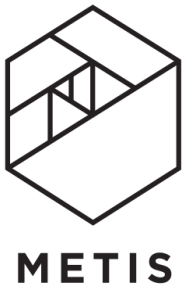
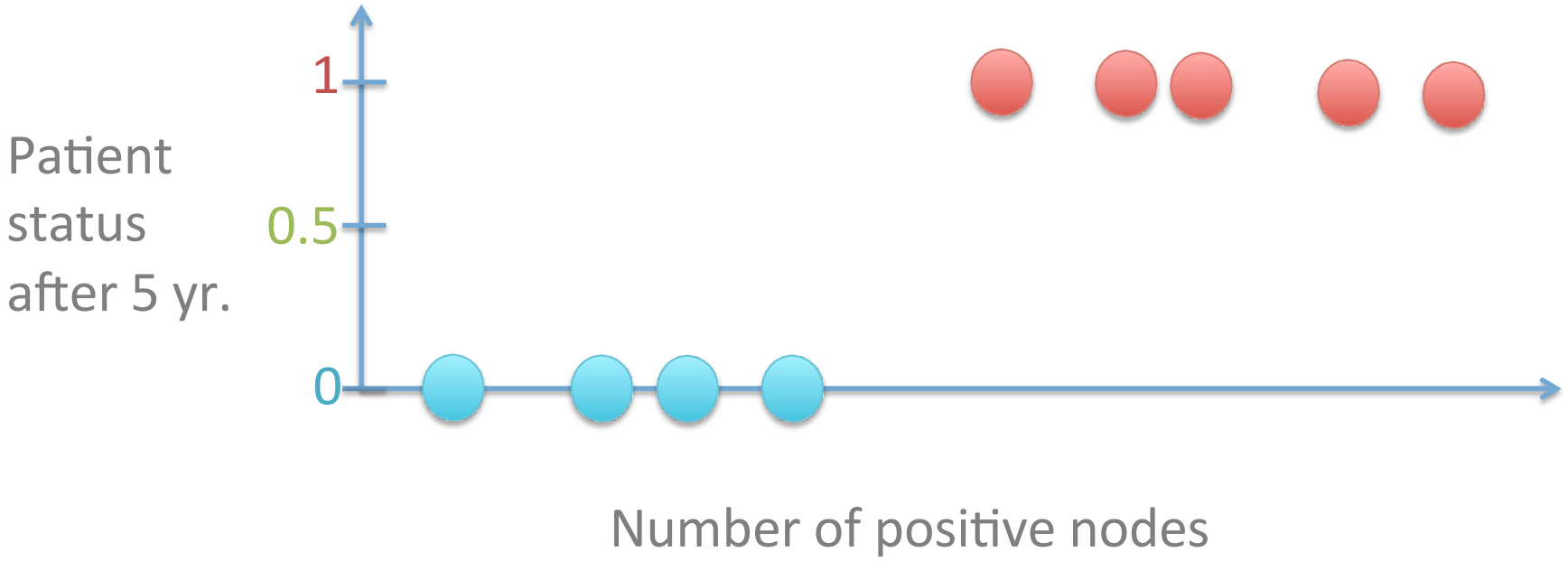
Support Vector Machines

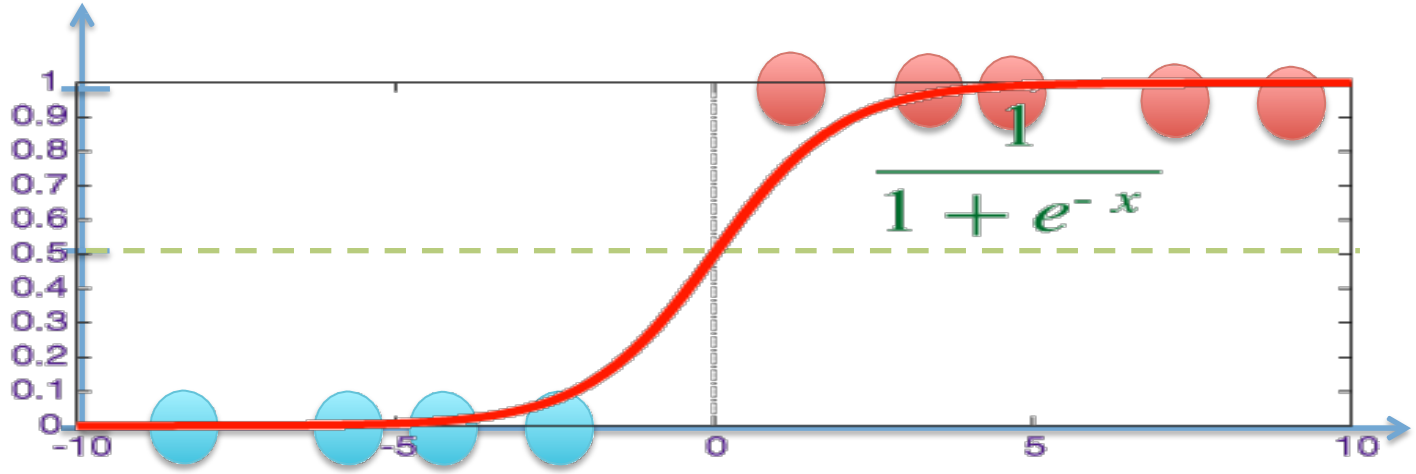


1 feature (num. nodes), 2 labels (survived/not)



Logistic Regression

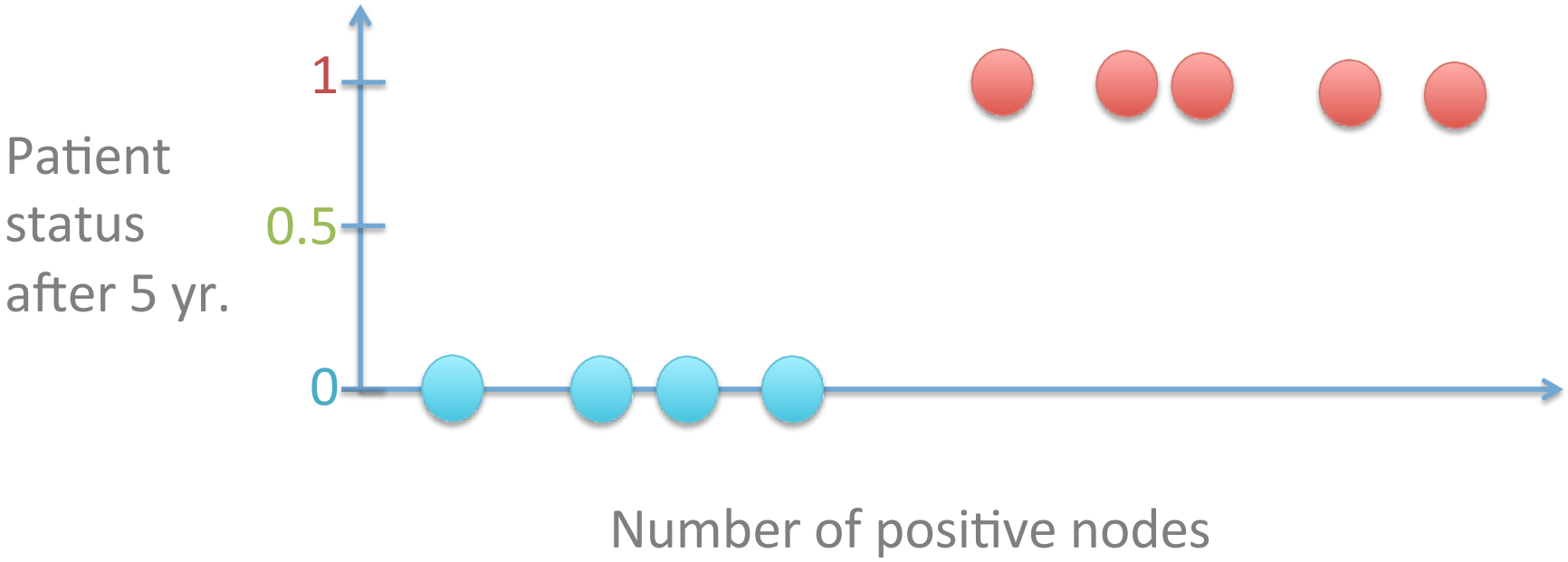
Patient status after 5 yr.



Number of positive nodes

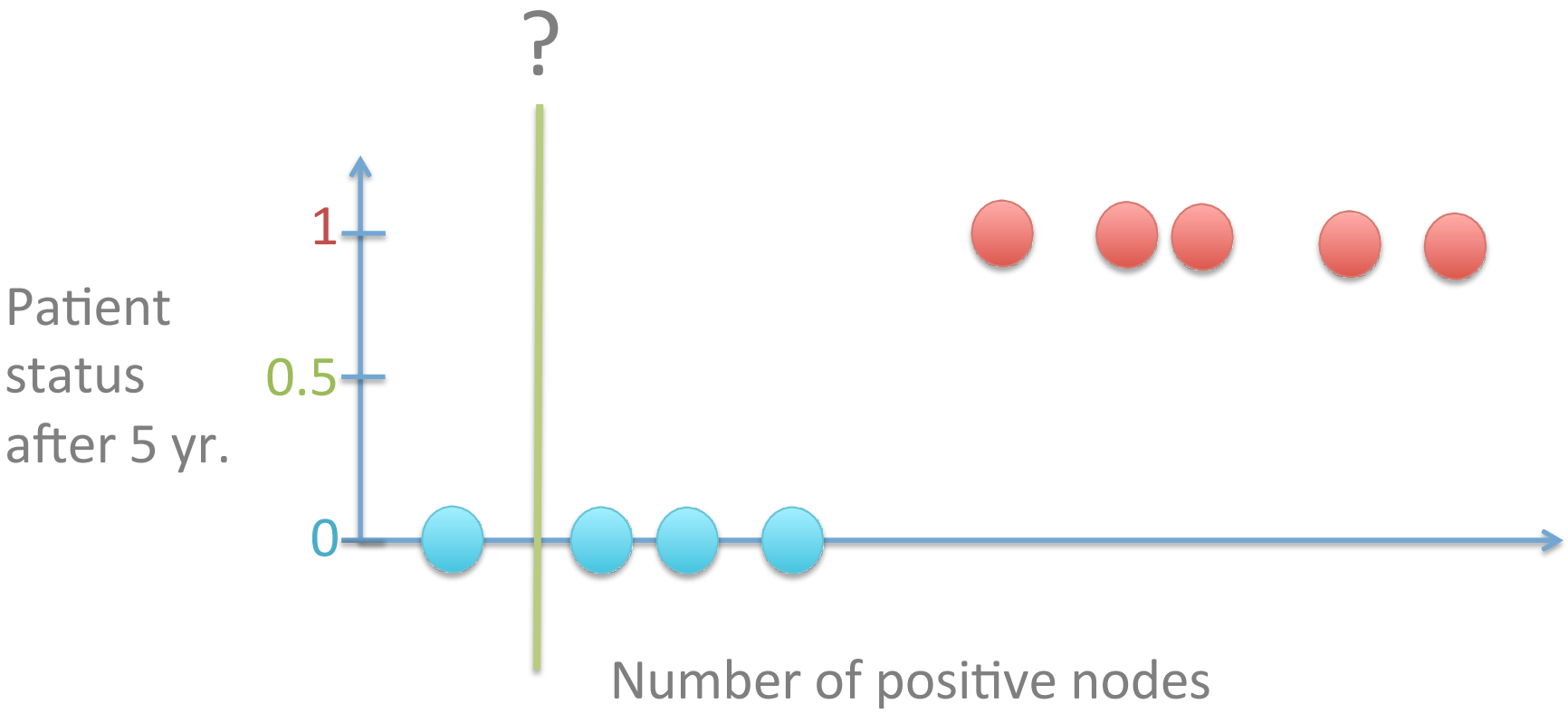
$$y_\beta(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x+\varepsilon)}}$$

Support Vector Machine (SVM)



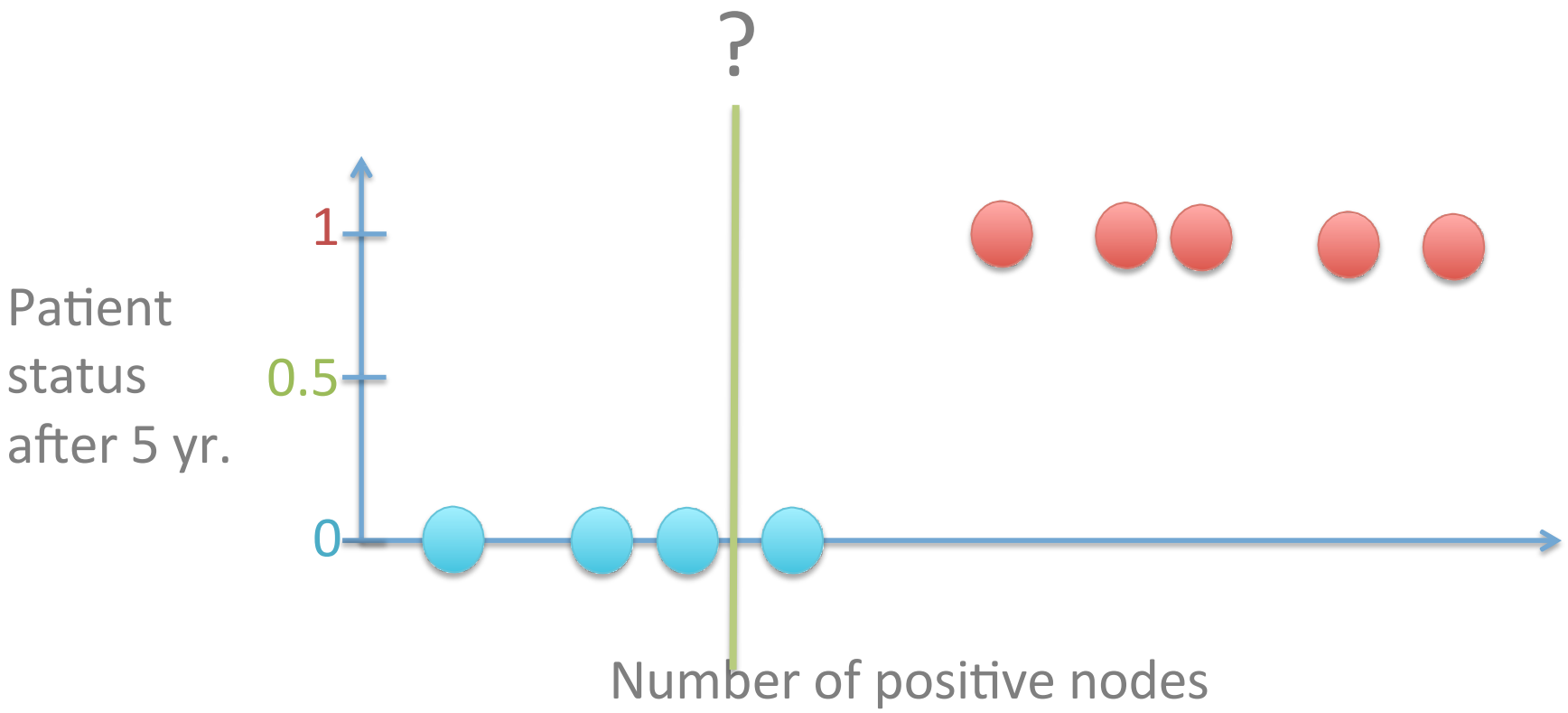
Find the best boundary that separates two classes

Support Vector Machine (SVM)



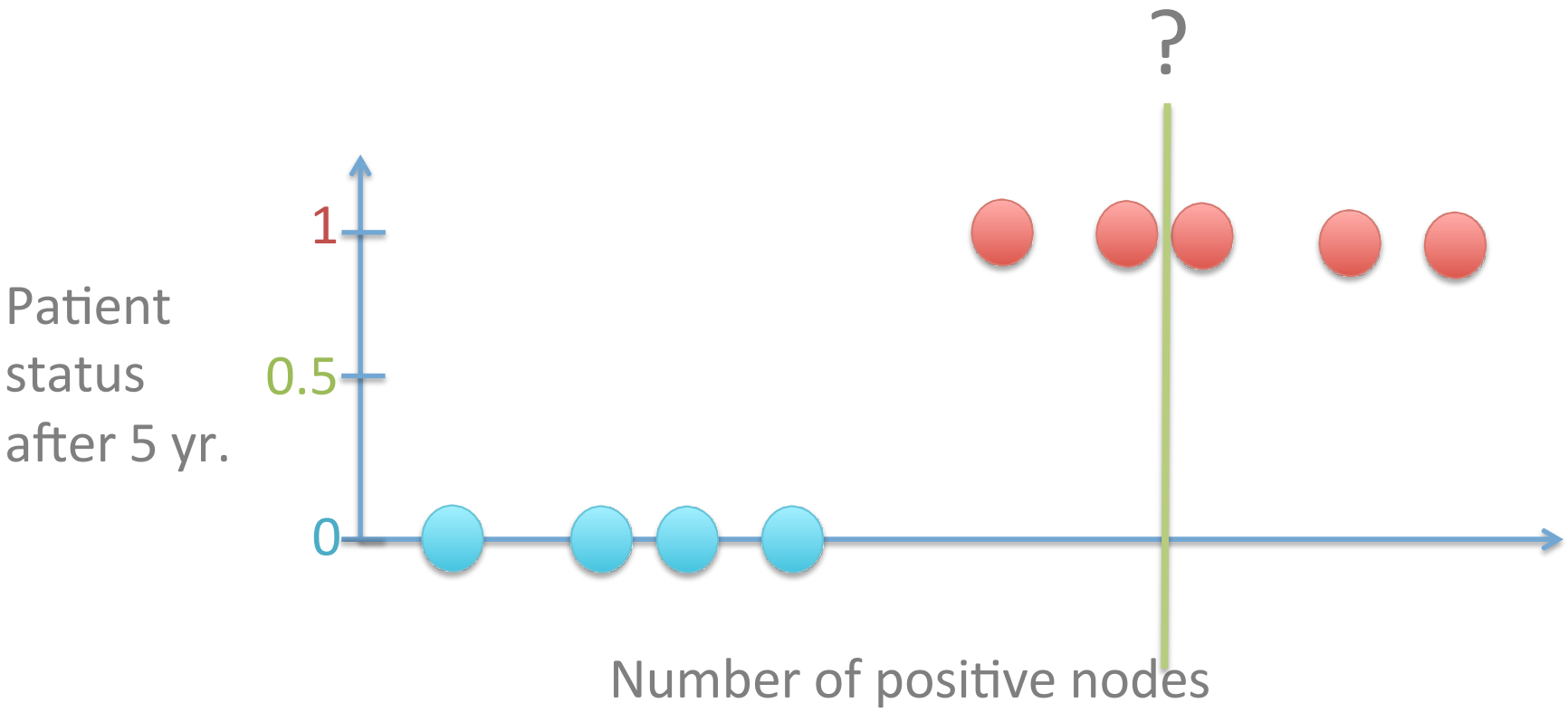
Bad: 3 misclassifications, accuracy 67%

Support Vector Machine (SVM)



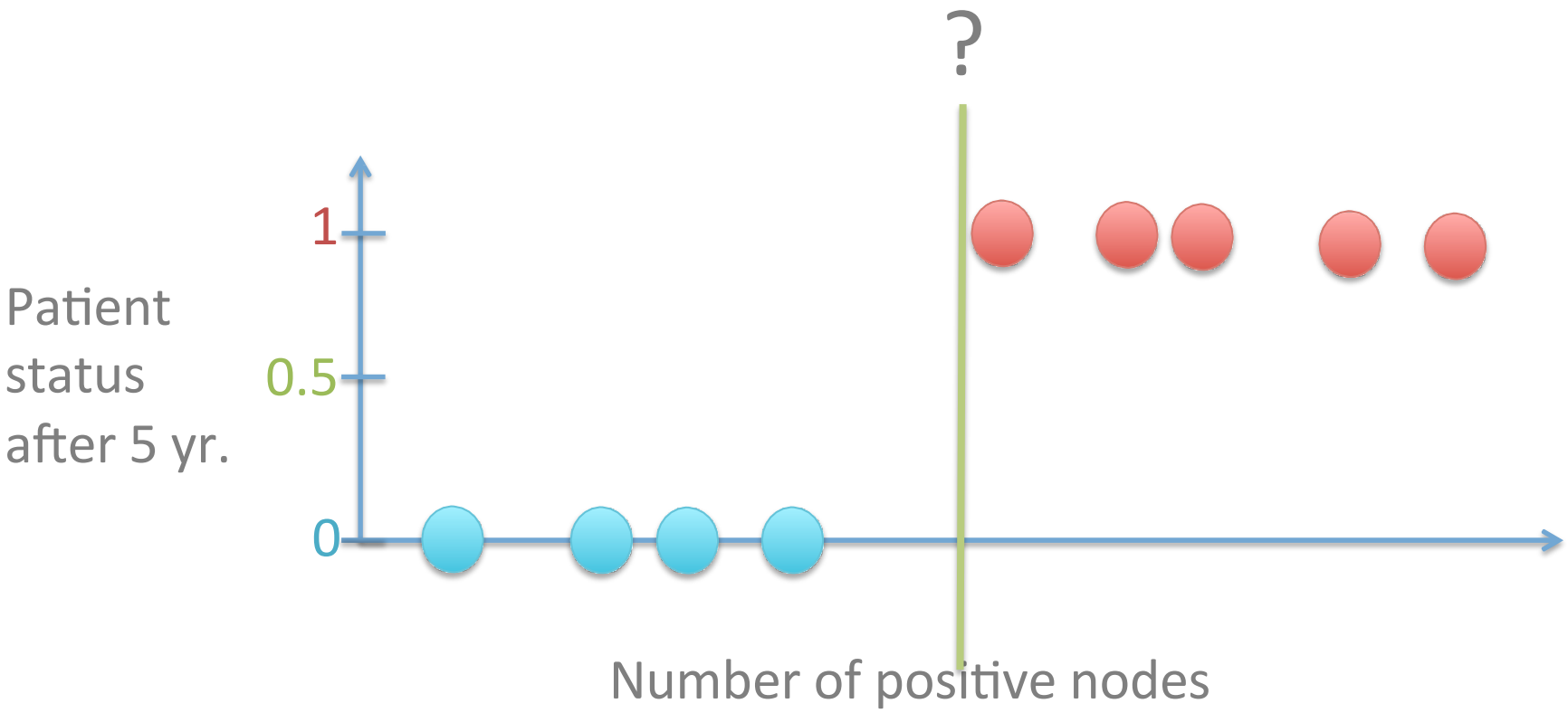
One misclassification, accuracy 89%

Support Vector Machine (SVM)



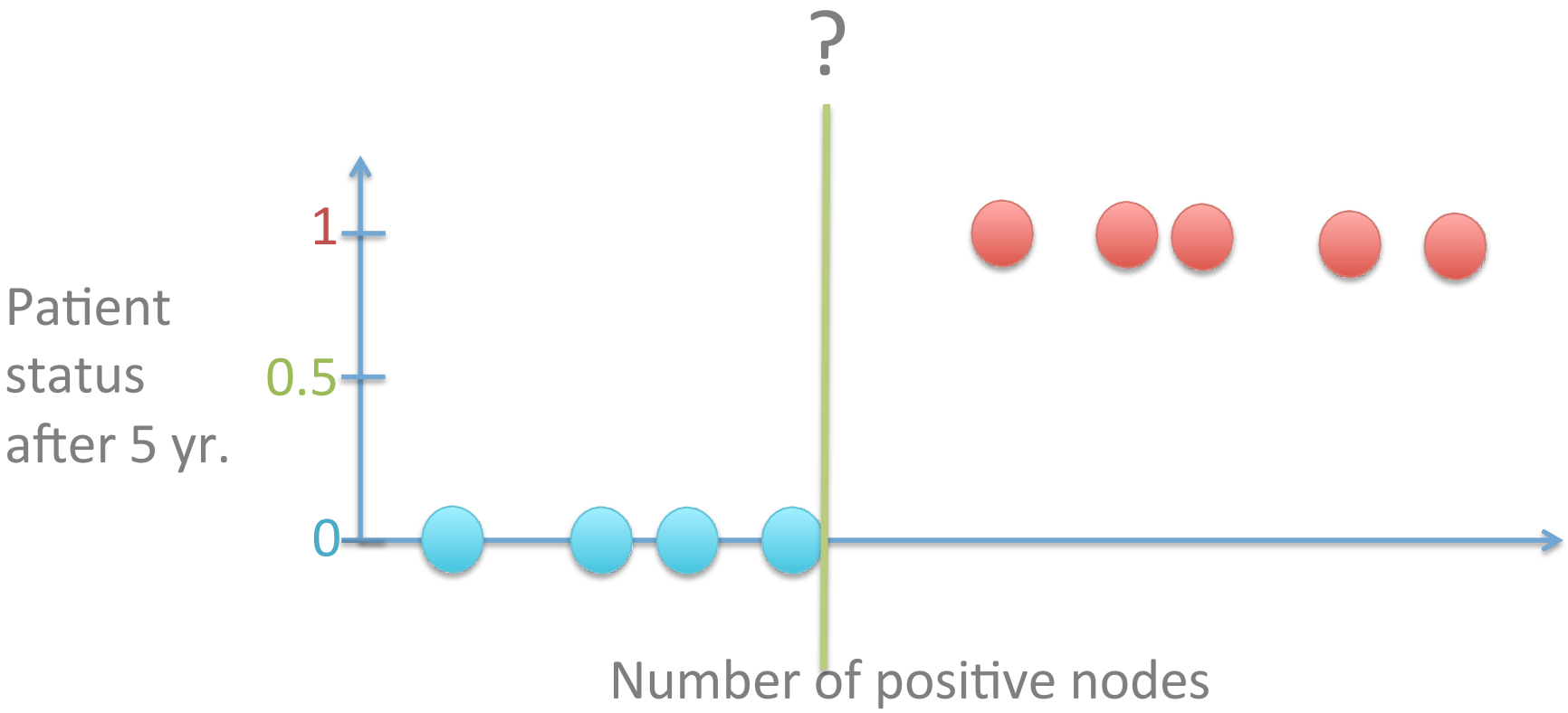
Accuracy: 78%

Support Vector Machine (SVM)



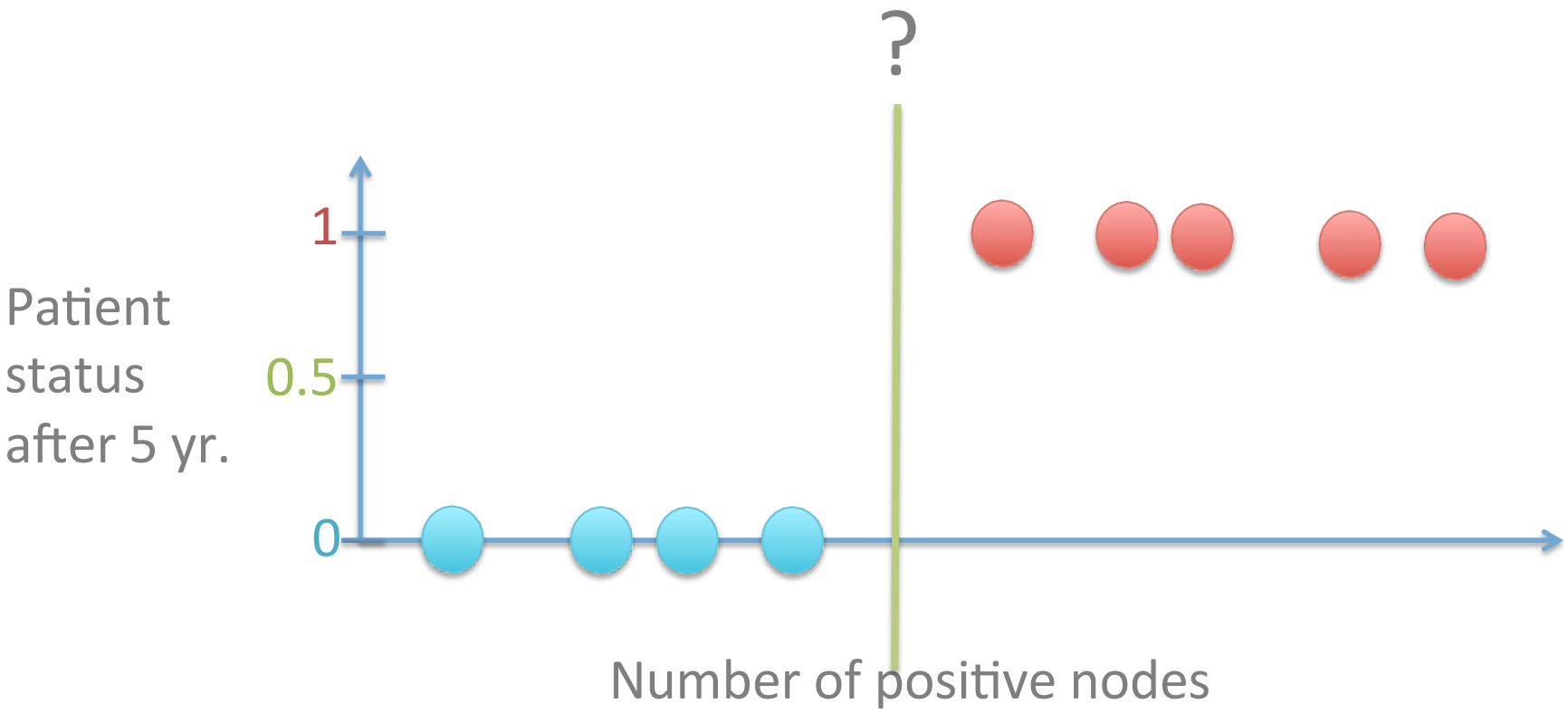
Accuracy: 100%

Support Vector Machine (SVM)



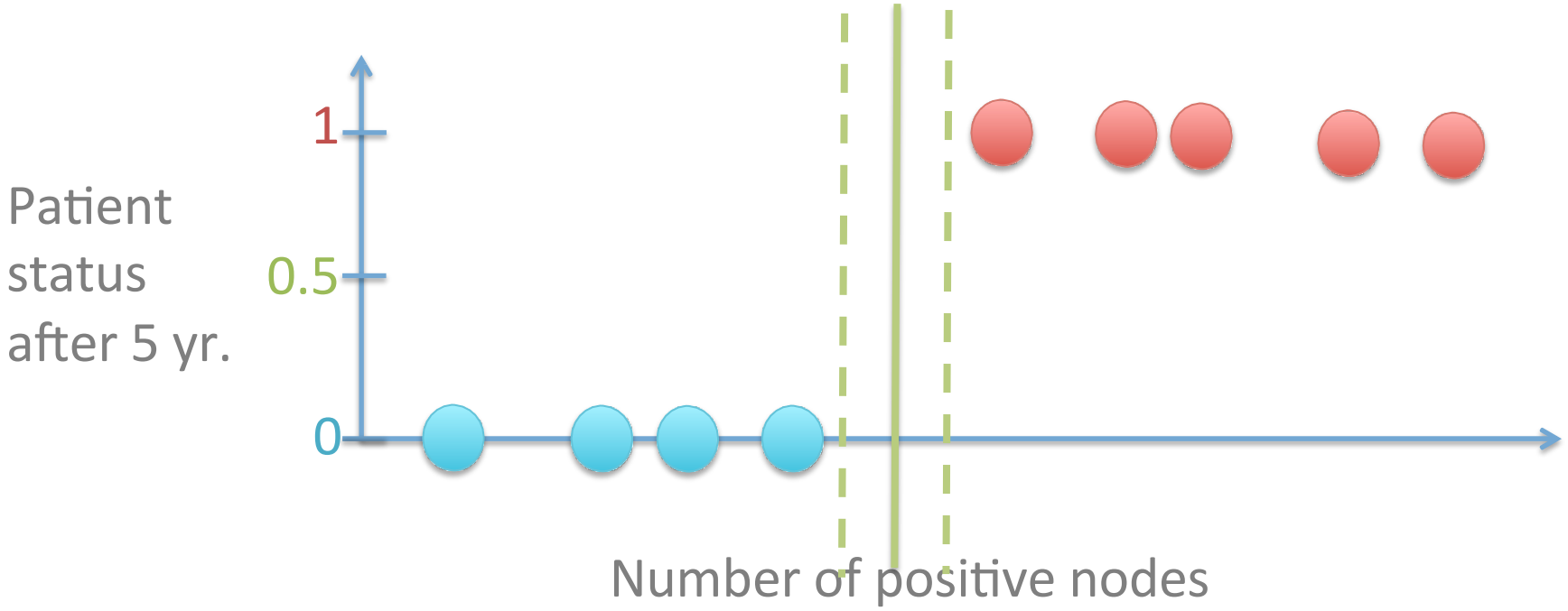
Accuracy: 100%

Support Vector Machine (SVM)



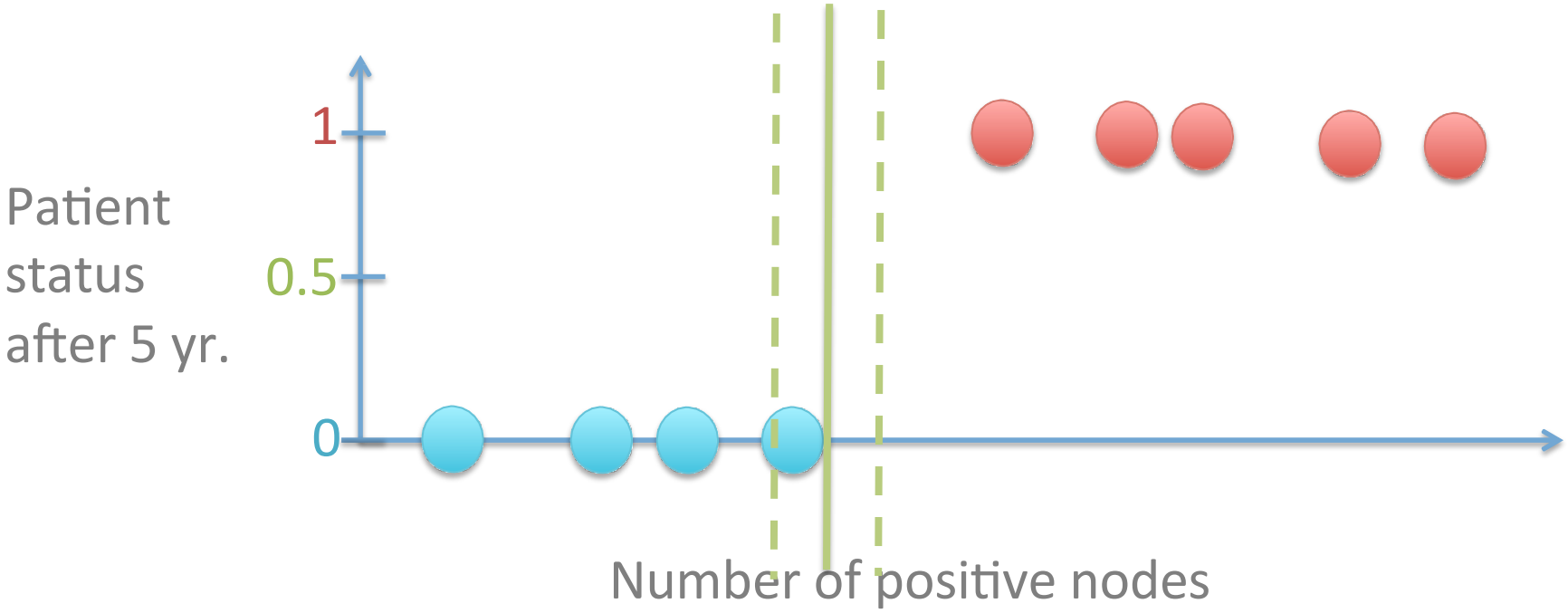
Accuracy: 100%

Support Vector Machine (SVM)



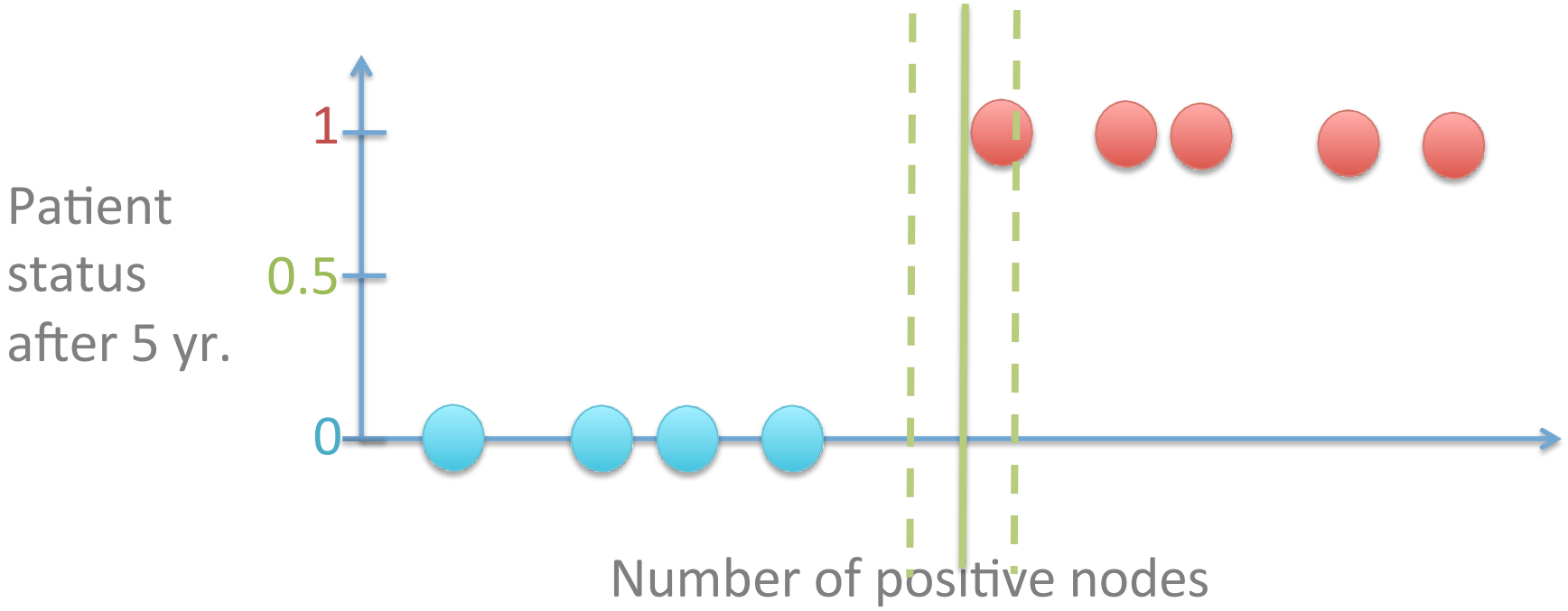
The margin: No man's land

Support Vector Machine (SVM)



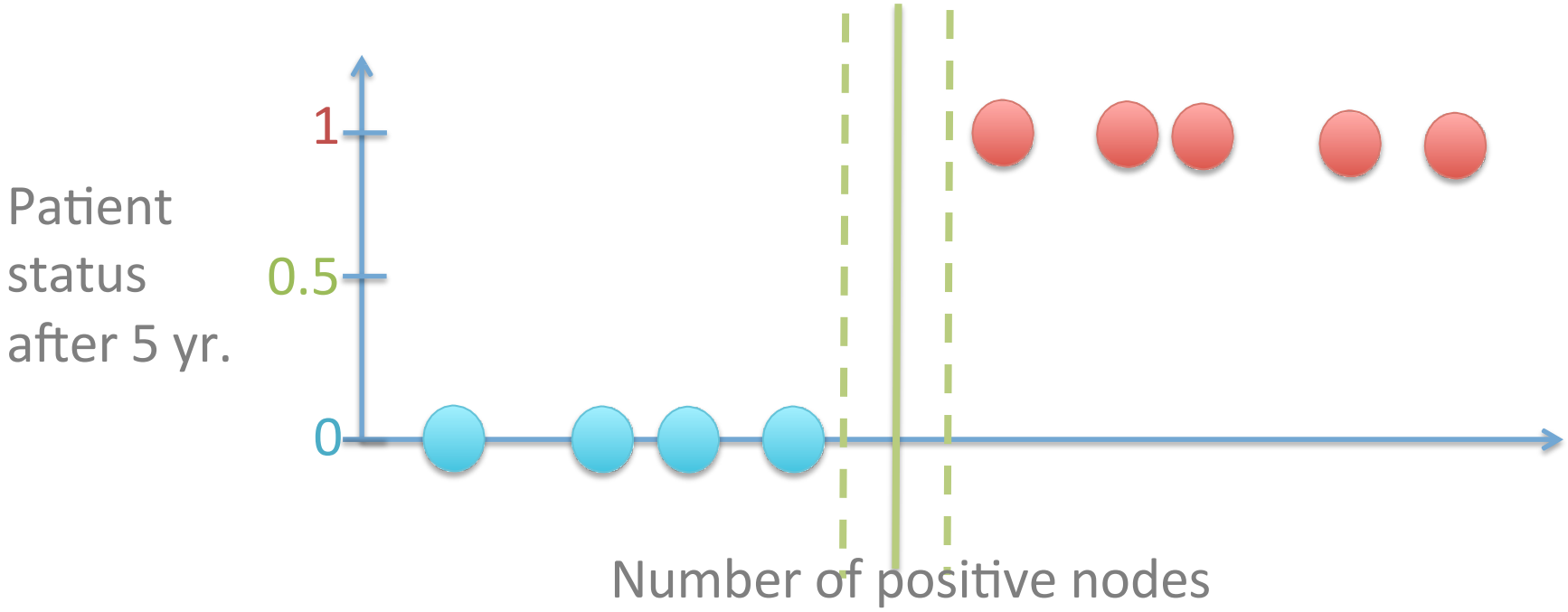
Extra error due to point in margin

Support Vector Machine (SVM)



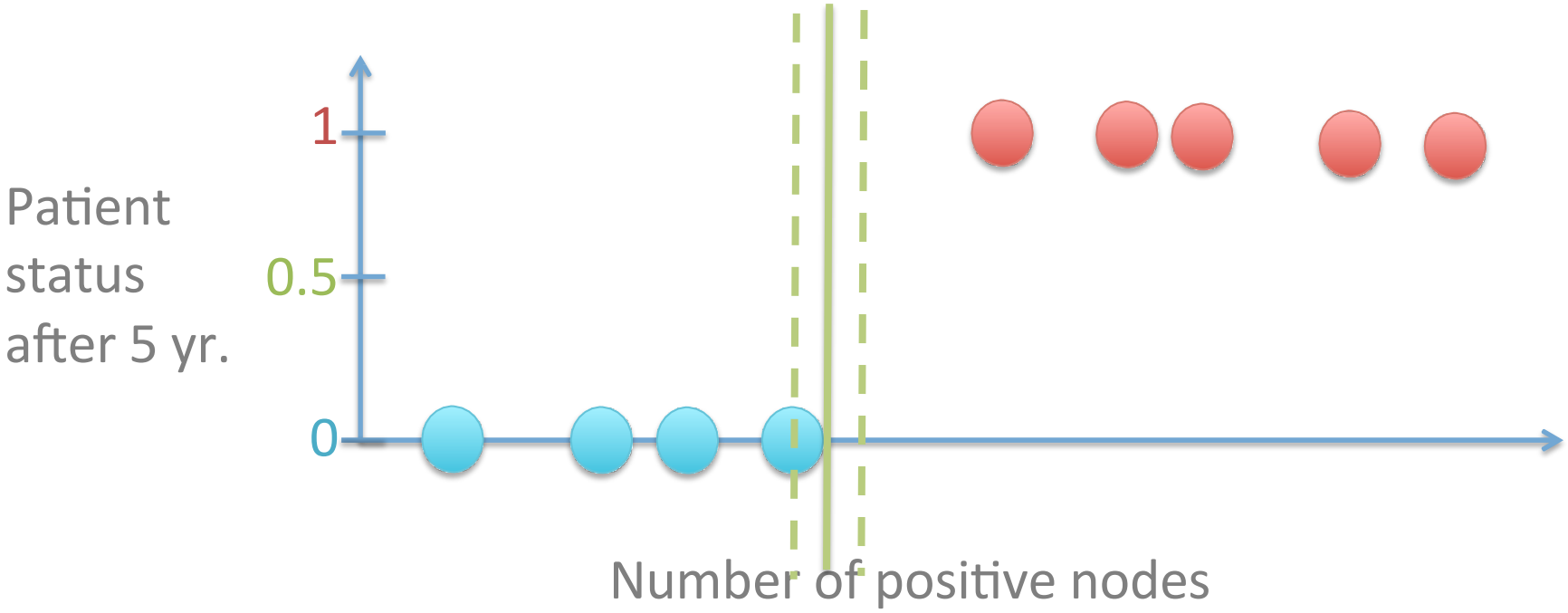
Extra error due to point in margin

Support Vector Machine (SVM)



