```

Python:

```
from sklearn.neighbors import KNeighborsClassifier
knnFit = KNeighborsClassifier(n_neighbors=3)
knnFit.fit(array(xtrain), array(ytrain))
knnTest = Series(knnFit.predict(xtest),
                 index = ytest.index)
nCorrect = sum(diag(pandas.crosstab(knnTest, ytest)))
```

k-nearest-neighbors (knn)



knn and the curse of dimensionality

Volume of p-dimensional hypersphere of radius r scales as

$$V_p(r) \propto r^p$$

If dimensionality p is high, $V_p(r)$ shrinks rapidly as $r \to 0$.

knn and the curse of dimensionality

Volume of p-dimensional hypersphere of radius r scales as

$$V_p(r) \propto r^p$$

If dimensionality p is high, $V_p(r)$ shrinks rapidly as $r \to 0$.

- hard to find k neighbors close by when p large.

knn and the curse of dimensionality

Volume of p-dimensional hypersphere of radius r scales as

$$V_p(r) \propto r^p$$

If dimensionality p is high, $V_p(r)$ shrinks rapidly as $r \to 0$.

- hard to find k neighbors close by when p large.
- So must use points far away to guess what's going on at x.

knn and the curse of dimensionality

Volume of p-dimensional hypersphere of radius r scales as

$$V_p(r) \propto r^p$$

If dimensionality p is high, $V_p(r)$ shrinks rapidly as $r \to 0$.

- hard to find k neighbors close by when p large.
- So must use points far away to guess what's going on at x.
- Not surprisingly this doesn't always work ...

knn and the curse of dimensionality

Volume of p-dimensional hypersphere of radius r scales as

$$V_p(r) \propto r^p$$

If dimensionality p is high, $V_p(r)$ shrinks rapidly as $r \to 0$.

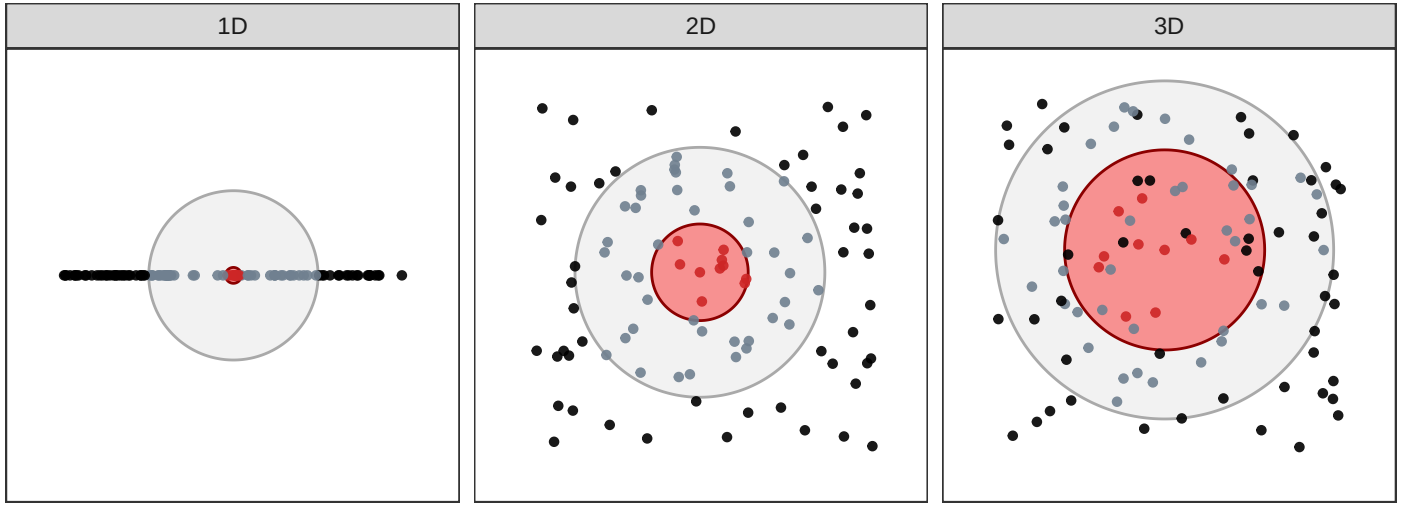
- hard to find k neighbors close by when p large.
- So must use points far away to guess what's going on at x.
- Not surprisingly this doesn't always work ...

May be better to do

- **feature selection** or
- **feature extraction**

and then fit model using reduced feature set.

knn and the curse of dimensionality



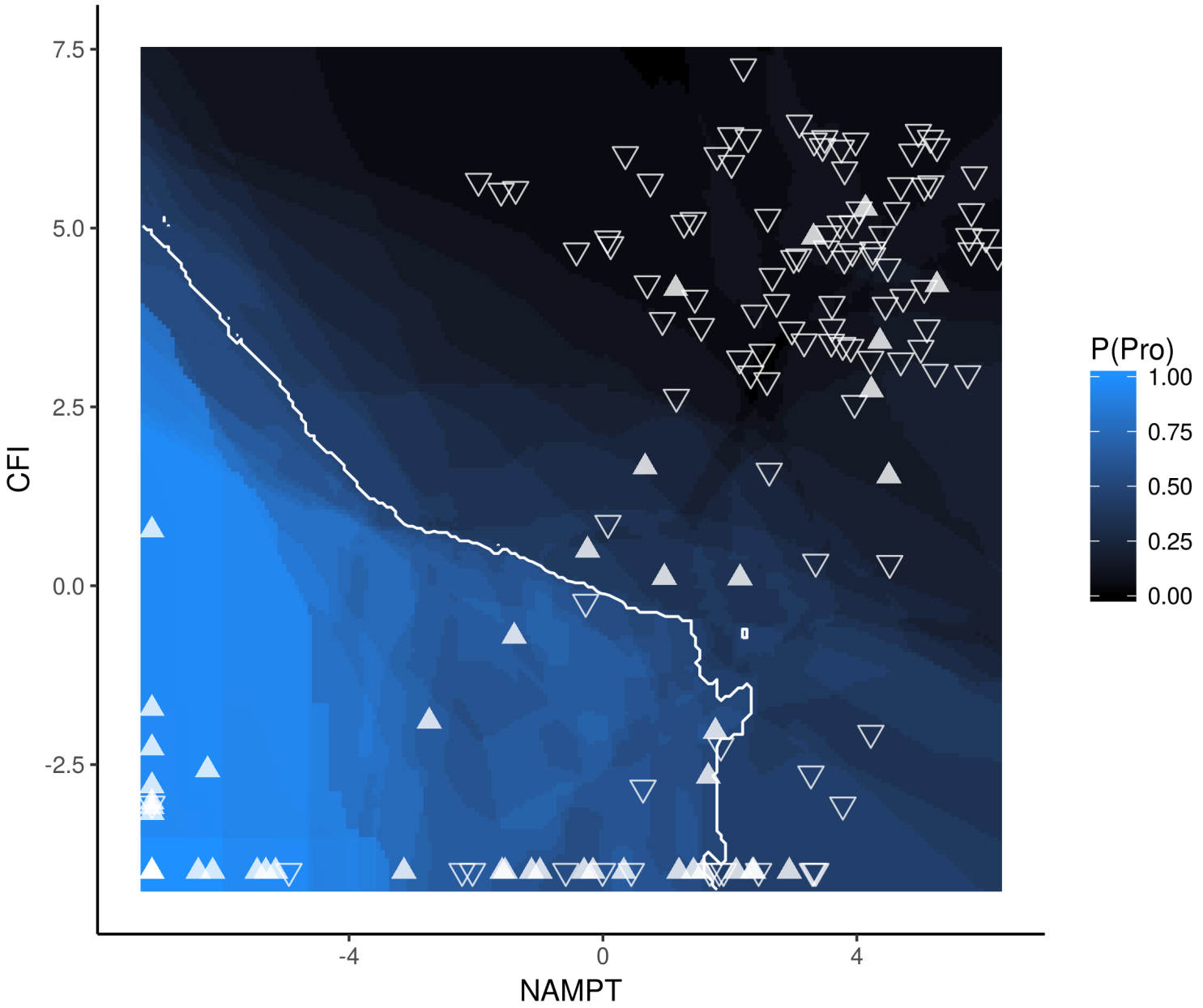
- hard to find k neighbors close by when p large.
- So must use points far away to guess what's going on at x.
- Not surprisingly this doesn't always work . . .

May be better to do

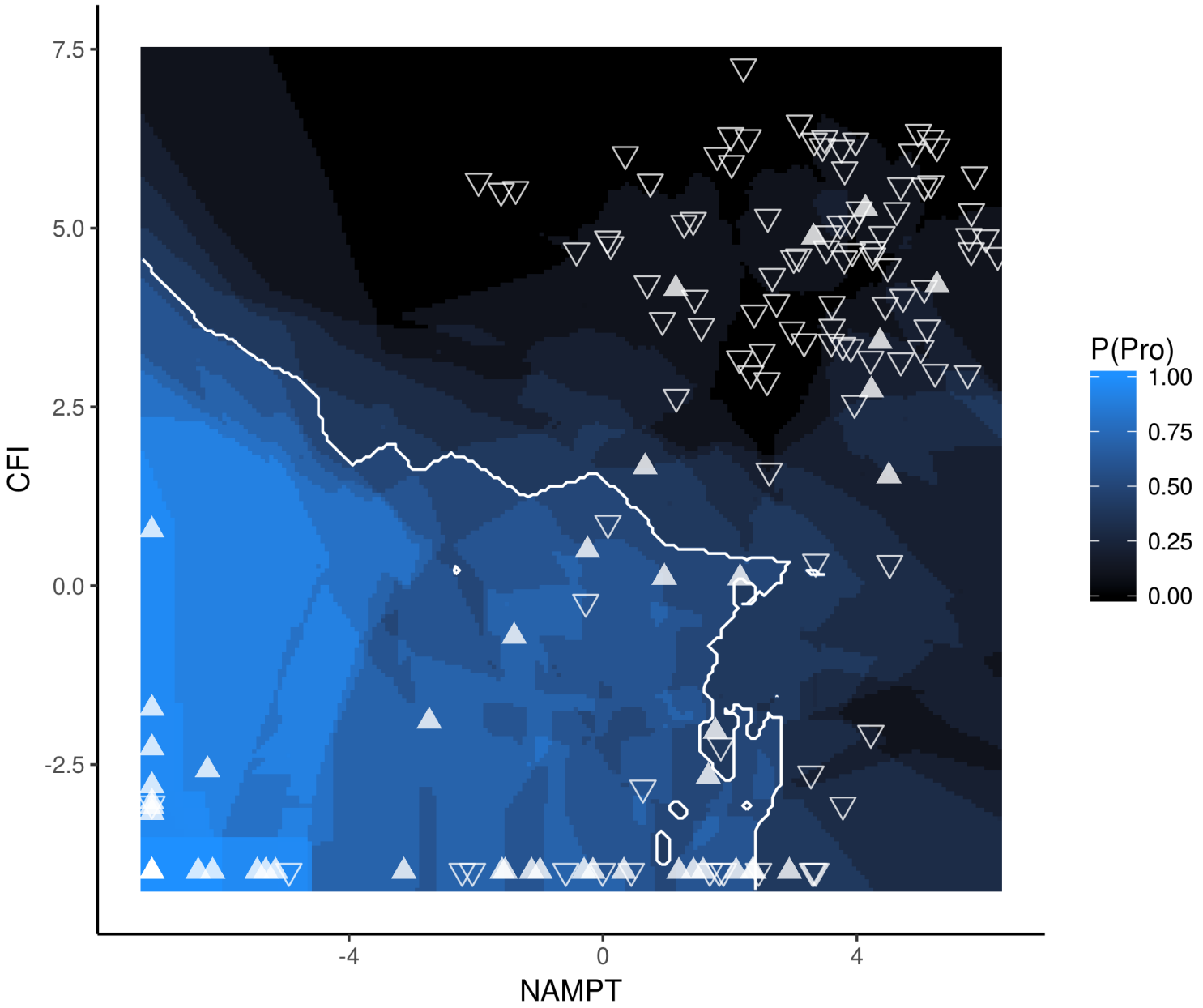
- **feature selection** or
- **feature extraction**

and then fit model using reduced feature set.

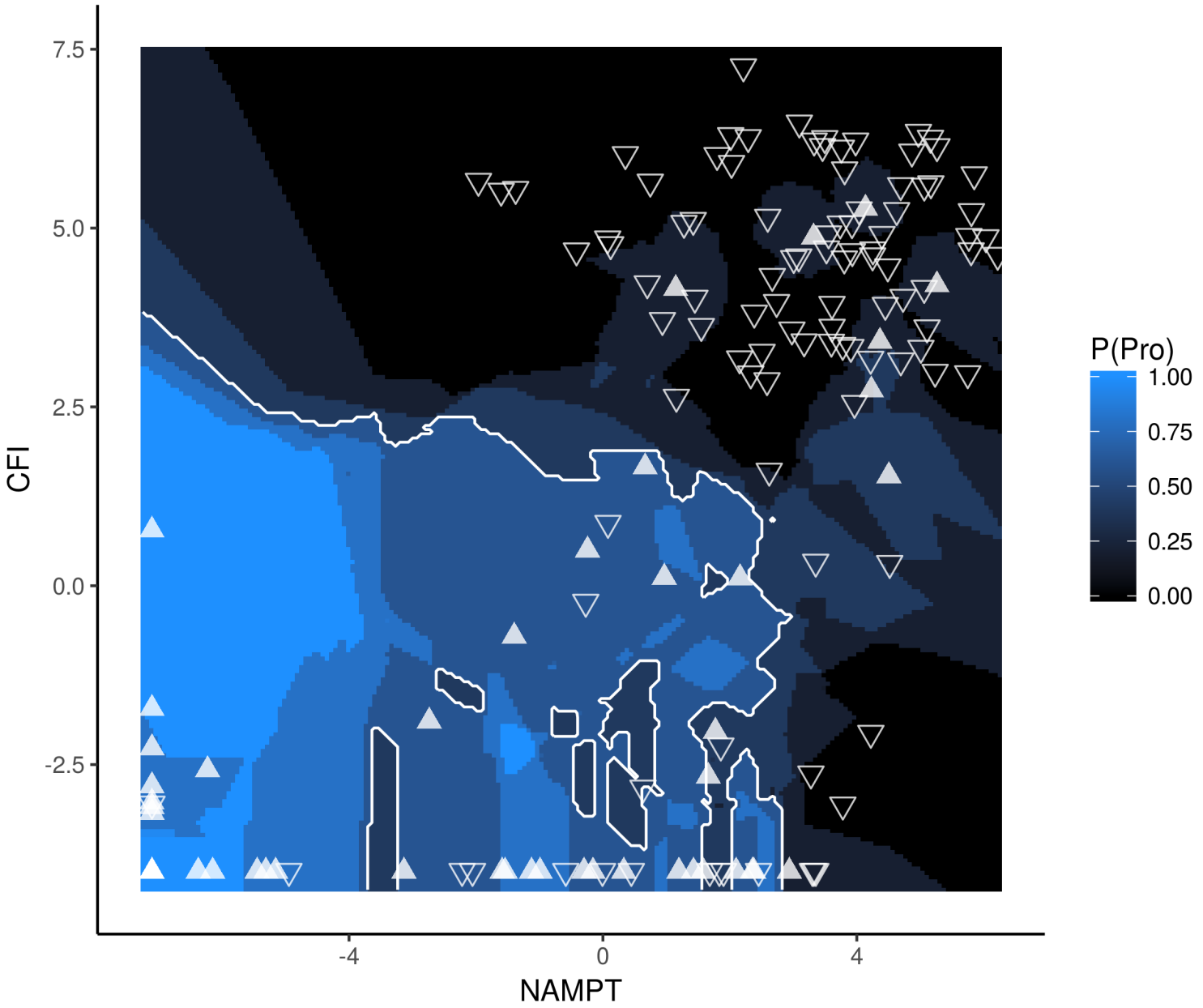
Overfitting: K=20



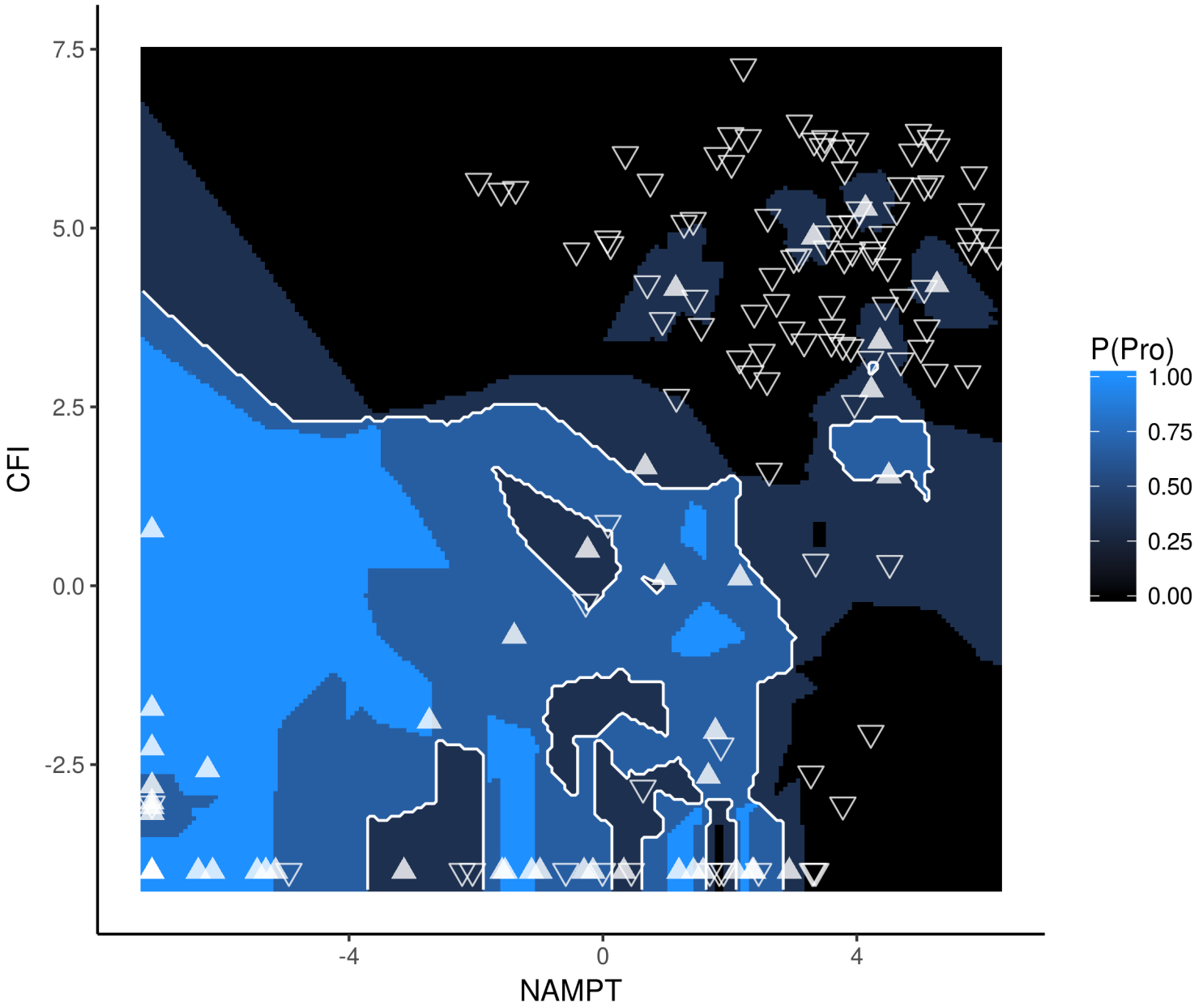
Overfitting: K=10



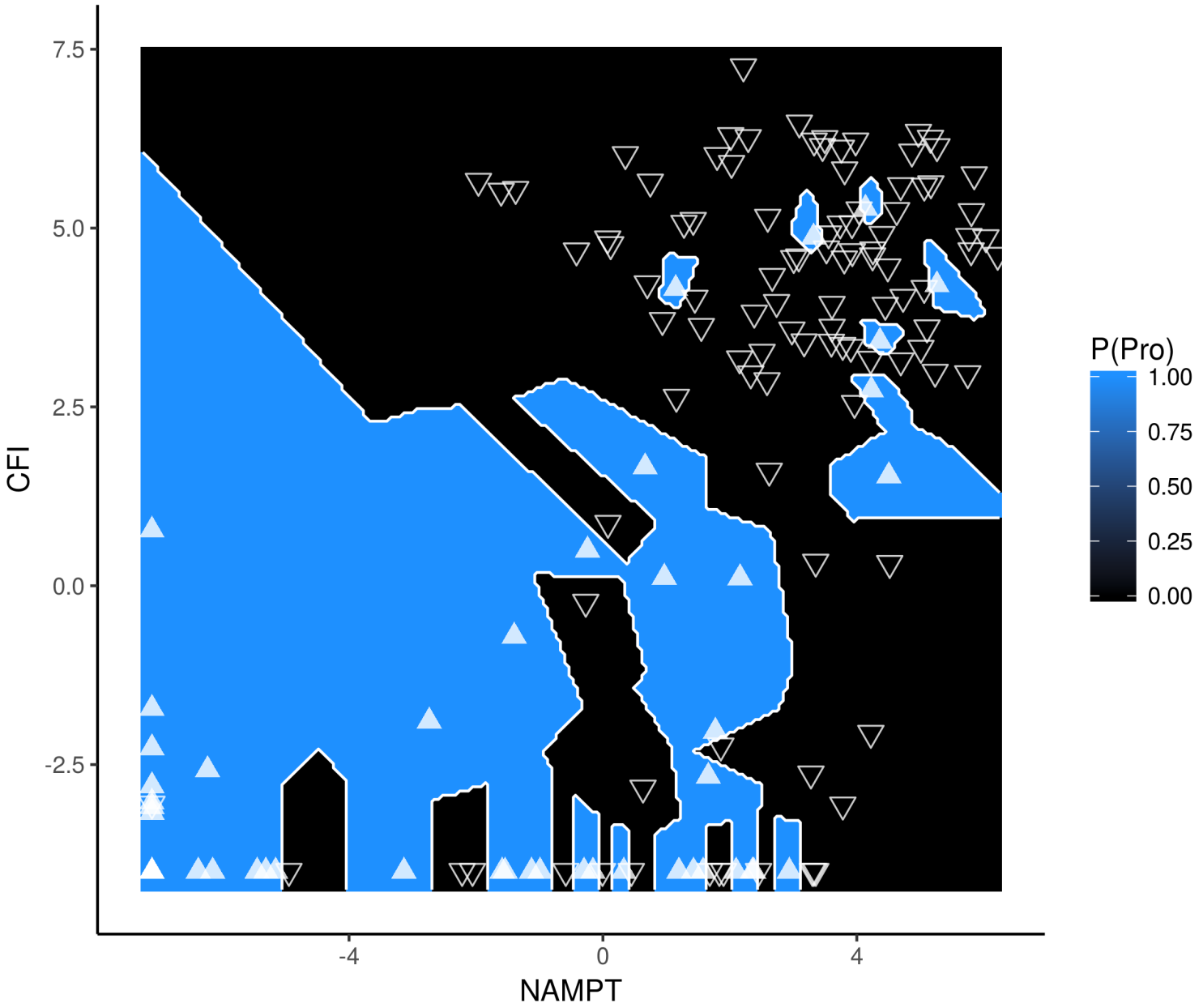
Overfitting: K=5