Best boundary

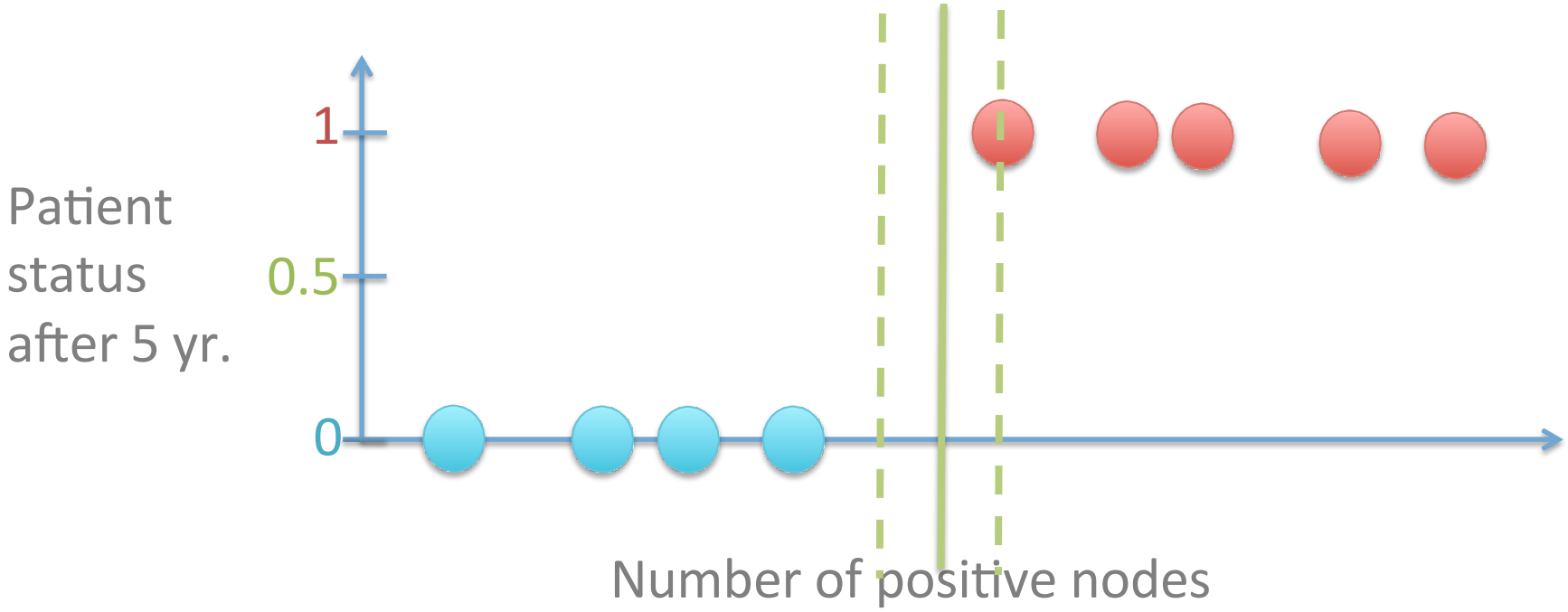
Support Vector Machine (SVM)



An even better way of doing this:

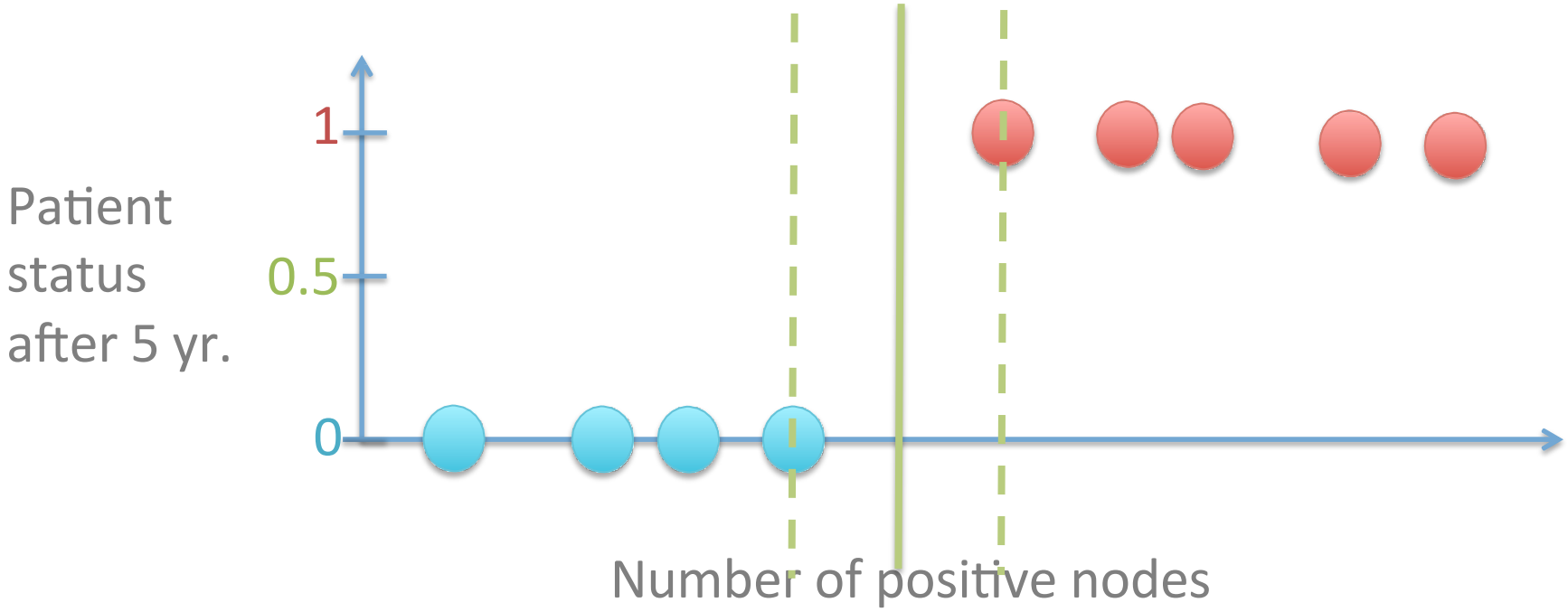
Find the boundary with the largest margin

Support Vector Machine (SVM)



Extra error due to point in margin

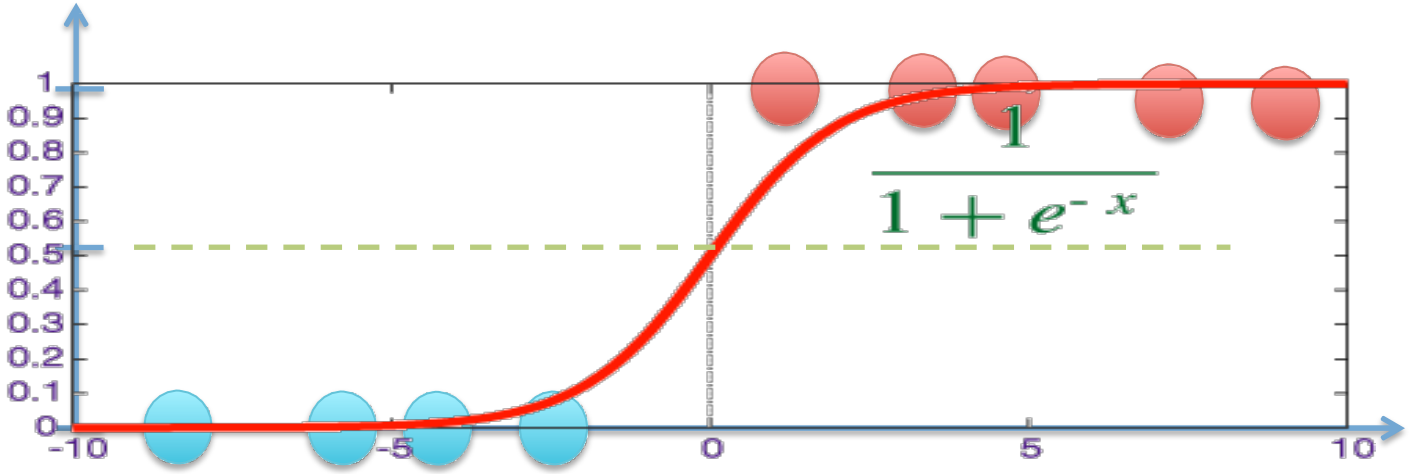
Support Vector Machine (SVM)



Best boundary

Logistic Regression

Patient status after 5 yr.

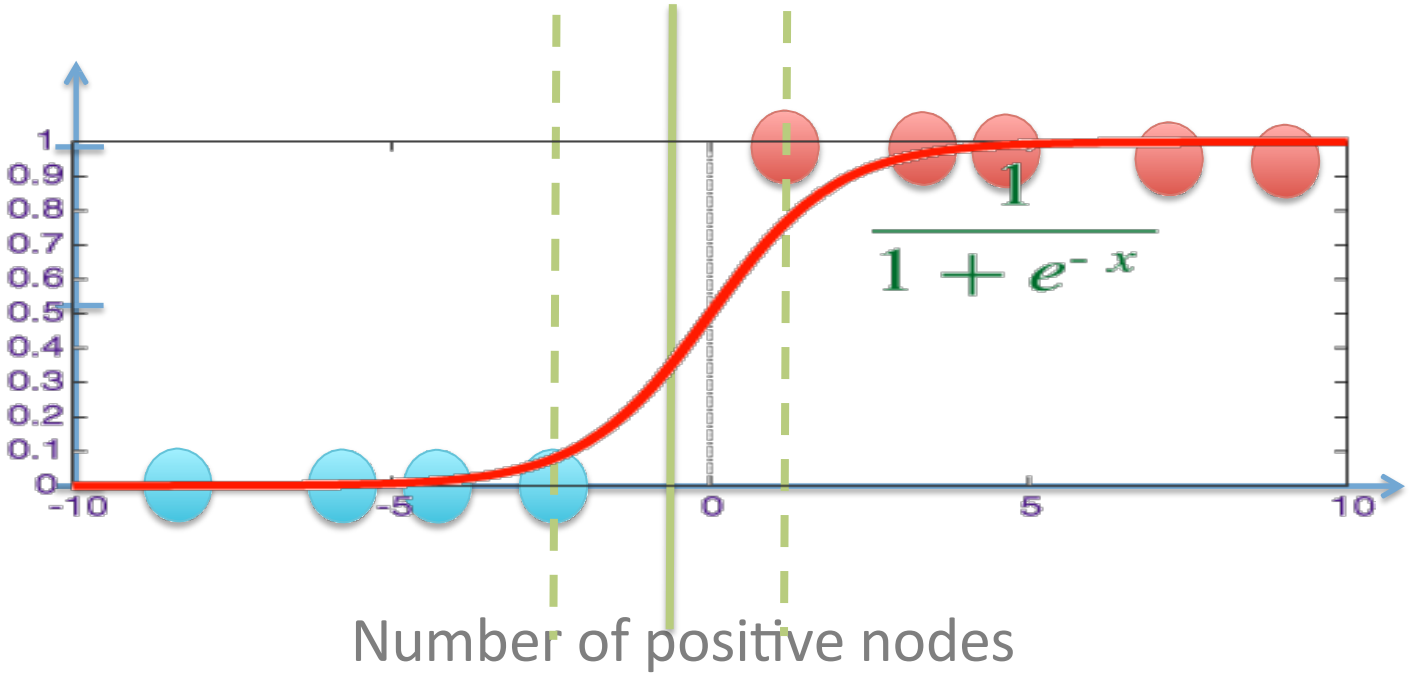


Number of positive nodes

$$y_\beta(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x+\varepsilon)}}$$

Logistic Regression and SVM are not very different

Patient status after 5 yr.

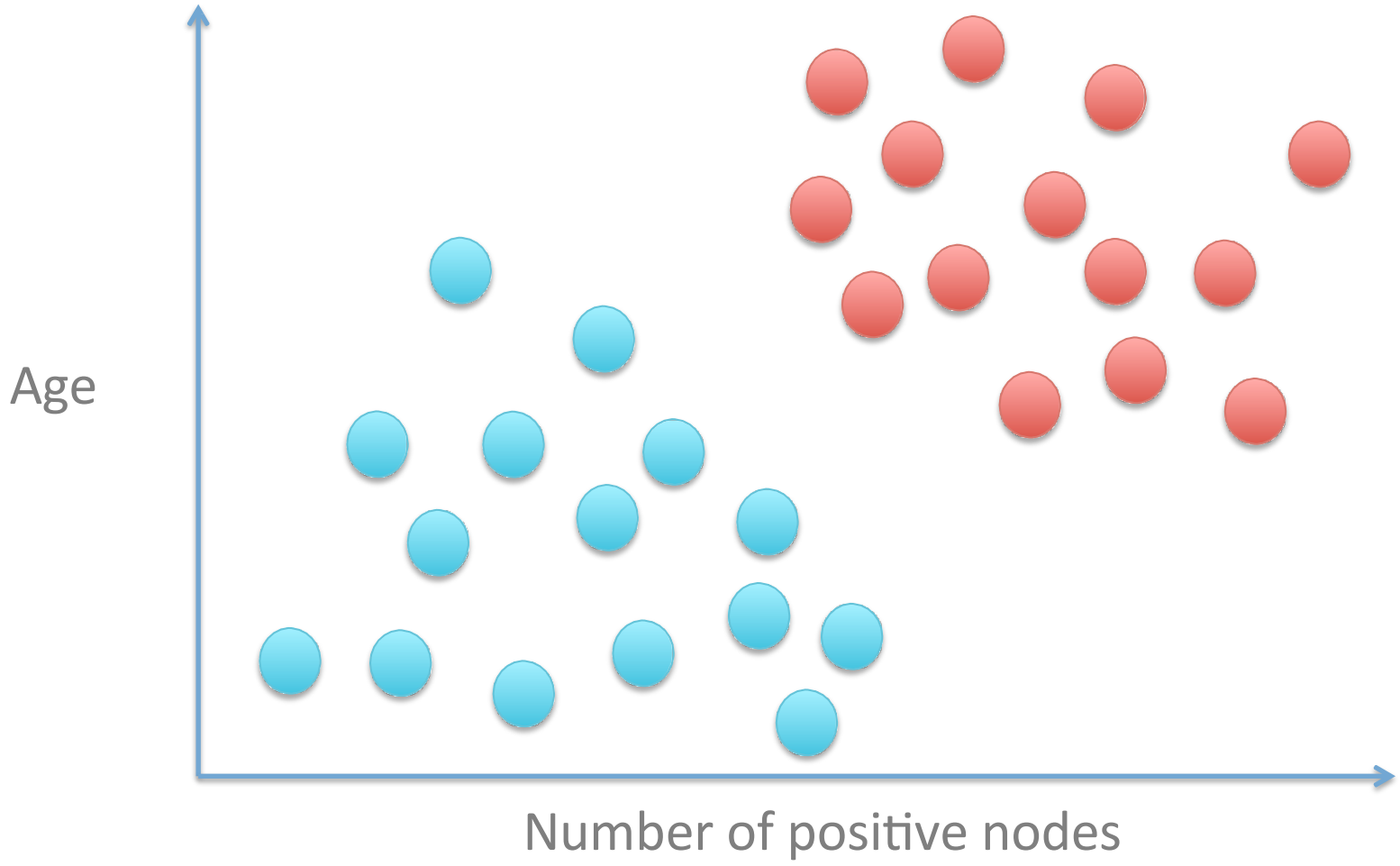


Number of positive nodes

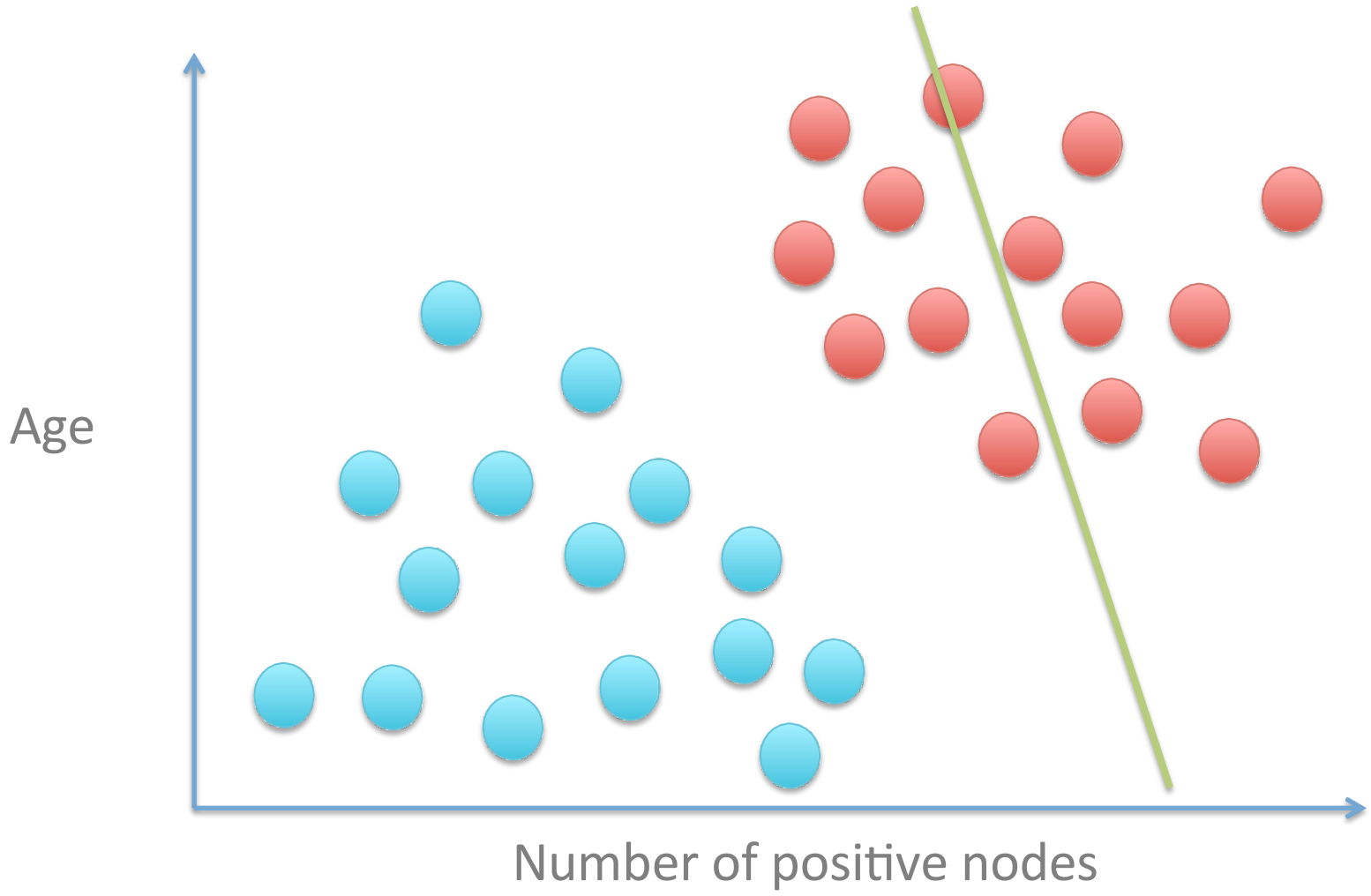
Best boundary

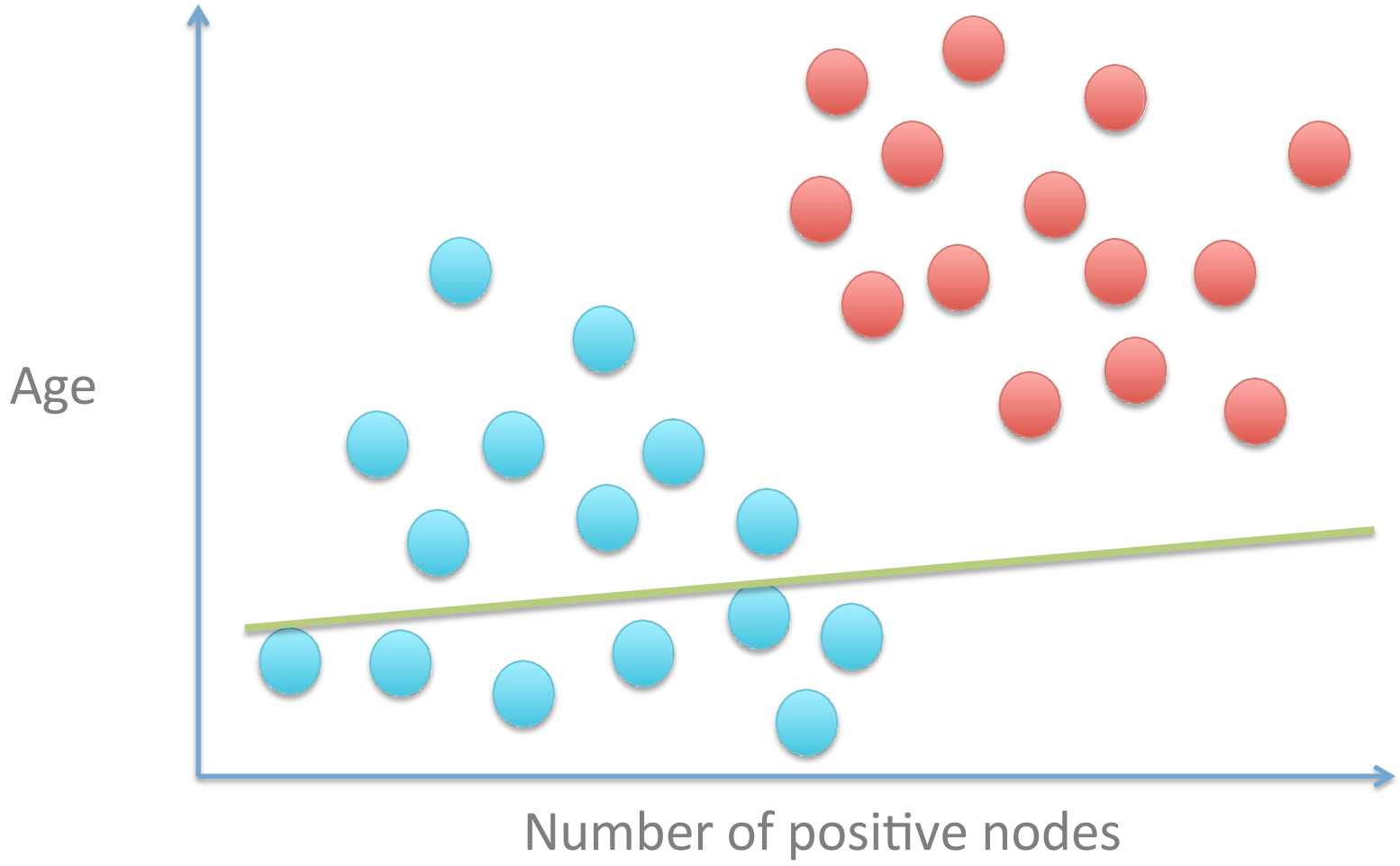
2 Features: Number of + nodes, Age

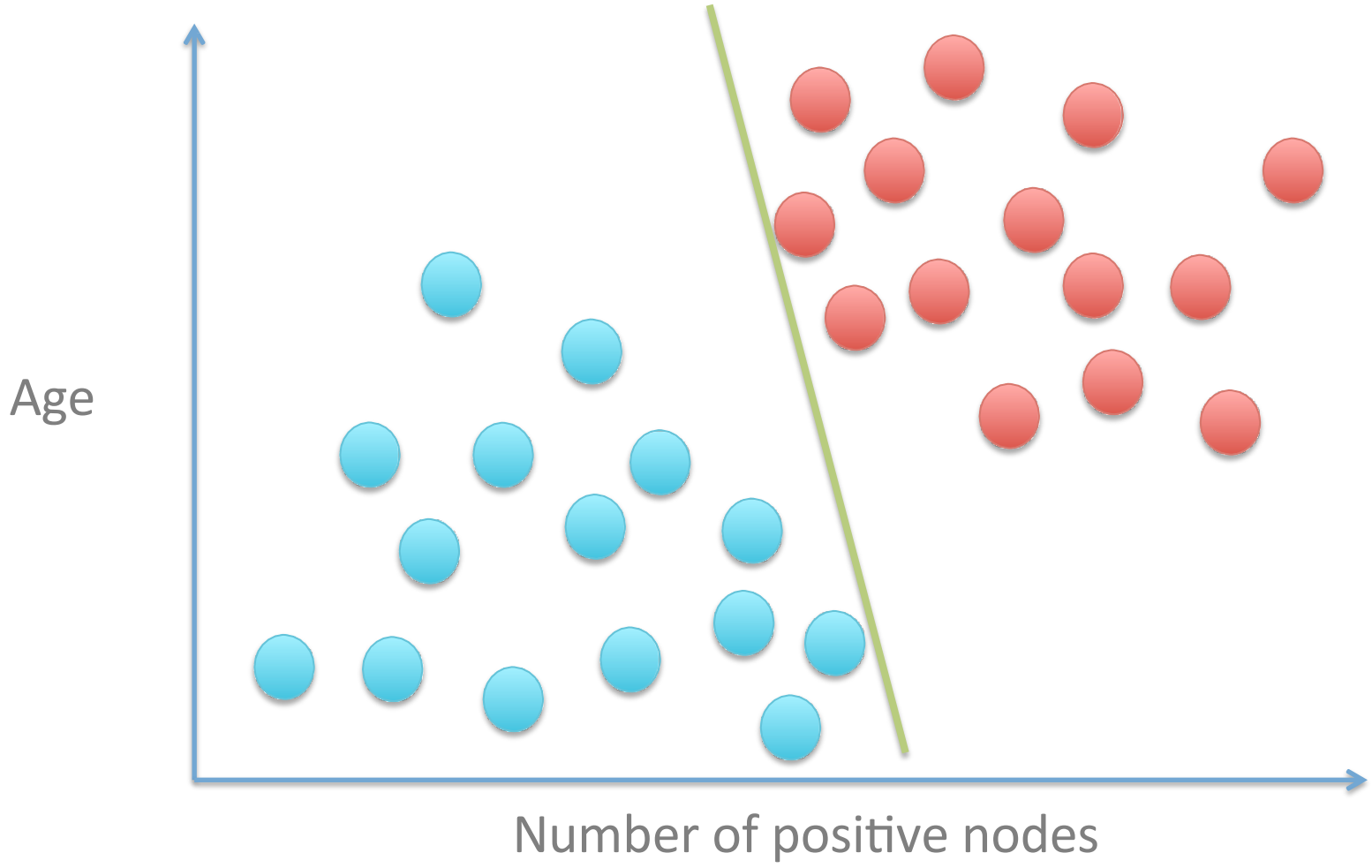
2 Labels: Survived / Lost

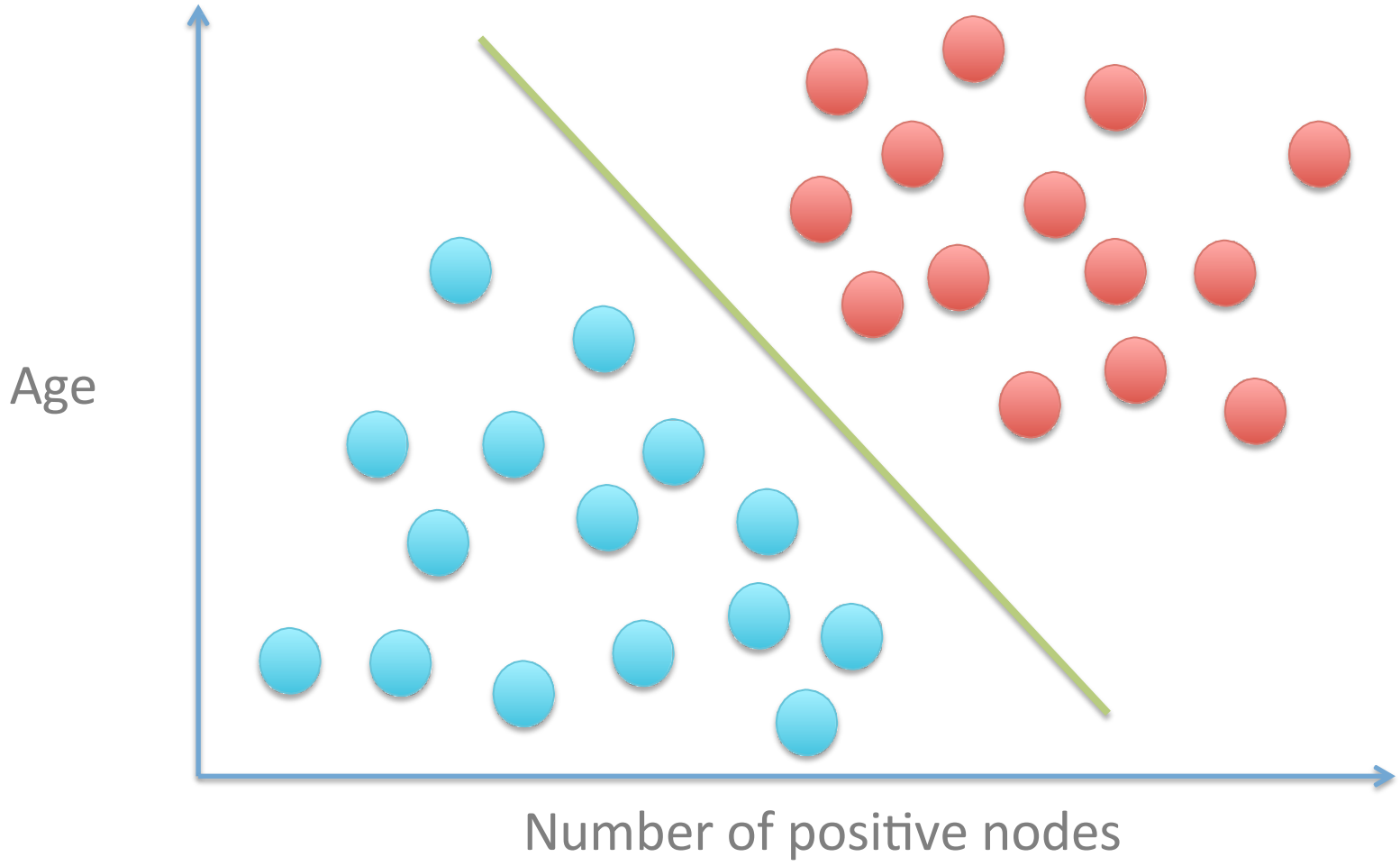


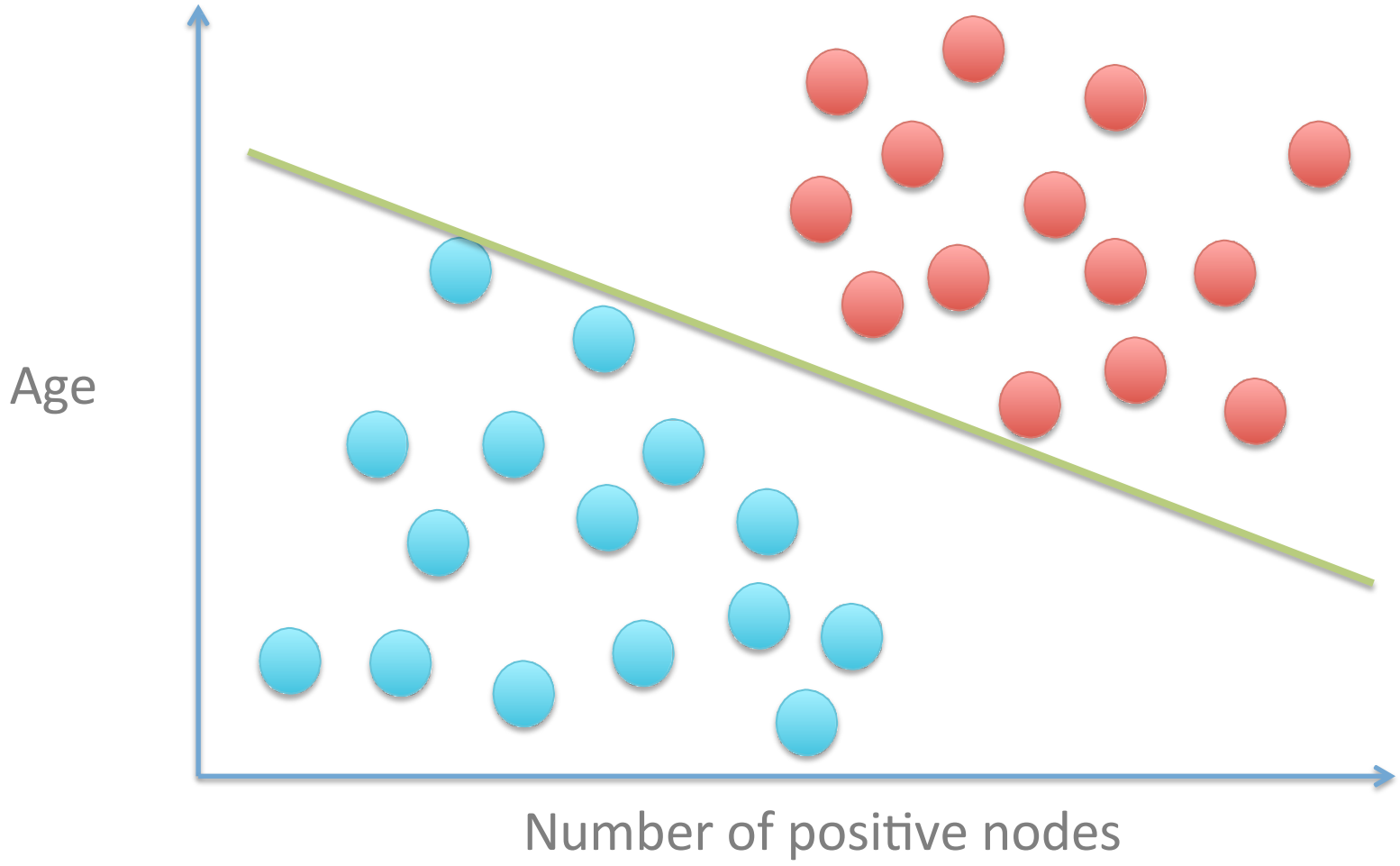
Find the line that separates the classes best

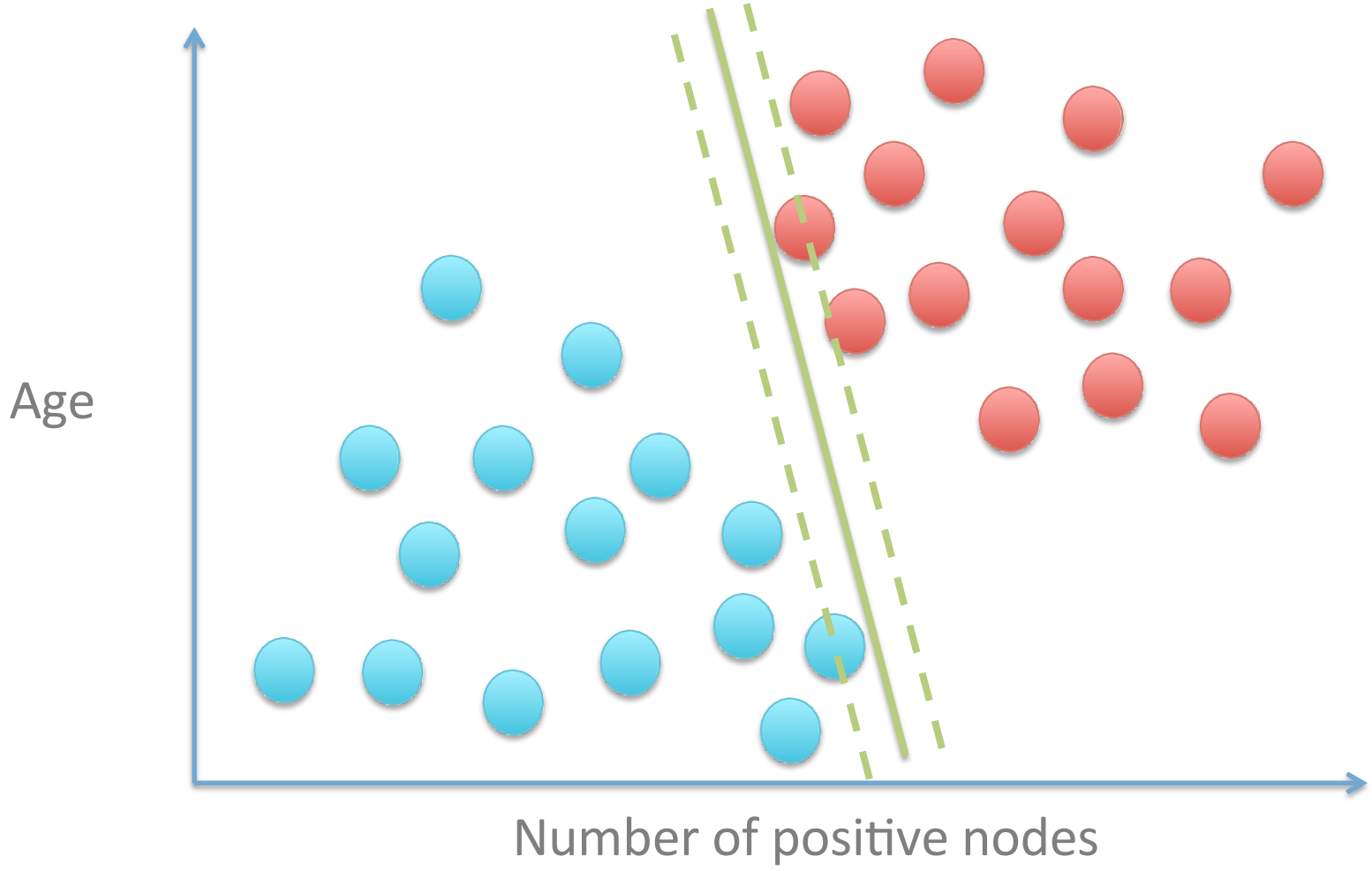


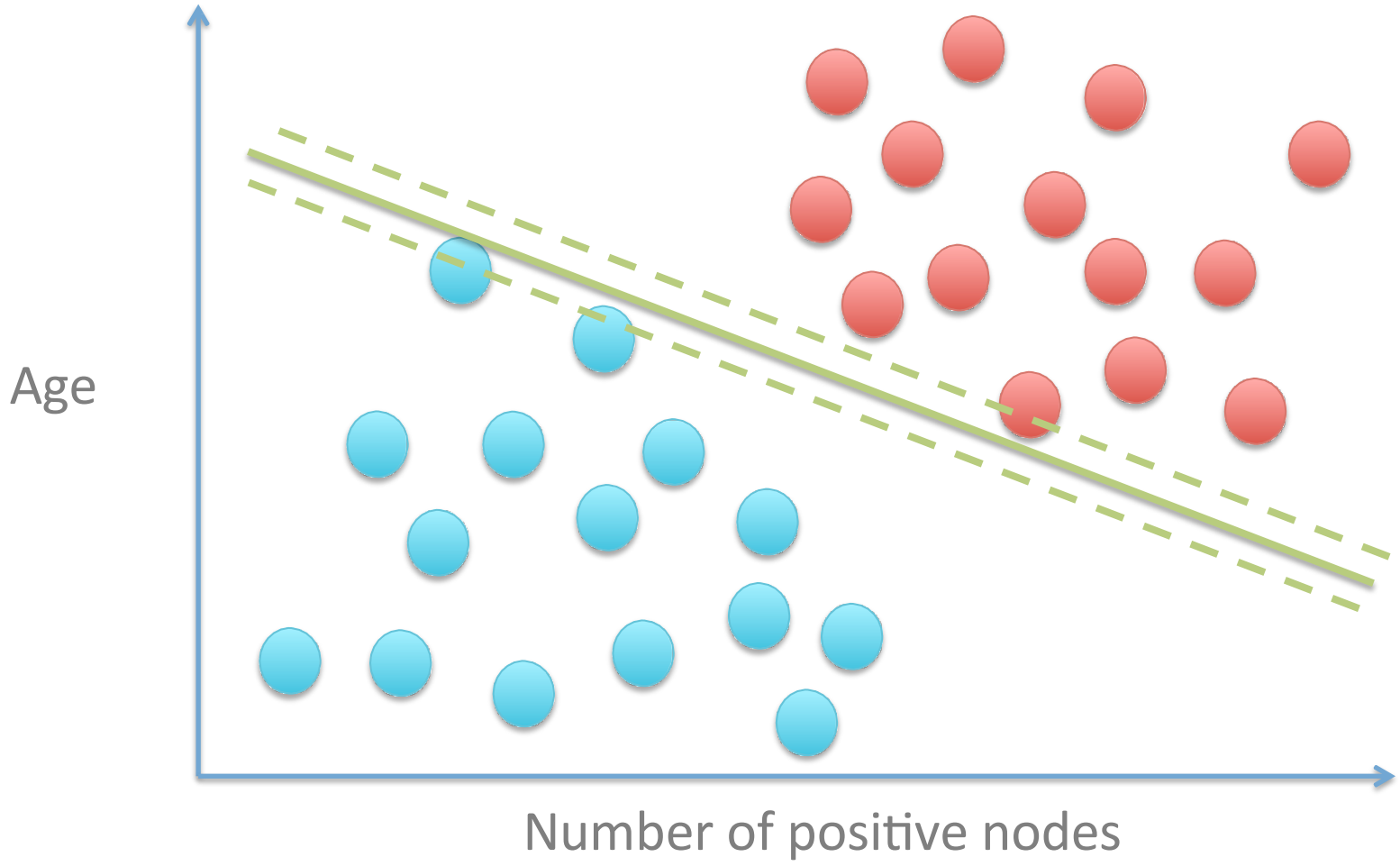

Age
Number of positive nodes


Age
Number of positive nodes

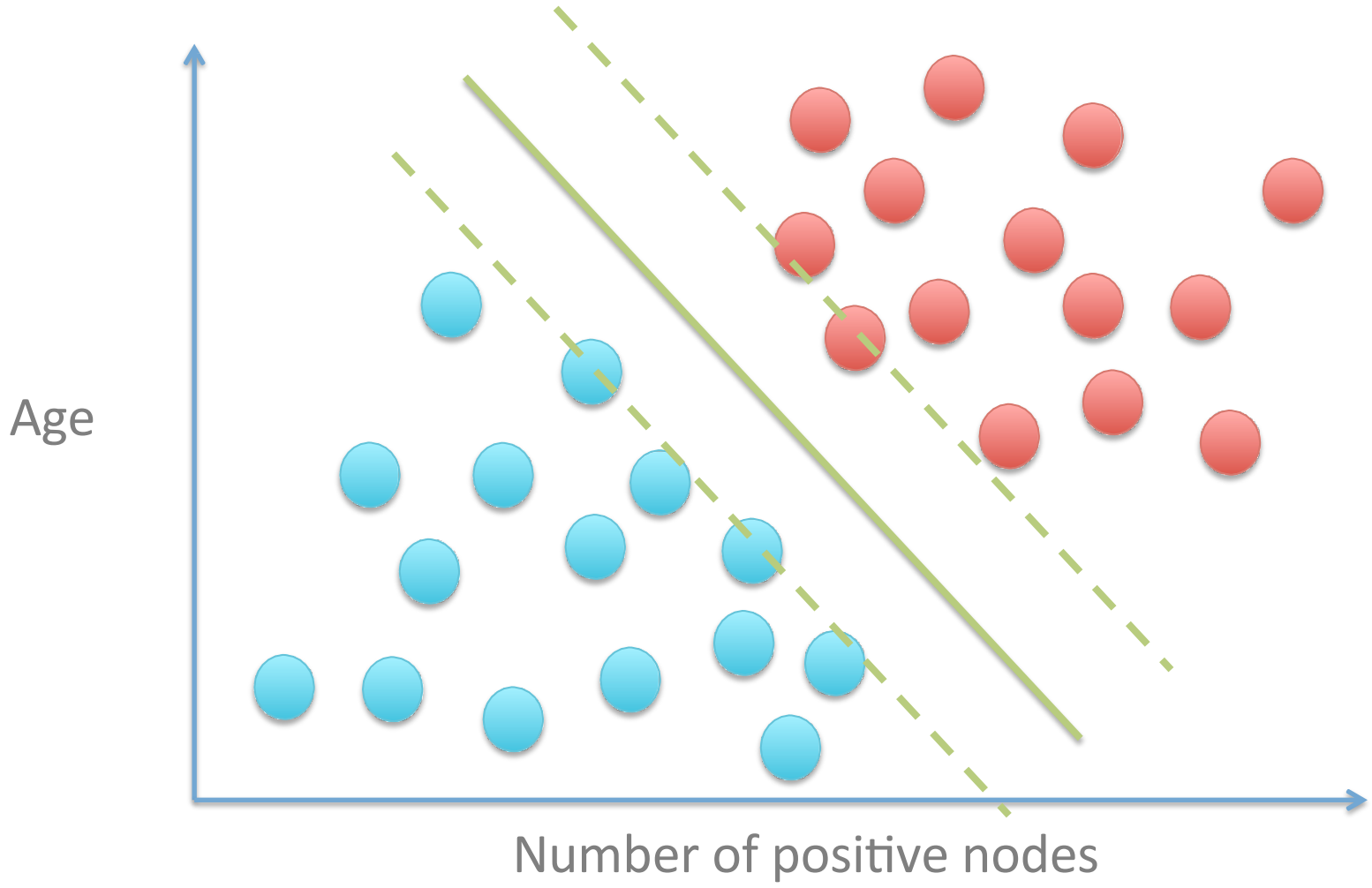

Age
Number of positive nodes


Age
Number of positive nodes


Age
Number of positive nodes

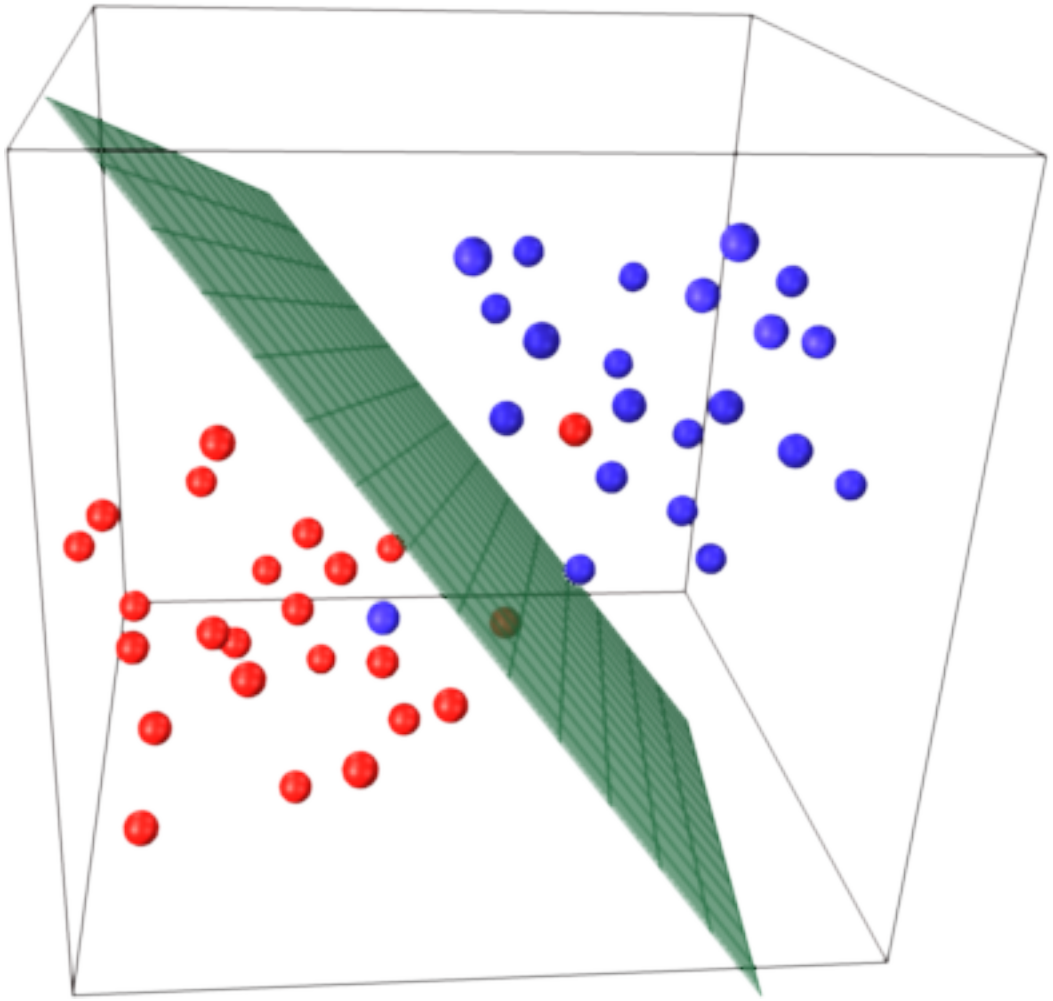

Age
Number of positive nodes

Best boundary



3 features: Find the best boundary plane

(More features: hyperplane)