Overfitting: K=3



Overfitting: K=1



Cross-Validation (CV)

Know evaluating performance by resubstitution suffers from bias.

But what if we don't have a test set S_{test} lying around?

Cross-Validation (CV)

Know evaluating performance by resubstitution suffers from bias.

But what if we don't have a test set S_{test} lying around?

Could split sample set S into disjoint test and training sets ...

Cross-Validation (CV)

Know evaluating performance by resubstitution suffers from bias.

But what if we don't have a test set S_{test} lying around?

Could split sample set S into disjoint test and training sets ...

If limited samples available, might partition $S = S_1 \cup S_2$ and try:

1. train M on S_1 to obtain $(M, \boldsymbol{\theta}_1)$ for testing on S_2;
2. then train on S_2 to obtain model $(M, \boldsymbol{\theta}_2)$ for testing on S_1.

Unbiased performance estimate could then be obtained using:

- predictions $\mathbb{P}_{M,\boldsymbol{\theta}_2}(Y \mid \mathrm{X})$ for samples in S_1, and
- predictions $\mathbb{P}_{M,\boldsymbol{\theta}_1}(Y \mid \mathrm{X})$ for samples in S_2.

K-Fold Cross-Validation

This procedure can be generalized to split S up into K subsets S_k for each of which:

1. a model $(M, \boldsymbol{\theta}_{-k})$ is trained using training set $S_{-k} = \bigcup_{q \neq k} S_q$
2. predictions $\mathbb{P}_{M,\boldsymbol{\theta}_{-k}}(Y \mid \mathrm{X} = \mathrm{x}_i)$ are made for samples $i \in S_k$
3. performance estimates are made for each S_k based on $\mathbb{P}_{M,\boldsymbol{\theta}_{-k}}(Y \mid \mathrm{X} = \mathrm{x}_i)$ and then averaged over all K folds.

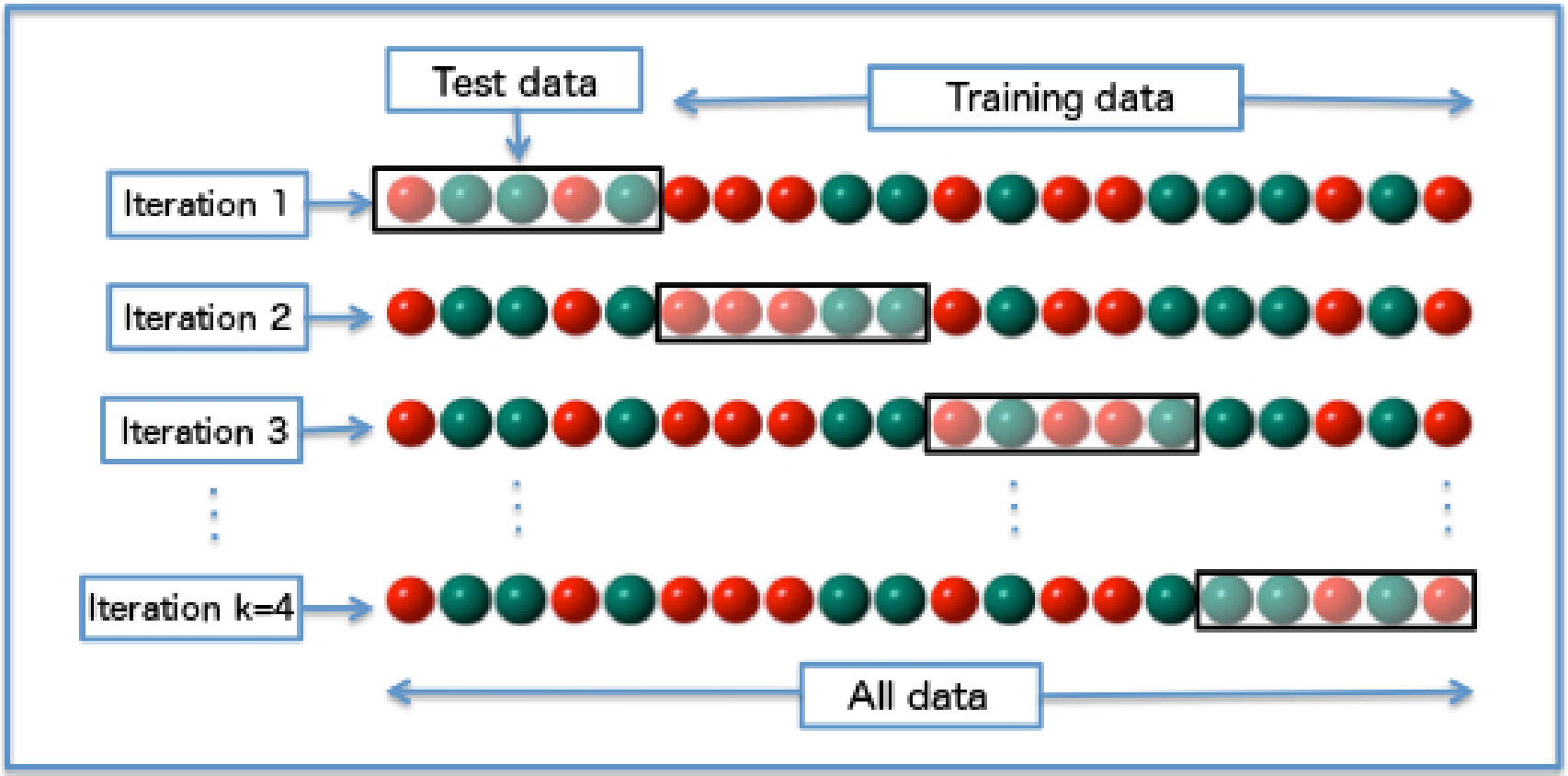
K-Fold Cross-Validation

This procedure can be generalized to split S up into K subsets S_k for each of which:

1. a model $(M, \boldsymbol{\theta}_{-k})$ is trained using training set $S_{-k} = \bigcup_{q \neq k} S_q$
2. predictions $\mathbb{P}_{M,\boldsymbol{\theta}_{-k}}(Y \mid \mathrm{X} = \mathrm{x}_i)$ are made for samples $i \in S_k$
3. performance estimates are made for each S_k based on $\mathbb{P}_{M,\boldsymbol{\theta}_{-k}}(Y \mid \mathrm{X} = \mathrm{x}_i)$ and then averaged over all K folds.

Very important:
Cross-validation is only valid if all *supervised* steps performed in building a classification model are conducted separately in each of the k folds.

K-Fold Cross-Validation



https://en.wikipedia.org/wiki/File:K-fold_cross_validation_EN.jpg

5-Fold Cross-Validation

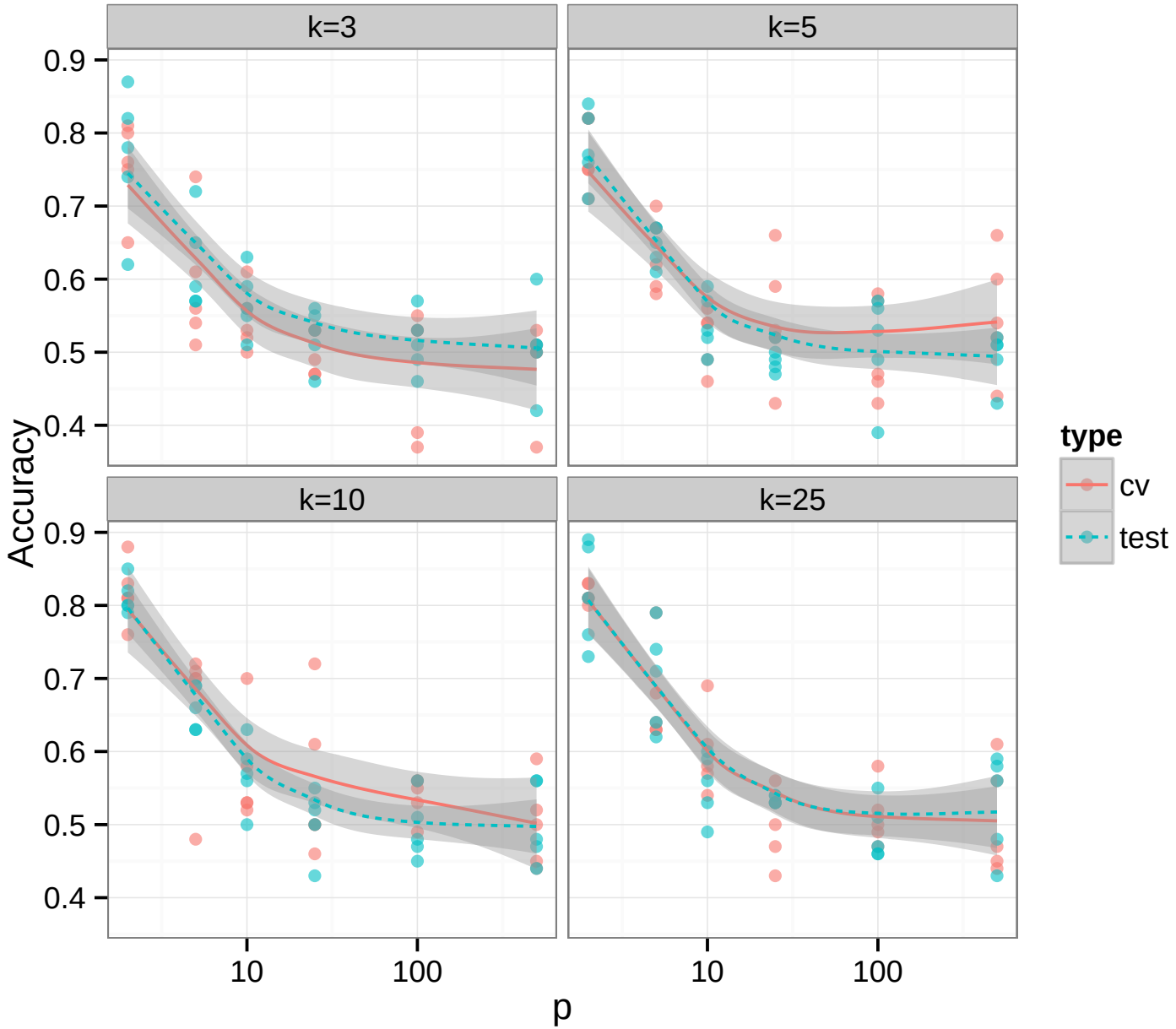
R:

```
library(caret)
knnCV = train(
    x = xtrain,
    y = ytrain,
    method = "knn",
    tuneGrid = data.frame(k=3),
    trControl = trainControl(method="cv", number=5)
)
cvAccuracyEstimate = knnCV$results[ , "Accuracy"]
```

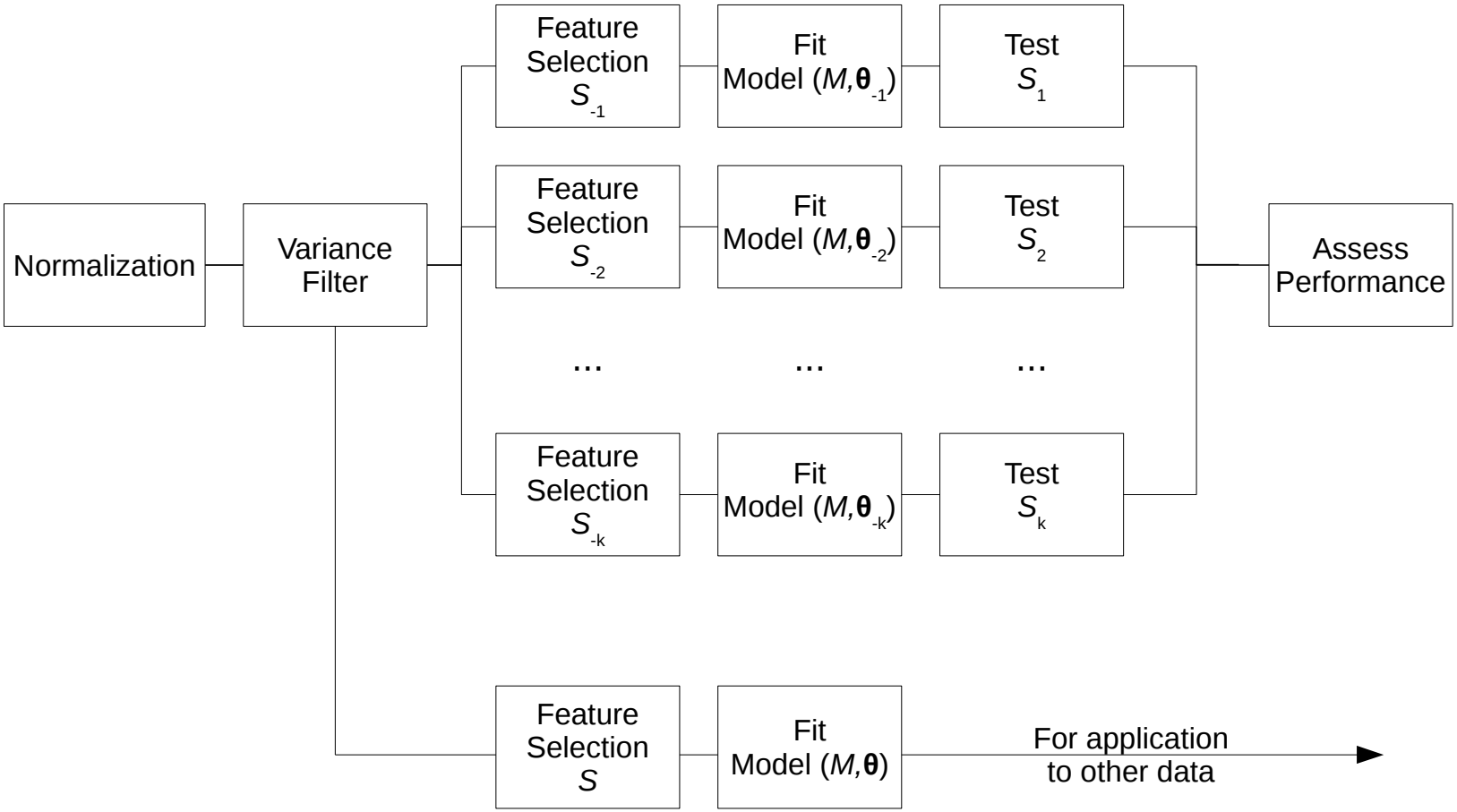
Python:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cross_validation import cross_val_score
knnClass = KNeighborsClassifier(n_neighbors=3)
cvAccs = cross_val_score(estimator = knnClass,
                         X = array(xtrain),
                         y = array(ytrain),
                         cv = 5)
cvAccuracyEstimate = mean(cvAccs)
```