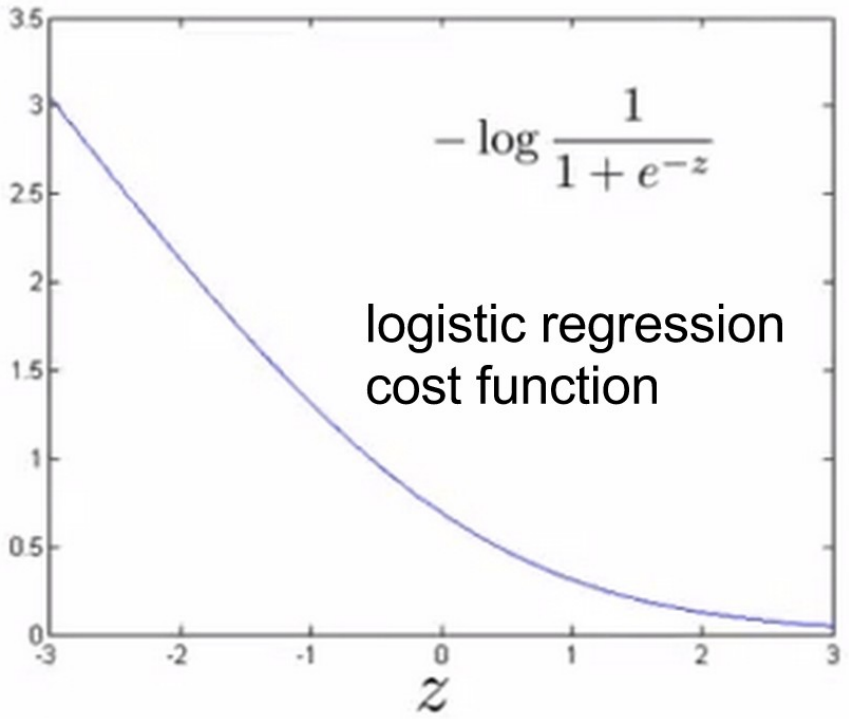
Logistic Regression Cost Function

$$y_{pred}(x) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(\beta_0+\beta_1 x+\ldots+\varepsilon)}}$$

$$Cost(y_{pred}, y_{true}) = \begin{cases} -\log(y_{pred}) & if \quad y_{true} = 1 \\ -\log(1-y_{pred}) & if \quad y_{true} = 0 \end{cases}$$

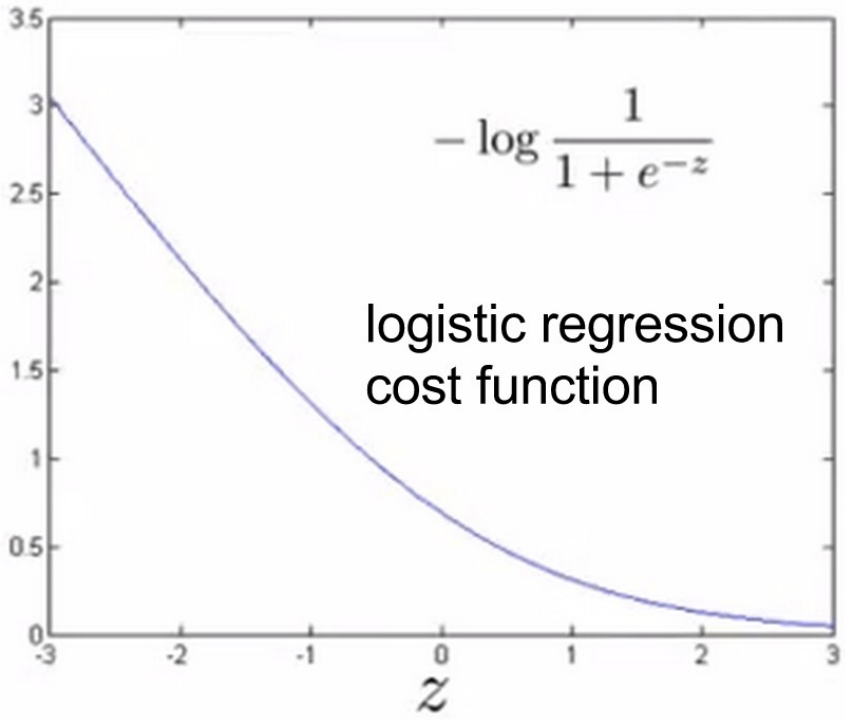
Logistic Regression Cost Function

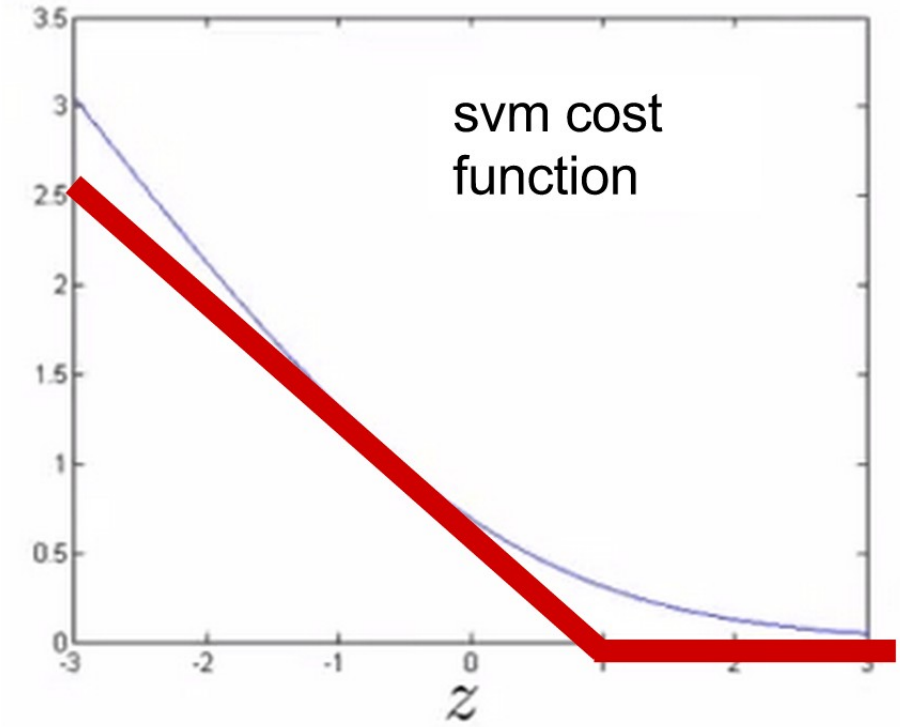
$$Cost(y_{pred}, y_{true}) = \begin{cases} -\log(y_{pred}) & if \quad y_{true} = 1 \\ -\log(1 - y_{pred}) & if \quad y_{true} = 0 \end{cases}$$



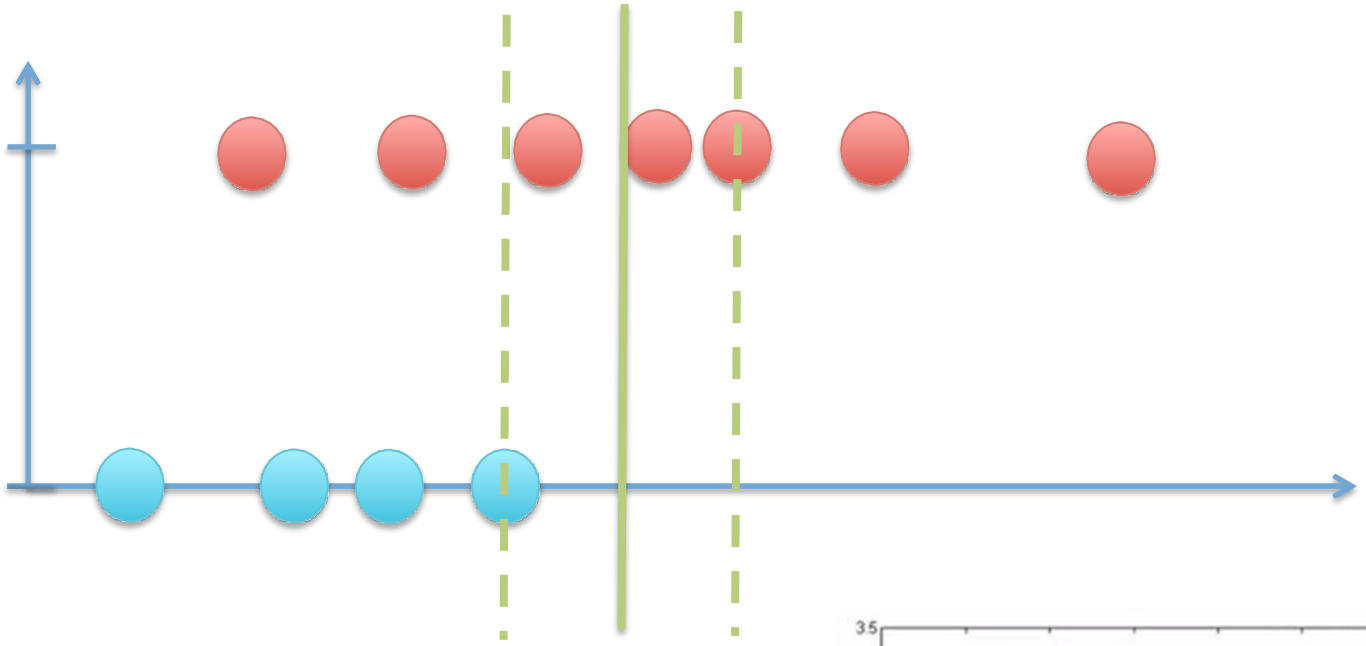
Logistic Regression Cost Function

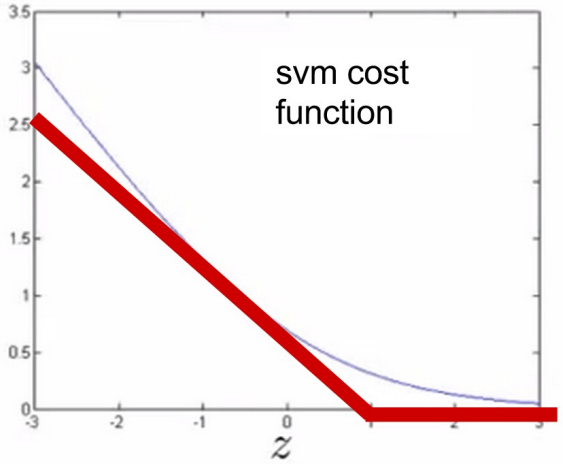
$$Cost(y_{pred}, y_{true}) = \begin{cases} -\log(y_{pred}) & if \quad y_{true} = 1 \\ -\log(1 - y_{pred}) & if \quad y_{true} = 0 \end{cases}$$



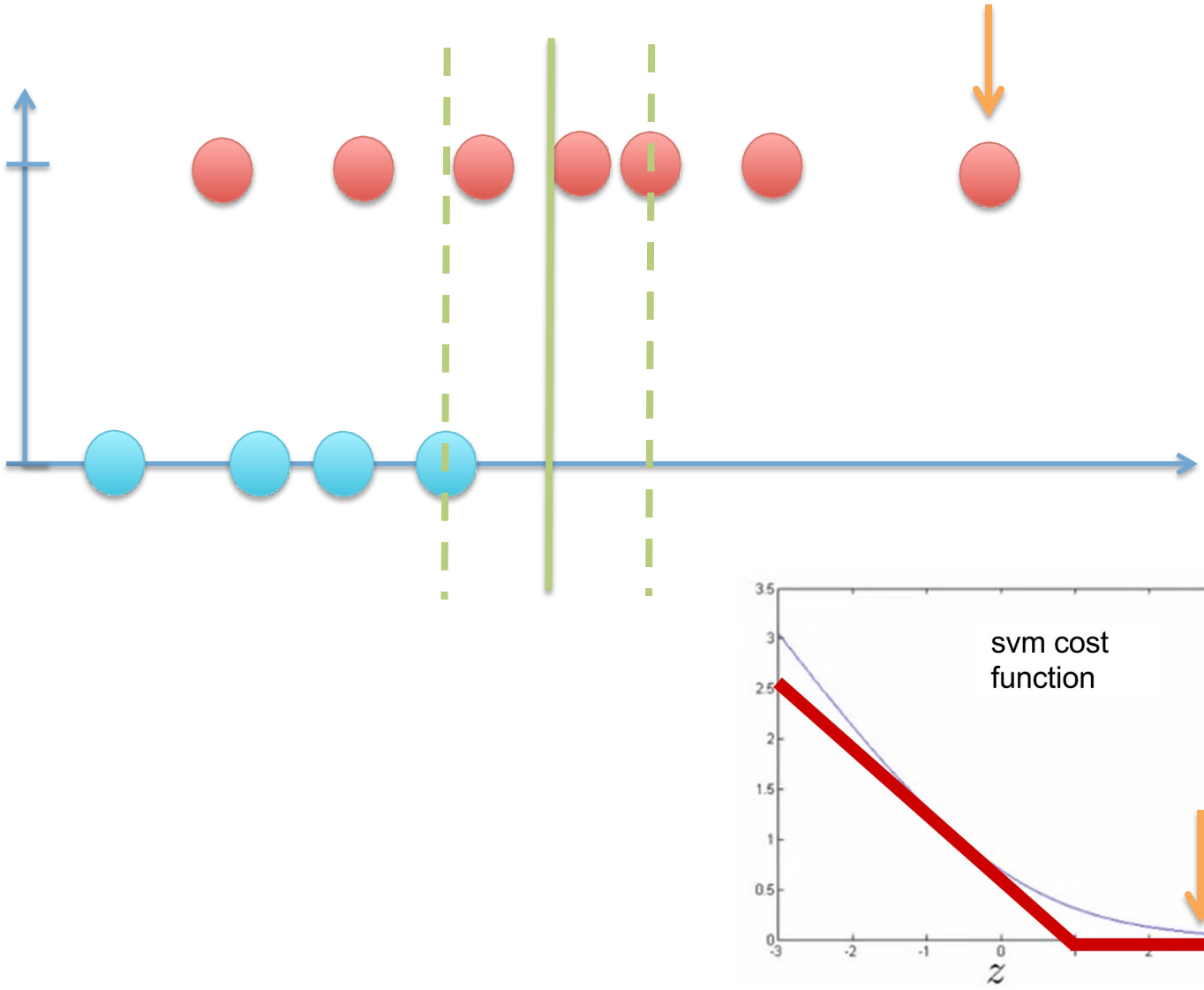


SVM Cost Function

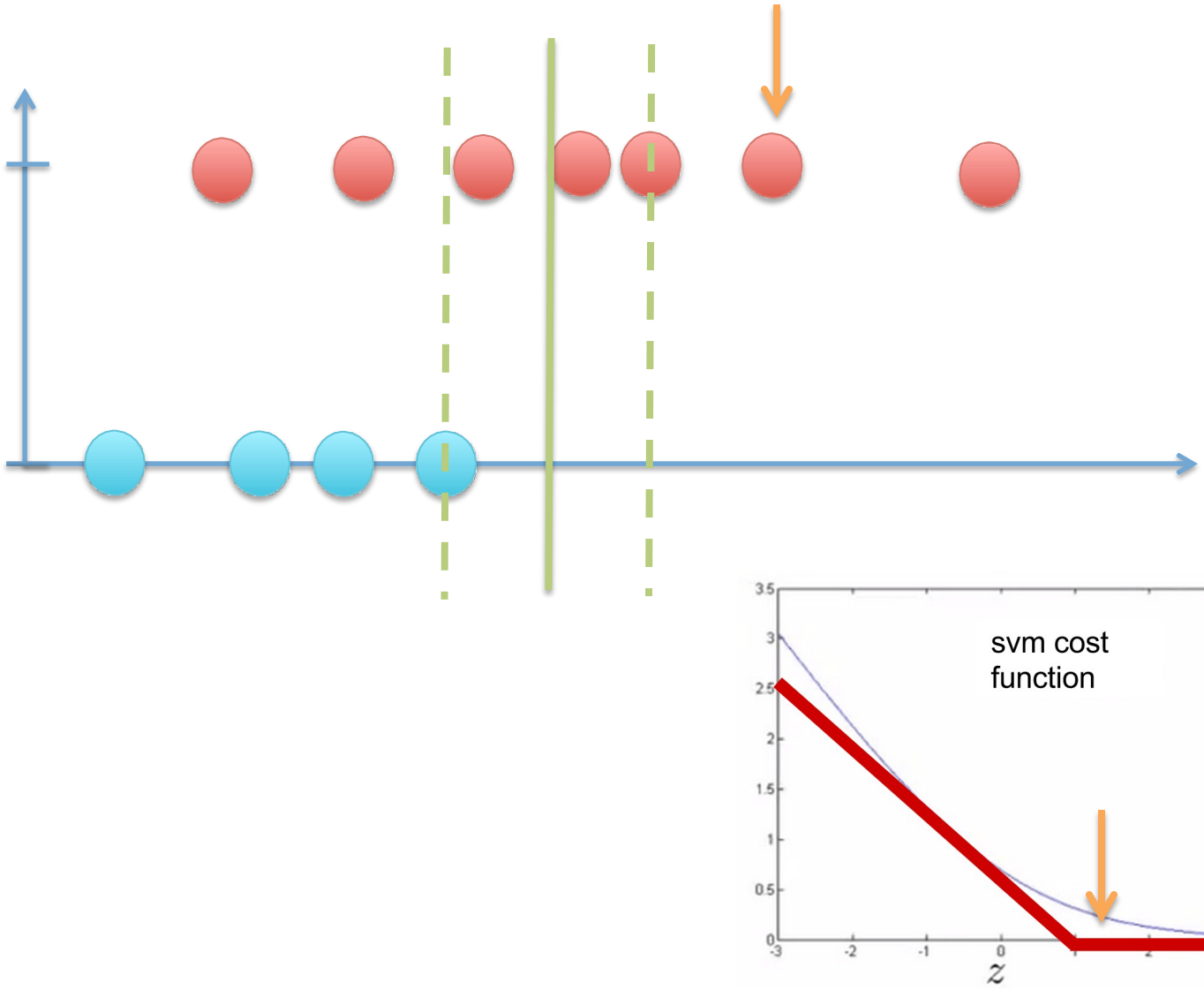




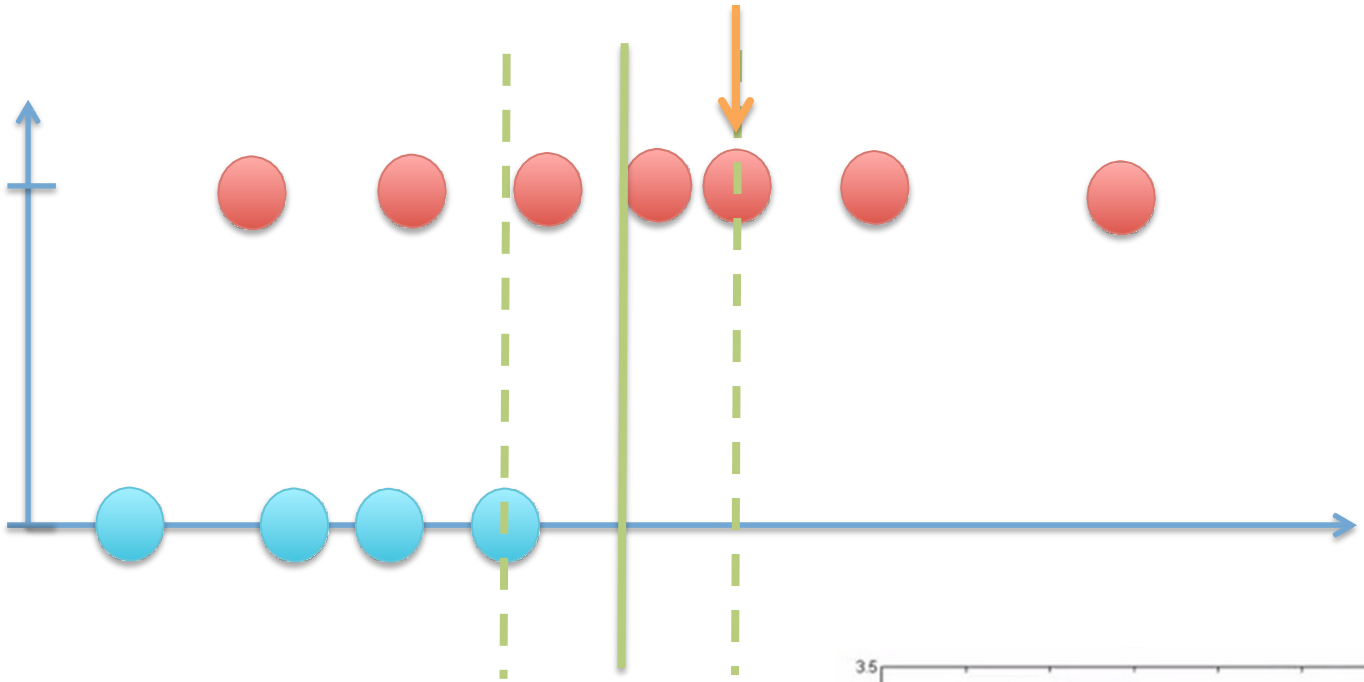
SVM Cost Function

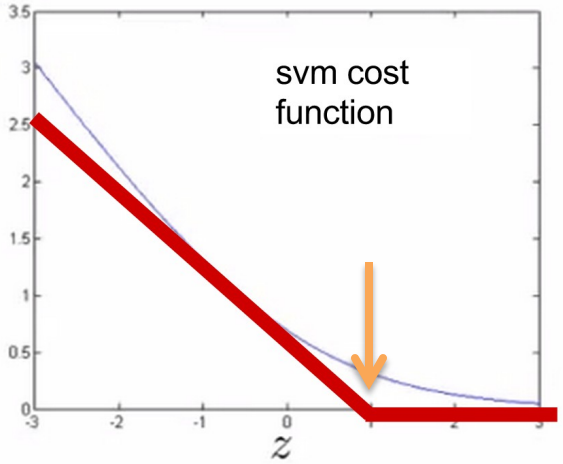


SVM Cost Function

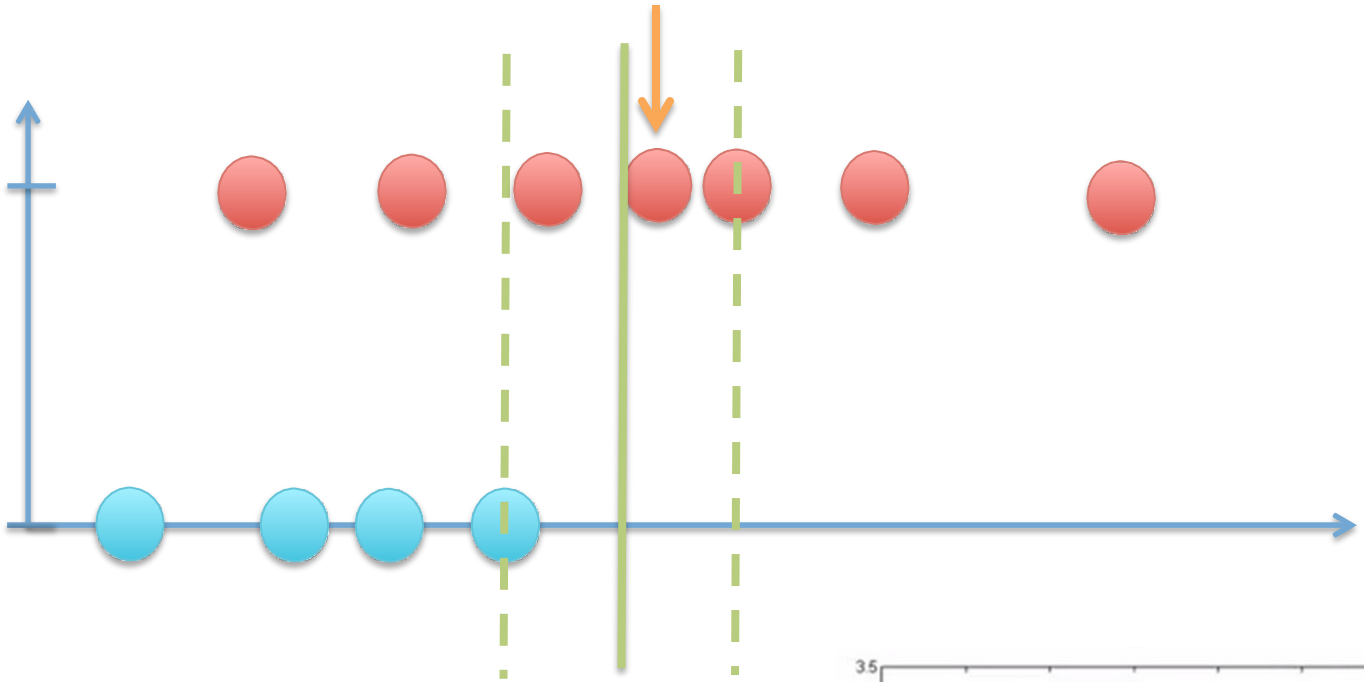


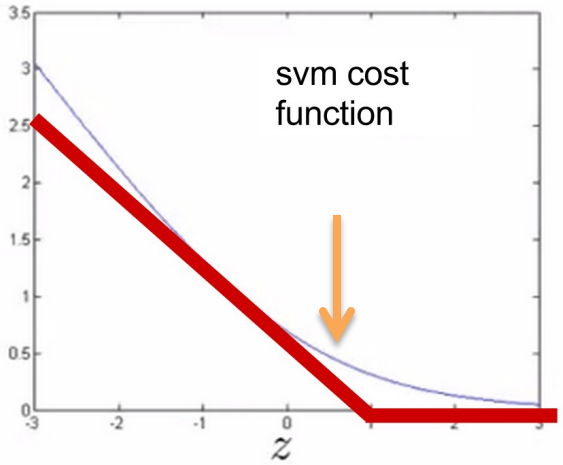
SVM Cost Function



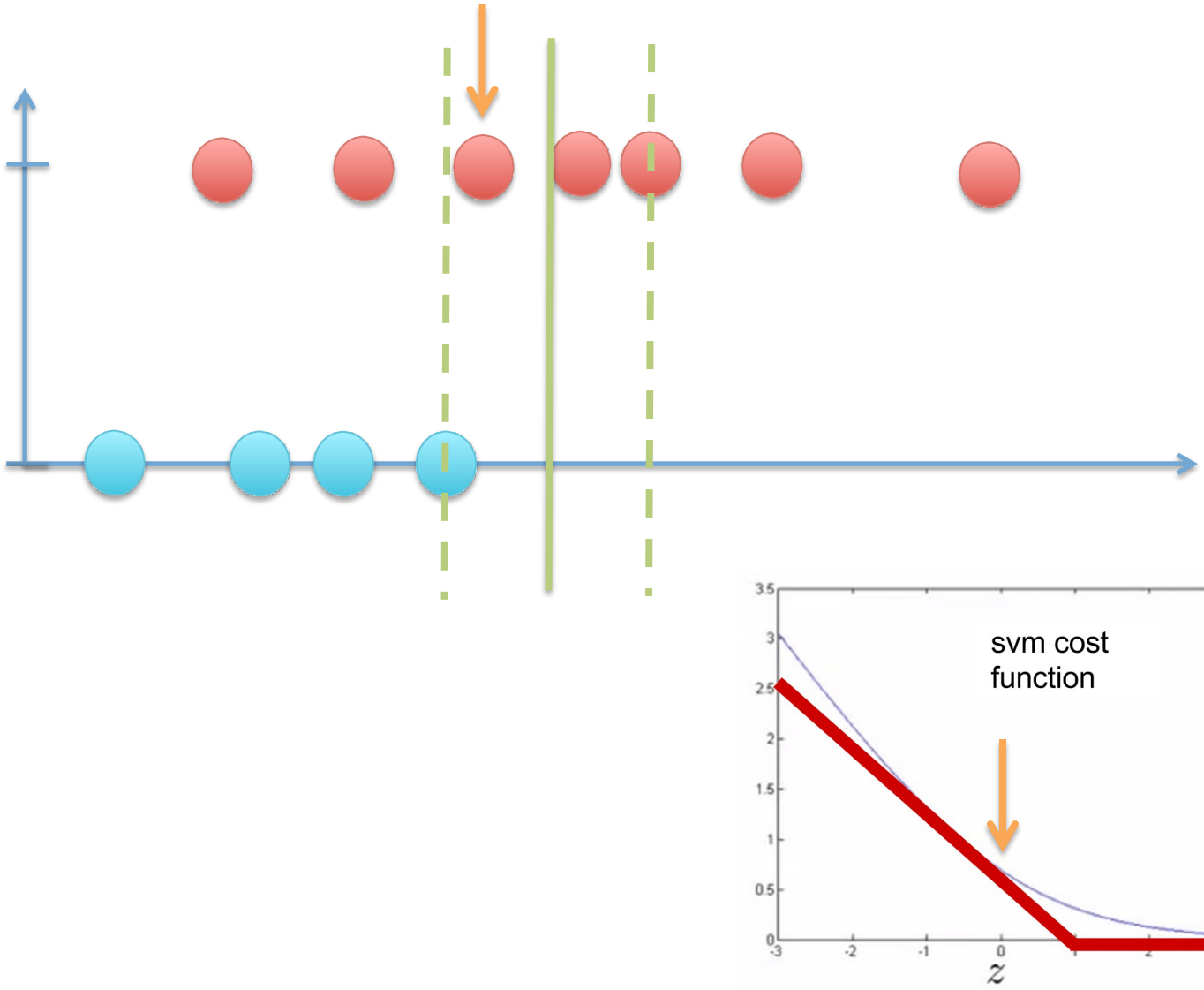


SVM Cost Function

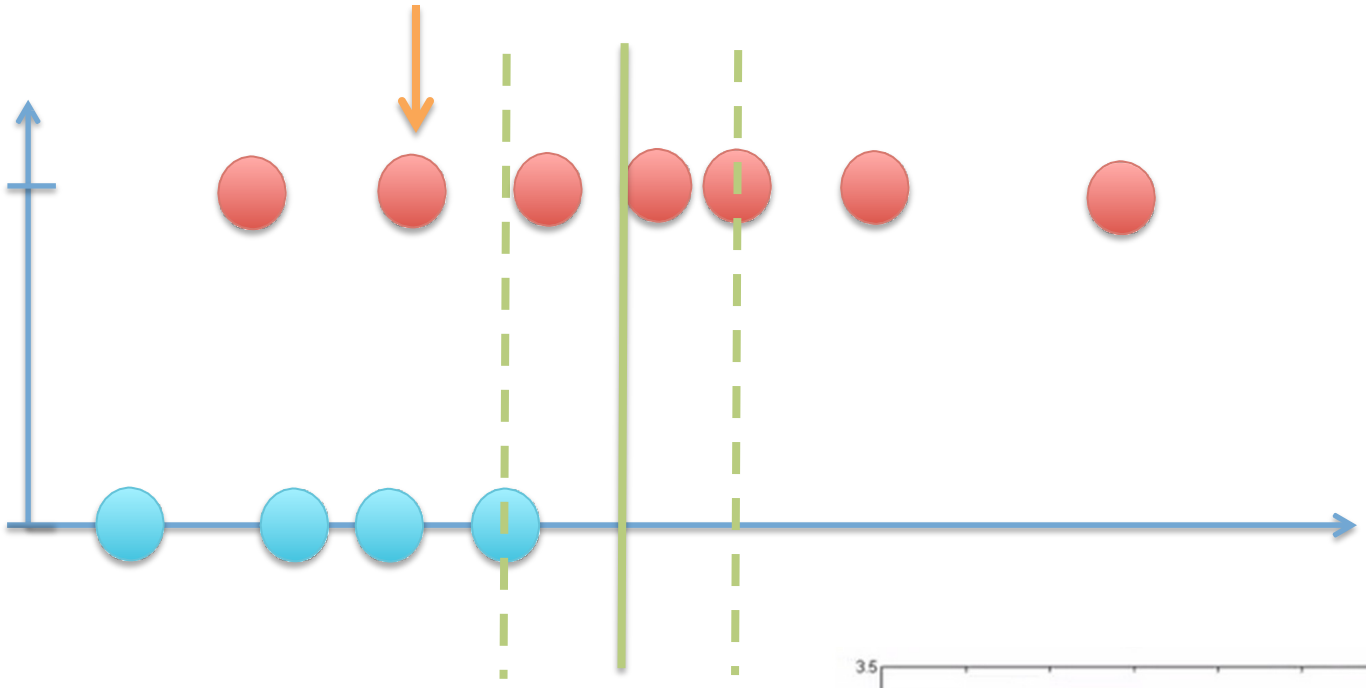


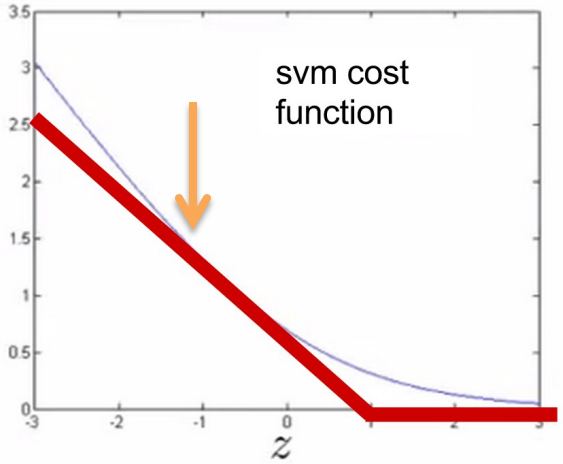


SVM Cost Function

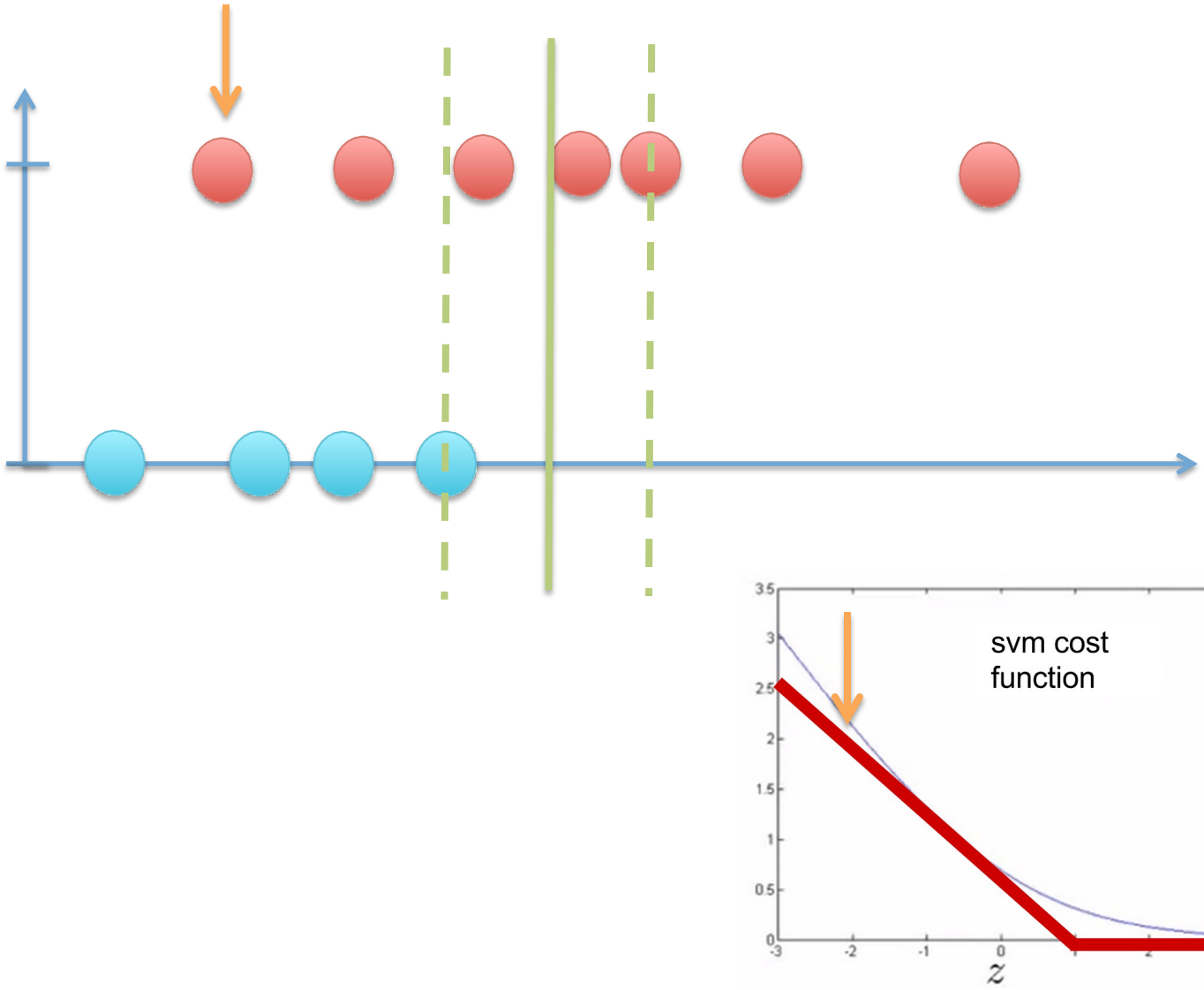


SVM Cost Function





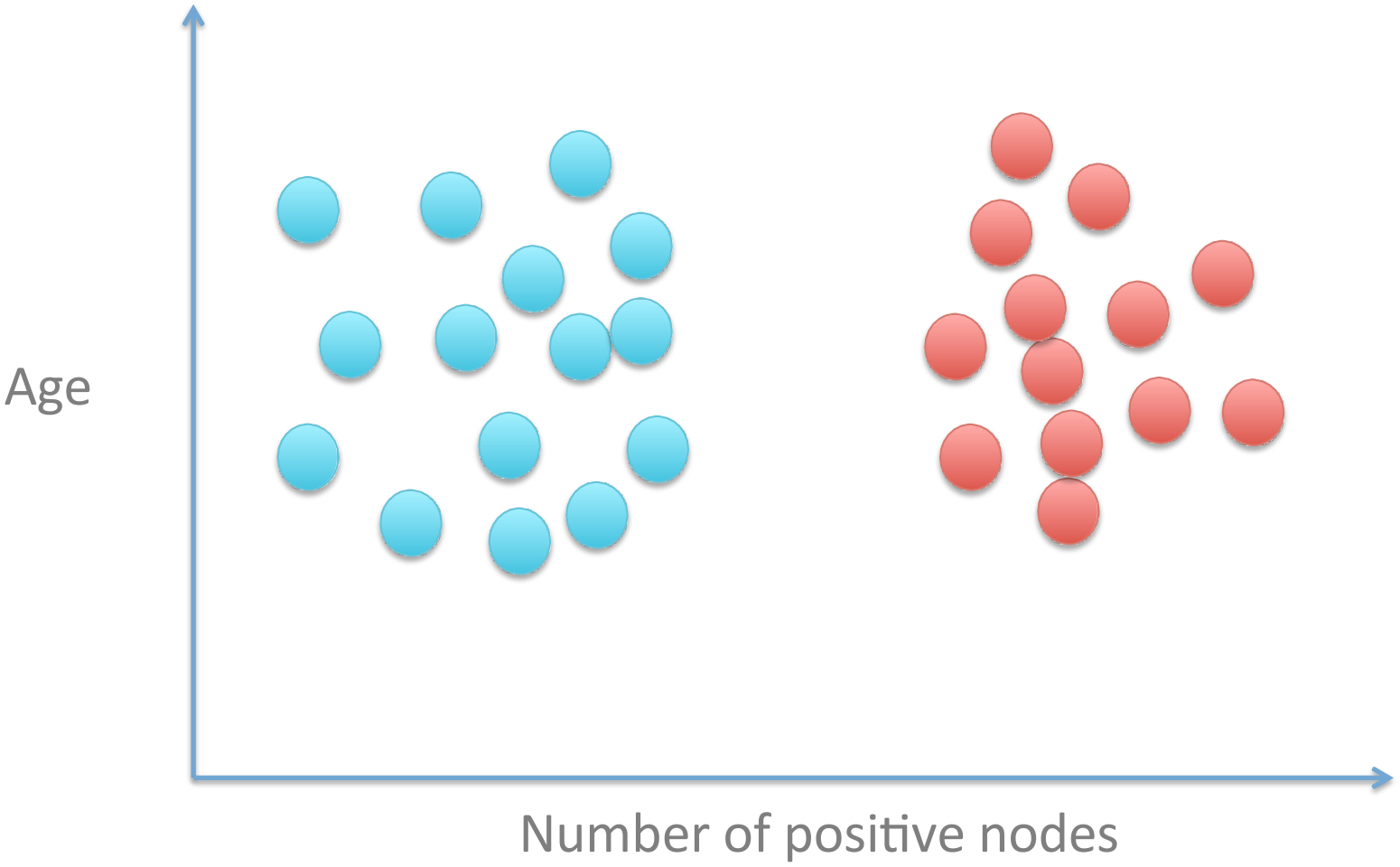
SVM Cost Function



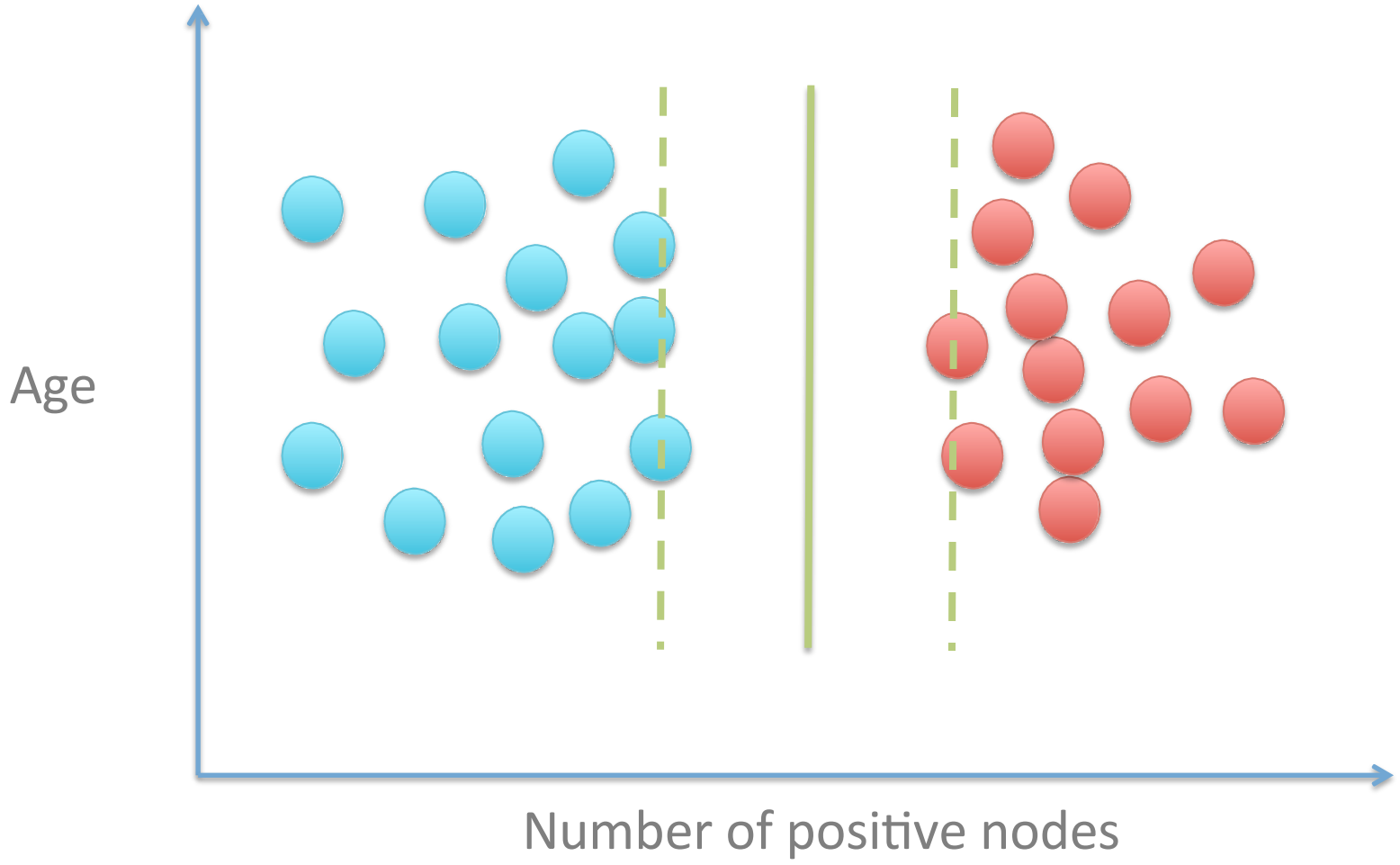
```
from sklearn.svm import SVC

from sklearn.svm import LinearSVC
```

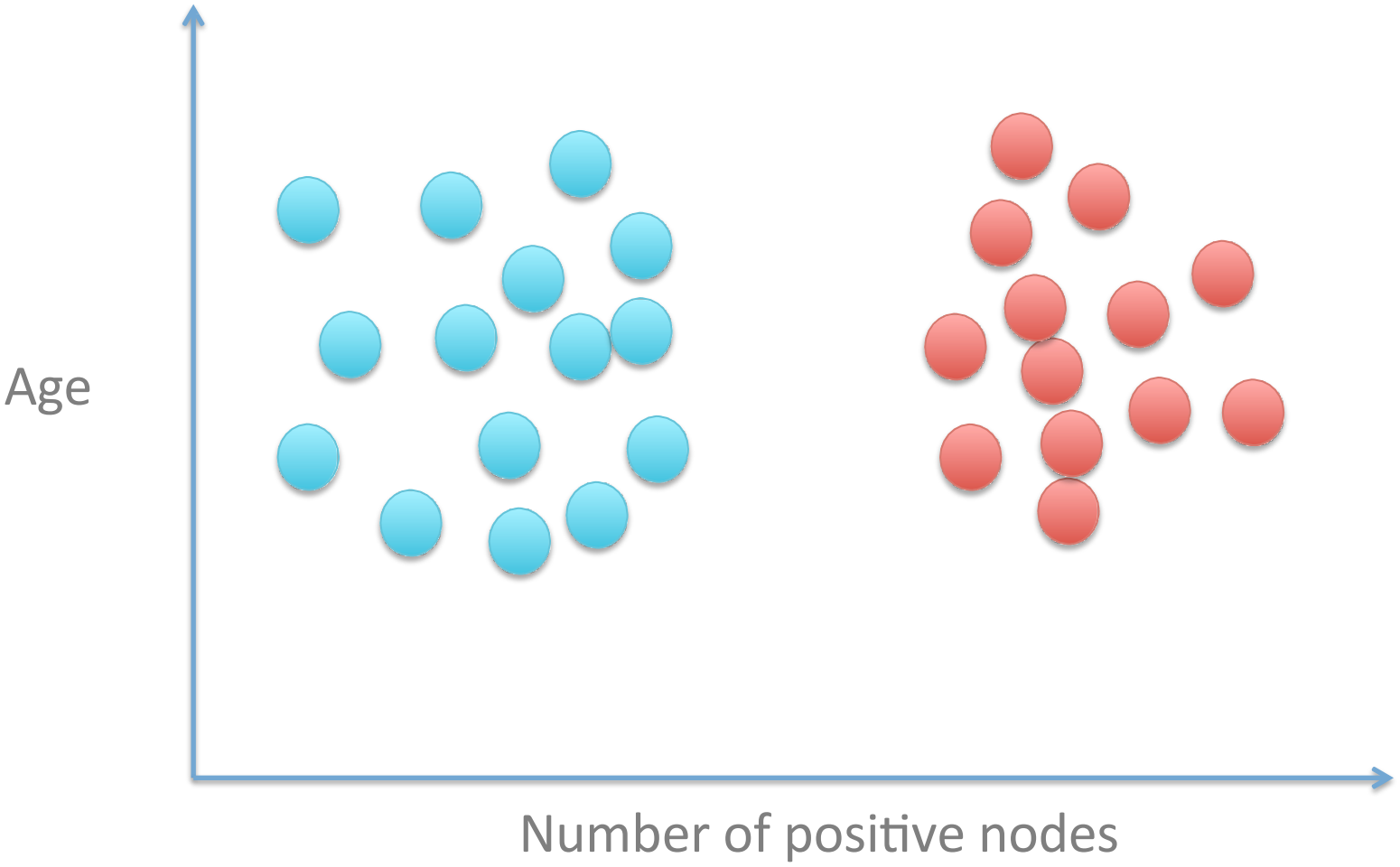
Outlier problems



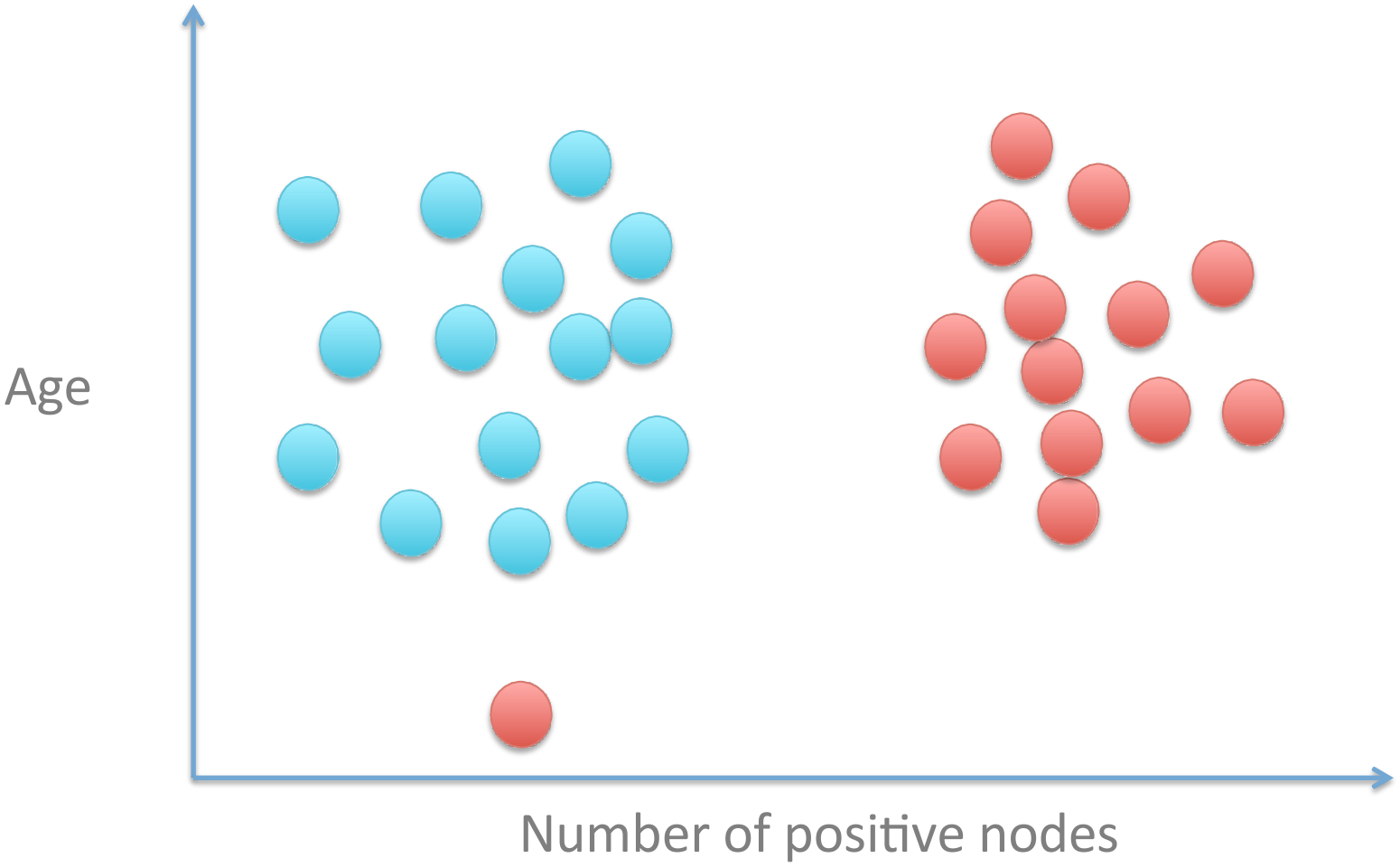
Outlier problems



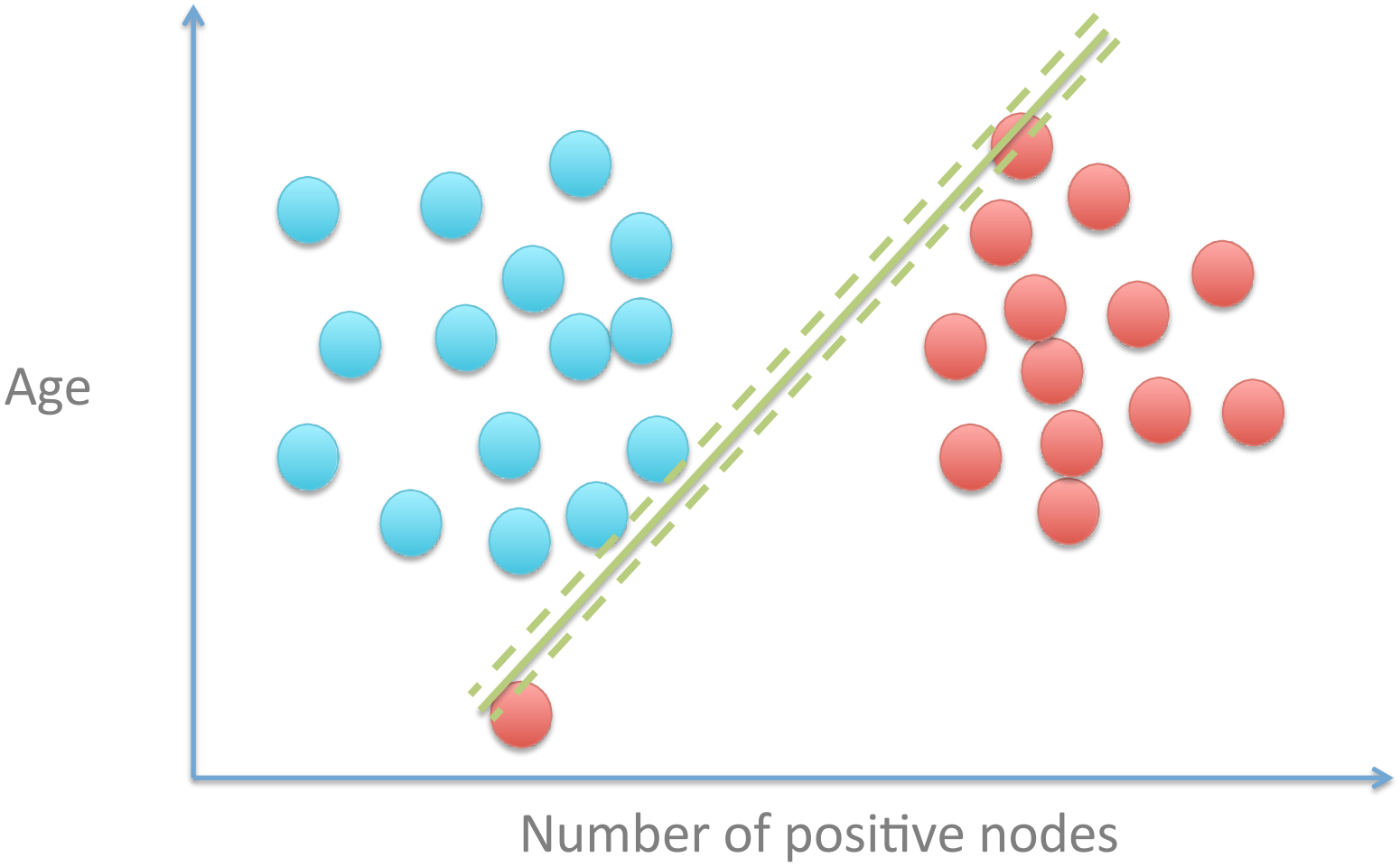
Outlier problems



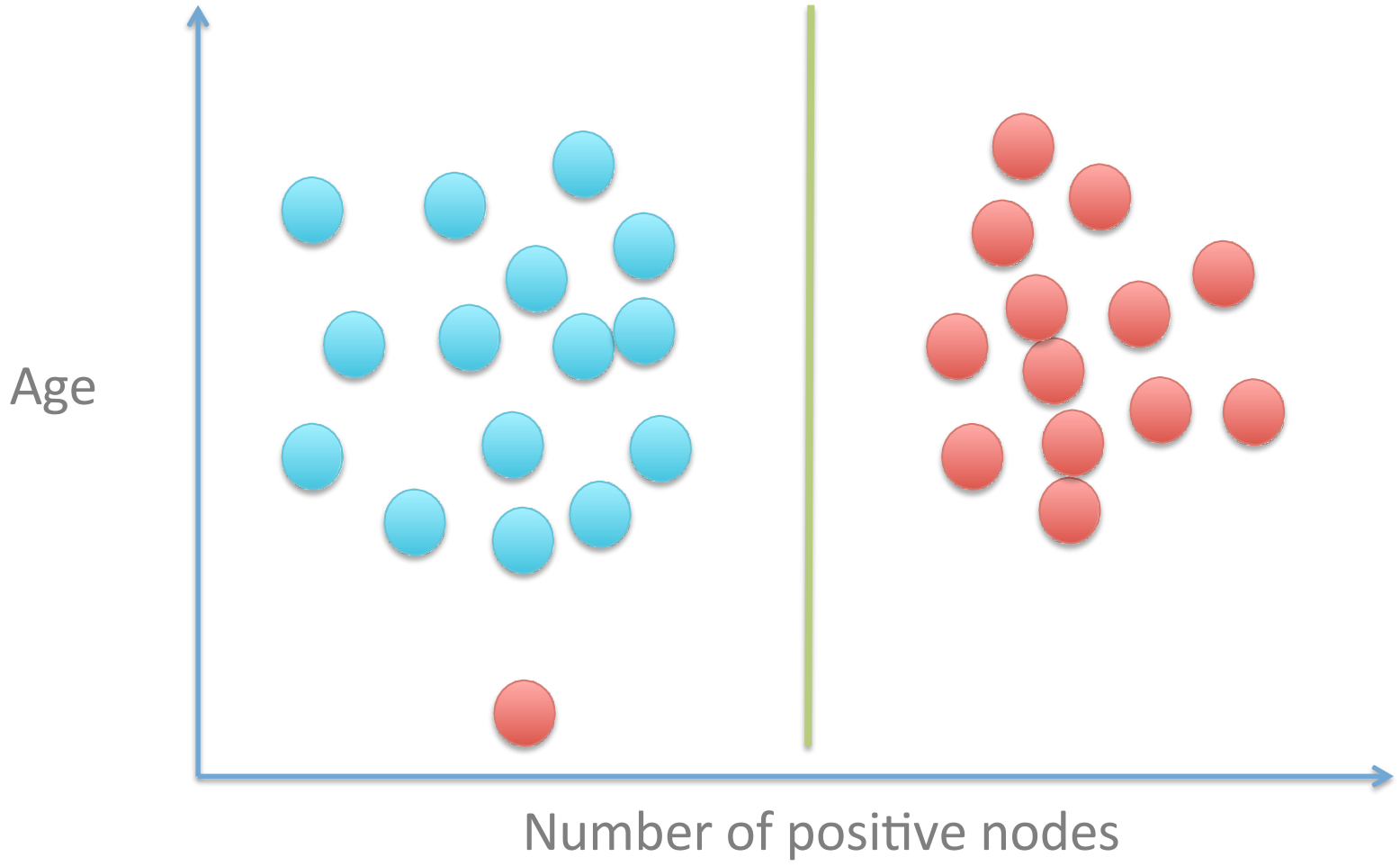
Outlier problems



Outlier problems

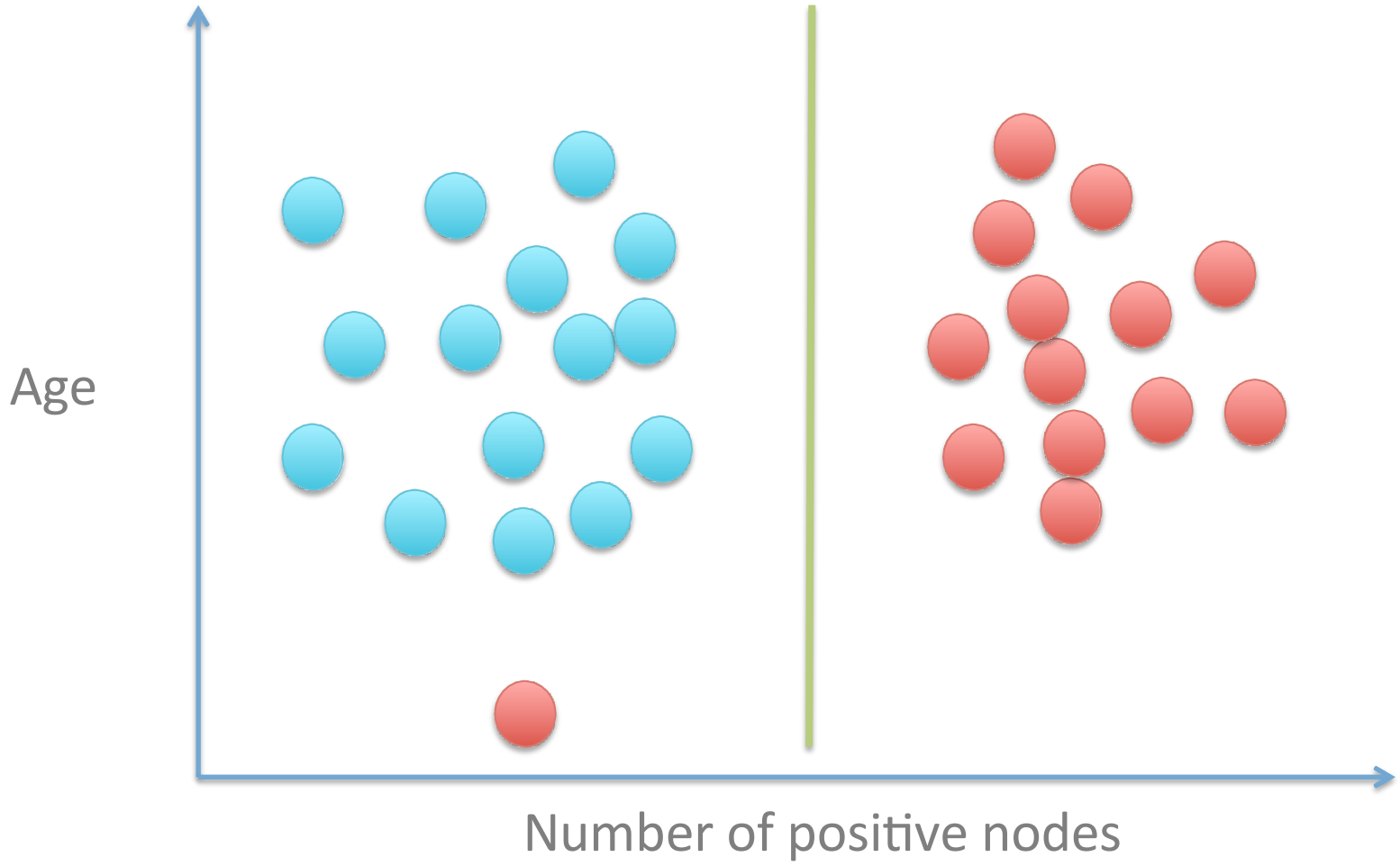


This is probably still the best boundary



Regularization in the Cost Function handles this

$$Cost(y_{pred}, y_{true}) = SVMCost(\beta_i) + \frac{1}{C}\sum_i \beta_i^2$$



Regularization in the Cost Function handles this

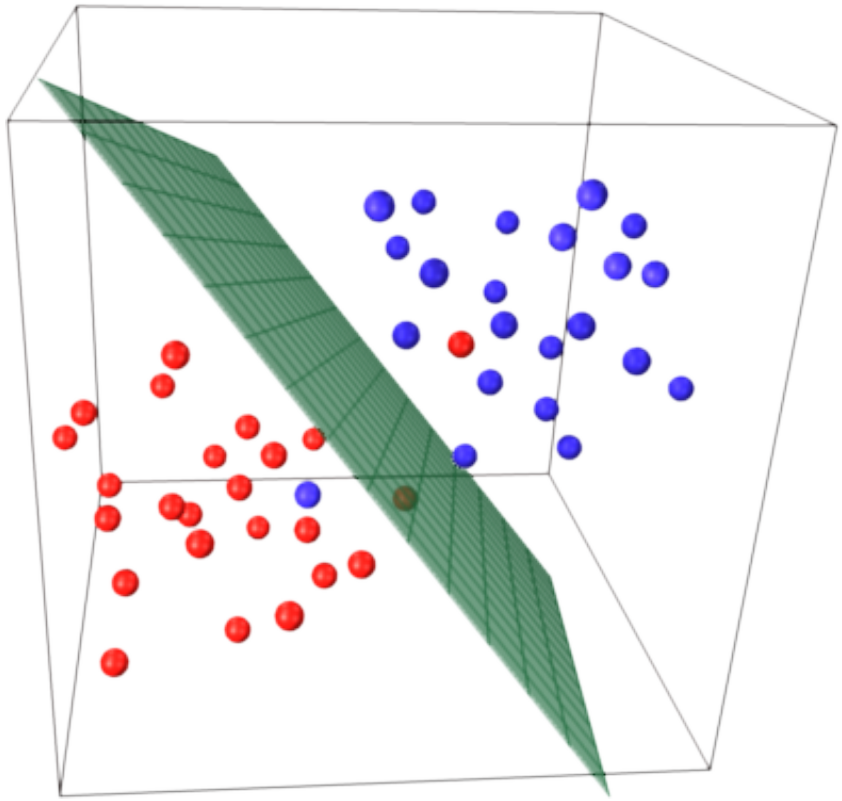
$$Cost(y_{pred}, y_{true}) = SVMCost(\beta_i) + \frac{1}{C}\sum_i \beta_i^2$$

But first, what do coefficients mean for SVMs?

Regularization in the Cost Function handles this

$$Cost(y_{pred}, y_{true}) = SVMCost(\beta_i) + \frac{1}{C}\sum_i \beta_i^2$$

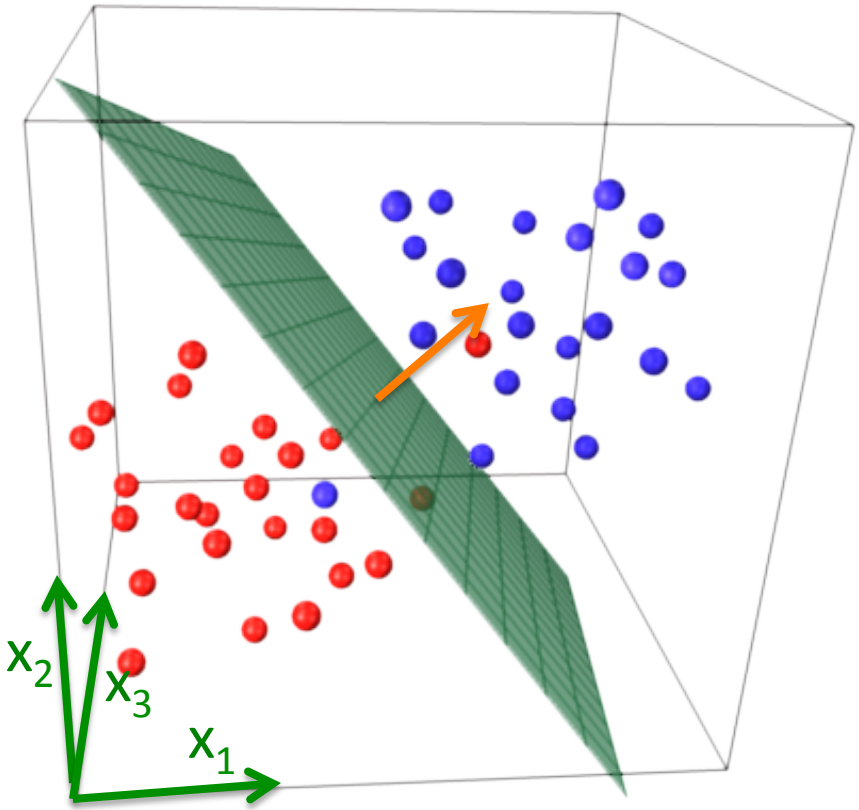
But first, what do coefficients mean for SVMs?



Regularization in the Cost Function handles this

$$Cost(y_{pred}, y_{true}) = SVMCost(\beta_i) + \frac{1}{C}\sum_i \beta_i^2$$

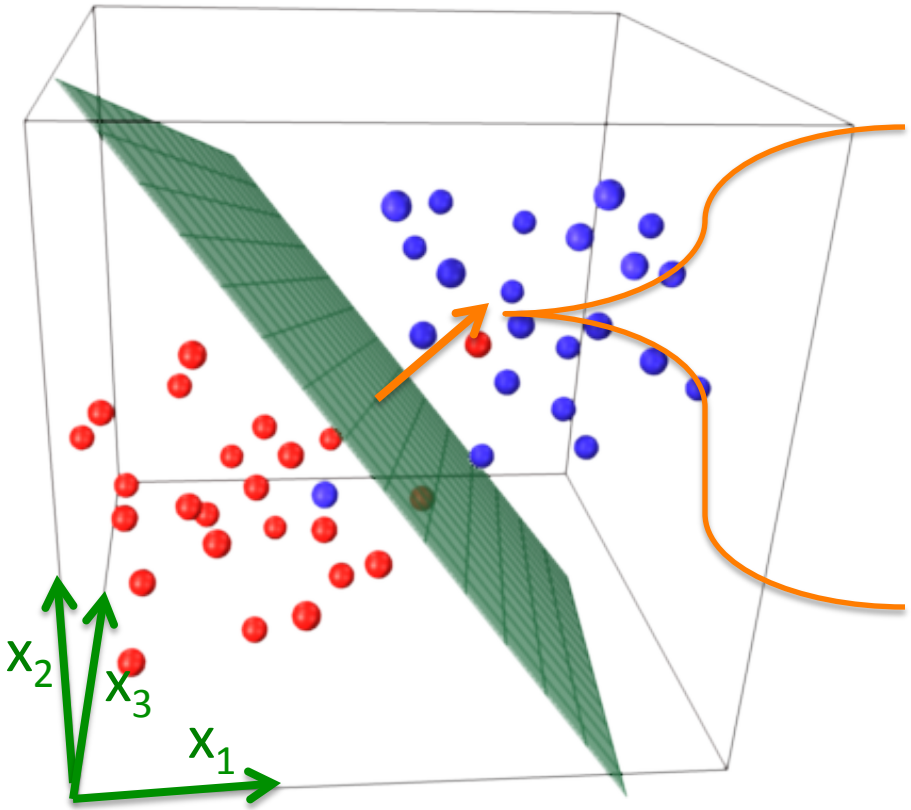
But first, what do coefficients mean for SVMs?



Regularization in the Cost Function handles this

$$Cost(y_{pred}, y_{true}) = SVMCost(\beta_i) + \frac{1}{C}\sum_i \beta_i^2$$

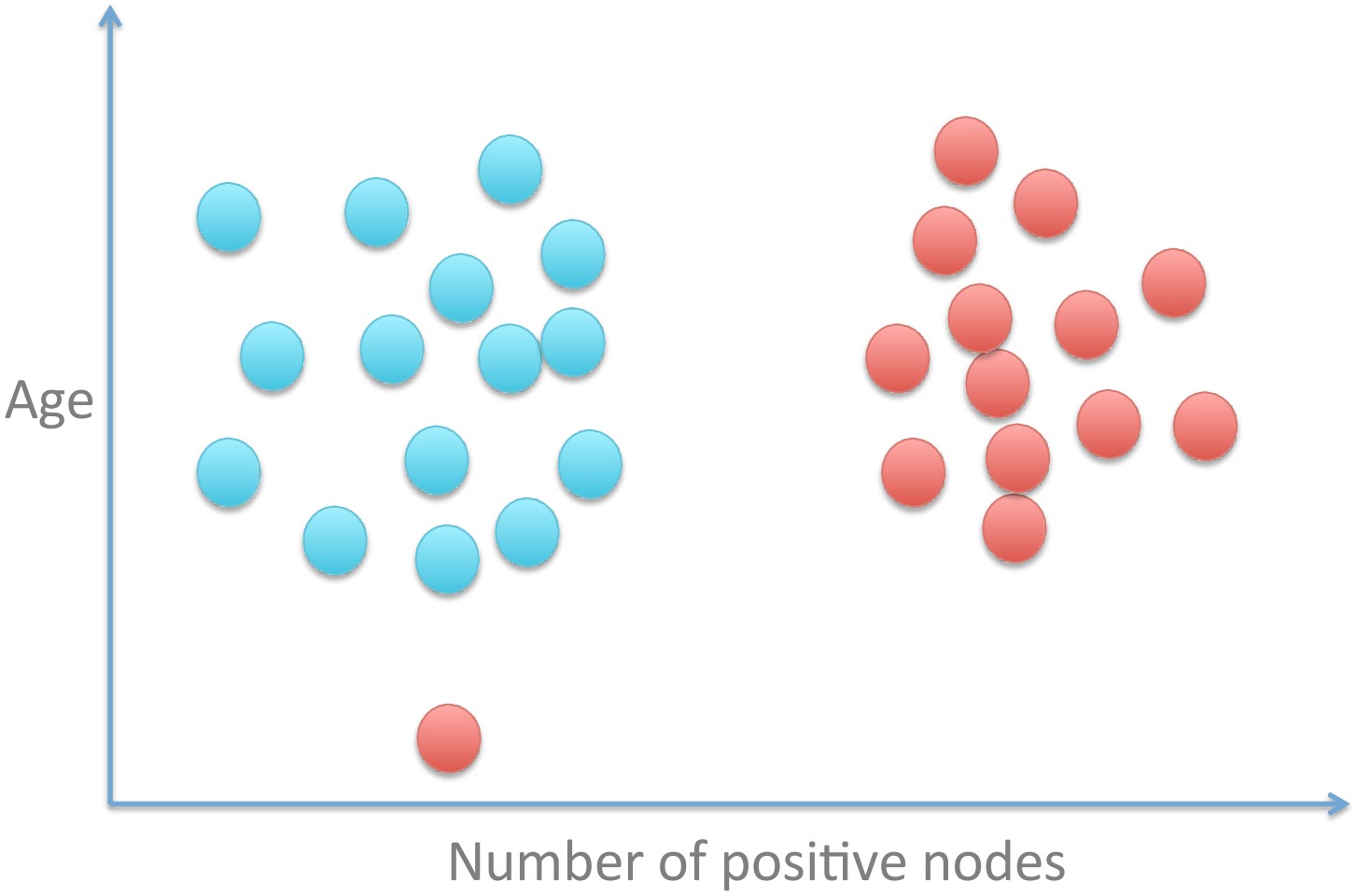
But first, what do coefficients mean for SVMs?



$$\begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix}$$

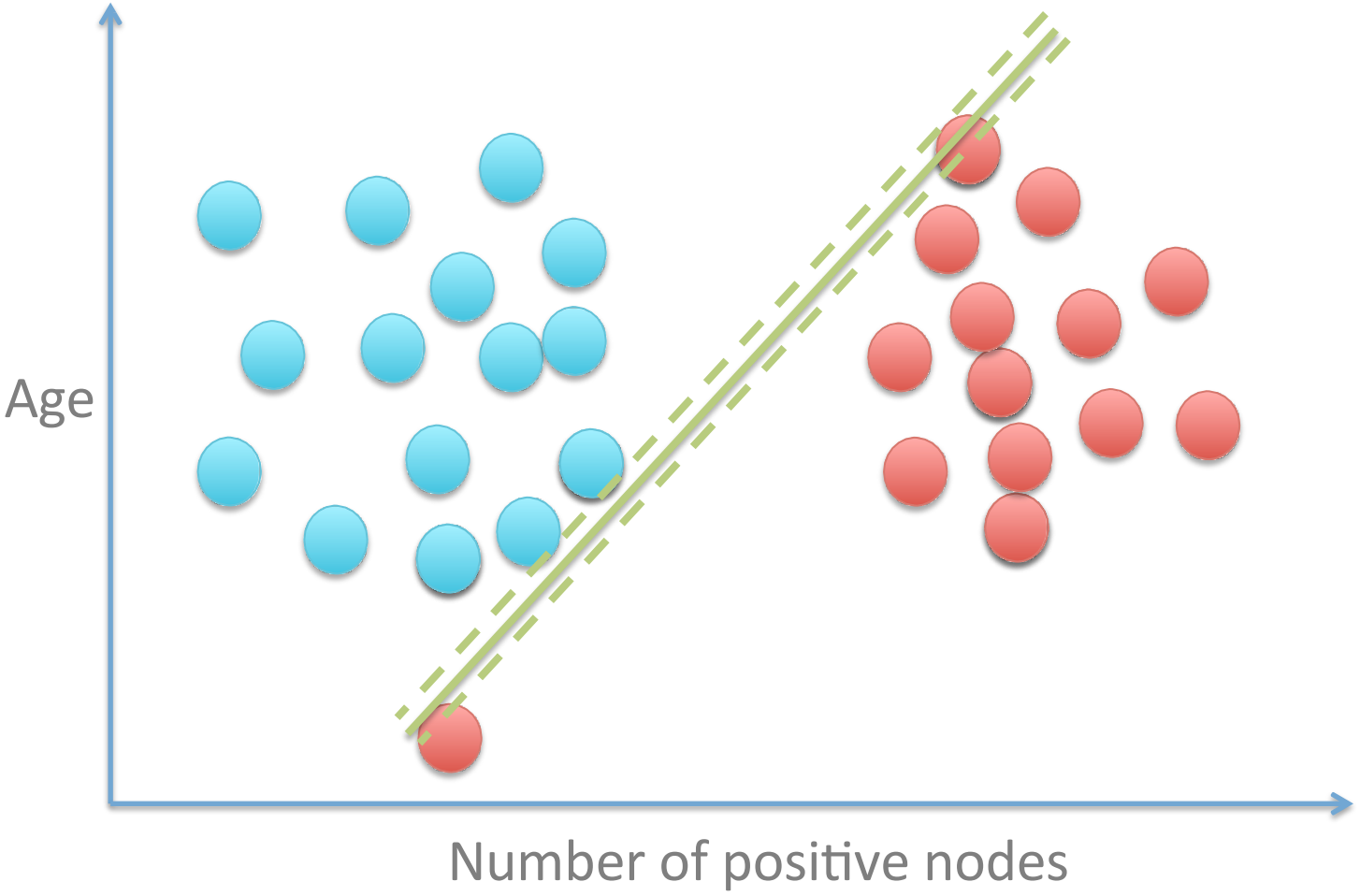
Vector orthogonal to the separating hyperplane

$$Cost(y_{pred}, y_{true}) = SVMCost(\beta_i) + \frac{1}{C}\sum_i \beta_i^2$$



$$Cost(y_{pred}, y_{true}) = SVMCost(\beta_i) + \frac{1}{C}\sum_i \beta_i^2$$

Model 1

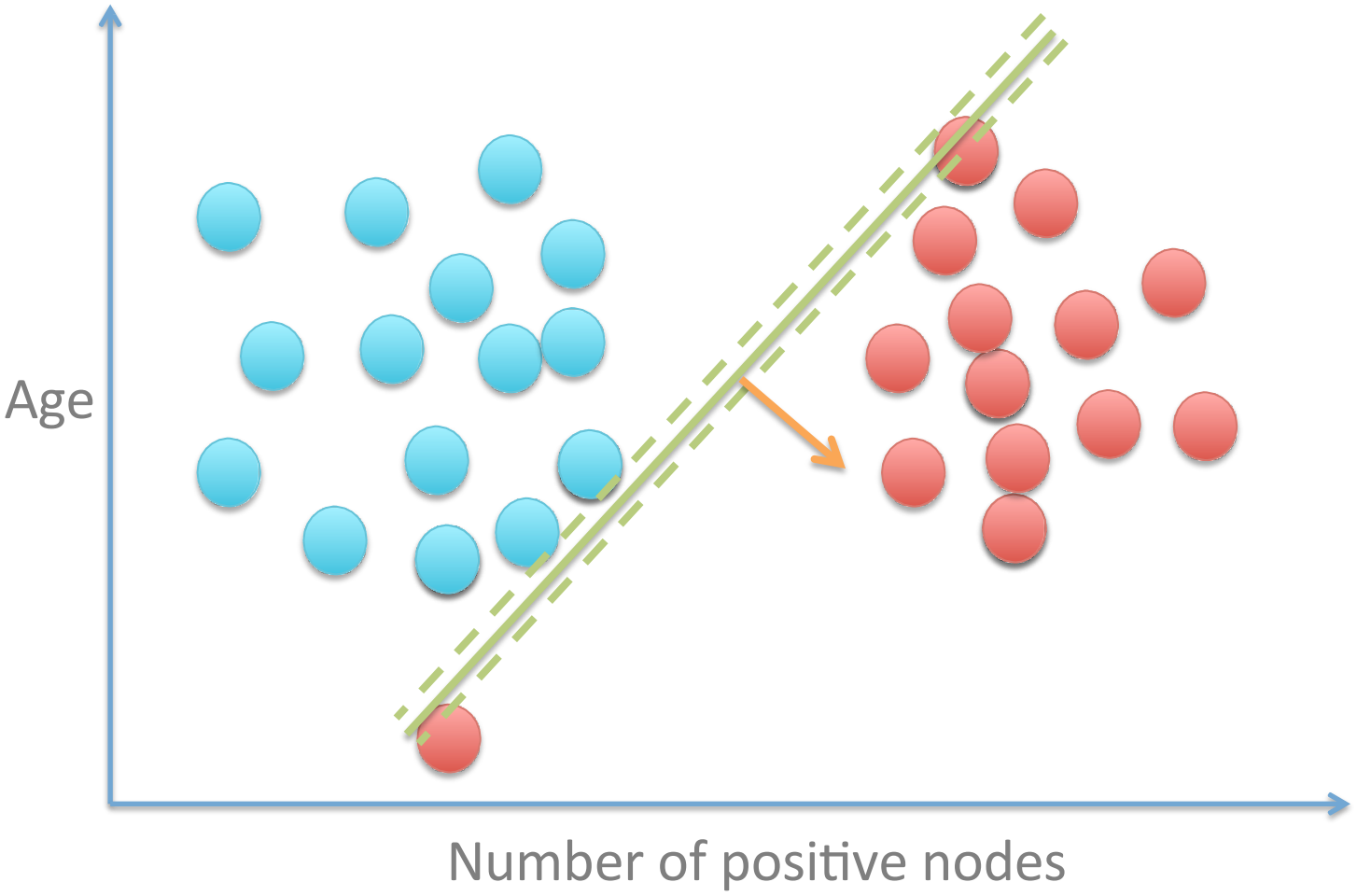


$$Cost(y_{pred}, y_{true}) = SVMCost(\beta_i) + \frac{1}{C}\sum_i \beta_i^2$$

Model 1

vector:
(1,-1.5)

$\beta_{nodes} = 1$
$\beta_{age} = -1.5$



$$Cost(y_{pred}, y_{true}) = SVMCost(\beta_i) + \frac{1}{C}\sum_i \beta_i^2$$

optimal (best fit)