5-Fold Cross-Validation



Cross-Validation Flow



Feature selection

In many cases, expression patterns of most genes either:

1. are uninformative, or
2. contain only information redundant with a small number of maximally useful markers

for a particular classification task.

Feature selection attempts to identify useful markers for inclusion in classifier.

Feature selection

In many cases, expression patterns of most genes either:

1. are uninformative, or
2. contain only information redundant with a small number of maximally useful markers

for a particular classification task.

Feature selection attempts to identify useful markers for inclusion in classifier.

Feature selection not always required, but may:

1. reduce computational workload,
2. help to avoid overfitting
   - though feature selection can itself overfit,

   and
3. facilitate model platform migration.

Taxonomy (adapted from Saeys *et al.* (2007))

Filter

- Selection done before and independently of classifier construction.
- Can be univariate or multivariate.

Taxonomy (adapted from Saeys *et al.* (2007))

Filter

- Selection done before and independently of classifier construction.
- Can be univariate or multivariate.

Wrapper

- Multiple fits using different feature sets.
- Select feature set for which fit model optimizes specified criterion.
- Often iterative; may add and/or remove features at each iteration.

Taxonomy (adapted from Saeys *et al.* (2007))

Filter

- Selection done before and independently of classifier construction.
- Can be univariate or multivariate.

Wrapper

- Multiple fits using different feature sets.
- Select feature set for which fit model optimizes specified criterion.
- Often iterative; may add and/or remove features at each iteration.