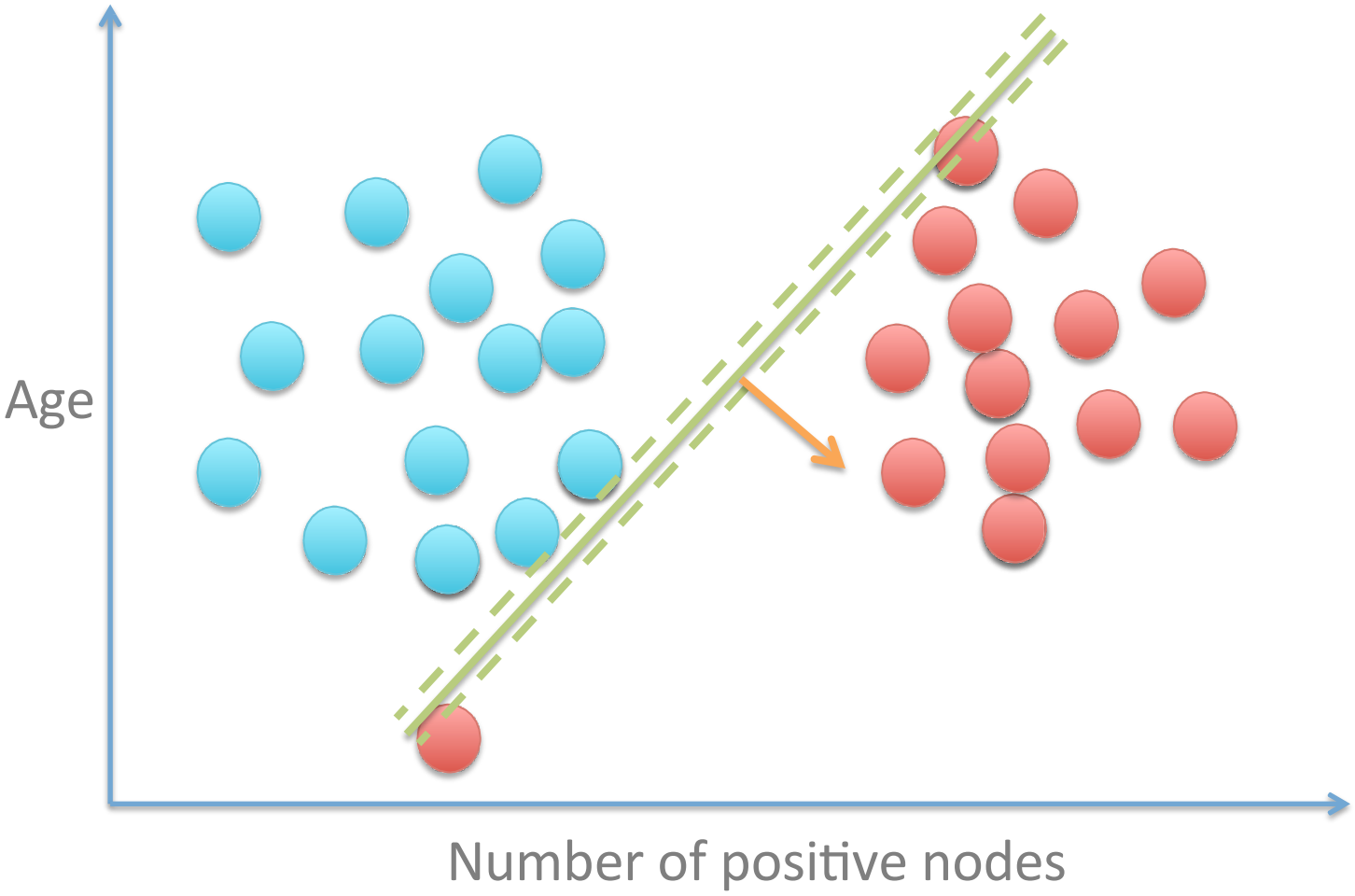
high (large beta vector)

Model 1

vector: (1,-1.5)

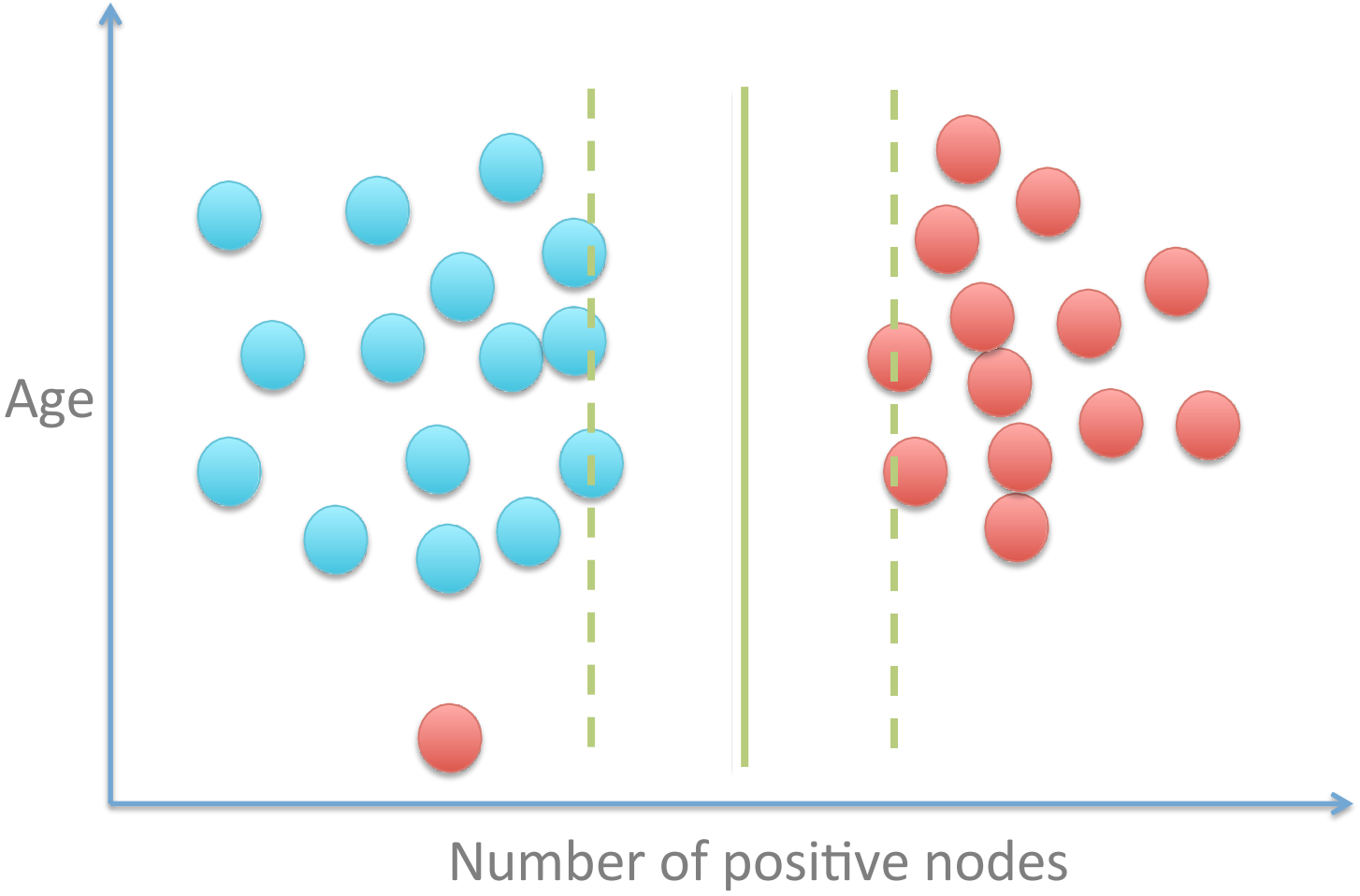
$\beta_{nodes} = 1$

$\beta_{age} = -1.5$



$$Cost(y_{pred}, y_{true}) = SVMCost(\beta_i) + \frac{1}{C}\sum_i \beta_i^2$$

Model 2

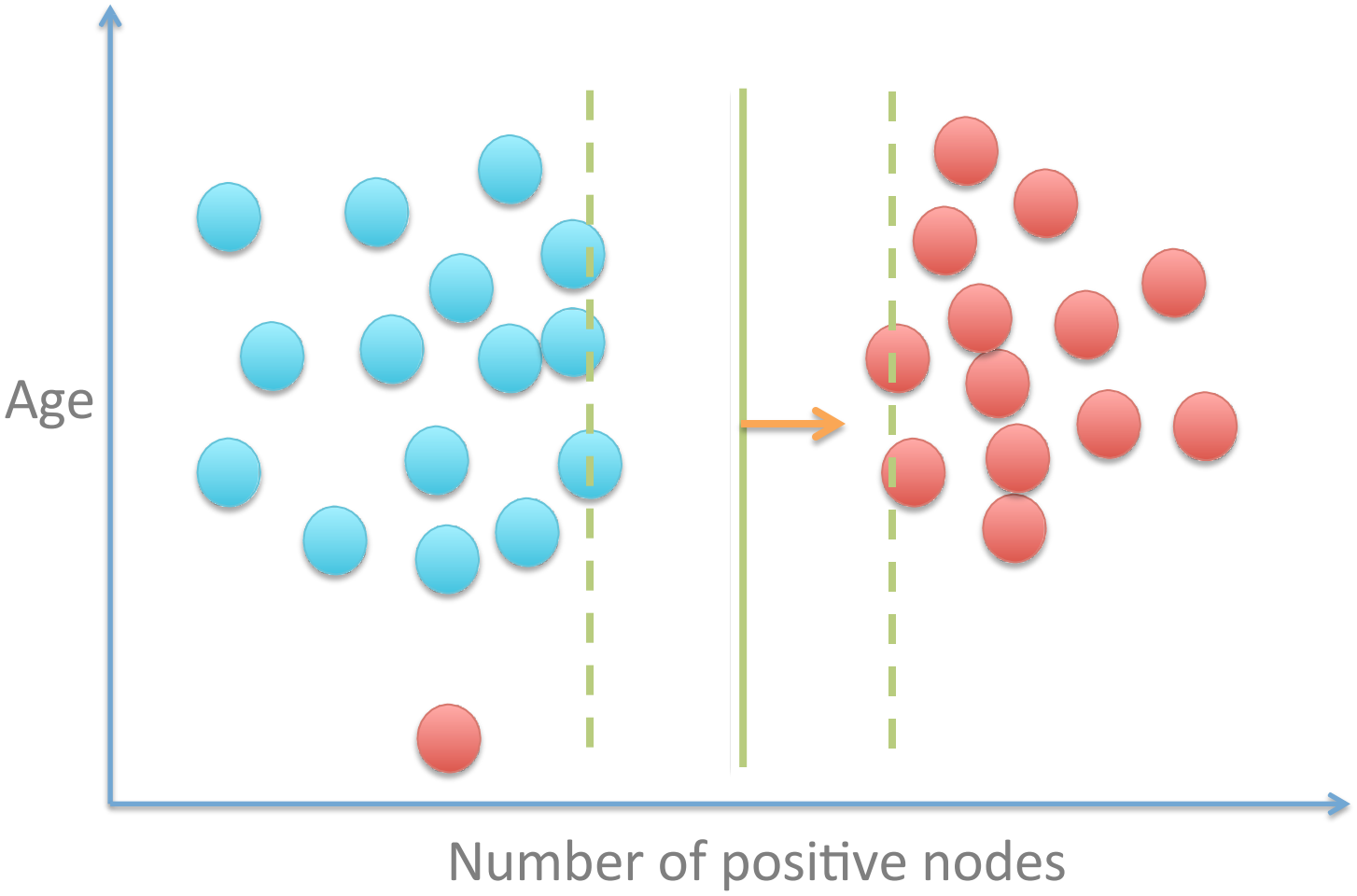


$$Cost(y_{pred}, y_{true}) = SVMCost(\beta_i) + \frac{1}{C}\sum_i \beta_i^2$$

Model 2

vector:
(1,0)

$\beta_{nodes} = 1$
$\beta_{age} = 0$



$$Cost(y_{pred}, y_{true}) = SVMCost(\beta_i) + \frac{1}{C}\sum_i \beta_i^2$$

slightly higher (close but not best fit)

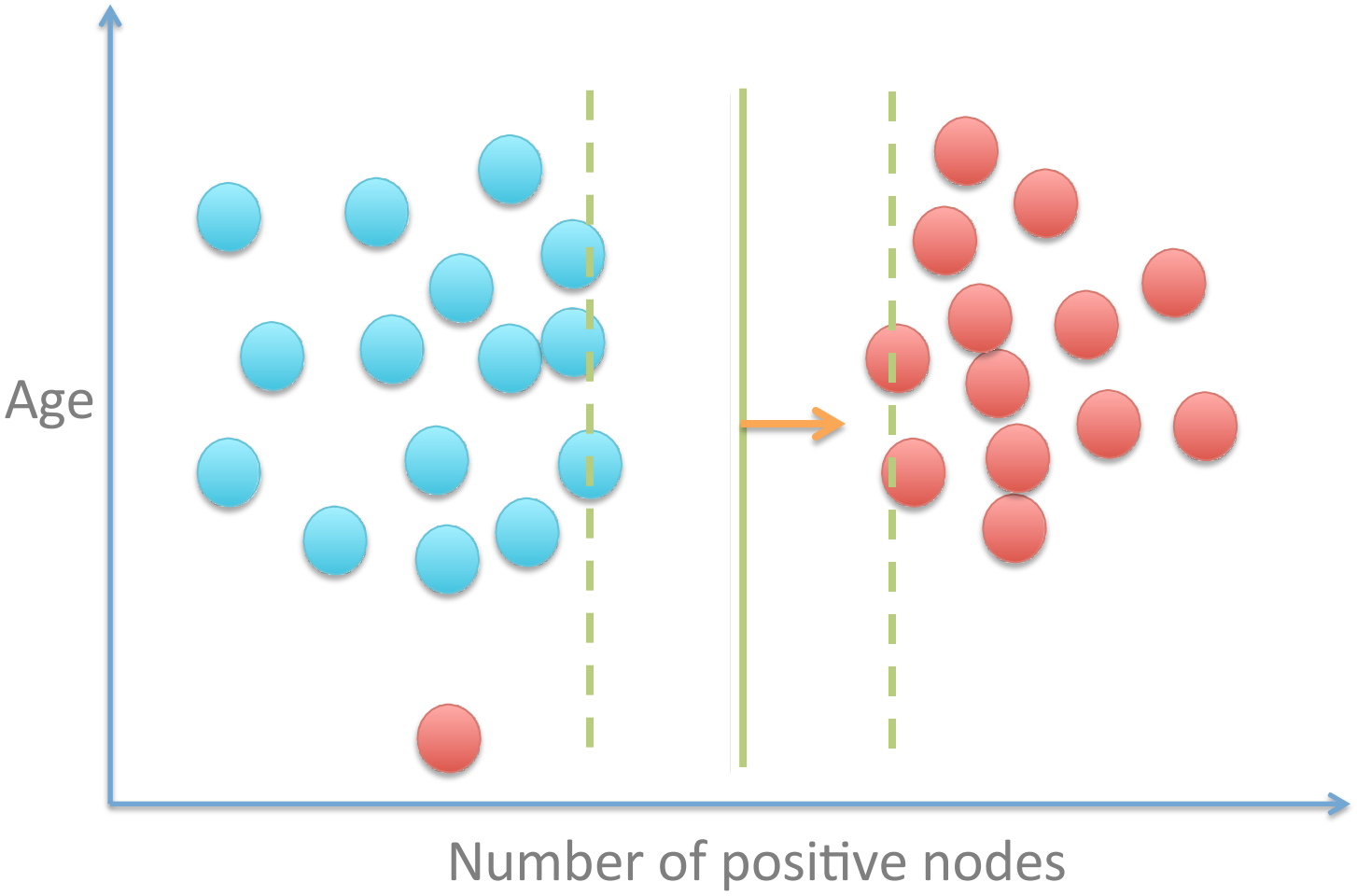
lower (smaller beta vector, simpler model)

Model 2

vector: (1,0)

$\beta_{nodes} = 1$

$\beta_{age} = 0$



$$Cost(y_{pred}, y_{true}) = SVMCost(\beta_i) + \frac{1}{C}\sum_i \beta_i^2$$

slightly higher (close but not best fit)

lower (smaller beta vector, simpler model)

Total cost is lower for this simpler model.

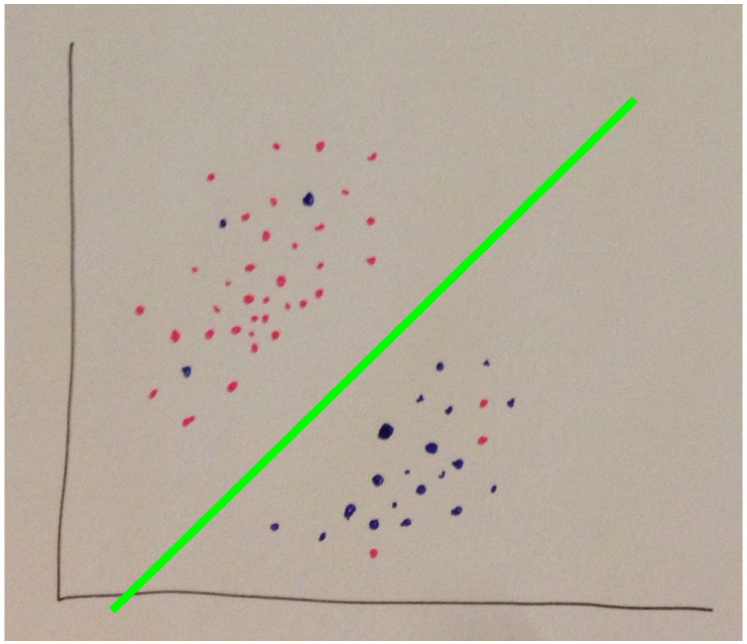
The more complex model improved fit, but not as much to overcome the extra regularization cost from the added term.

That's how regularization drives simpler models and avoids overfitting.

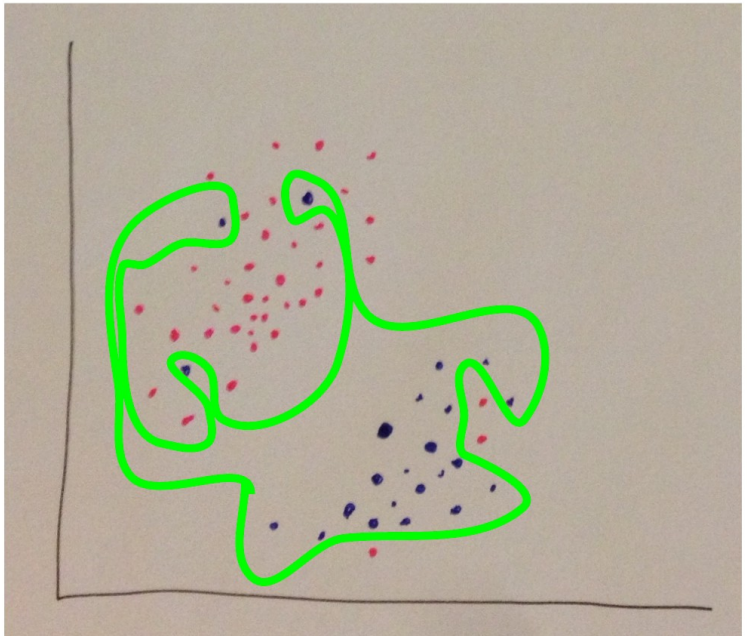
Tune regularization with C

$$Cost(y_{pred}, y_{true}) = SVMCost(\beta_i) + \frac{1}{C}\sum_i \beta_i^2$$

Low C makes the decision surface smooth



High C aims at classifying all training examples correctly.

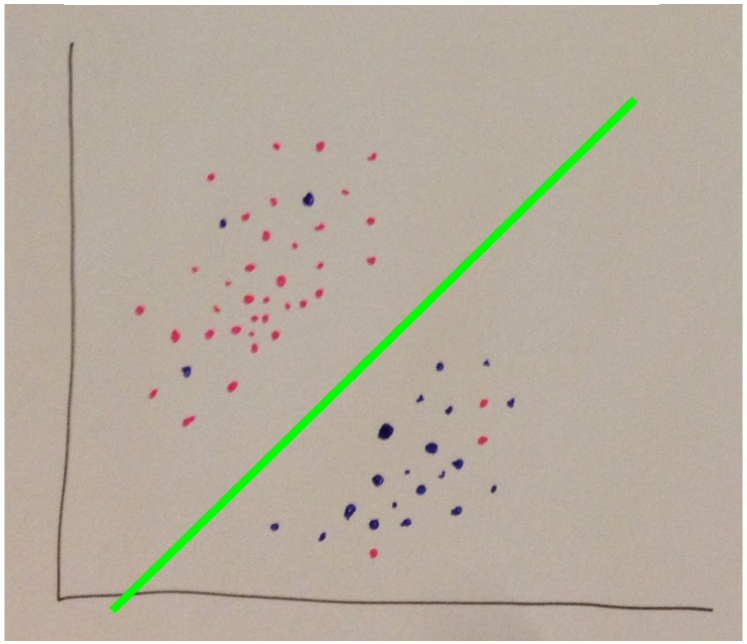


Tune regularization with C

$$Cost(y_{pred}, y_{true}) = SVMCost(\beta_i) + \frac{1}{C}\sum_i \beta_i^2$$

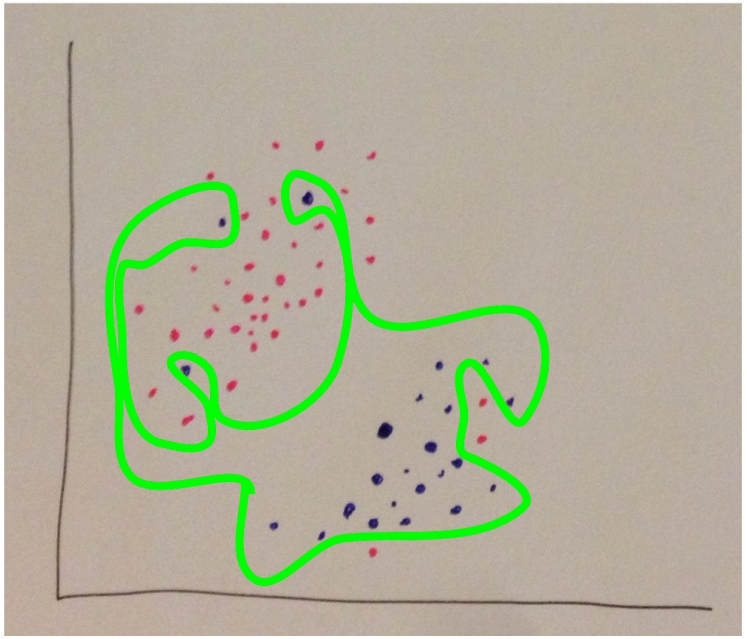
Low C

underfit



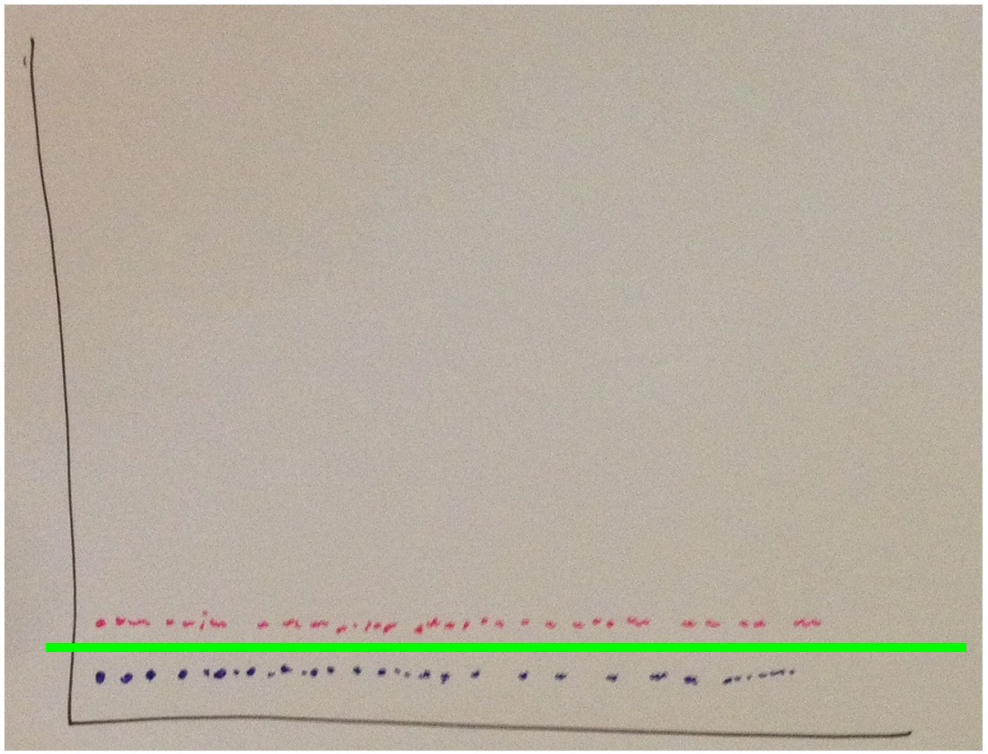
High C

overfit

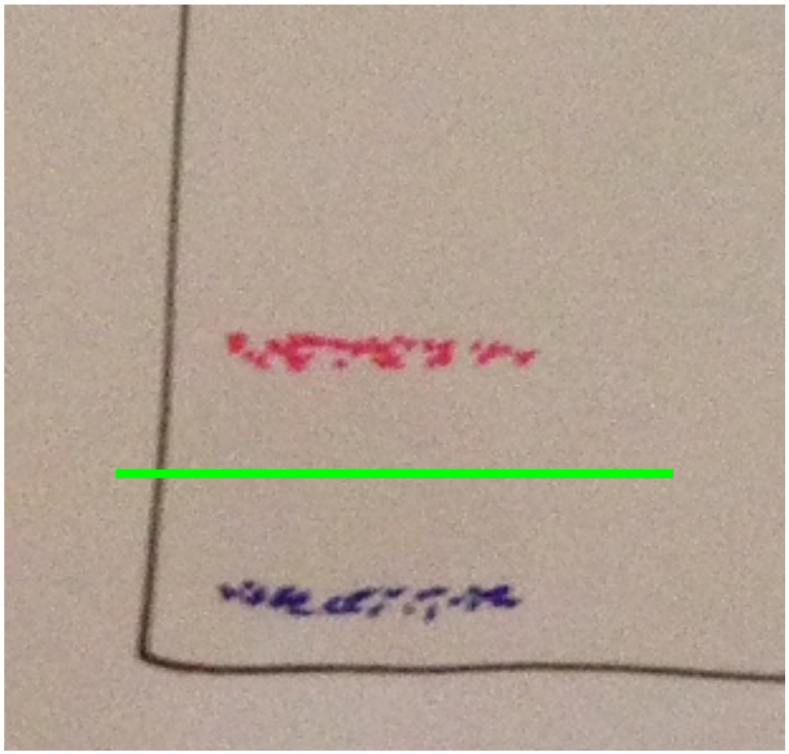


SVMs are not scale-invariant.

Scale matters



harder to fit



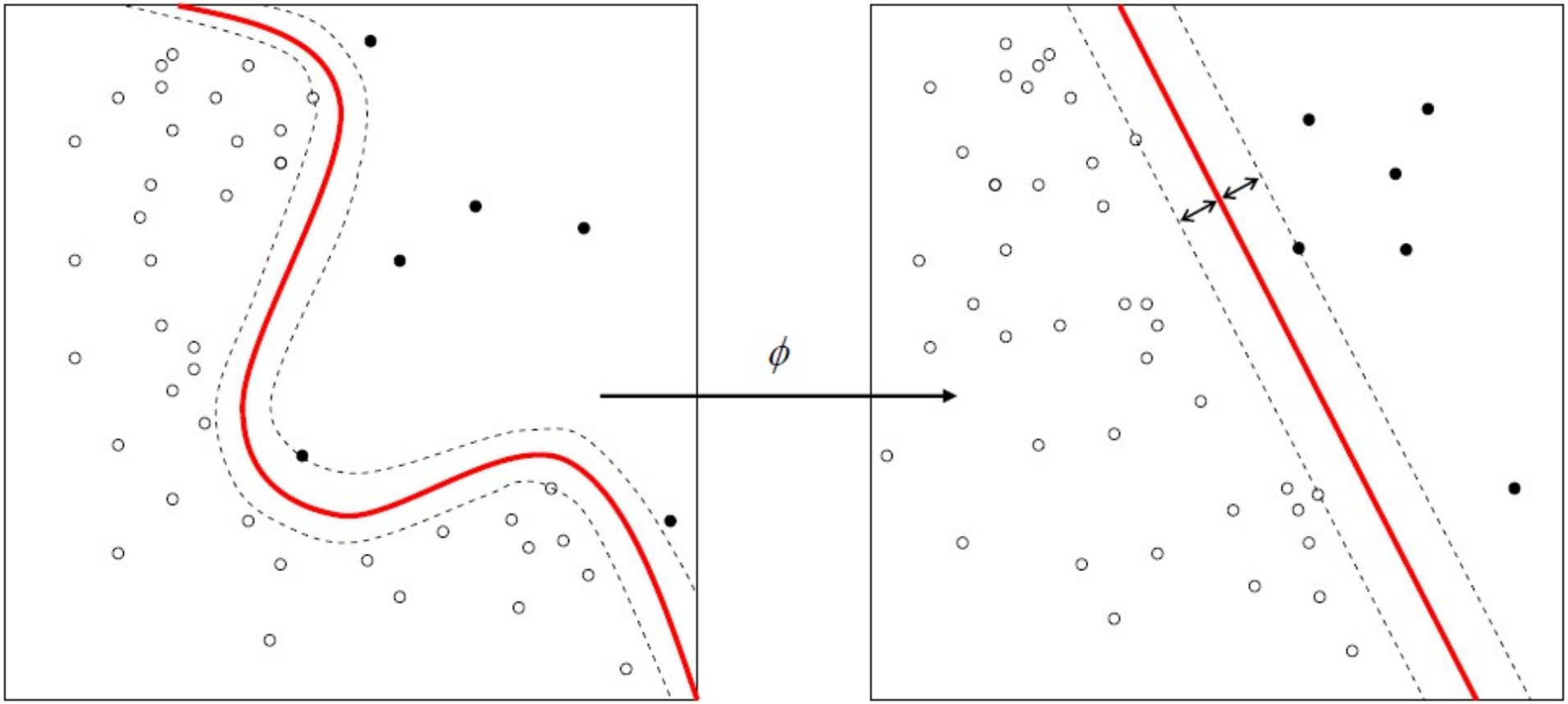
easier to fit

```
from sklearn import preprocessing

X_scaled = preprocessing.scale(X)
```

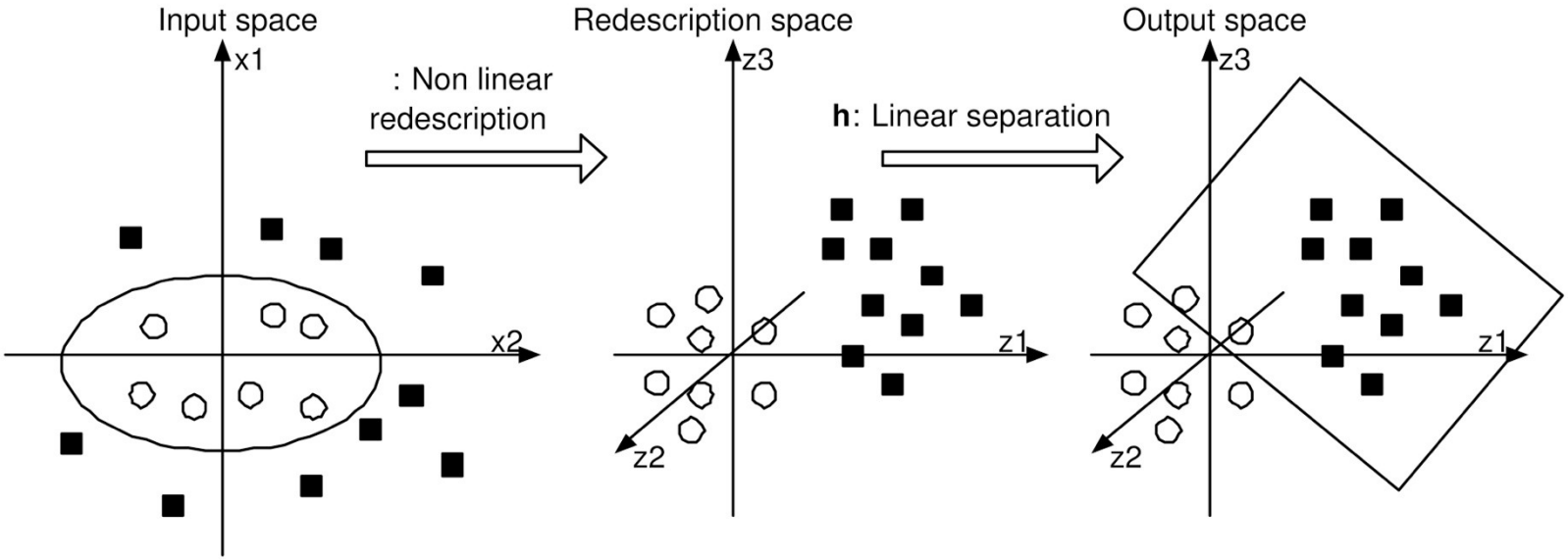
Nonlinear SVMs

Nonlinear data is linear in higher dimensions



Kernel Trick

Transform data so it is linearly separable



Different Kernels

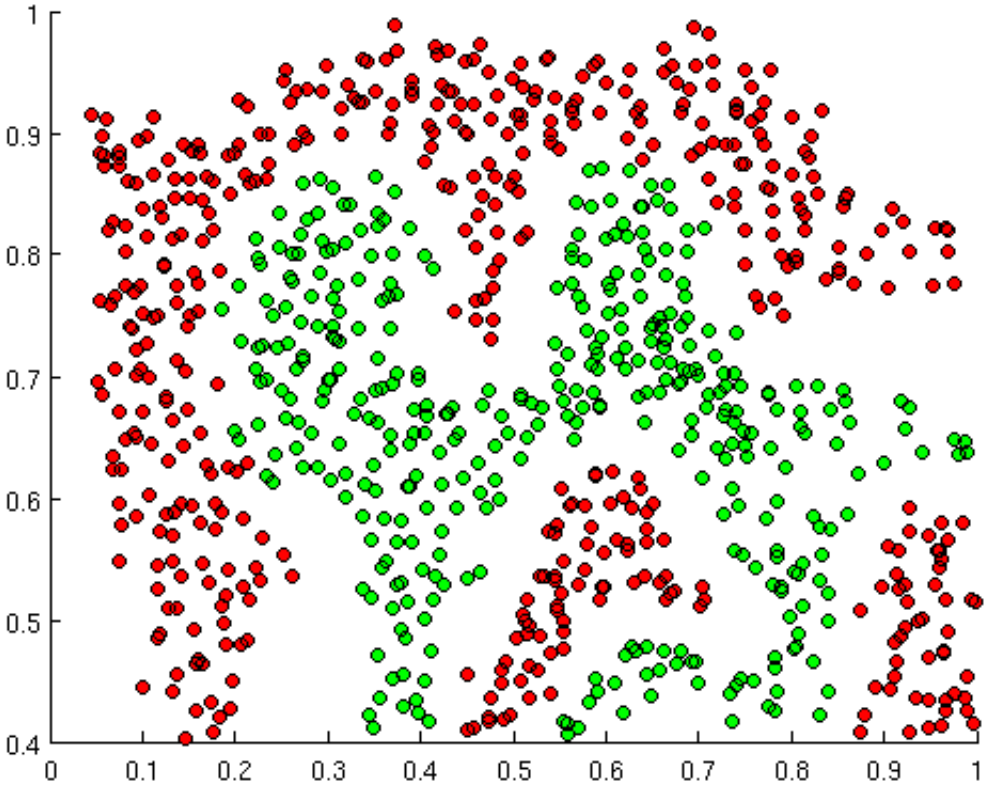
Linear kernel: No transformation

Type of Support Vector Machine	Inner Product Kernel $K(x,x_i)$, I = 1, 2, …, N	Comments
Polynomial learning machine	$(x^T x_i + 1)^p$	Power p is specified apriori by the user
Radial-basis function network	$\exp(1/(2\sigma^2)\|\|x-x_i\|\|^2)$	The width σ^2 is specified apriori
Two layer perceptron	$\tanh(\beta_0 x^T x_i + \beta_1)$	Mercer's theorem is satisfied only for some values of β_0 and β_1

Different Kernels

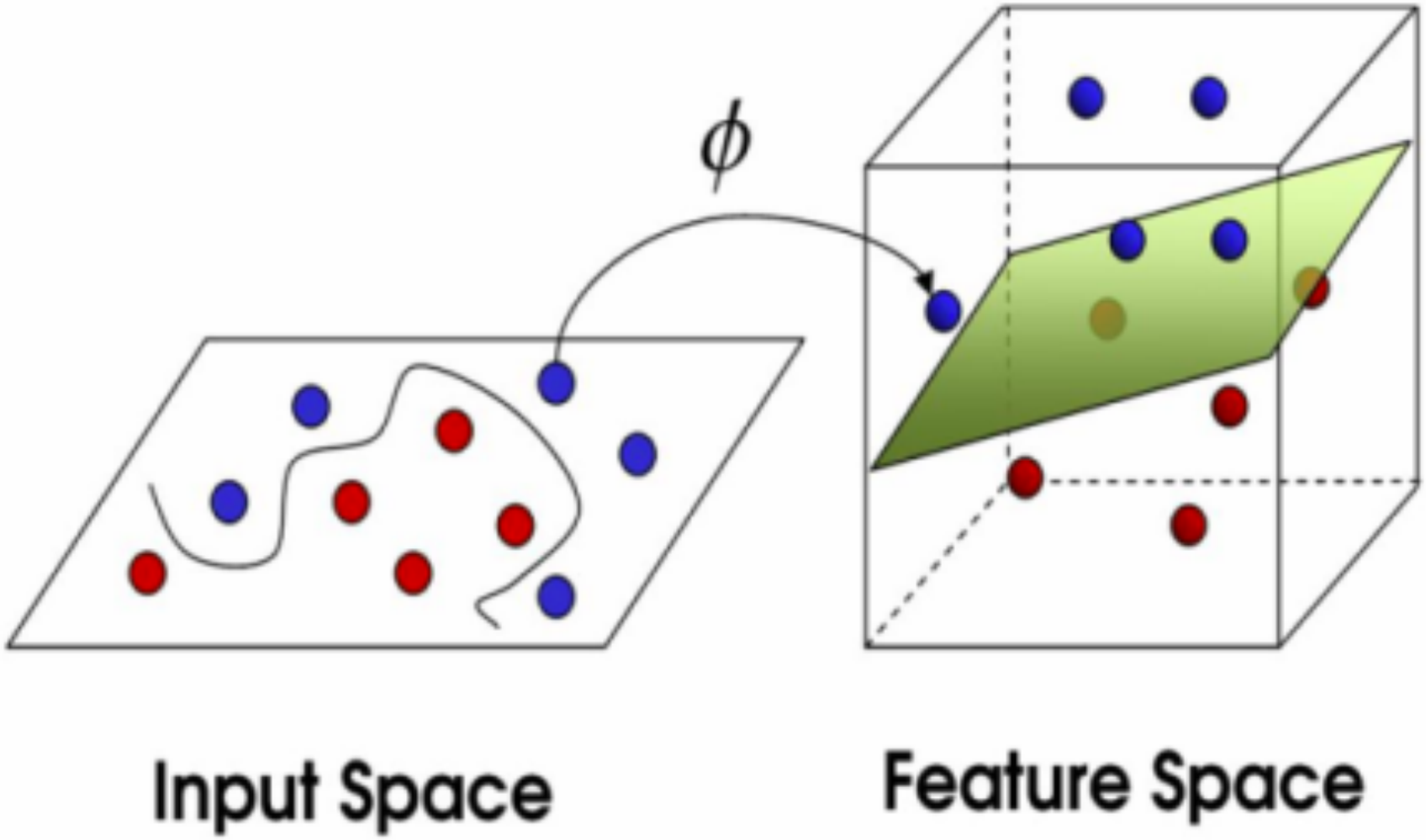
Type of Support Vector Machine	Inner Product Kernel $K(x,x_i)$, I = 1, 2, …, N	Comments
Polynomial learning machine	$(x^T x_i + 1)^p$	Power p is specified apriori by the user
Radial-basis function network	$\exp(1/(2\sigma^2)\|\|x-x_i\|\|^2)$	The width σ^2 is specified apriori
Two layer perceptron	$\tanh(\beta_0 x^T x_i + \beta_1)$	Mercer's theorem is satisfied only for some values of β_0 and β_1

SVMs can fit intricate boundaries with the Gaussian (RBF) kernel



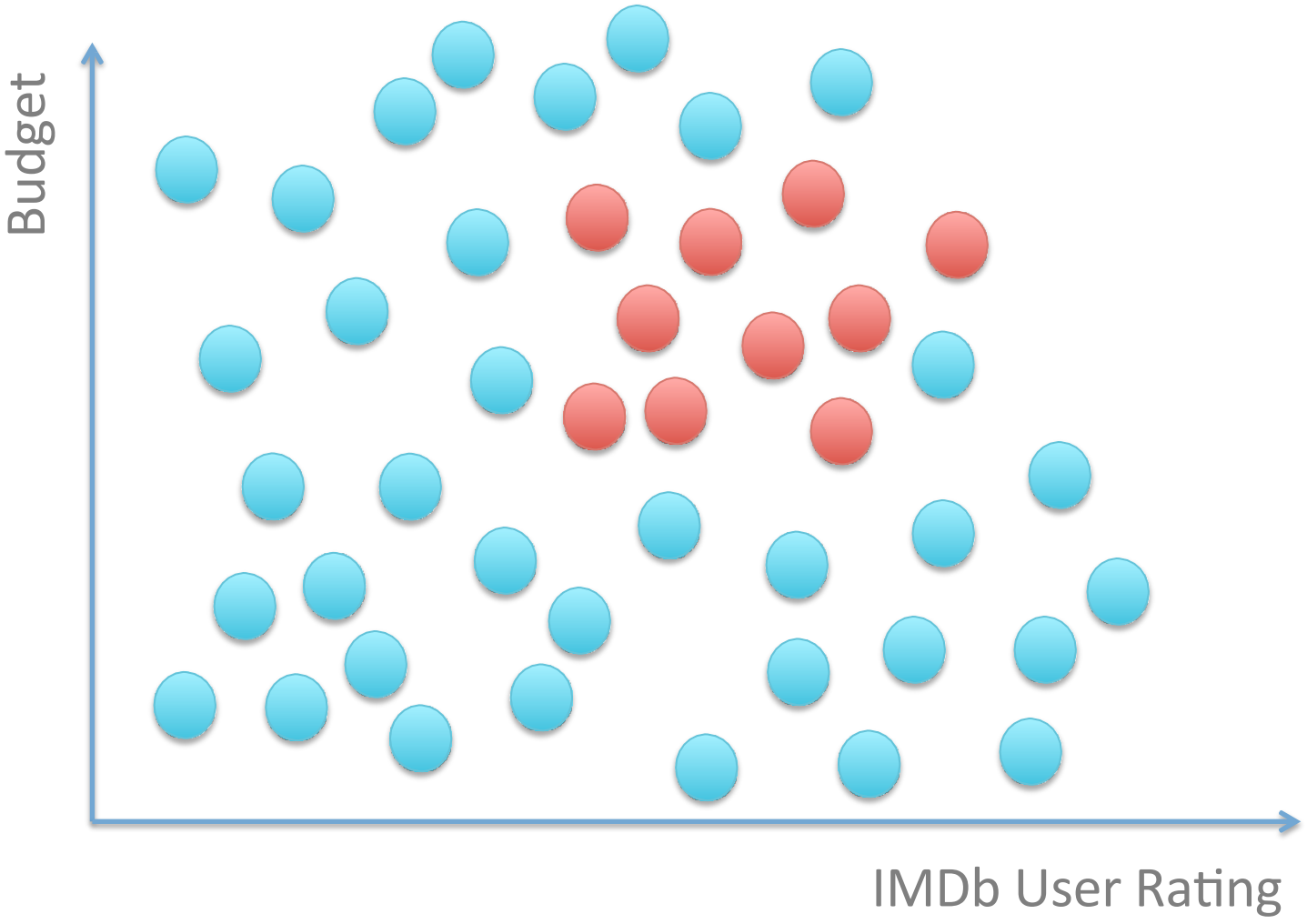
Gaussian Kernel

How does it work?



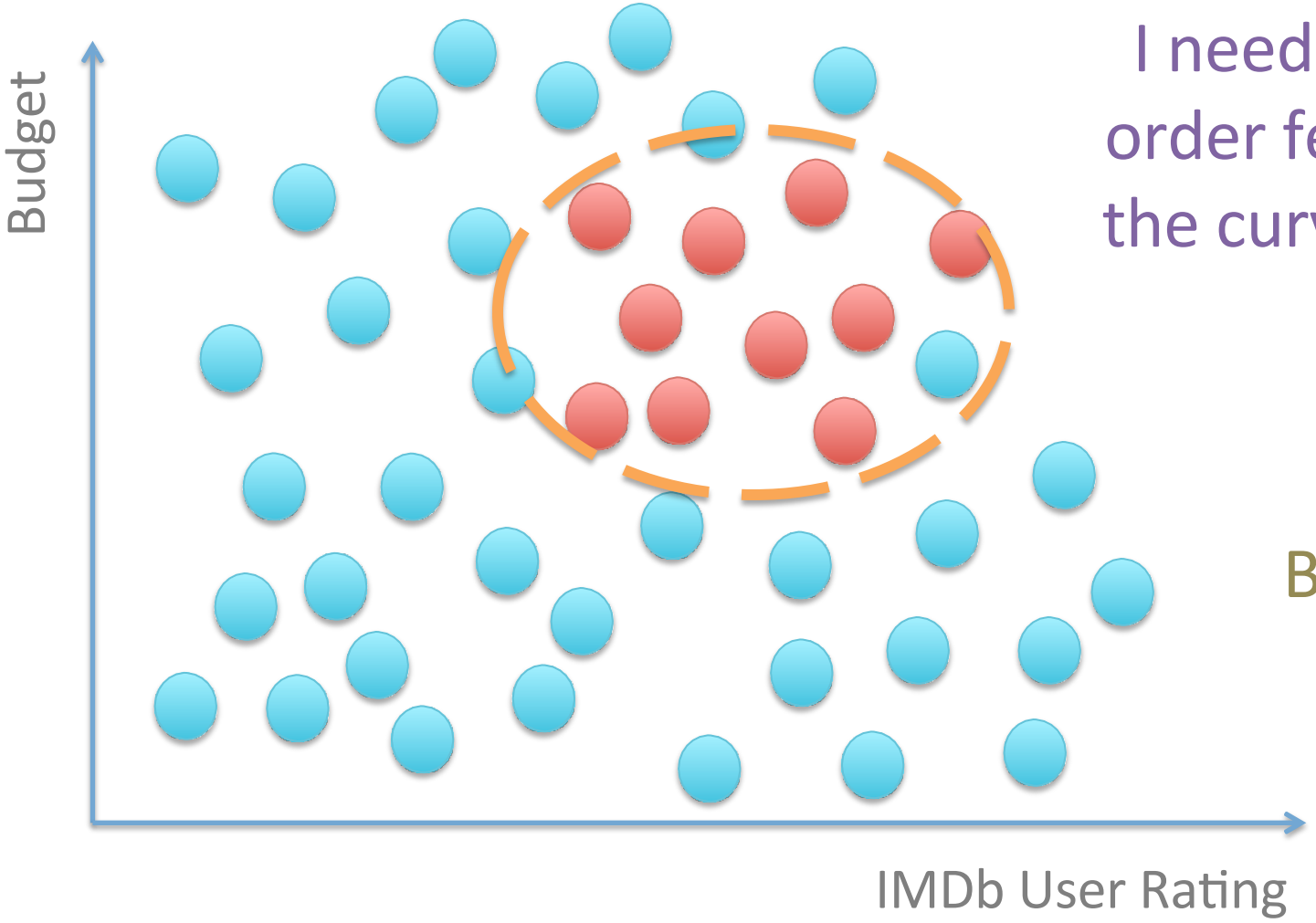
Palme d'Or Winners at Cannes

Pretty intricate boundary



Palme d'Or Winners at Cannes

Pretty intricate boundary



I need to use higher order features to get the curved boundary

$Budget^2$

$Rating^2$

$Budget*Rating$

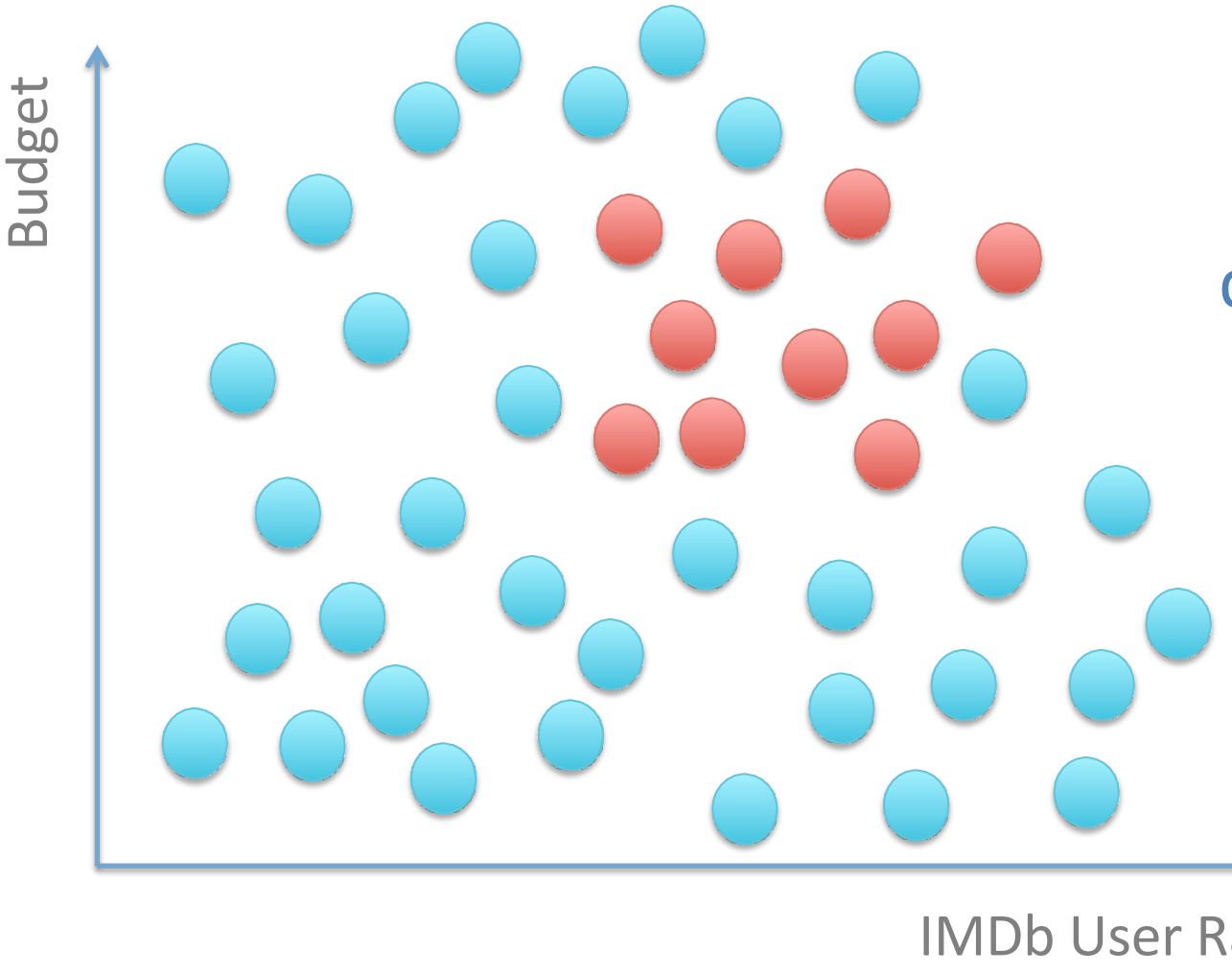
$Budget^3$

$Rating^3$

etc.

Palme d'Or Winners at Cannes

Different Approach: Transform the space



Two features describe this space

Can I describe it differently?

Palme d'Or Winners at Cannes

Different Approach: Transform the space

Two features describe this space

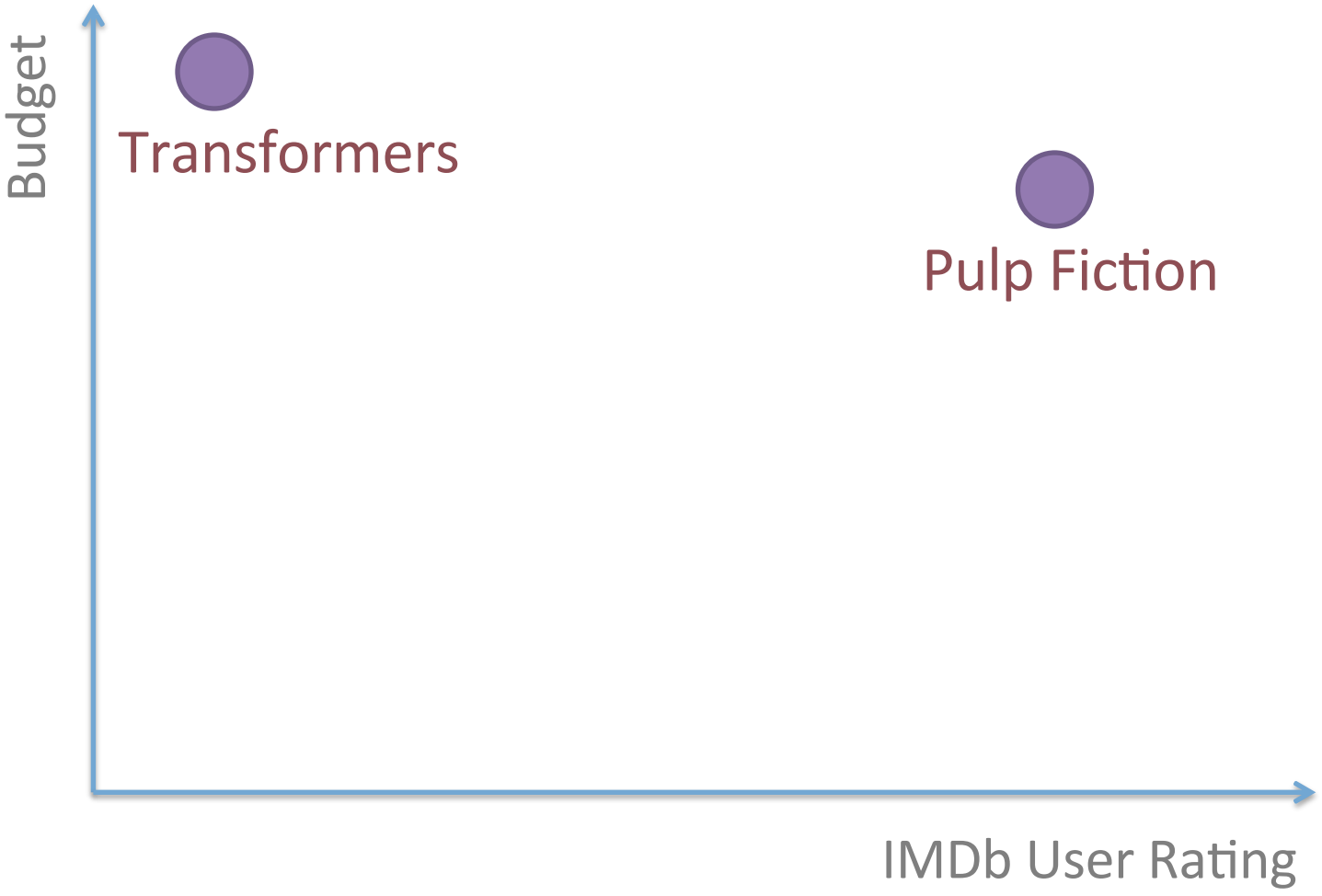
Can I describe it differently?

Palme d'Or Winners at Cannes

Different Approach: Transform the space

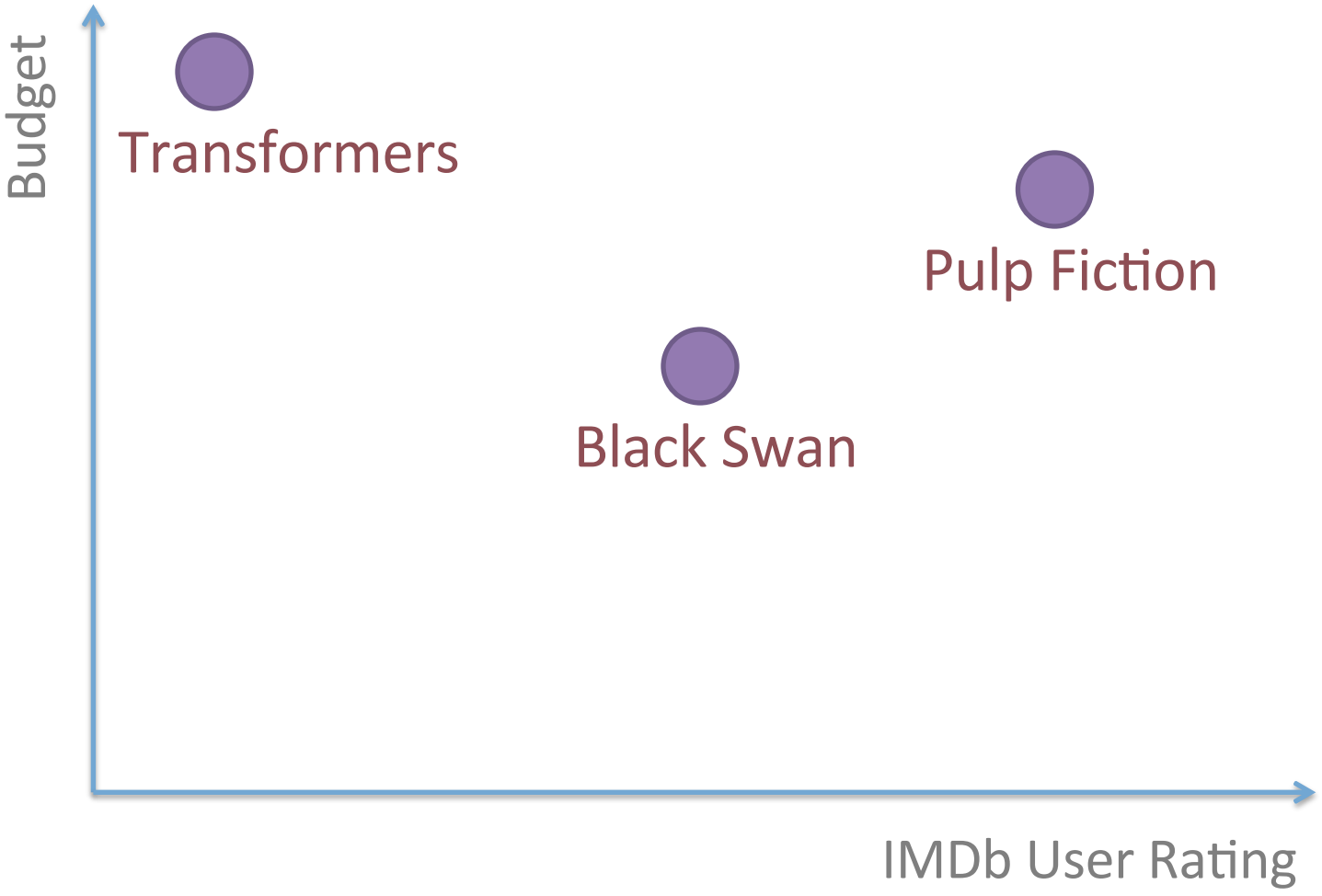
Palme d'Or Winners at Cannes

Different Approach: Transform the space



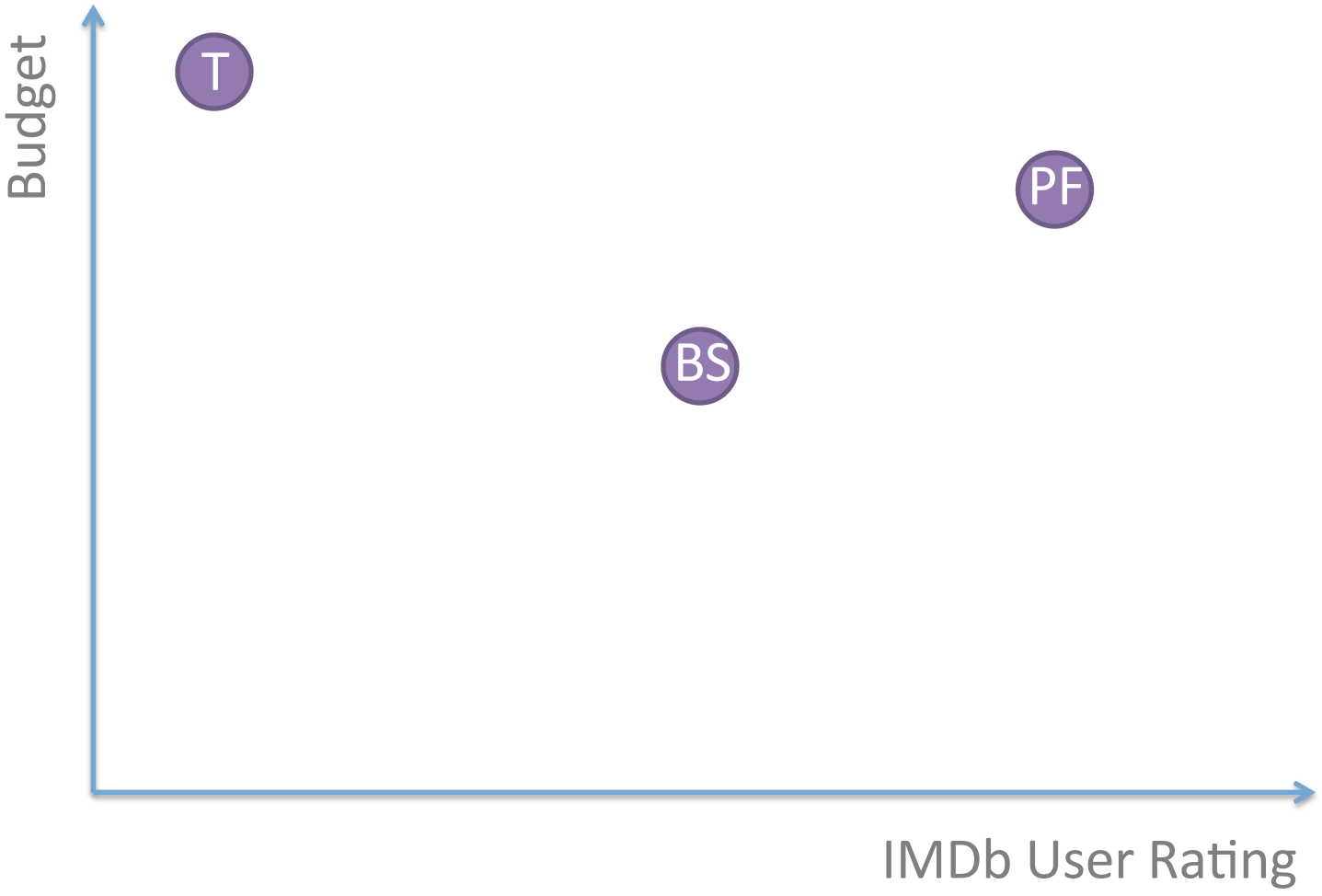
Palme d'Or Winners at Cannes

Different Approach: Transform the space



Define feature a_1: "Pulp Fiction"ness

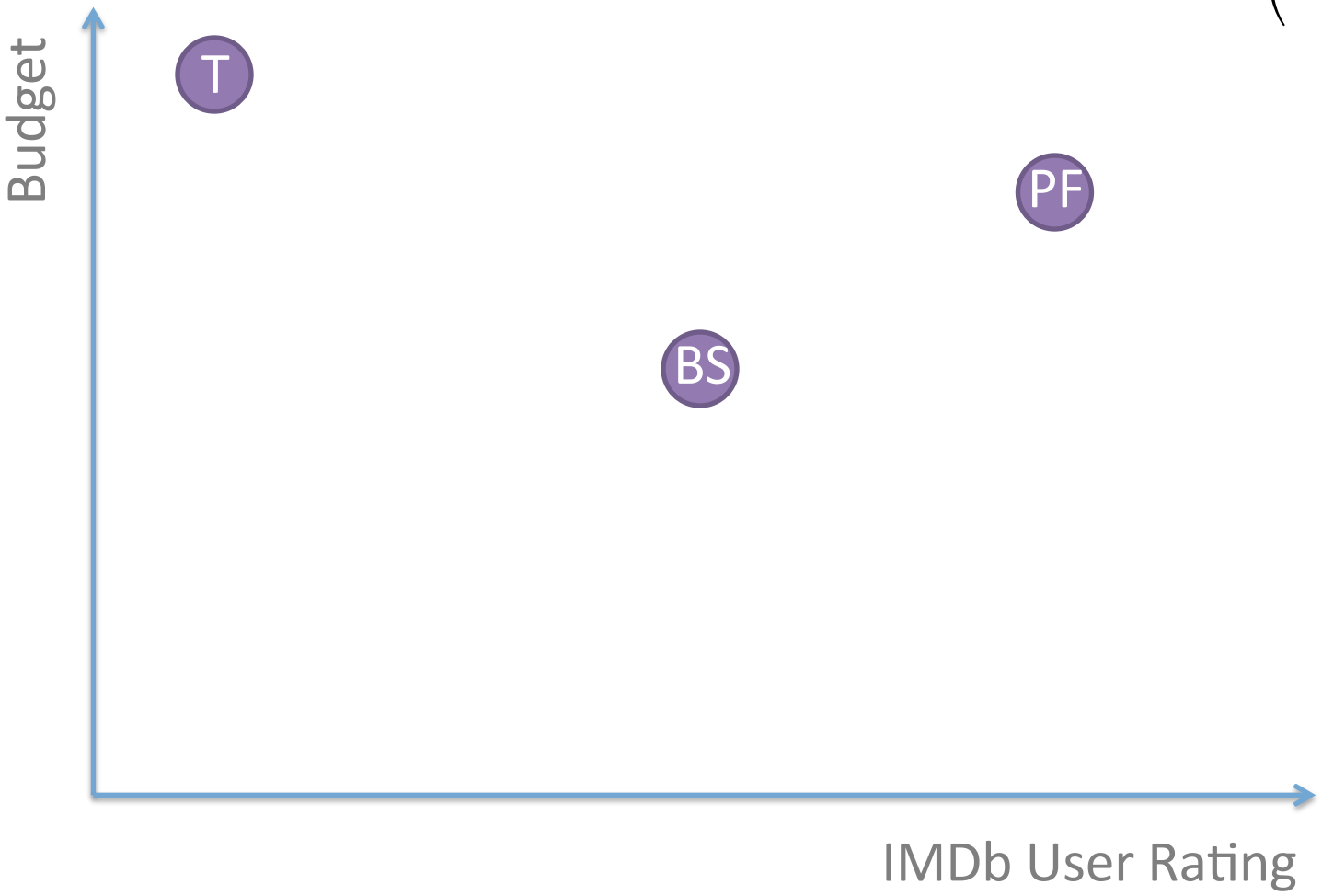
Are you close to Pulp Fiction?



Define feature a_1: "Pulp Fiction"ness

Are you close to Pulp Fiction?

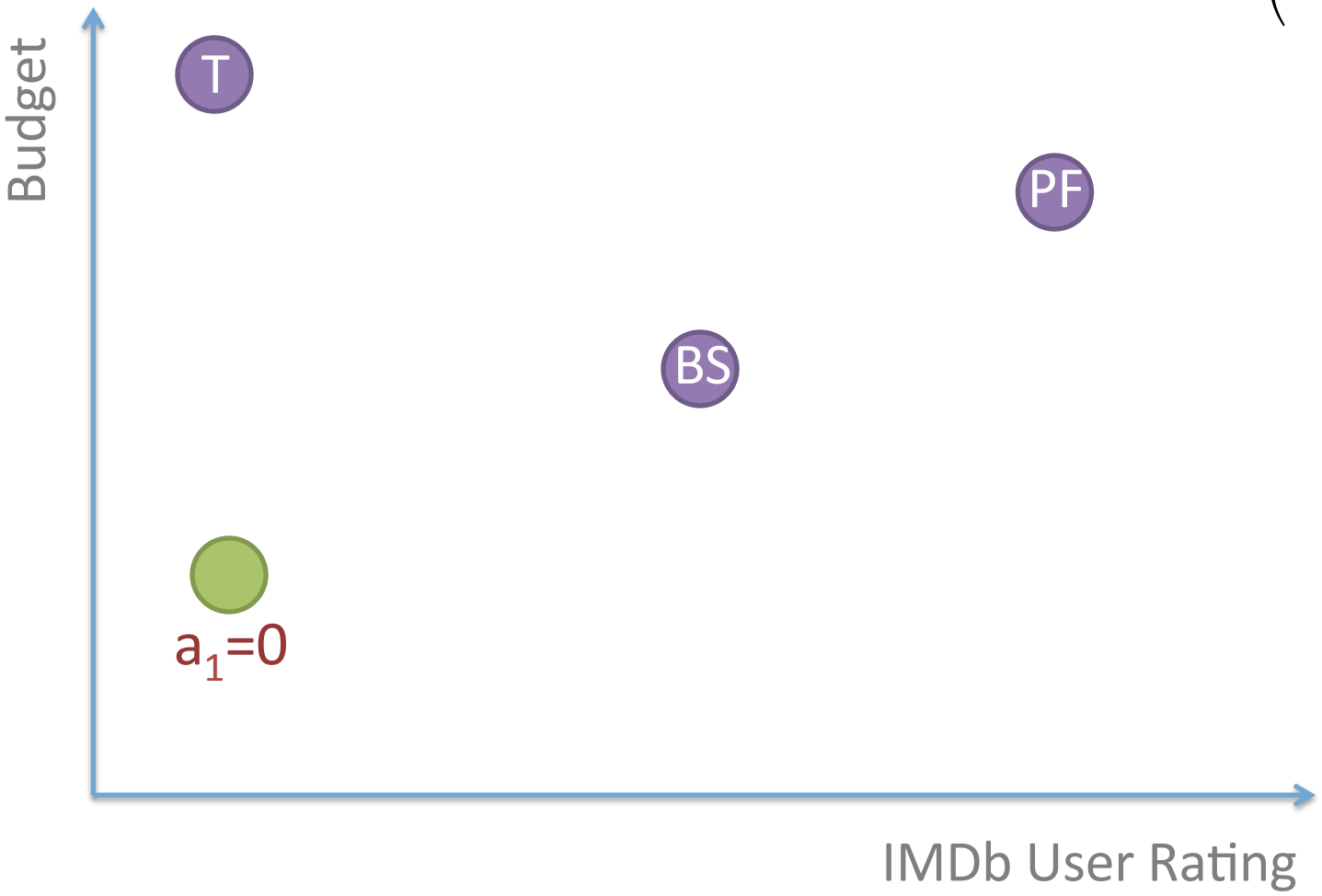
$$a_1(\overrightarrow{x^{obs}}) = \exp\left(\frac{\sum\left(x_i^{obs} - x_i^{PulpFiction}\right)^2}{2\sigma^2}\right)$$



Define feature a_1: "Pulp Fiction"ness

Are you close to Pulp Fiction?

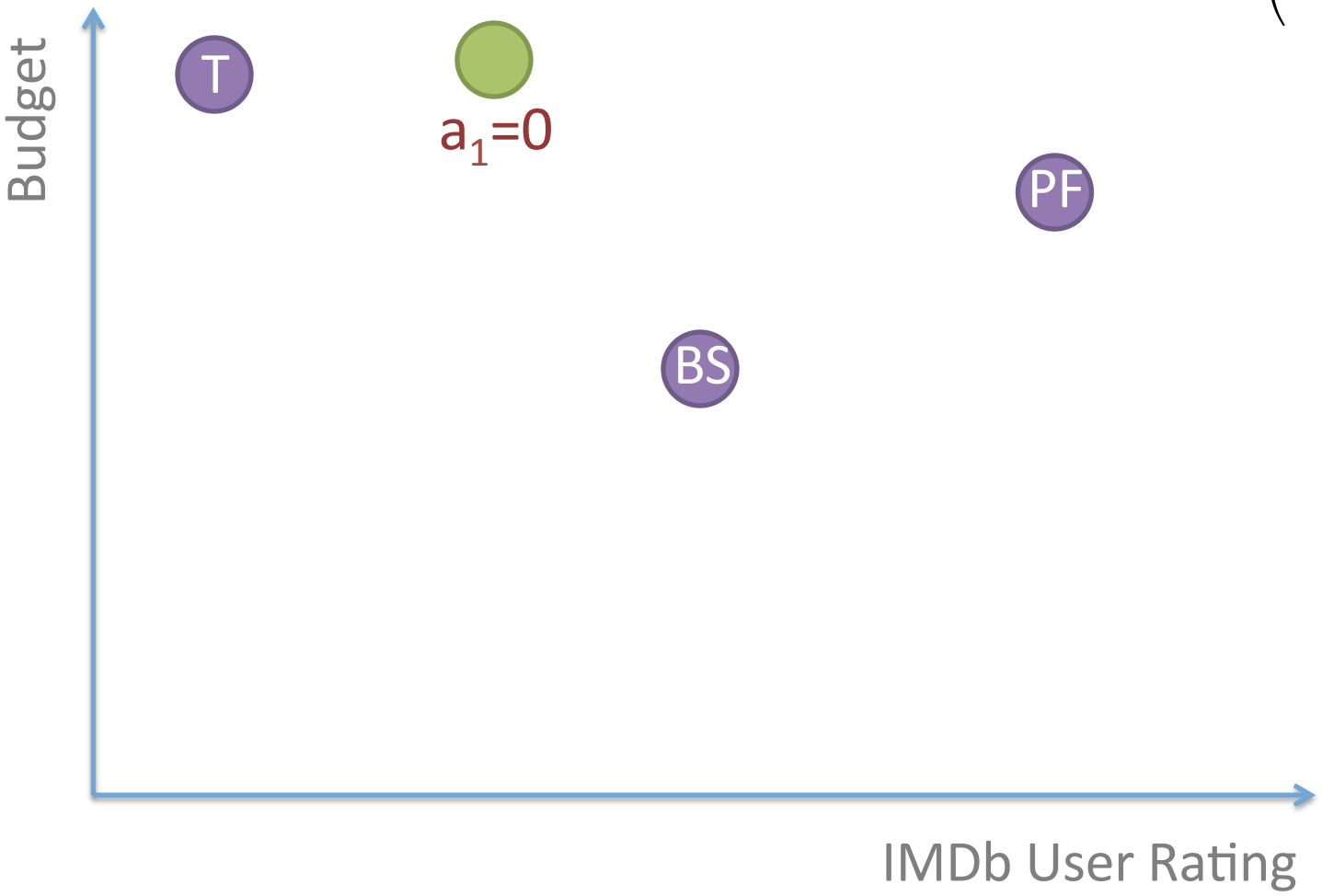
$$a_1(\overrightarrow{x^{obs}}) = \exp\left(\frac{\sum\left(x_i^{obs} - x_i^{PulpFiction}\right)^2}{2\sigma^2}\right)$$



Define feature a_1: "Pulp Fiction"ness

Are you close to Pulp Fiction?

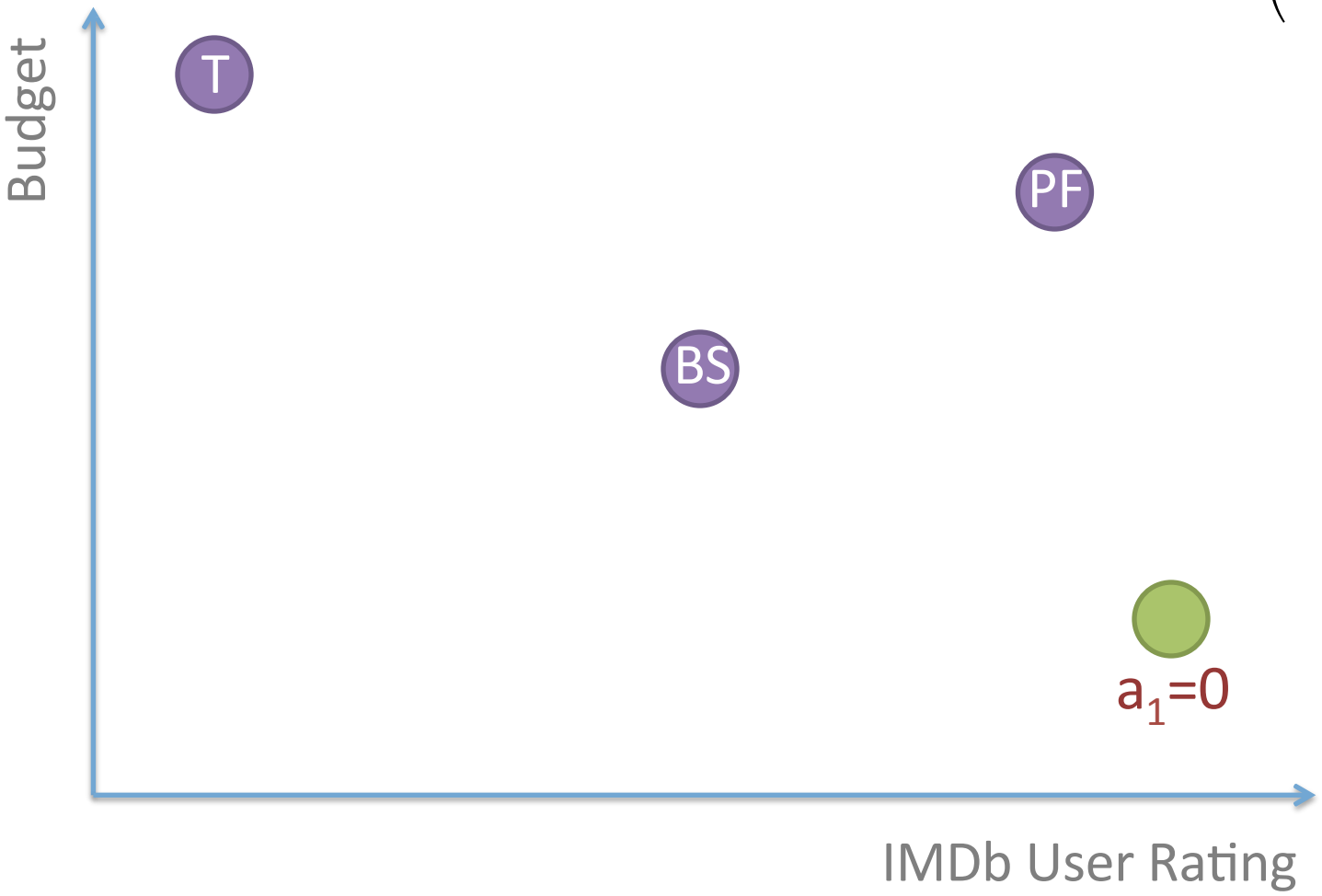
$$a_1(\overrightarrow{x^{obs}}) = \exp\left(\frac{\sum\left(x_i^{obs} - x_i^{PulpFiction}\right)^2}{2\sigma^2}\right)$$



Define feature a_1: "Pulp Fiction"ness

Are you close to Pulp Fiction?

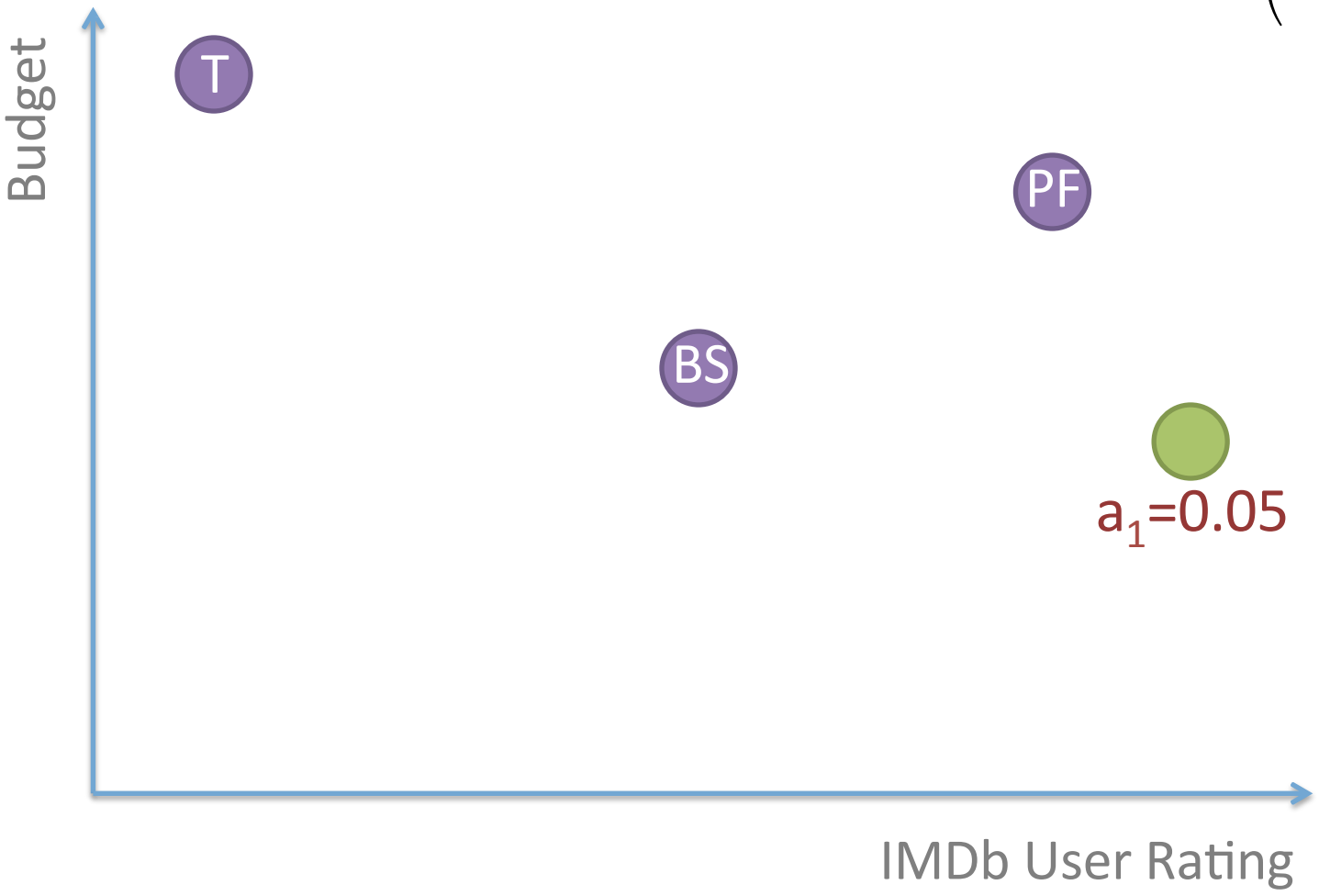
$$a_1(\overrightarrow{x^{obs}}) = \exp\left(\frac{\sum\left(x_i^{obs} - x_i^{PulpFiction}\right)^2}{2\sigma^2}\right)$$



Define feature a_1: "Pulp Fiction"ness

Are you close to Pulp Fiction?

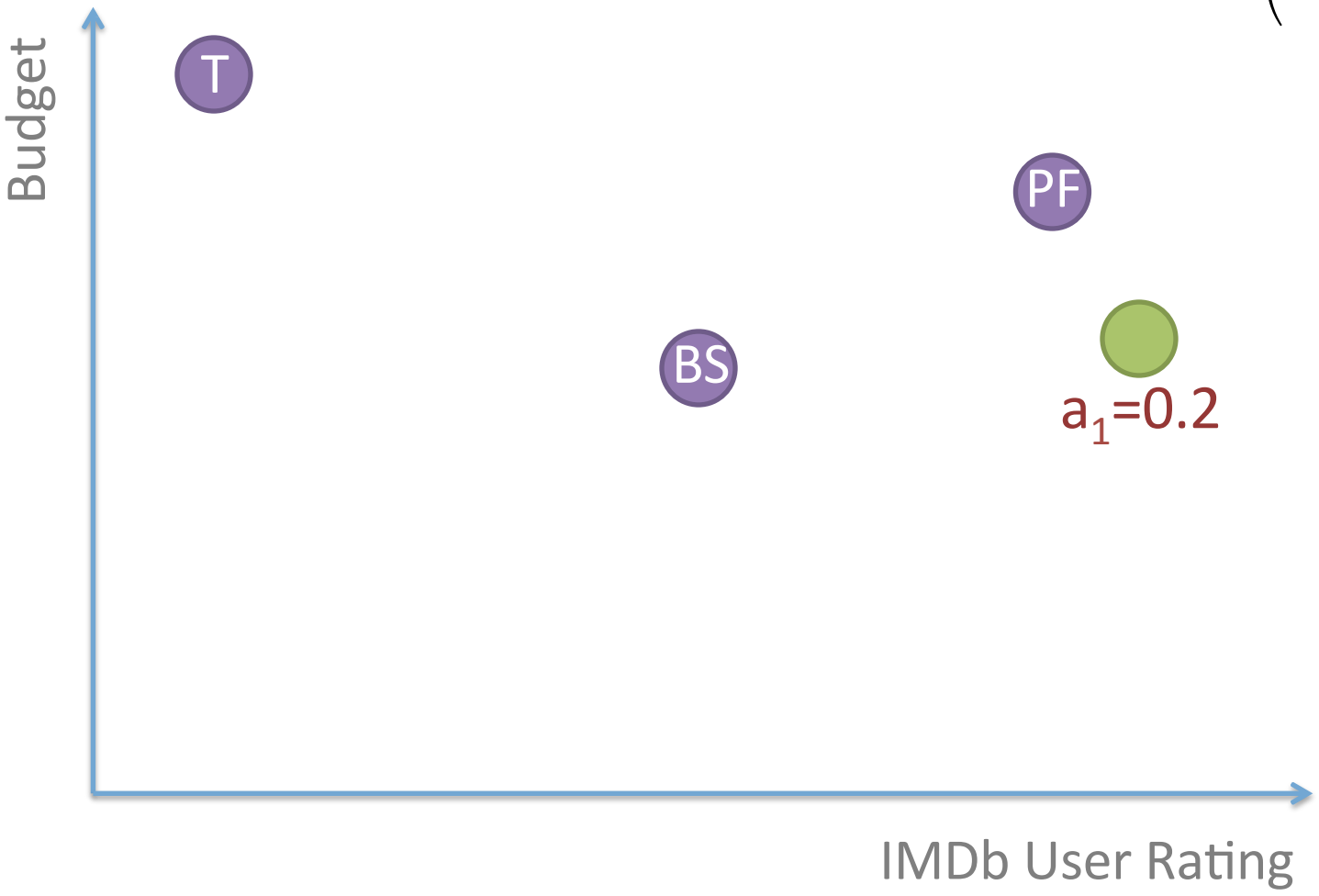
$$a_1(\overrightarrow{x^{obs}}) = \exp\left(\frac{\sum\left(x_i^{obs} - x_i^{PulpFiction}\right)^2}{2\sigma^2}\right)$$



Define feature a_1: "Pulp Fiction"ness

Are you close to Pulp Fiction?

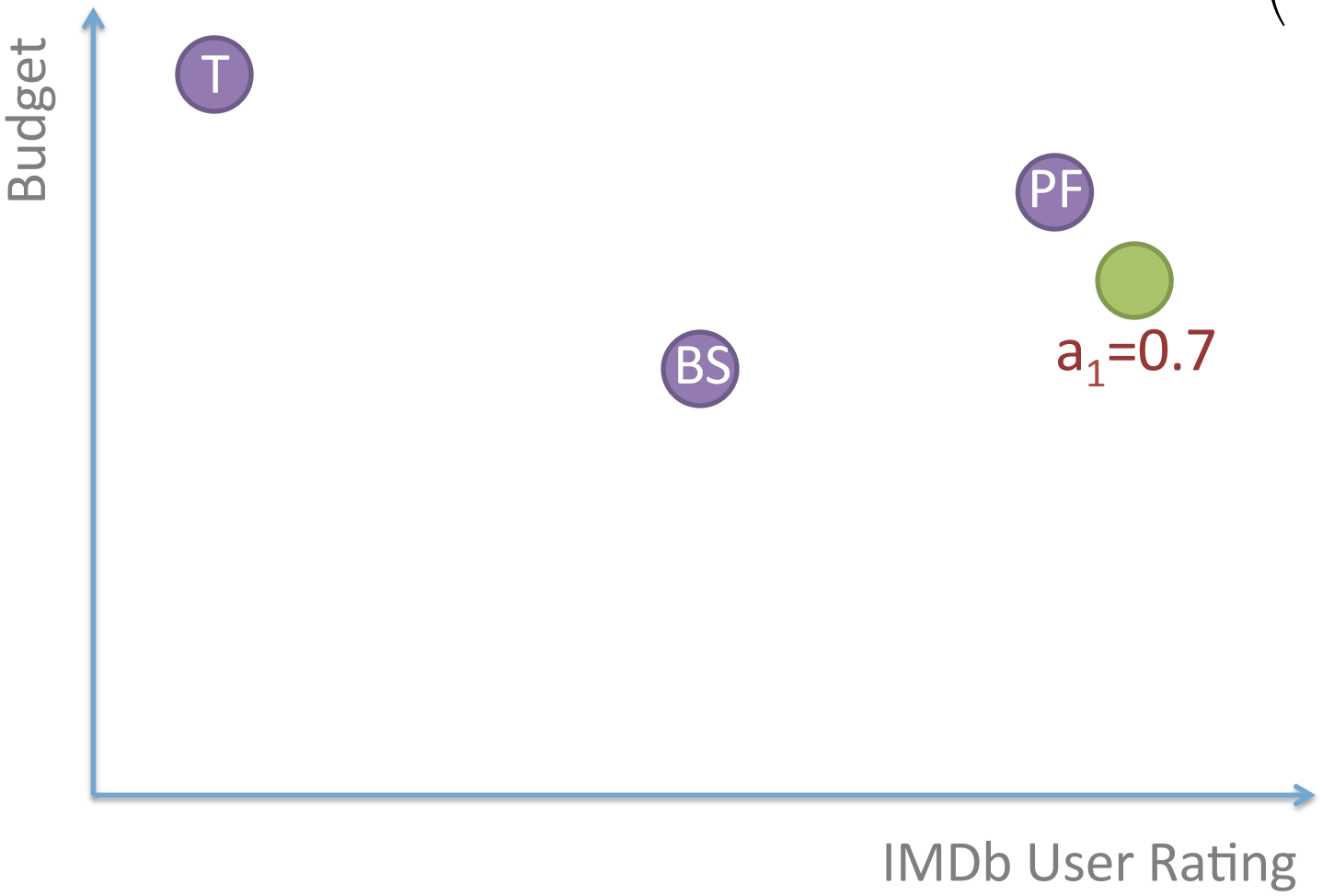
$$a_1(\overrightarrow{x^{obs}}) = \exp\left(\frac{\sum\left(x_i^{obs} - x_i^{PulpFiction}\right)^2}{2\sigma^2}\right)$$



Define feature a_1: "Pulp Fiction"ness

Are you close to Pulp Fiction?