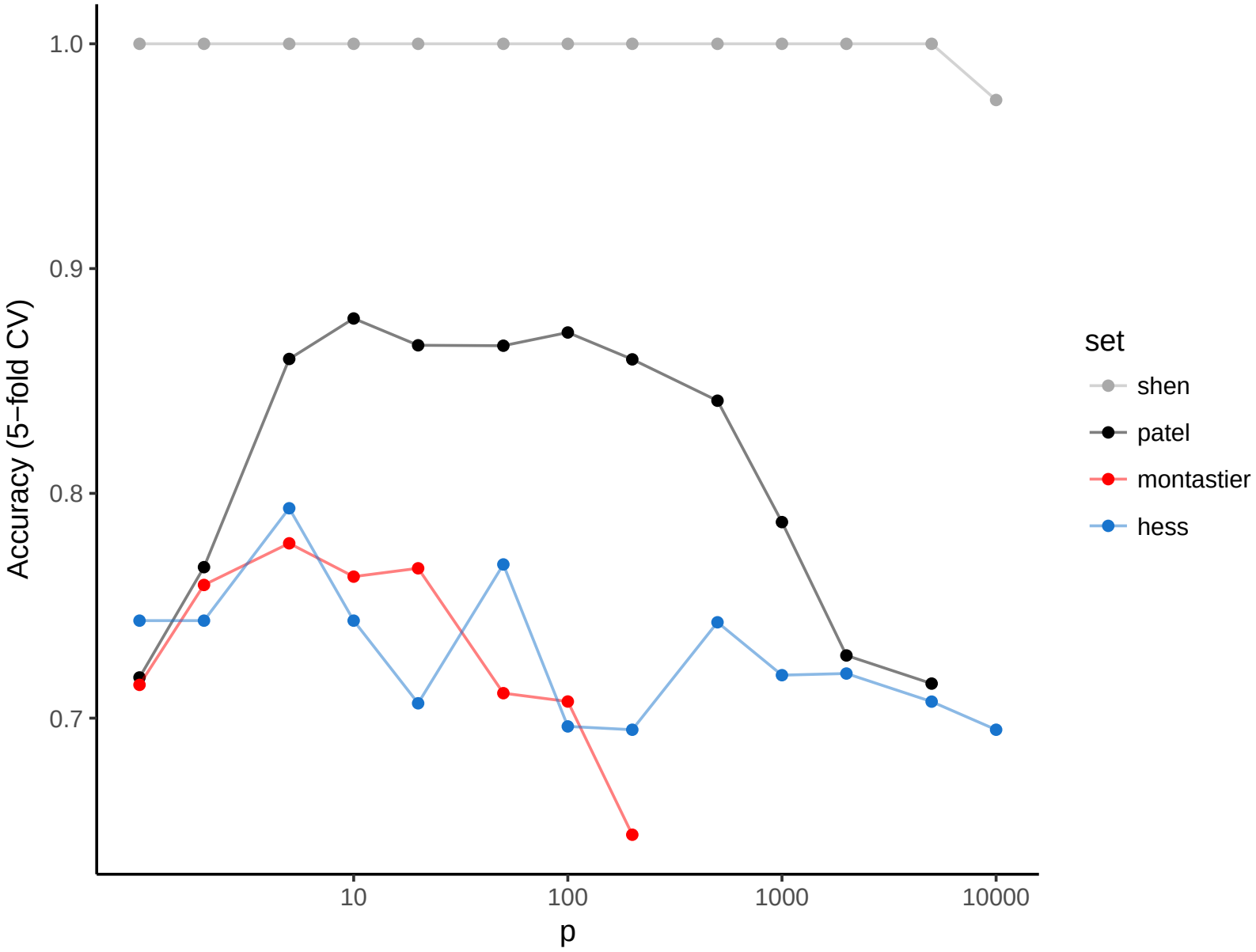
Embedded

- Feature selection inherently built into some classifier construction methods.
- Elegant but less flexible.

Taxonomy (adapted from Saeys *et al.* (2007))

Category	Advantages	Disadvantages	Examples
Filter	*Univariate*		
	Fast Scalable Independent of classifier	- feature dependencies - interaction w/classifier	t-test, ANOVA Wilcox test Rank Product
	Multivariate		
	+ feature dependencies Independent of classifier Intermediate complexity	Slower Less Scalable - interaction w/classifier	CFS Markov Blanket Filter
Wrapper	*Deterministic*		
	Simple + interaction w/classifier + feature dependencies	Risk of over-fitting Greedy (local optima) Classifier dependent selection	Forward Selection Backward Elimination Plus q minus r
	Randomized		
	Less prone to local optima + interaction w/classifier + feature dependencies	High risk over-fitting Computationally intensive Classifier dependent selection	Simulated Annealing Randomized Hill Climbing Genetic Algorithms
Embedded	+ interaction w/classifier + feature dependencies Intermediate complexity	No modularity Restrict algorithms	Decision trees Weighted Naive Bayes LASSO regression

knn accuracy With t-Test feature selection

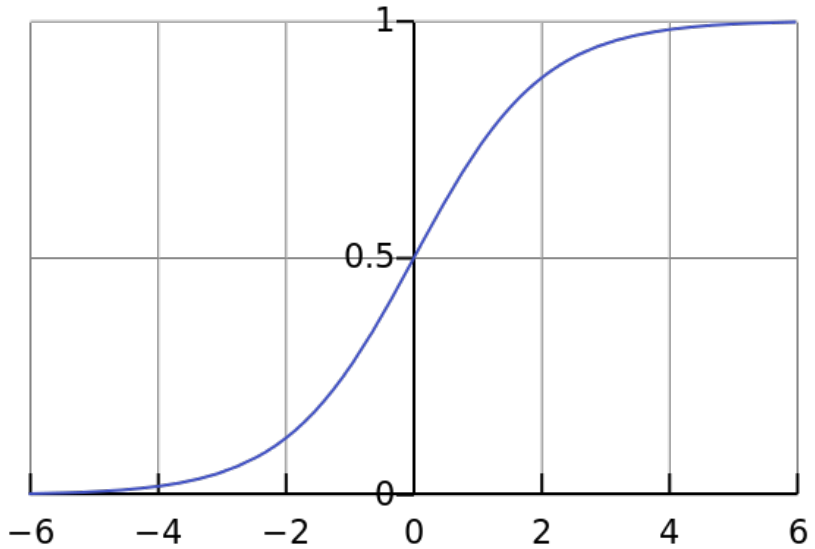


Linear models

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

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

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



Linear models

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

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

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 $\mathbb{P}(\mathsf{X} = \mathsf{x} | Y = y) \sim \mathcal{N}(\mu_y, \Sigma)$.
   - Fit by maximizing joint likelihood $\mathbb{P}(\mathsf{X} = \mathsf{x}, Y = y)$.
2. **logistic regression**
   - Conditional.
   - Makes no explicit distributional assumptions about X.
   - Maximizes likelihood of conditional $\mathbb{P}(Y = y \mid \mathsf{X} = \mathsf{x})$.

Naive Bayes

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

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

which can be substituted into Bayes' formula to yield:

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

Naive Bayes

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

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

which can be substituted into Bayes' formula to yield:

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

DLDA is a form of naive Bayes classification additionally assuming linearity.

Naive Bayes: does it work?

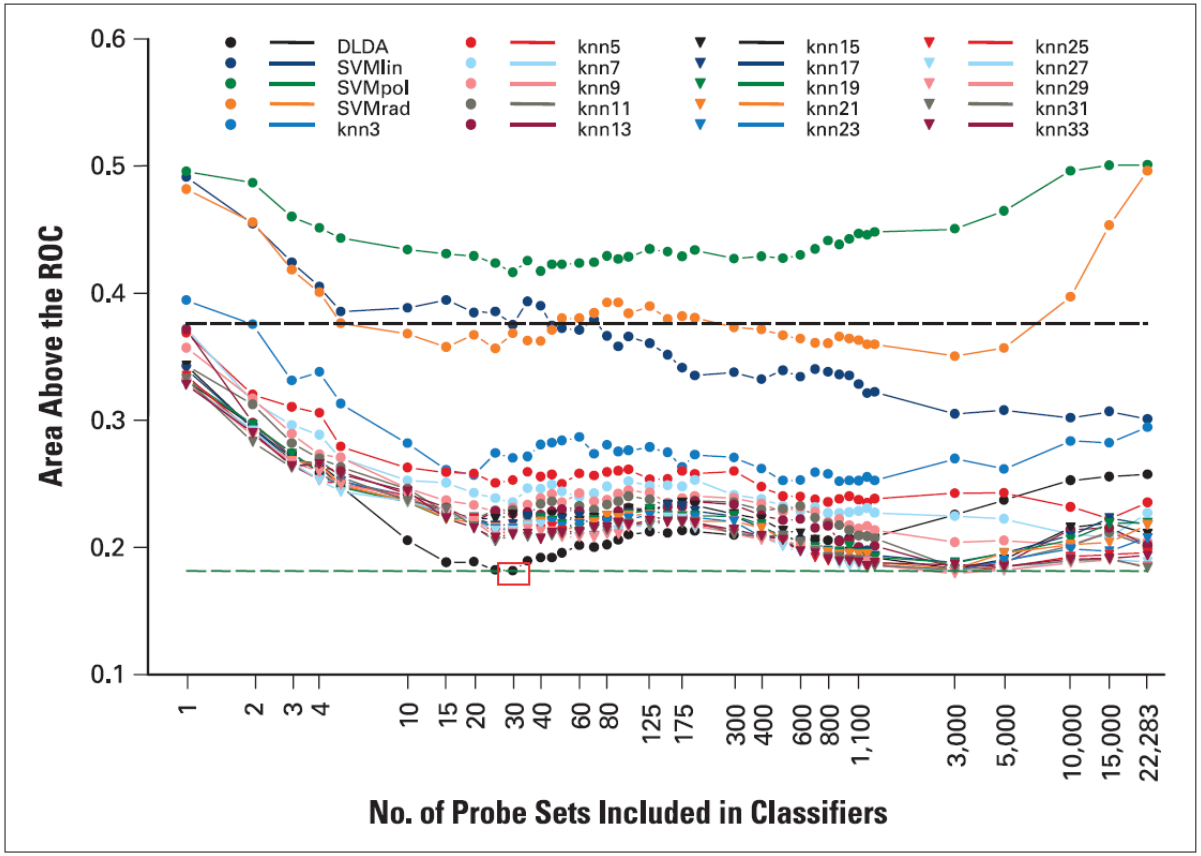


Fig 1. Mean area above the receiver operating characteristic (ROC) curves plotted against the number of top genes included in the classifiers. Complete 5-fold cross validation results (means over the 100 iterations) for 20 classifier algorithms including different numbers of probe sets (39 gene sets) 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.