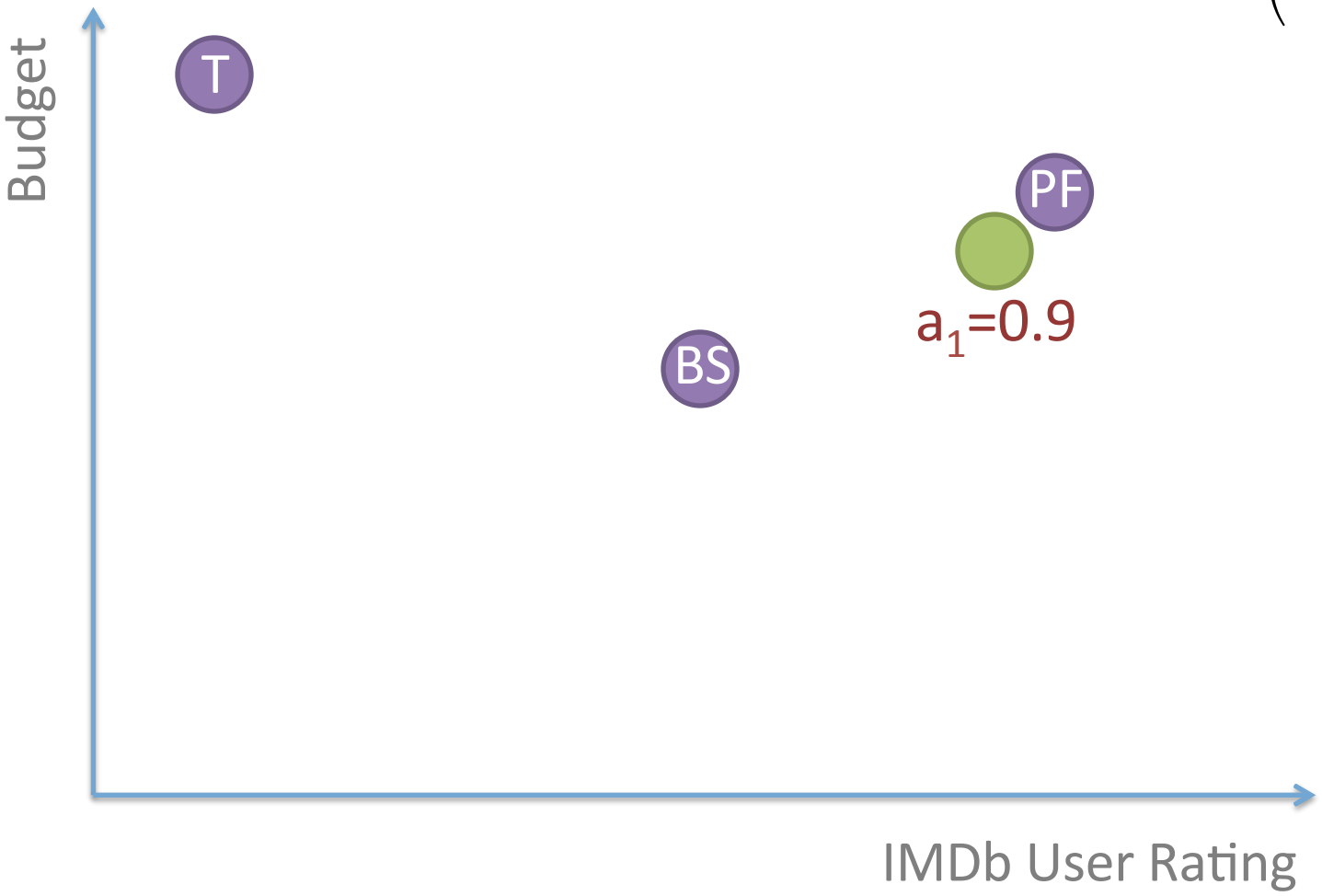
$$a_1(\overrightarrow{x^{obs}}) = \exp\left(\frac{\sum\left(x_i^{obs} - x_i^{PulpFiction}\right)^2}{2\sigma^2}\right)$$



Define feature a_1: "Pulp Fiction"ness

Are you close to Pulp Fiction?

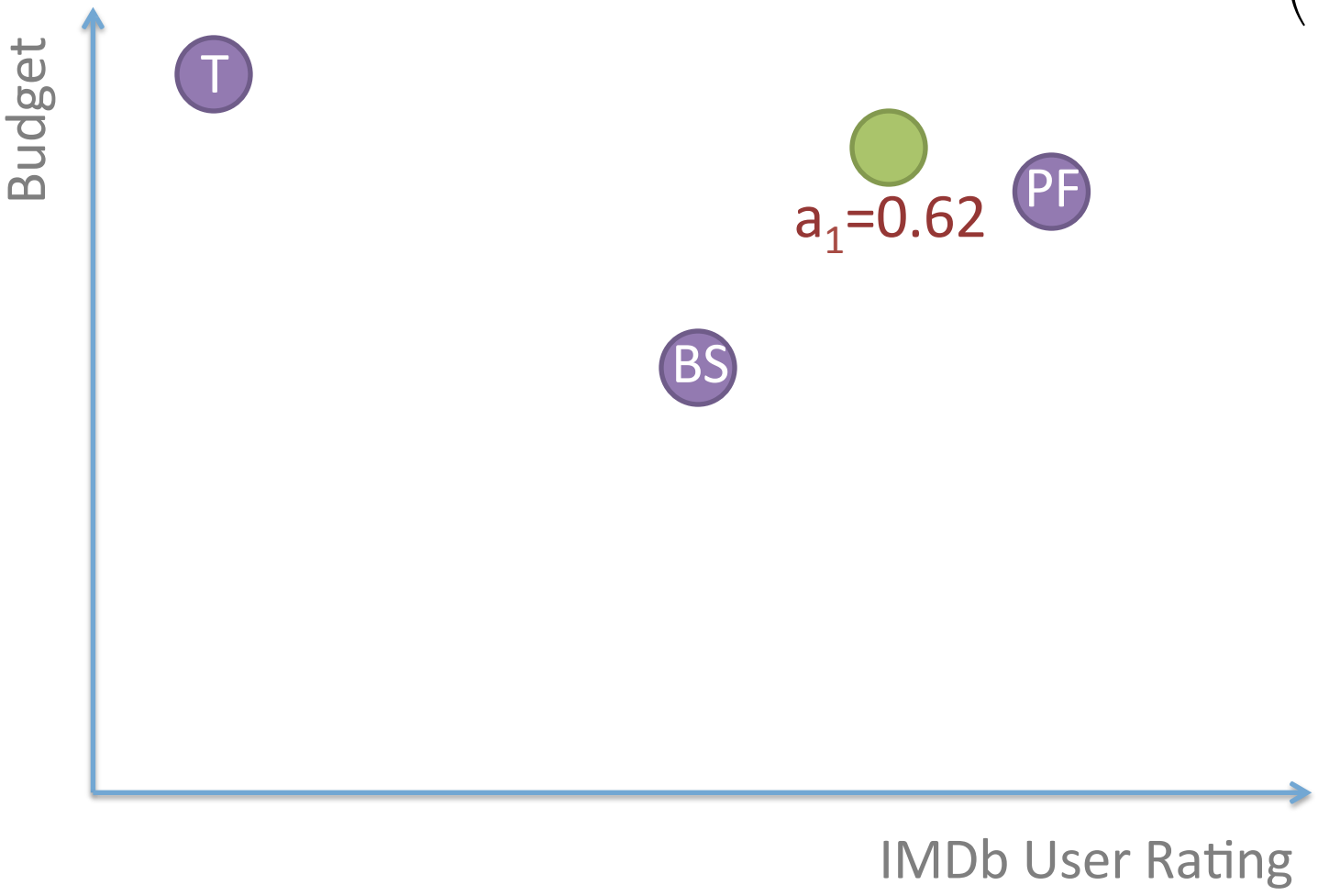
$$a_1(\overrightarrow{x^{obs}}) = \exp\left(\frac{\sum\left(x_i^{obs} - x_i^{PulpFiction}\right)^2}{2\sigma^2}\right)$$



Define feature a_1: "Pulp Fiction"ness

Are you close to Pulp Fiction?

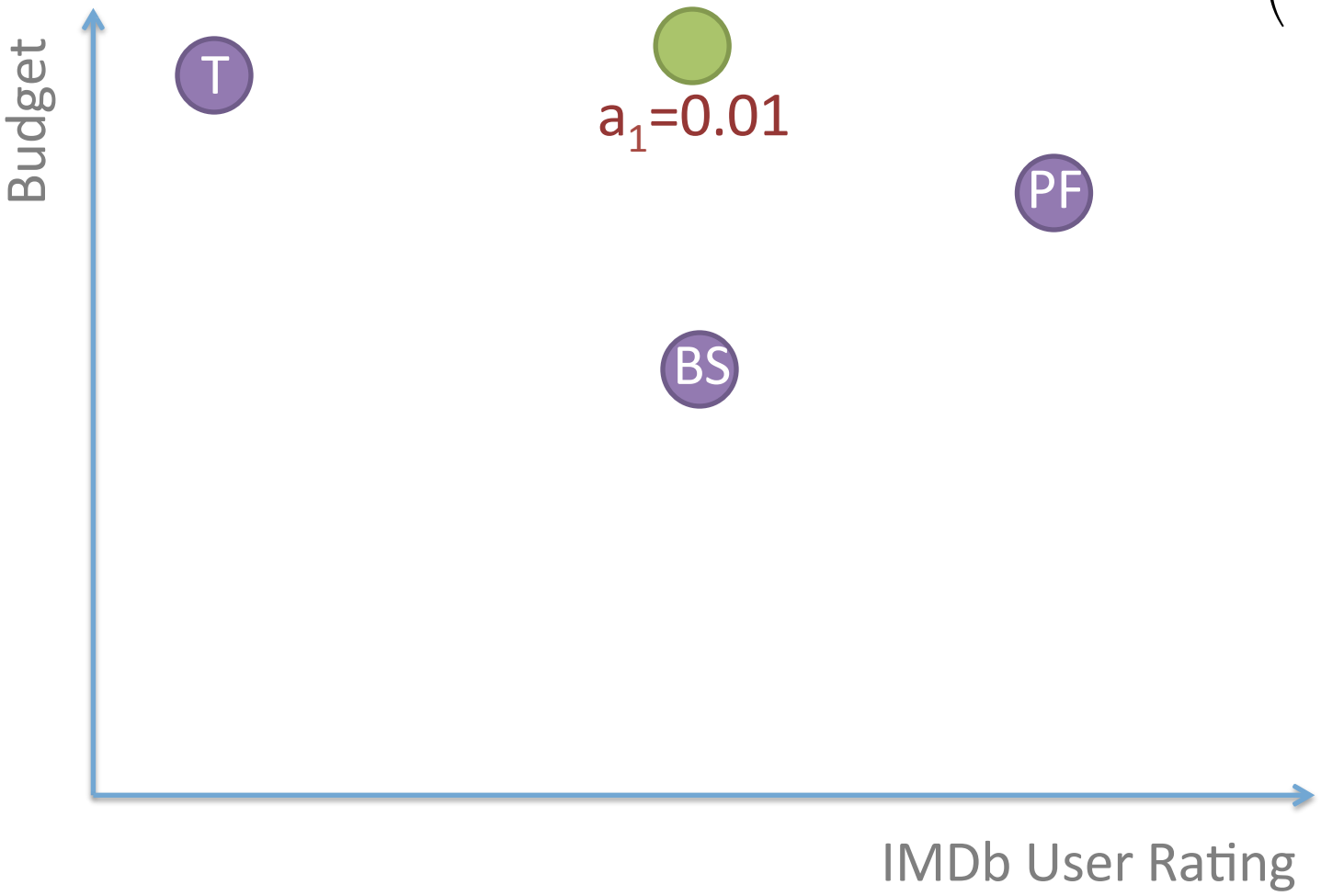
$$a_1(\vec{x}) = \exp\left(\frac{\sum\left(x_i - x_i^{PulpFiction}\right)^2}{2\sigma^2}\right)$$



Define feature a_1: "Pulp Fiction"ness

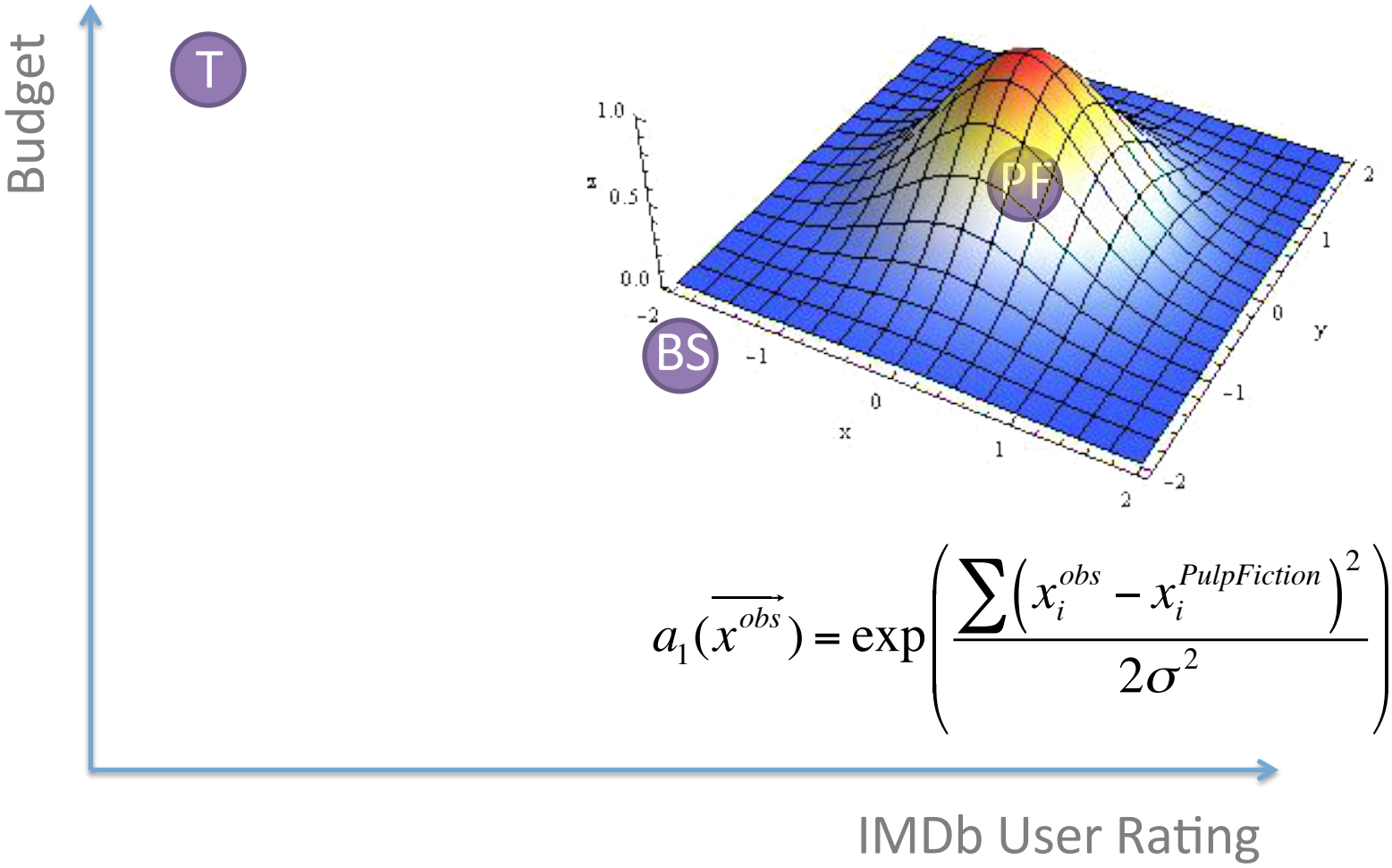
Are you close to Pulp Fiction?

$$a_1(\overrightarrow{x^{obs}}) = \exp\left(\frac{\sum\left(x_i^{obs} - x_i^{PulpFiction}\right)^2}{2\sigma^2}\right)$$

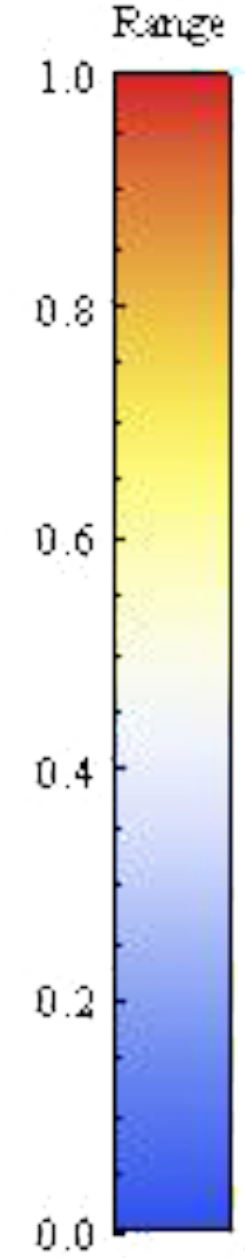


Define feature a_1: "Pulp Fiction"ness

Are you close to Pulp Fiction?

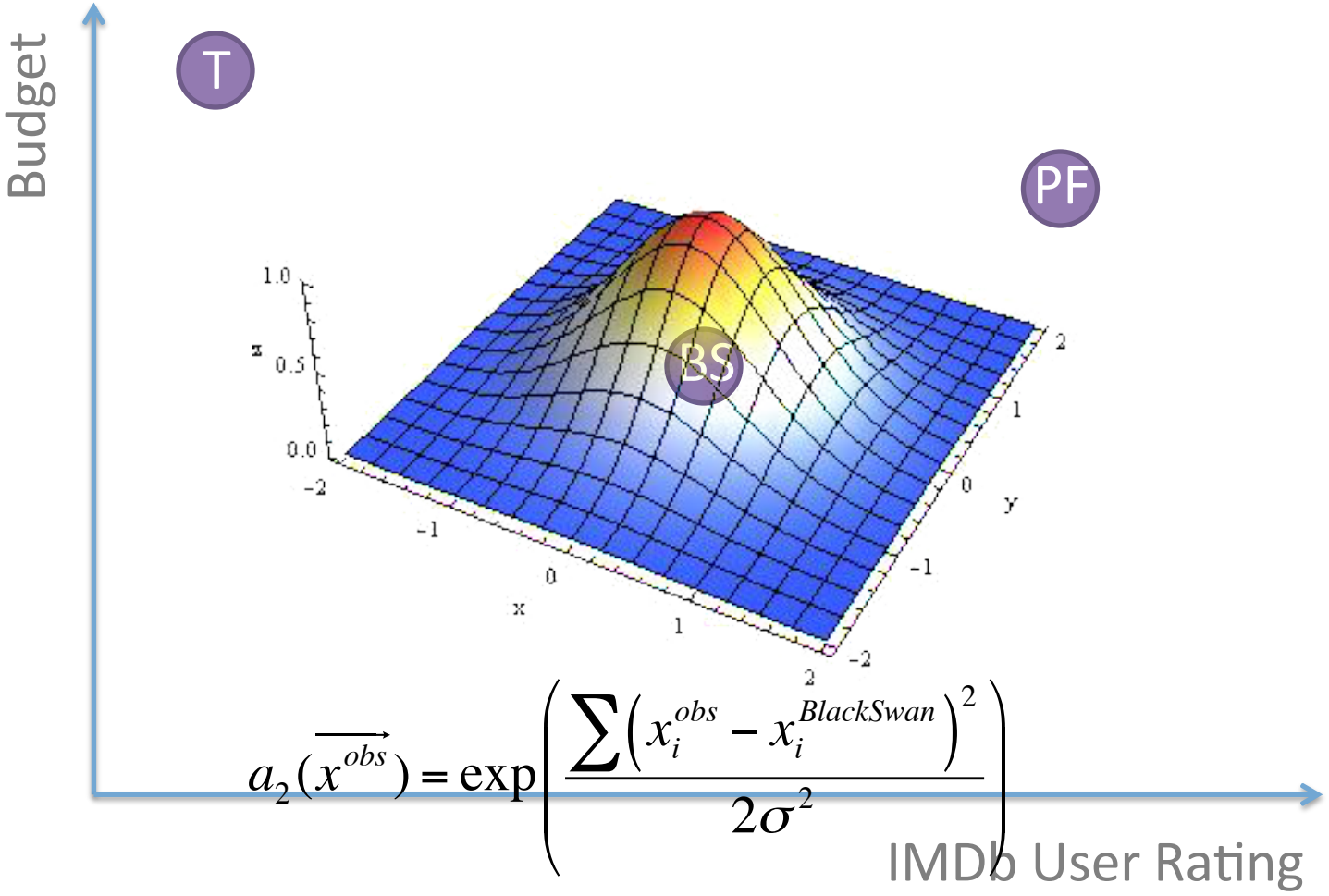


$$a_1(\overrightarrow{x^{obs}}) = \exp\left(\frac{\sum\left(x_i^{obs} - x_i^{PulpFiction}\right)^2}{2\sigma^2}\right)$$

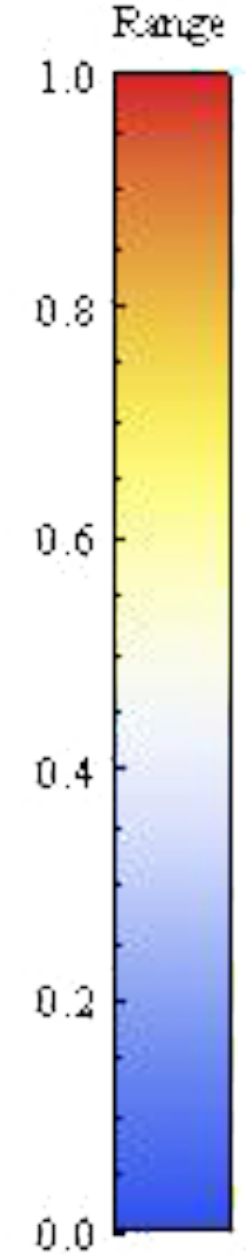


Define feature a_2: "Black Swan"ness

Are you close to Black Swan?

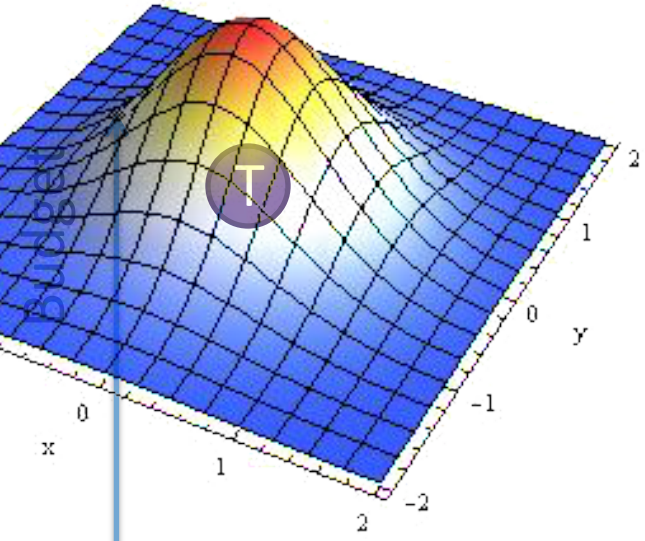


$$a_2(\overrightarrow{x^{obs}}) = \exp\left(\frac{\sum\left(x_i^{obs} - x_i^{BlackSwan}\right)^2}{2\sigma^2}\right)$$



Define feature a_3: "Transformers"ness

Are you close to Black Swan?

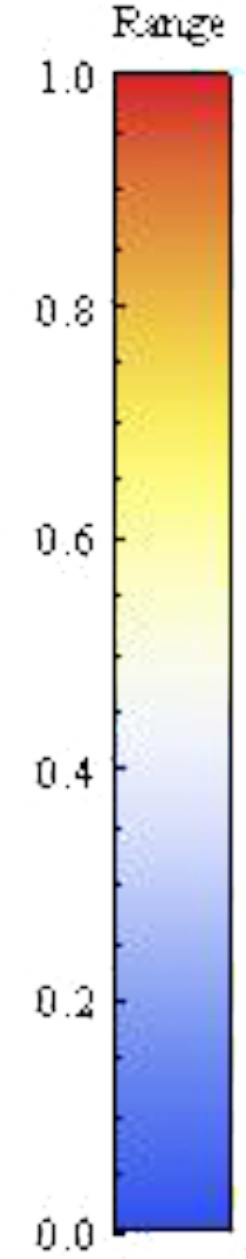


PF

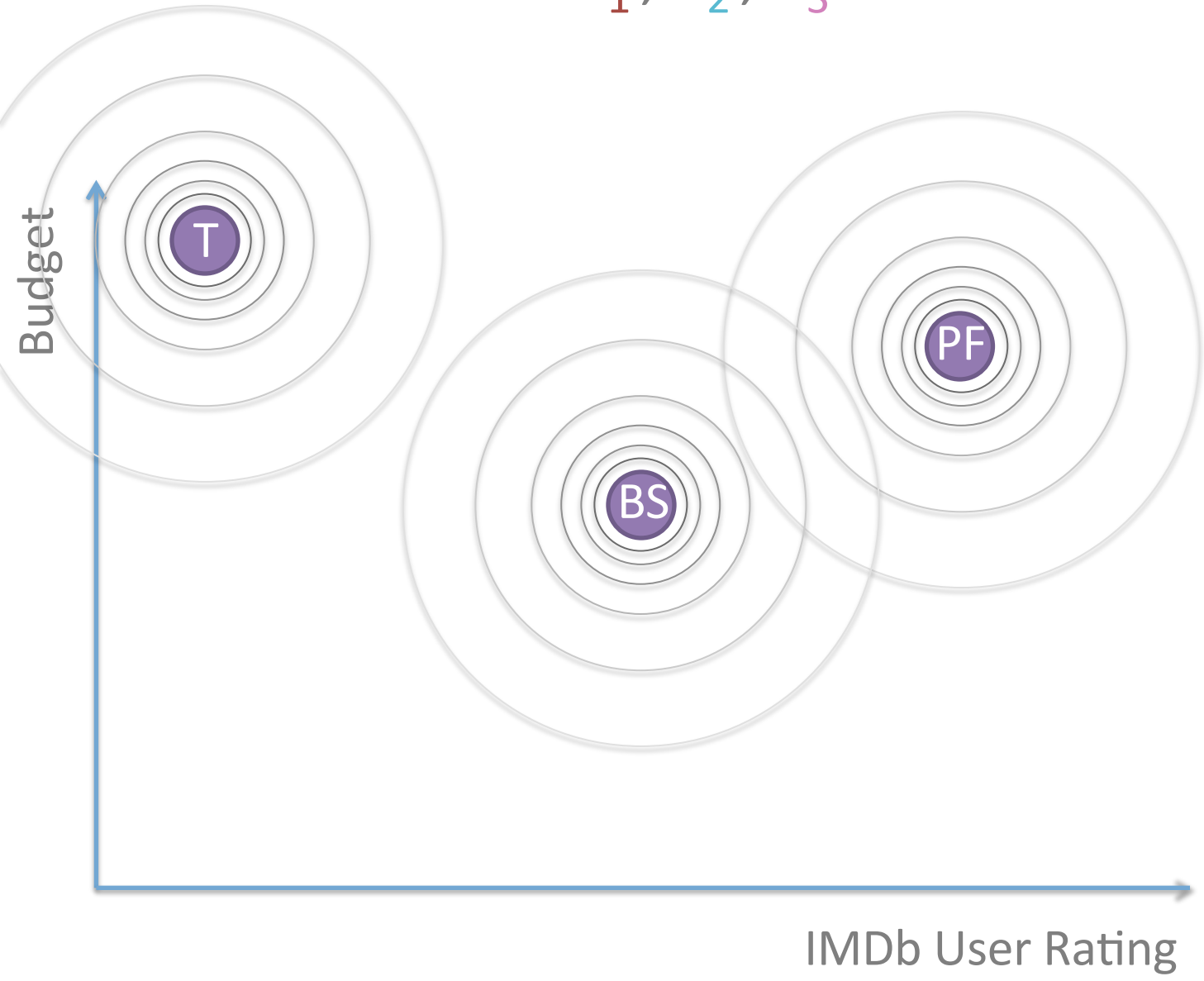
BS

$$a_3(\overrightarrow{x^{obs}}) = \exp\left(\frac{\sum\left(x_i^{obs} - x_i^{Transformers}\right)^2}{2\sigma^2}\right)$$

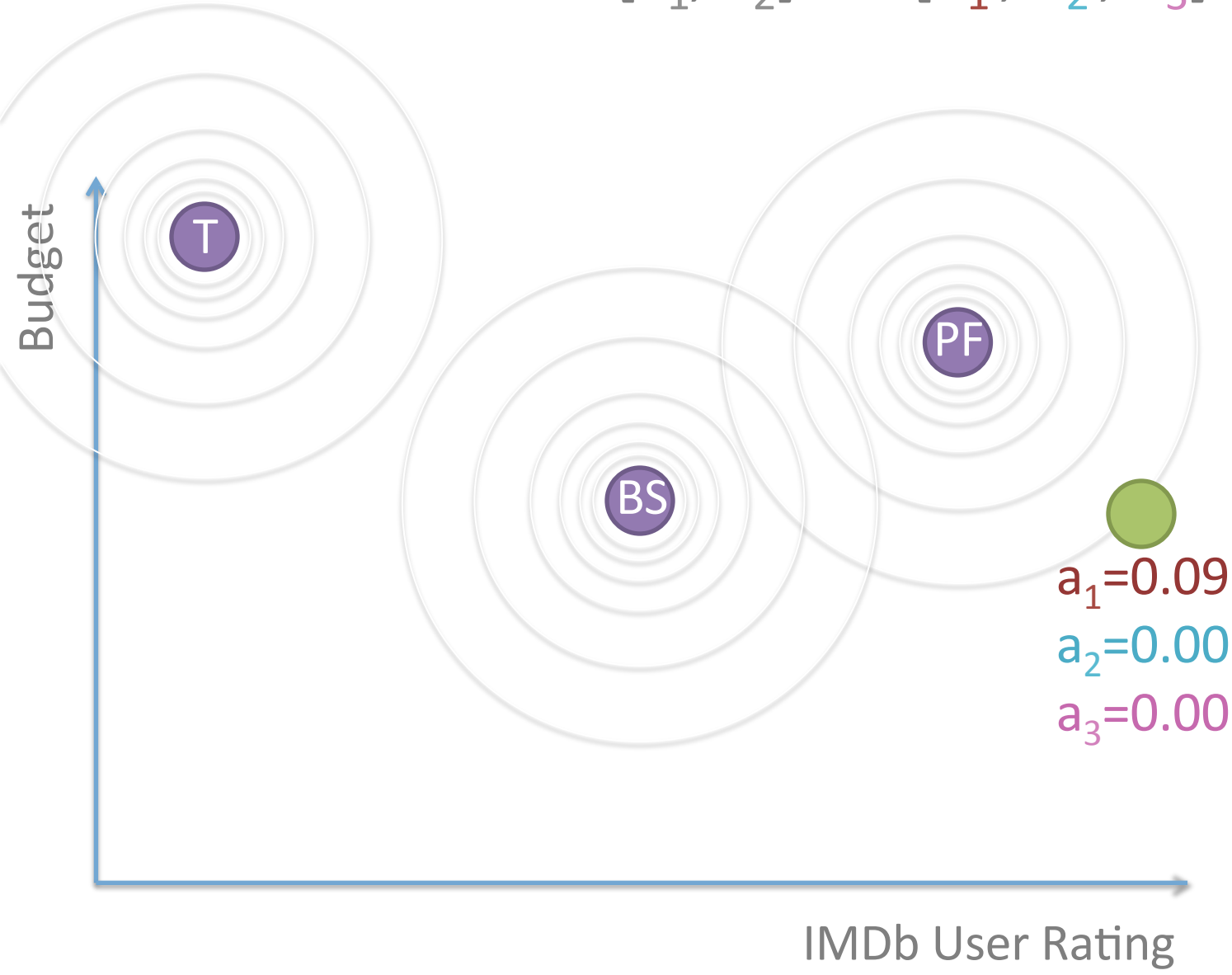
IMDb User Rating



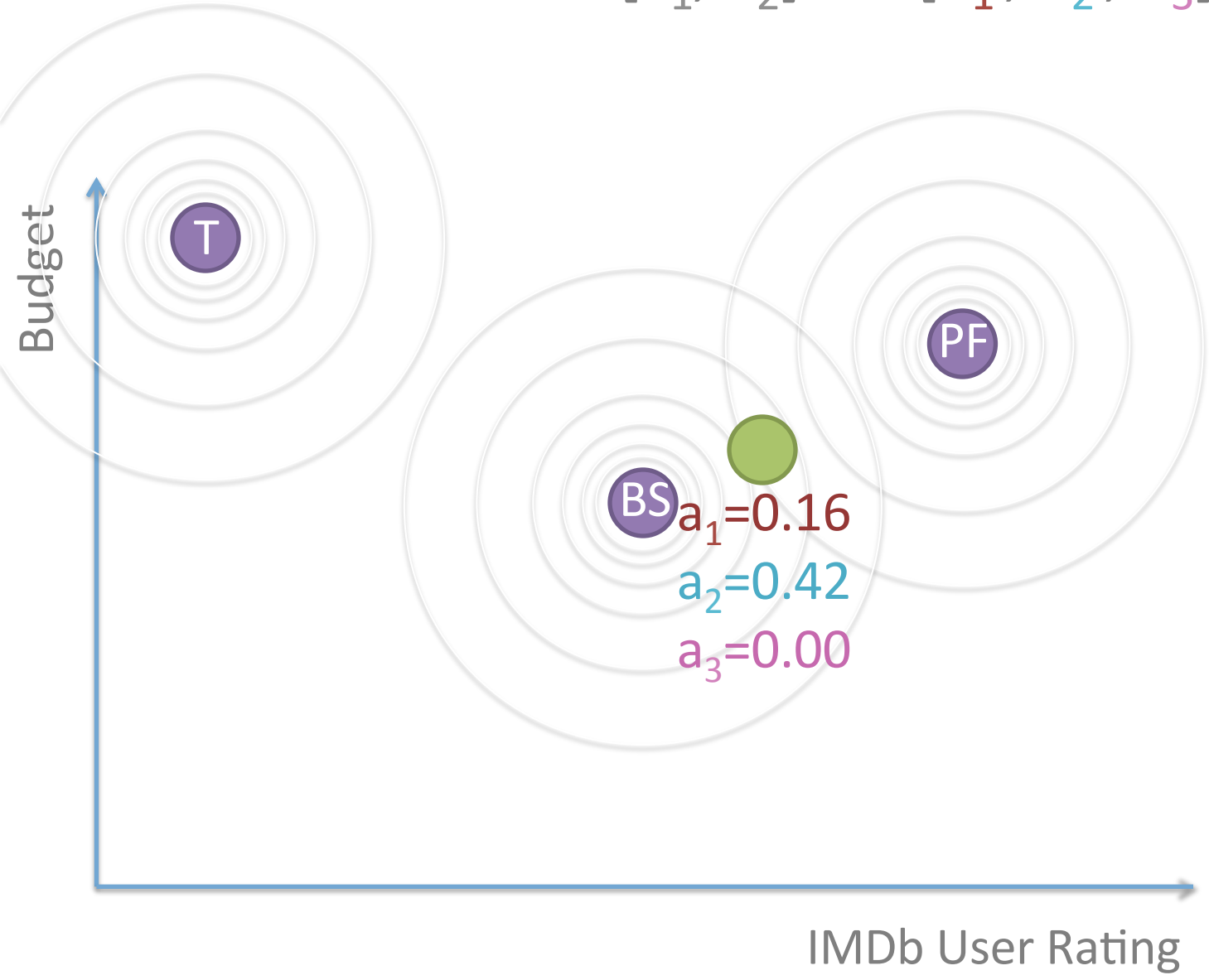
New features: a_1, a_2, a_3



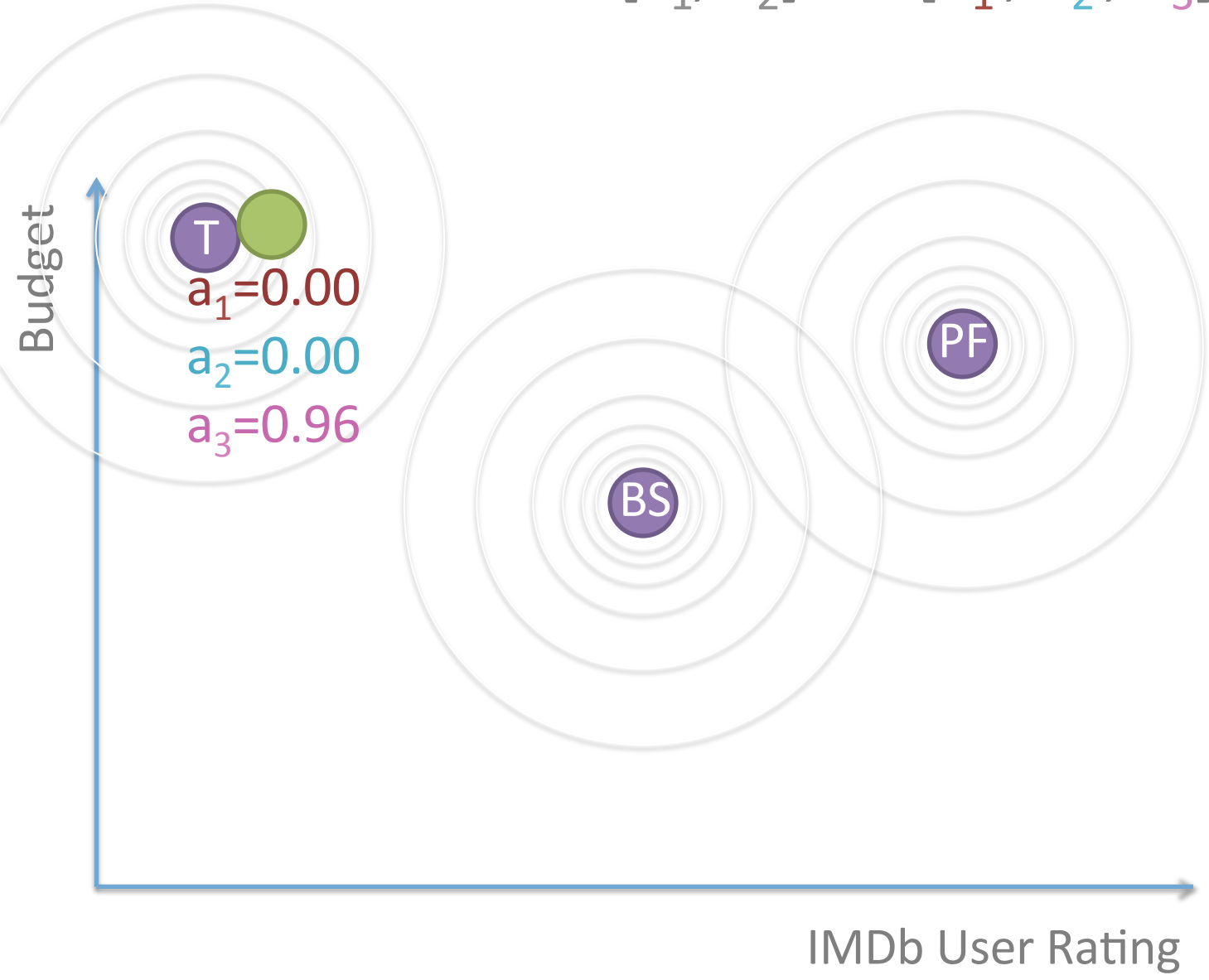
Transformation: $[x_1, x_2] \rightarrow [a_1, a_2, a_3]$



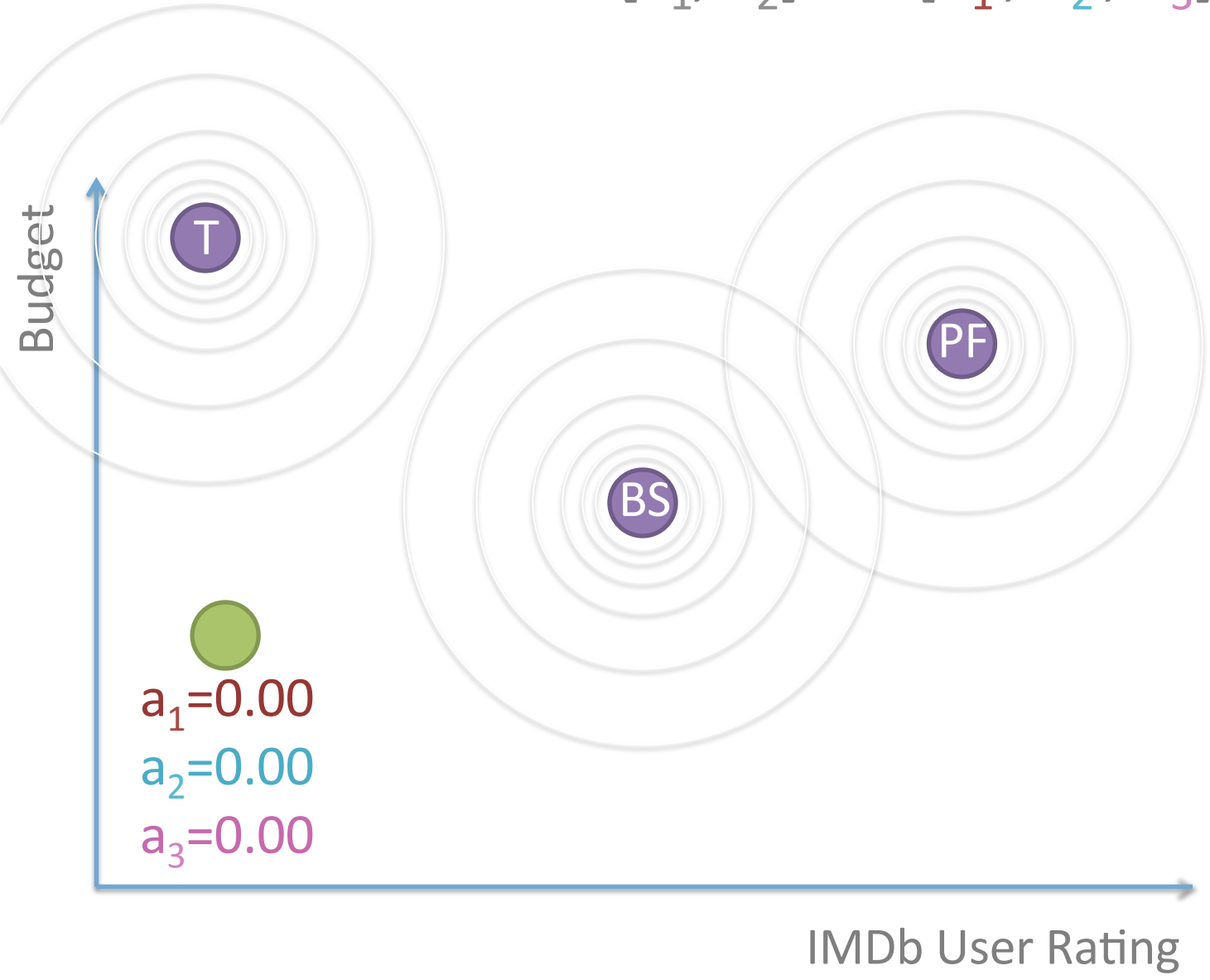
Transformation: $[x_1, x_2] \rightarrow [a_1, a_2, a_3]$