Support vector machines (SVMs)

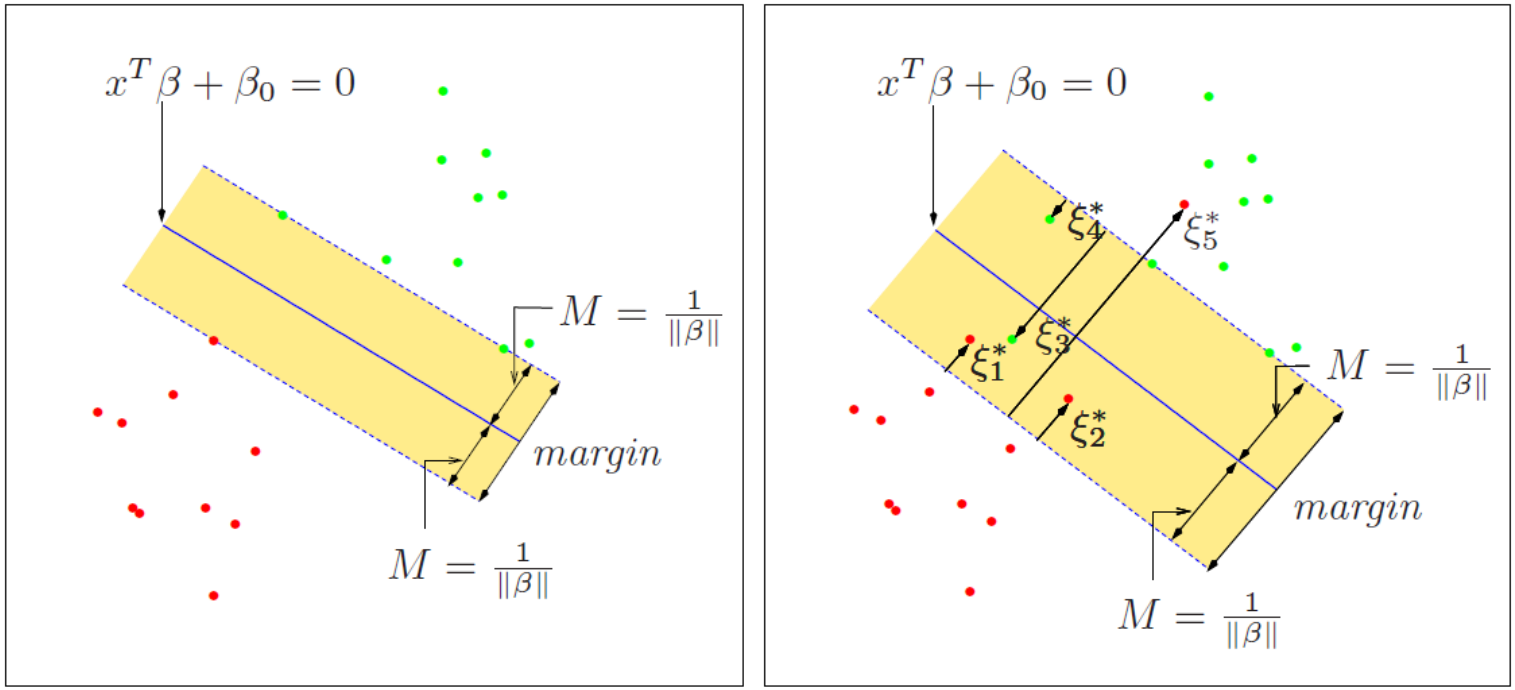
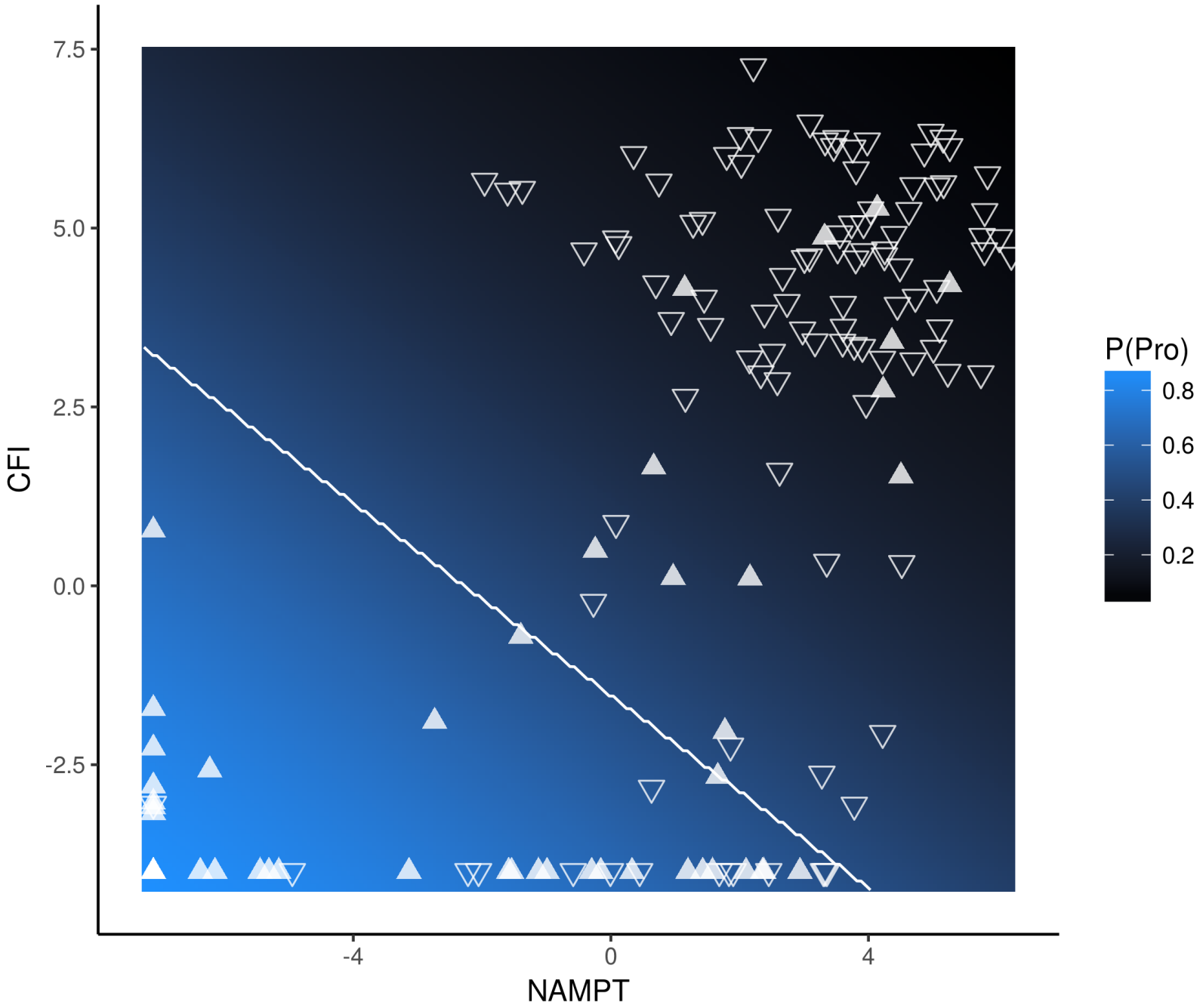


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width* $2M = 2/\|\beta\|$. *The right panel shows the nonseparable (overlap) case. The points labeled* ξ_j^* *are on the wrong side of their margin by an amount* $\xi_j^* = M\xi_j$; *points on the correct side have* $\xi_j^* = 0$. *The margin is maximized subject to a total budget* $\sum \xi_i \leq$ *constant. Hence* $\sum \xi_j^*$ *is the total distance of points on the wrong side of their margin.*

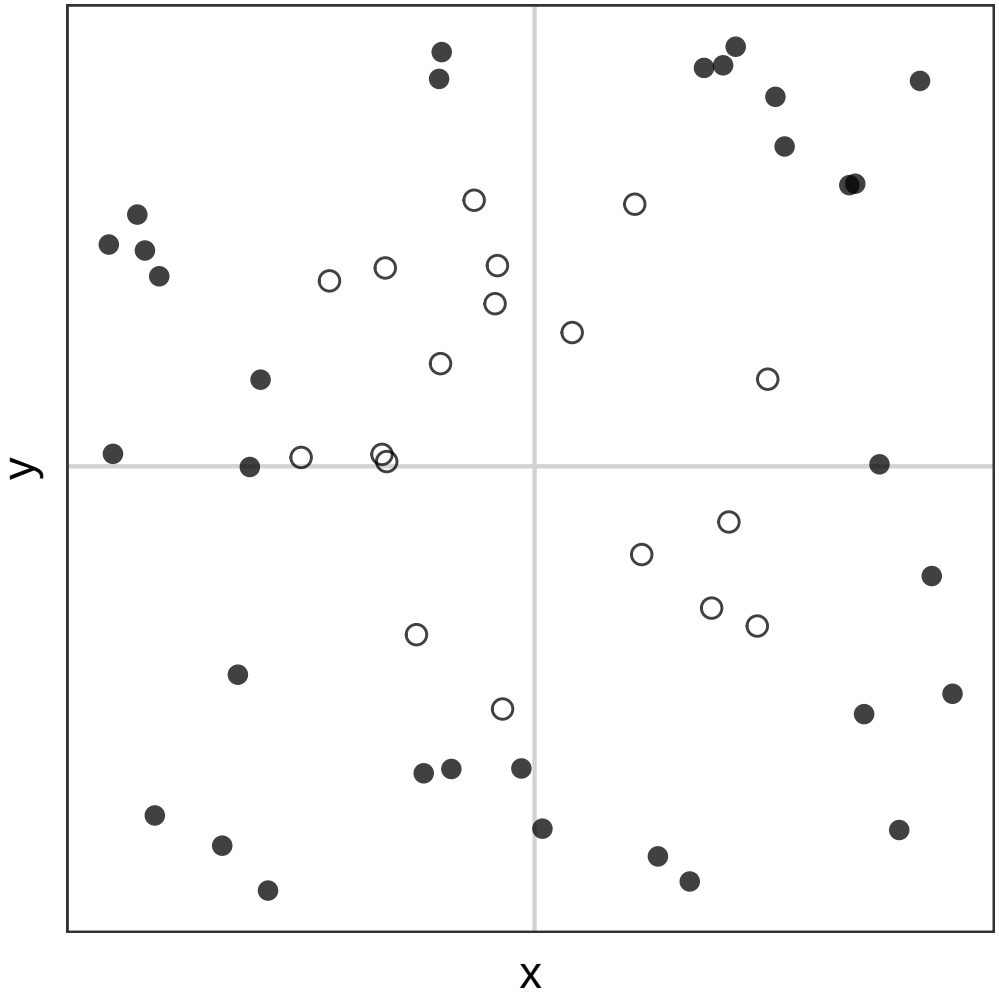
Taken from Hastie *et al.* (2009).

Linear SVM



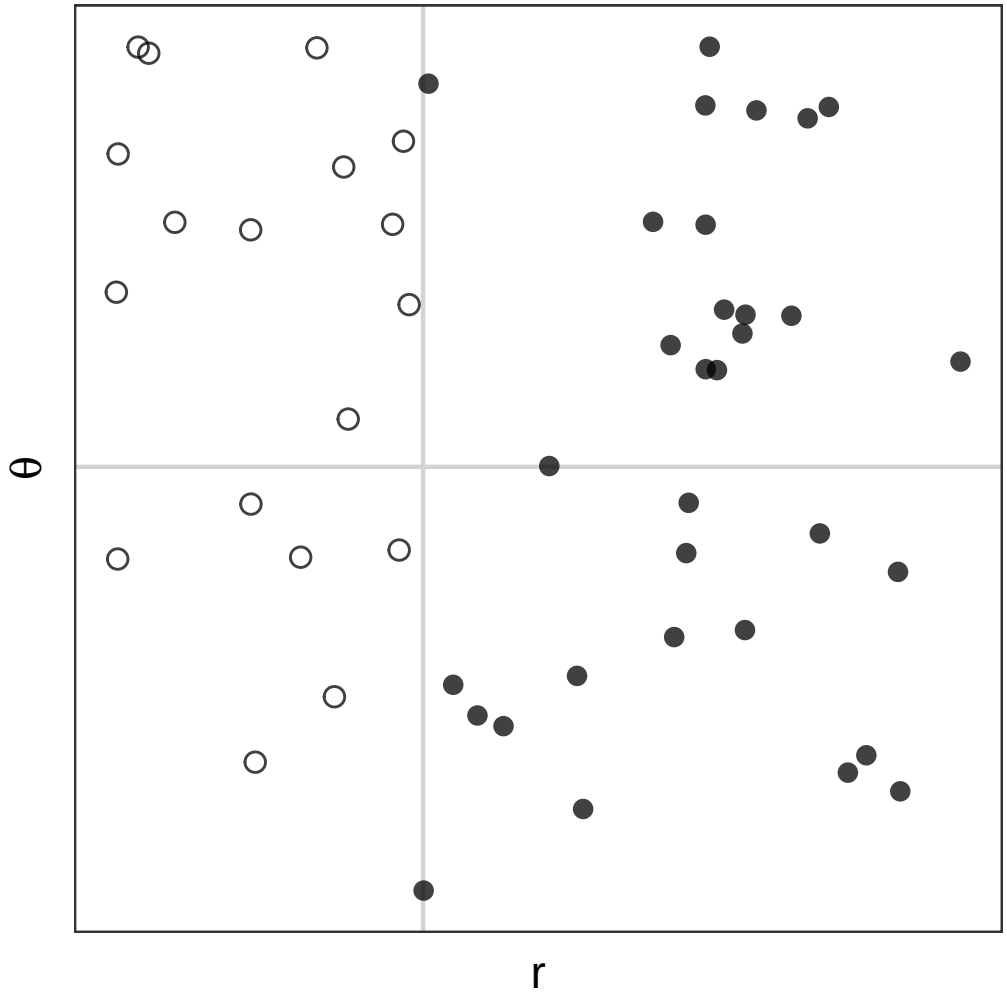
Nonlinear SVMs

Not every classification problem is linearly separable. . .



Nonlinear SVMs

Can fit SVM in nonlinearly transformed feature space.



Nonlinear SVMs

Can fit SVM in nonlinearly transformed feature space.

"Kernel trick" can be used to do this efficiently:

- Given transformation h, the kernel

$$k(\mathrm{x}, \mathrm{x}') = \langle h(\mathrm{x}), h(\mathrm{x}') \rangle$$

  is actually sufficient to fit SVM.

Nonlinear SVMs

Can fit SVM in nonlinearly transformed feature space.

"Kernel trick" can be used to do this efficiently:

- Given transformation h, the kernel

$$k(\mathrm{x}, \mathrm{x}') = \langle h(\mathrm{x}), h(\mathrm{x}') \rangle$$

  is actually sufficient to fit SVM.

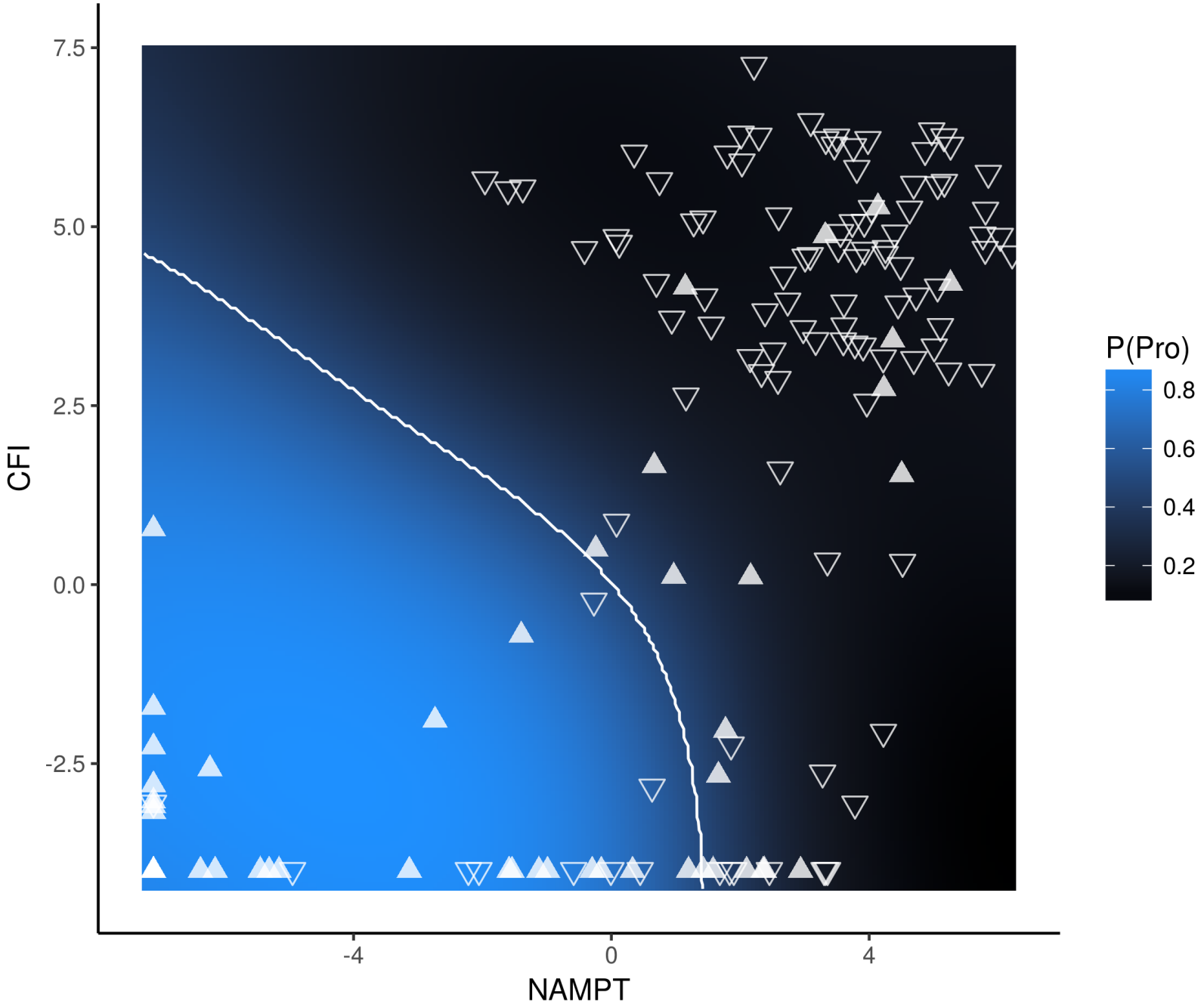
Most popular h is rather involved transformation designed to produce the radial basis kernel

$$k(\mathrm{x}, \mathrm{x}') = \exp\left(-\gamma \|\mathrm{x} - \mathrm{x}'\|^2\right)$$

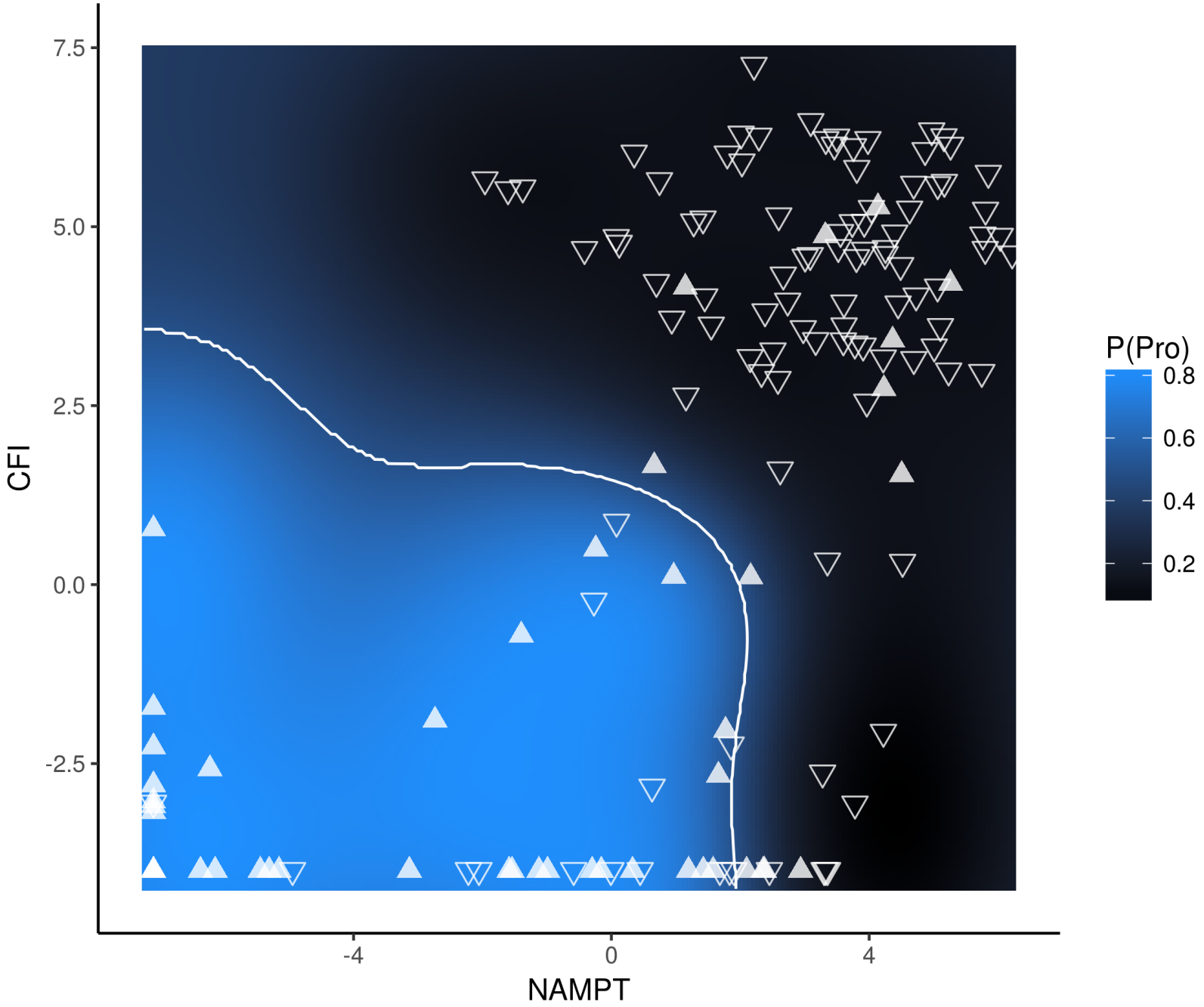
Intuition:

- SVMs classify a sample with features x based on (known) classes of similar training data x_i,
- where "similarity" is quantified by the kernel $k(\mathrm{x}, \mathrm{x}_i)$.

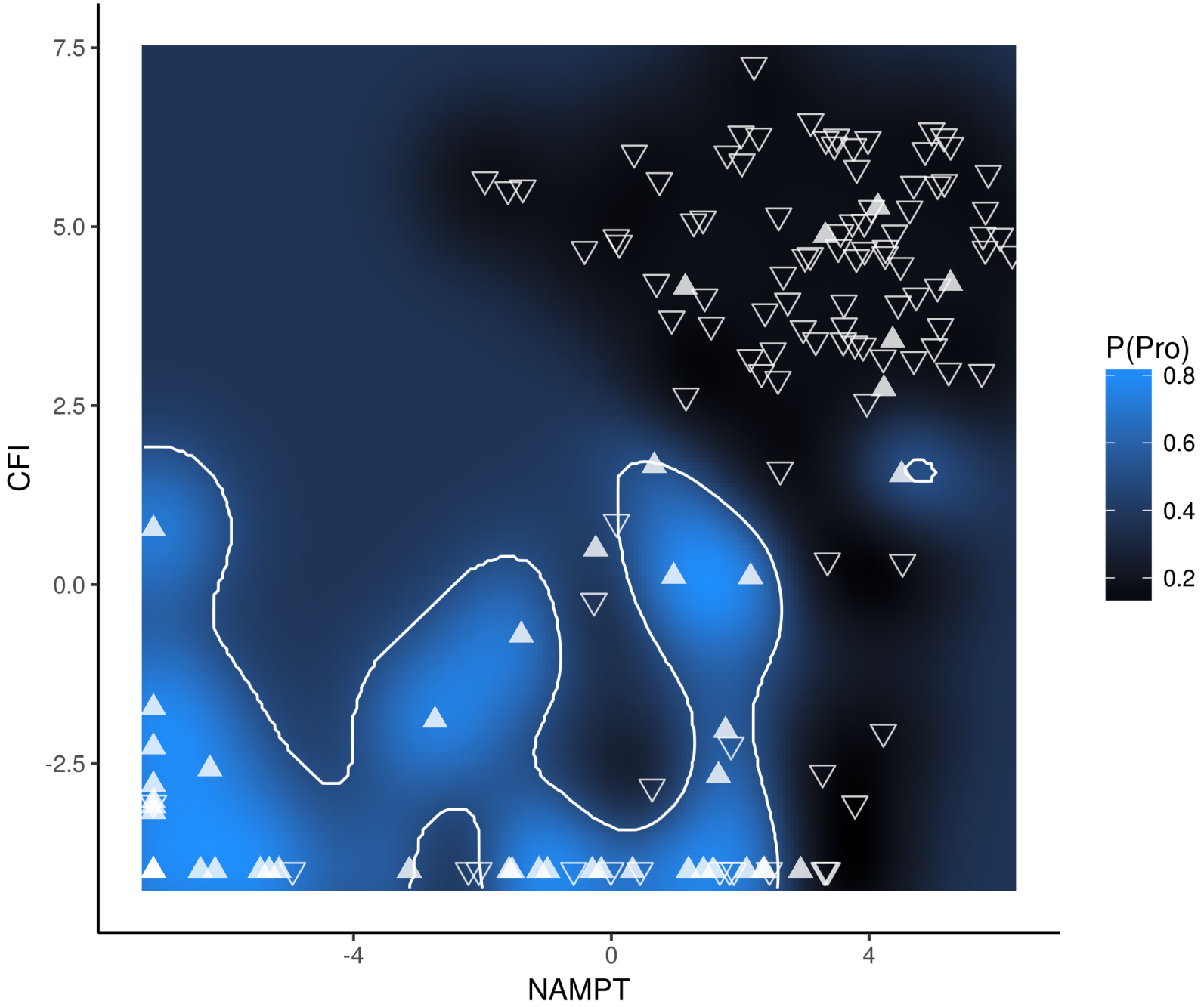
Radial SVM: $C = 1, \gamma = 0.5$



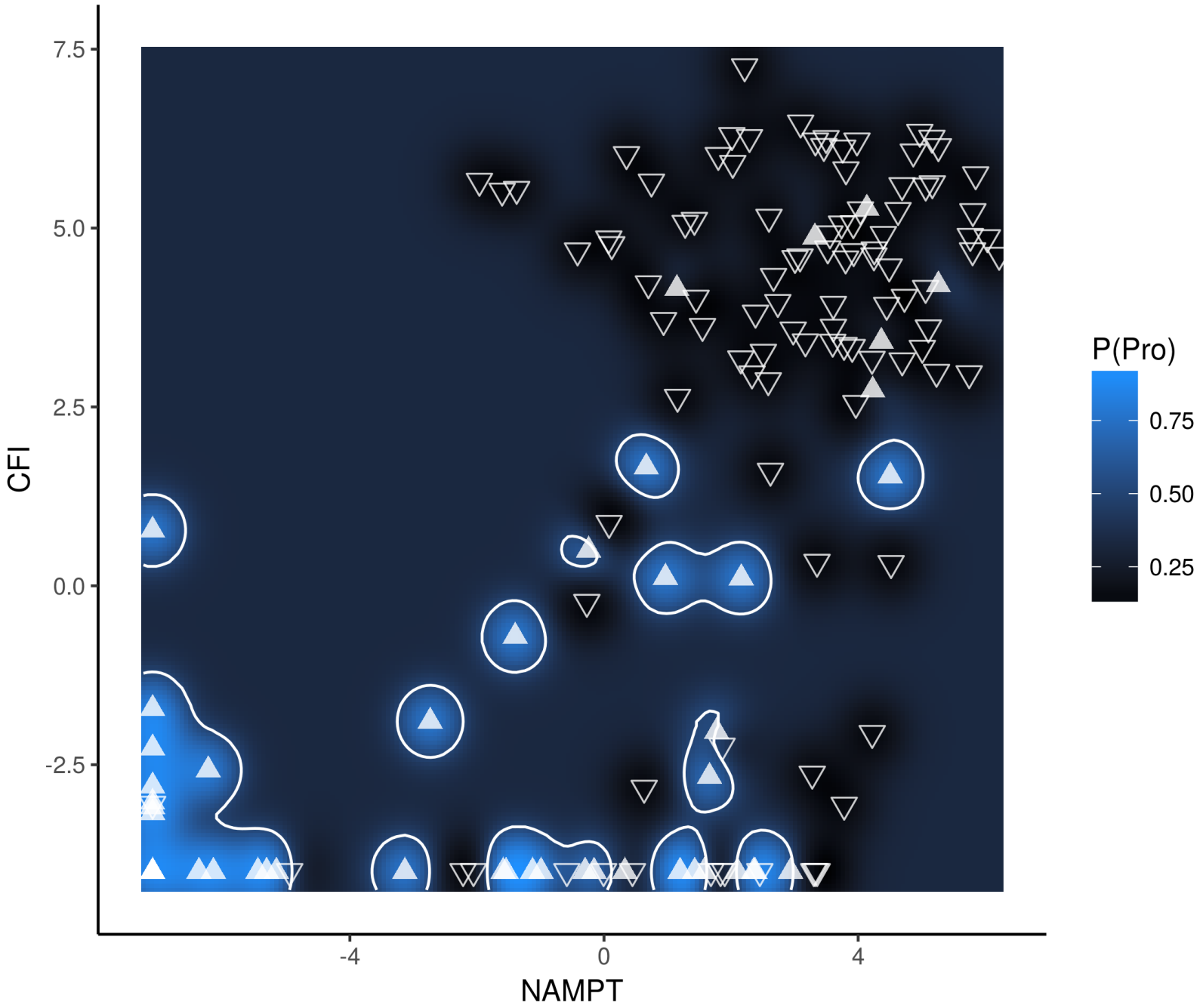
Radial SVM: $C = 1, \gamma = 2.5$



Radial SVM: $C = 1, \gamma = 12.5$



Radial SVM: $C = 1, \gamma = 62.5$



References I

Hastie, Trevor, Tibshirani, Robert, & Friedman, Jerome. 2009. *The Elements of Statistical Learning*. Springer.

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.

Kang, Huining, Chen, I-Ming, Wilson, Carla S, Bedrick, Edward J, Harvey, Richard C, Atlas, Susan R, Devidas, Meenakshi, Mullighan, Charles G, Wang, Xuefei, Murphy, Maurice, *et al.* . 2010. Gene expression classifiers for relapse-free survival and minimal residual disease improve risk classification and outcome prediction in pediatric B-precursor acute lymphoblastic leukemia. *Blood*, **115**(7), 1394–1405.

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

Saeys, Yvan, Inza, Iñaki, & Larrañaga, Pedro. 2007. A review of feature selection techniques in bioinformatics. *Bioinformatics*, **23**(19), 2507–2517.