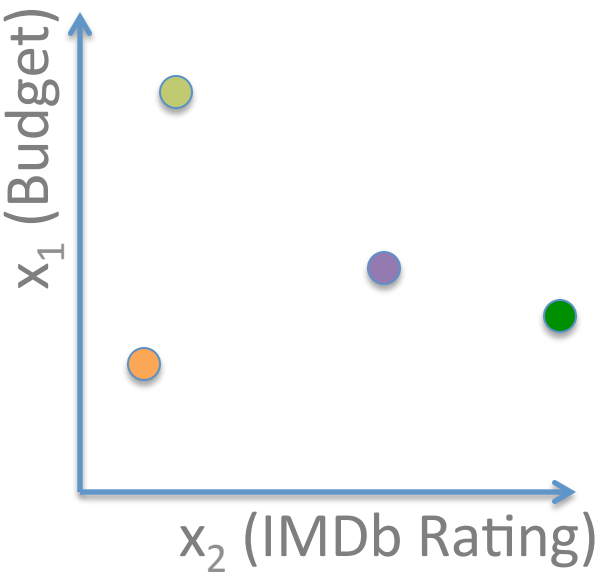
Transformation: $[x_1, x_2] \rightarrow [a_1, a_2, a_3]$



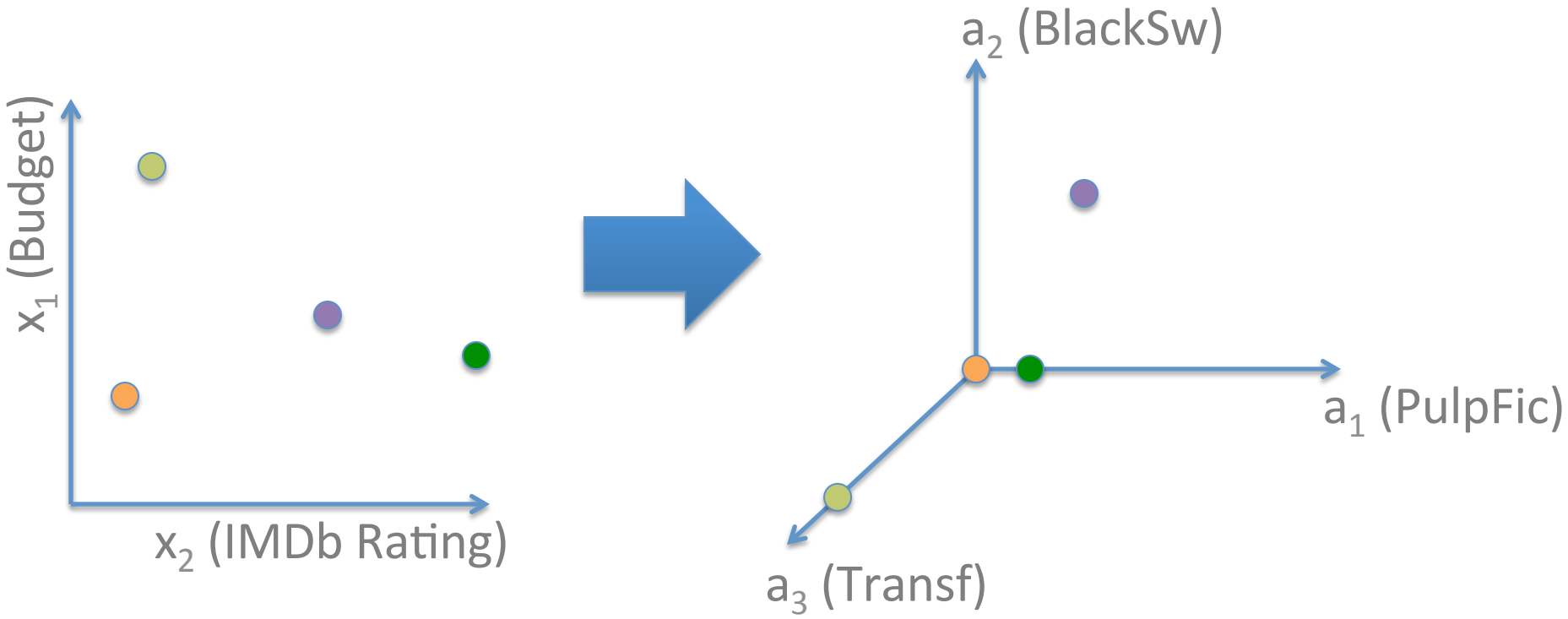
Transformation: $[x_1, x_2] \rightarrow [a_1, a_2, a_3]$



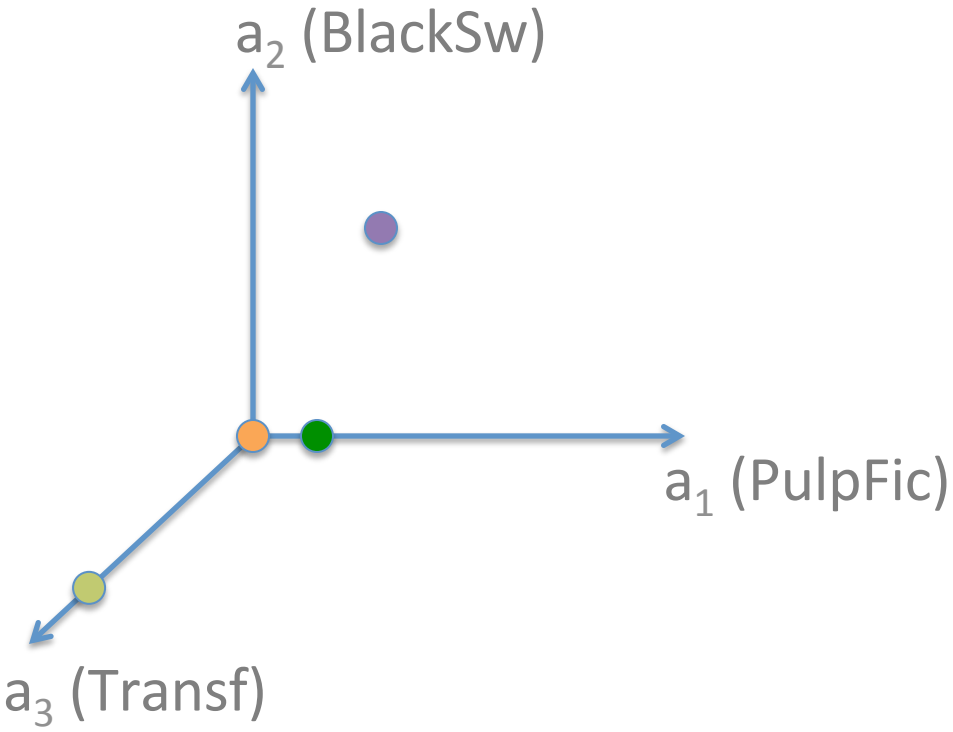
Transformation: $[x_1, x_2] \rightarrow [a_1, a_2, a_3]$



Transformation: $[x_1, x_2] \rightarrow [a_1, a_2, a_3]$

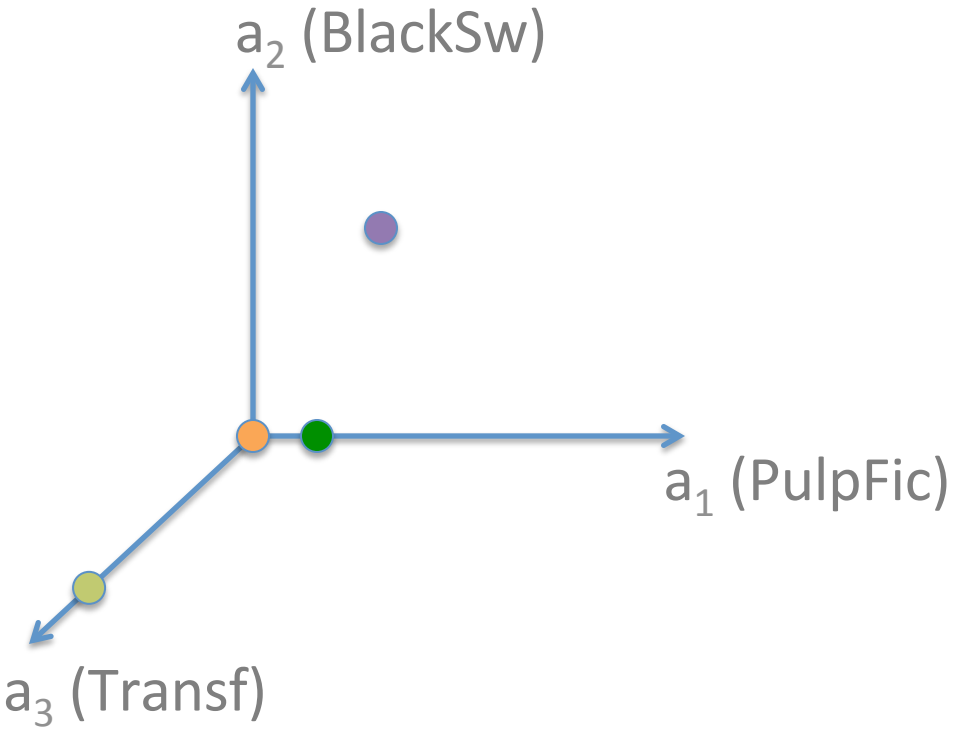


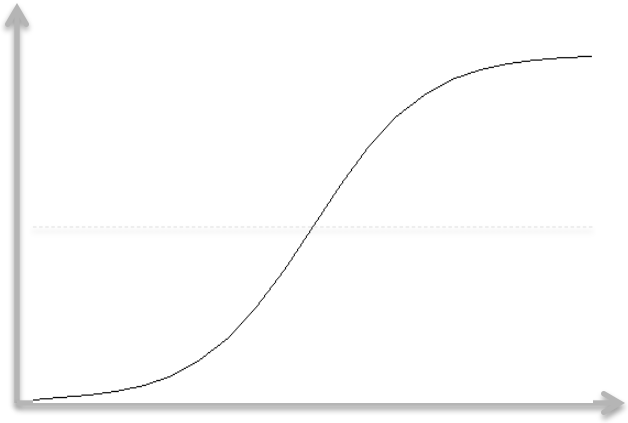
Classification in the new space



Classification in the new space

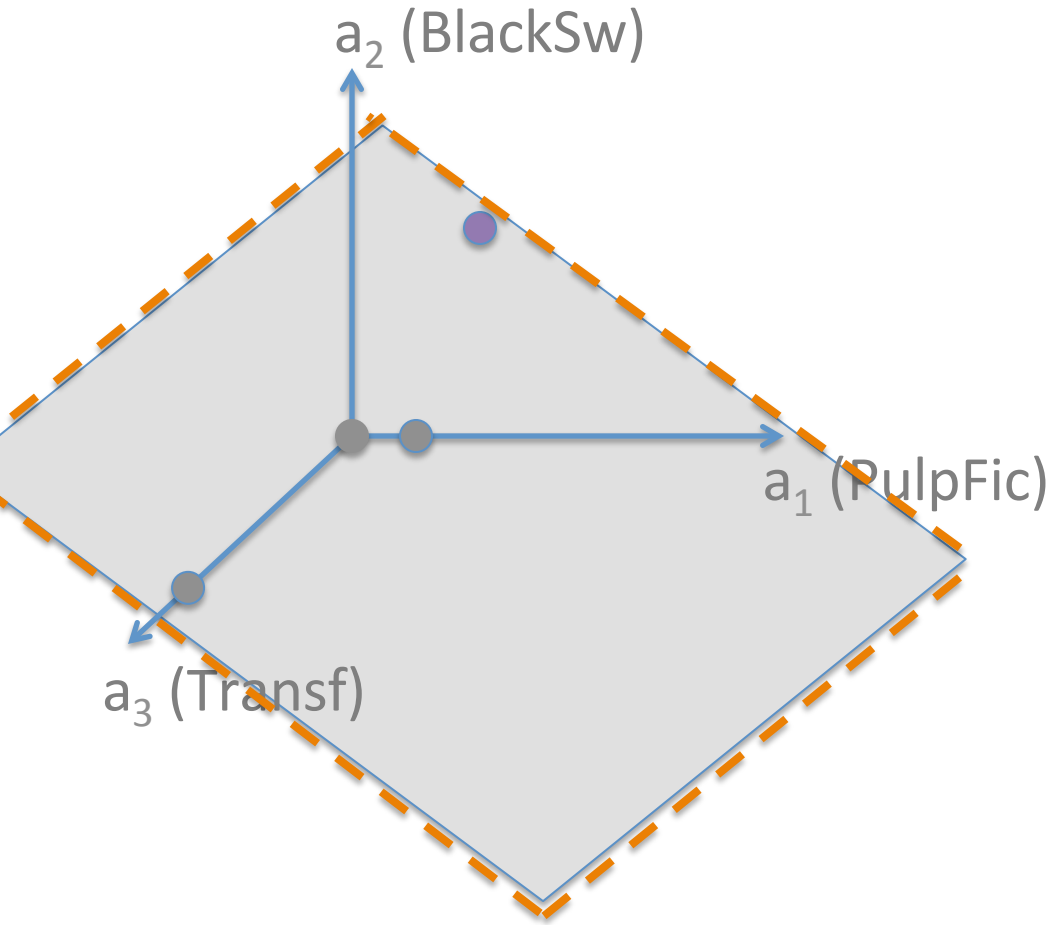
$$y = \frac{1}{1 + e^{-\beta_0 + \beta_1 a_1 + \beta_2 a_2 + \beta_3 a_3}}$$

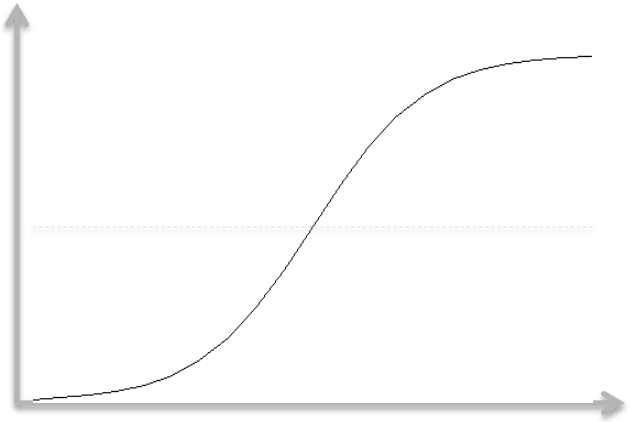




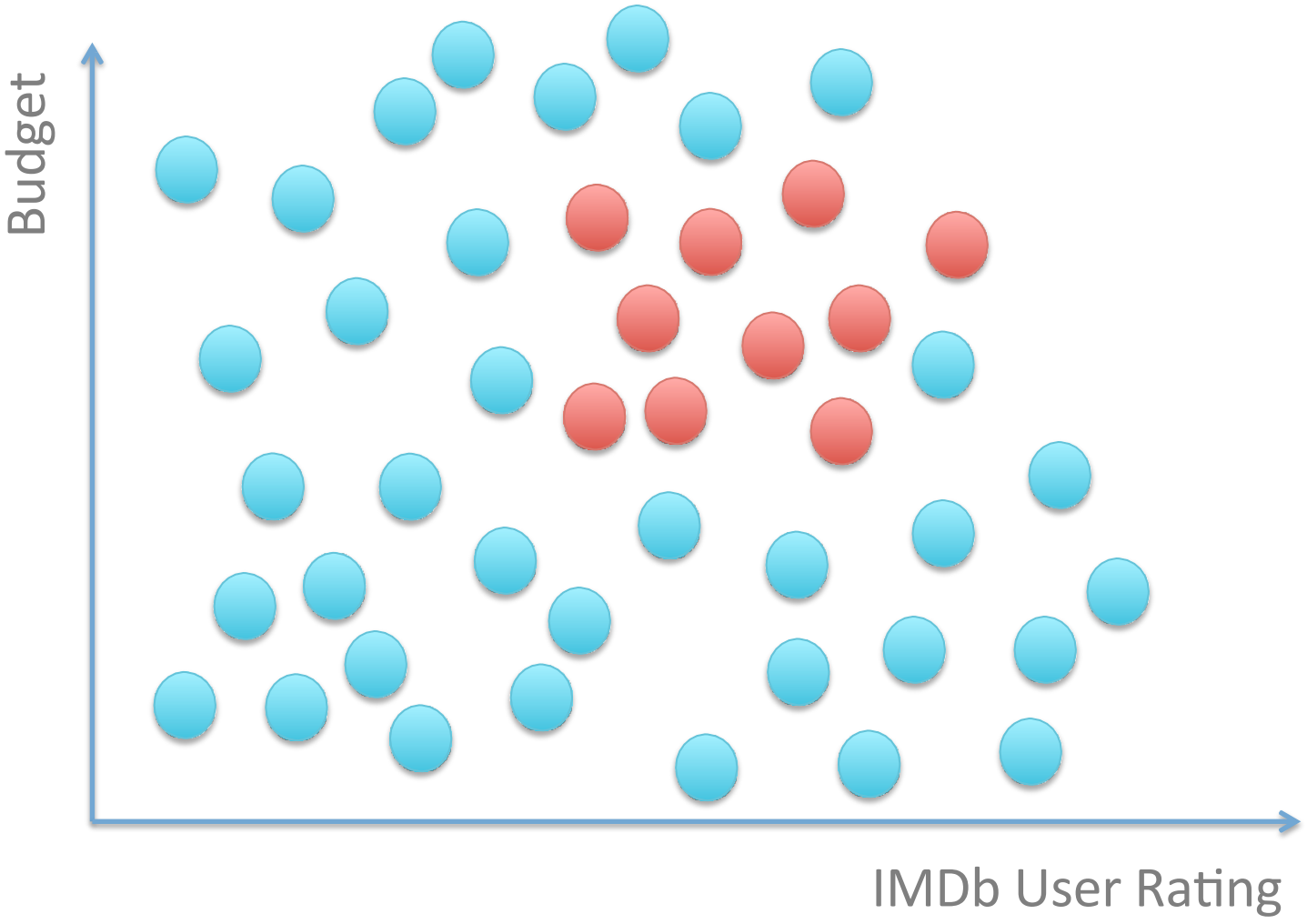
Classification in the new space

$$y = \frac{1}{1 + e^{-\beta_0 + \beta_1 a_1 + \beta_2 a_2 + \beta_3 a_3}}$$

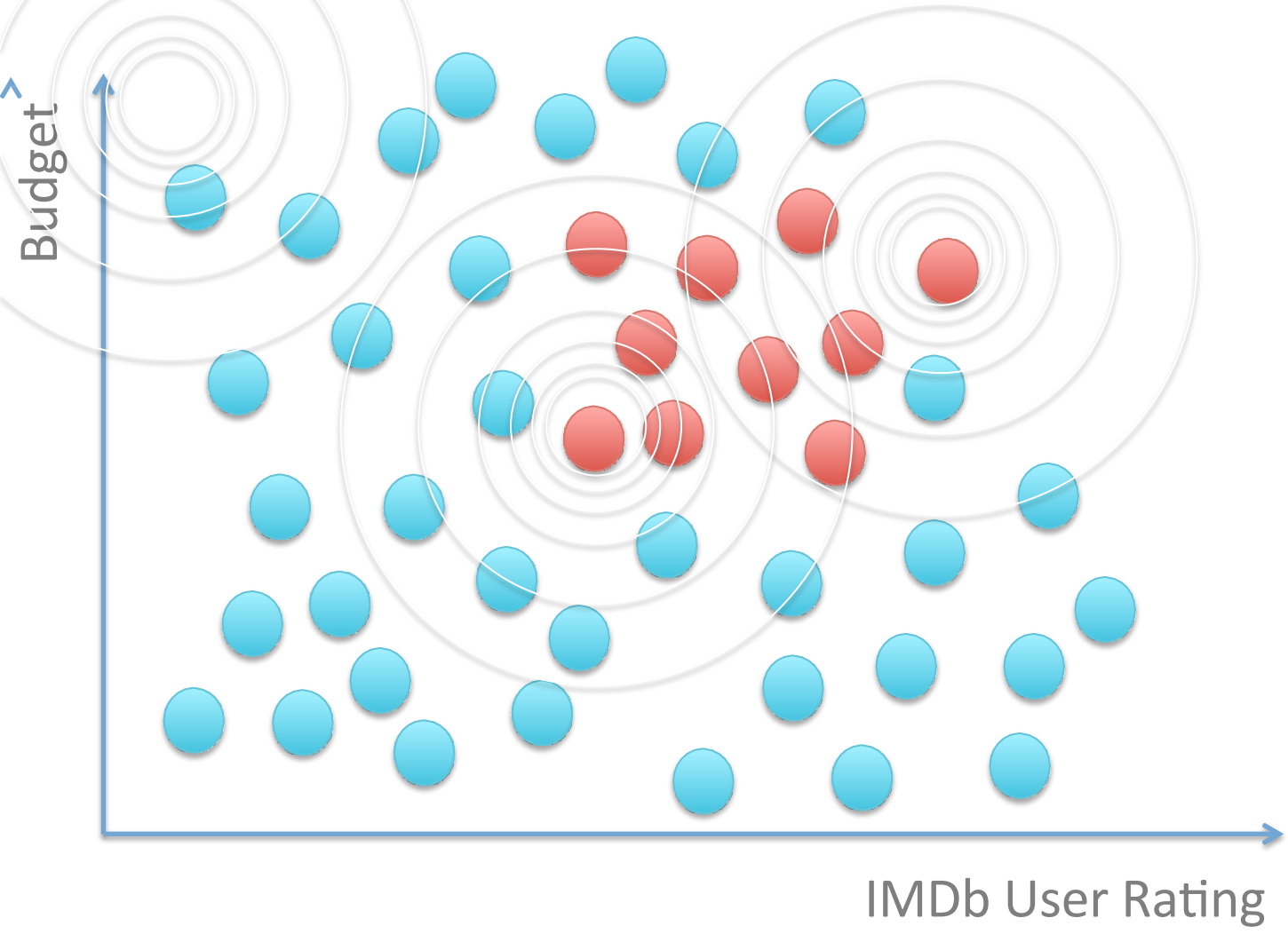




Palme d'Or Winners at Cannes

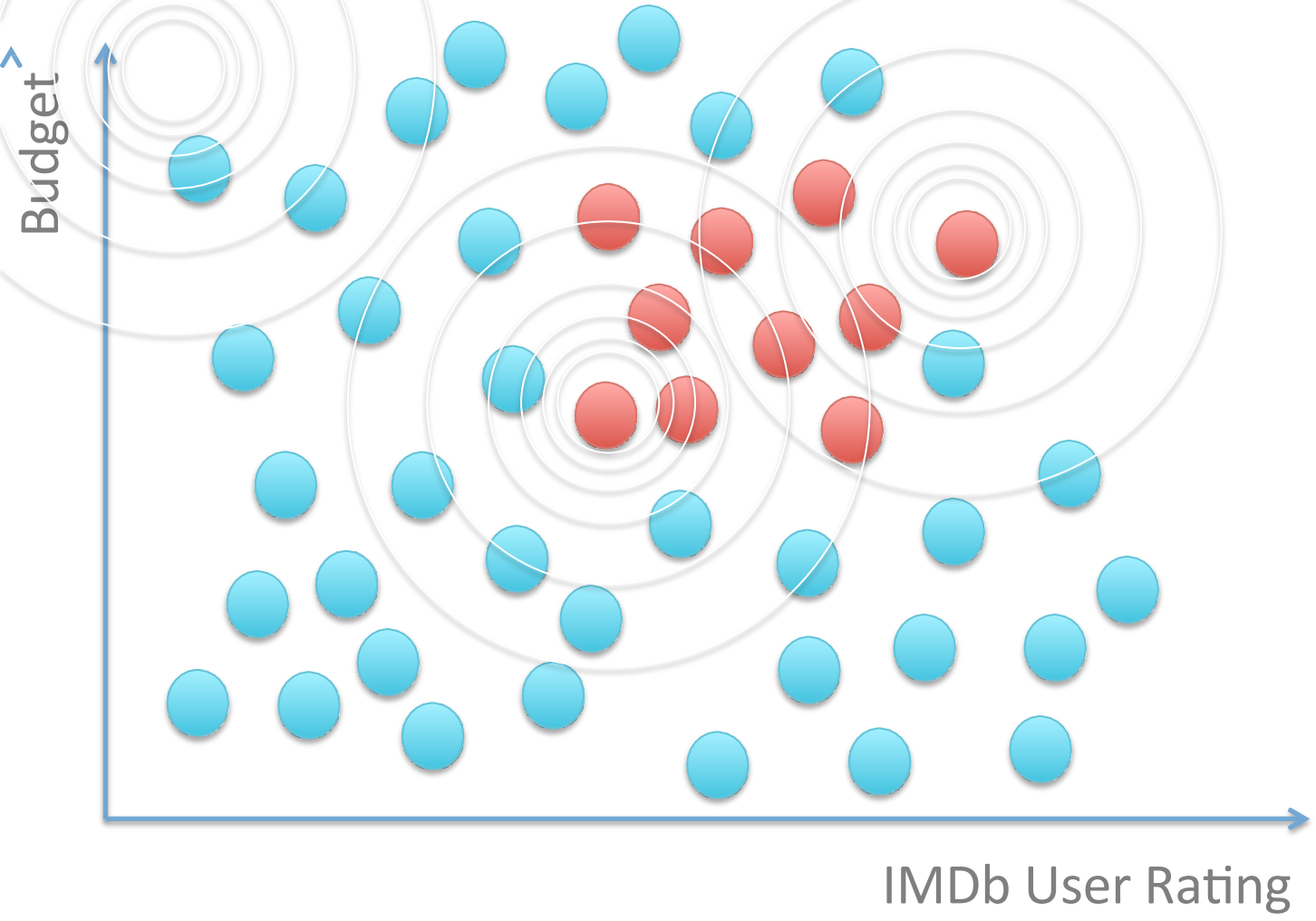


Palme d'Or Winners at Cannes



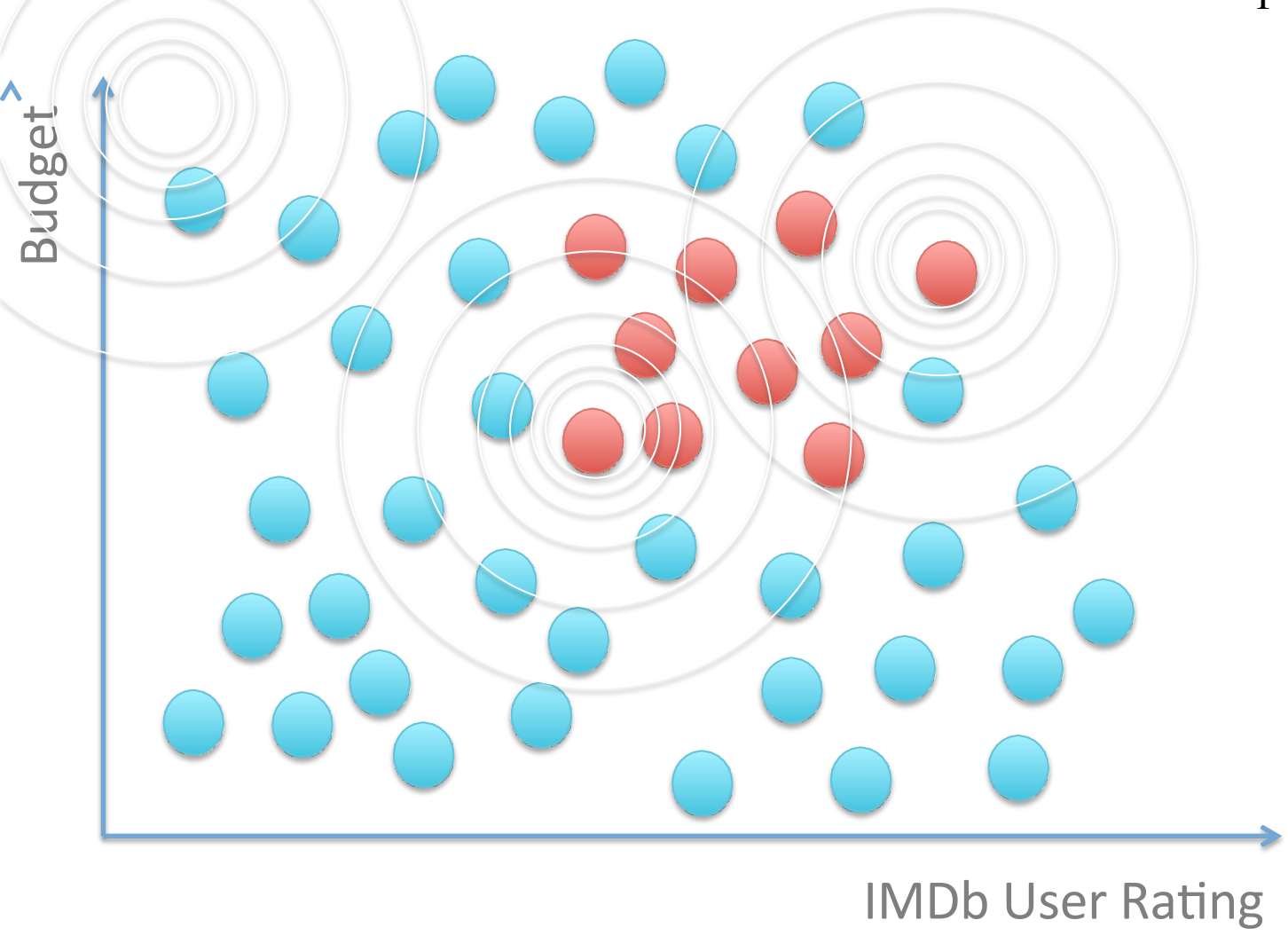
Palme d'Or Winners at Cannes

$$\beta_0 + \beta_1 a_1 + \beta_2 a_2 + \beta_3 a_3$$

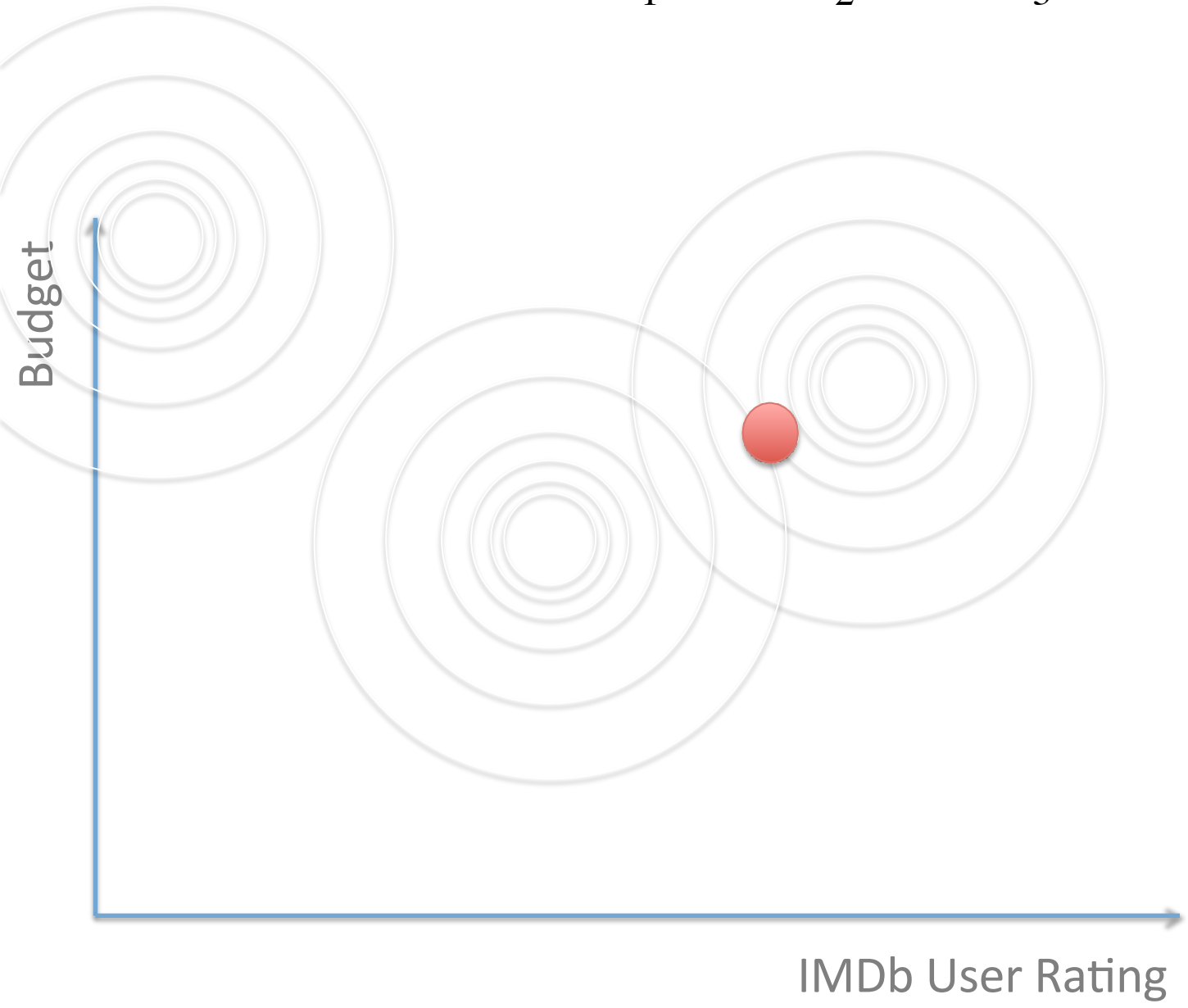


Palme d'Or Winners at Cannes

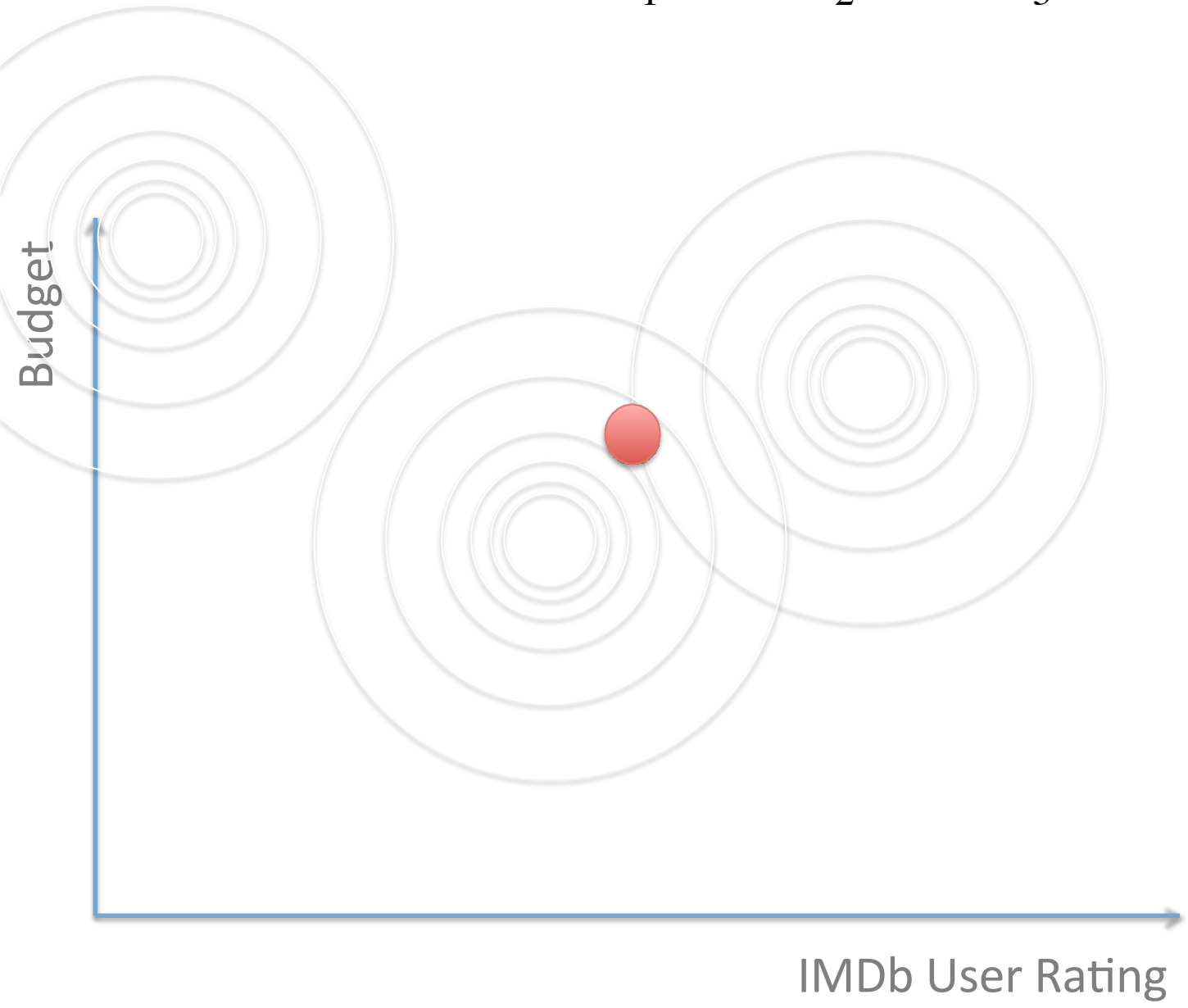
$$0 + 0.7a_1 + 0.9a_2 - 0.6a_3$$



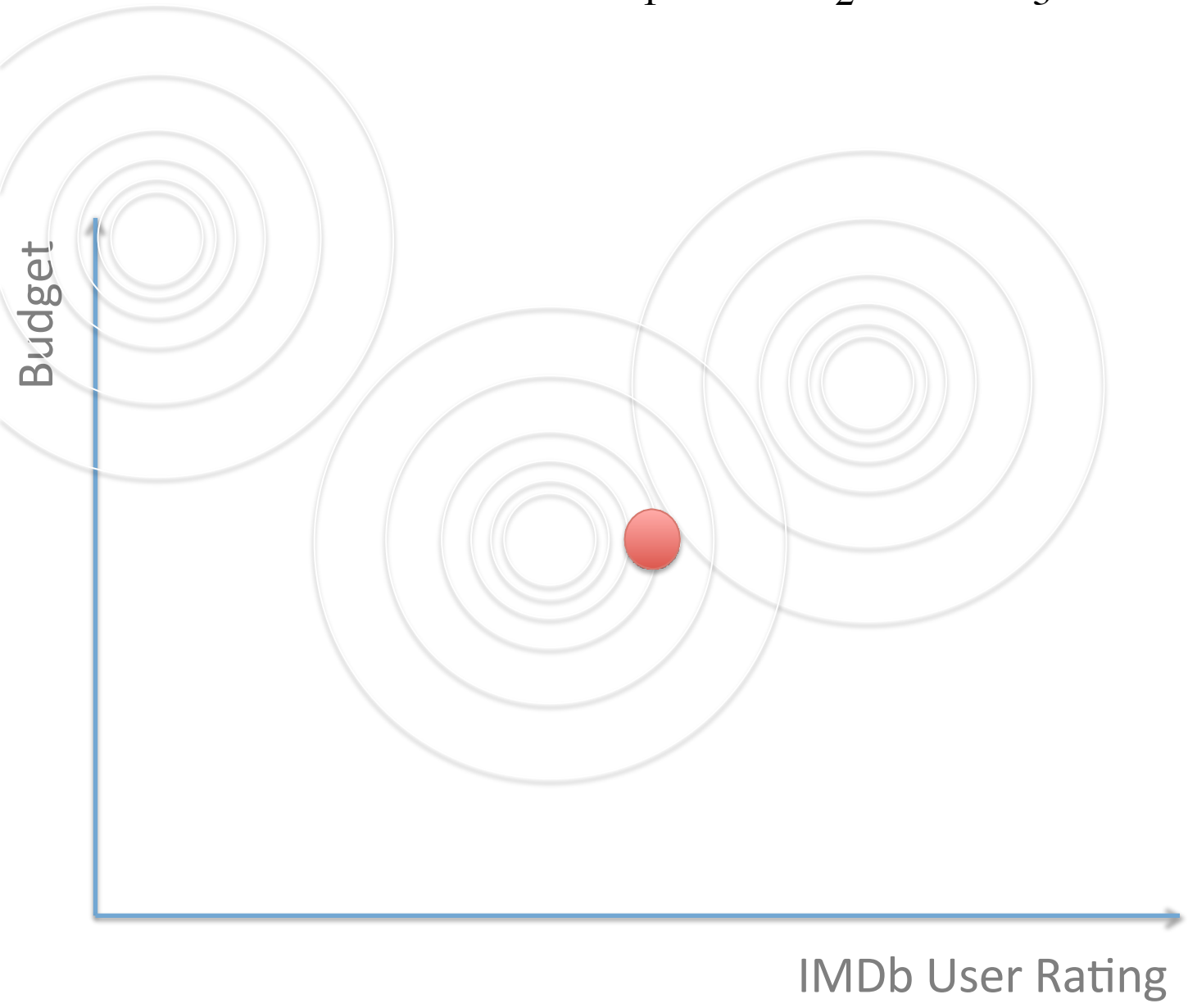
$$0 + 0.7a_1 + 0.9a_2 - 0.6a_3$$



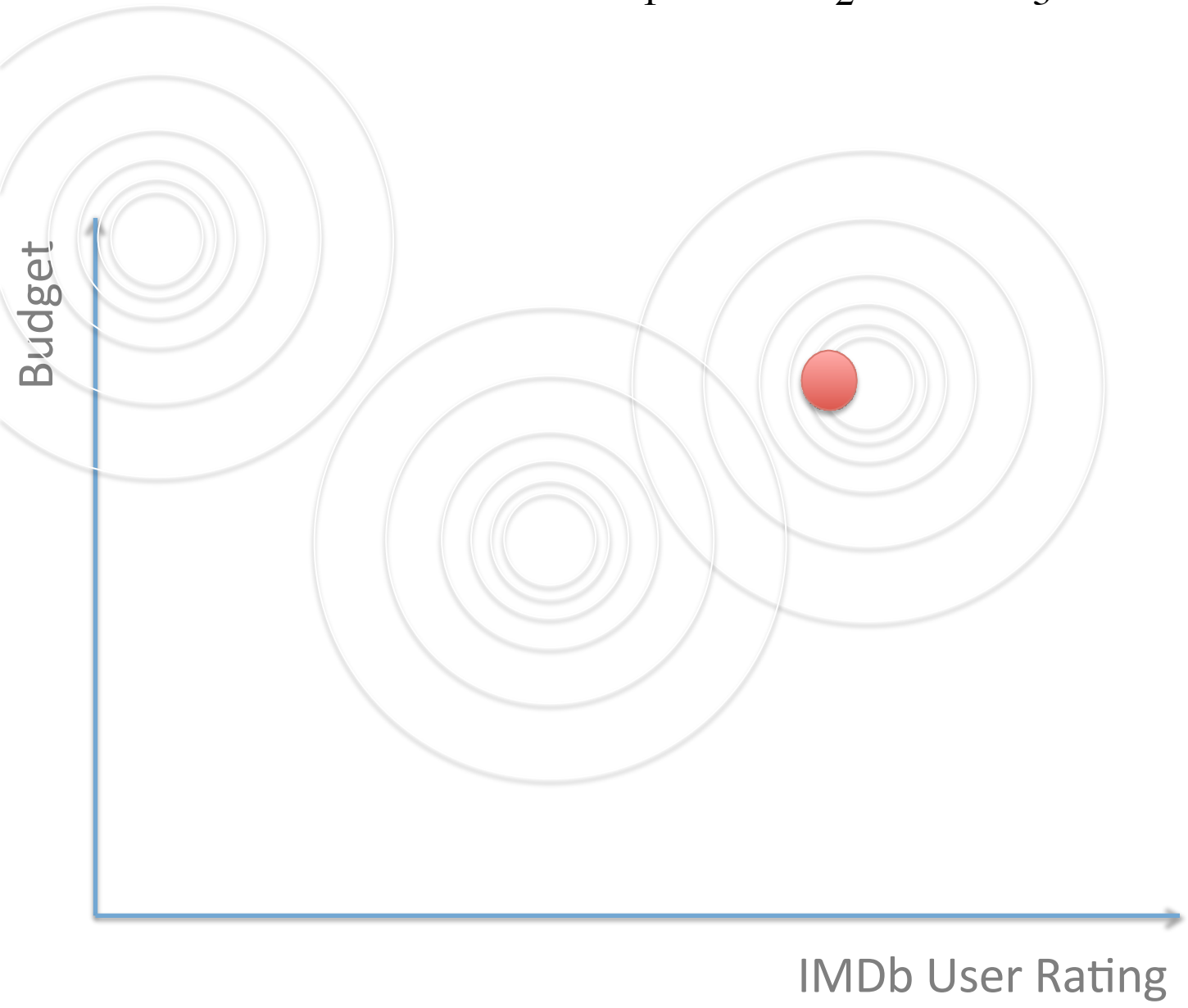
$$0 + 0.7a_1 + 0.9a_2 - 0.6a_3$$



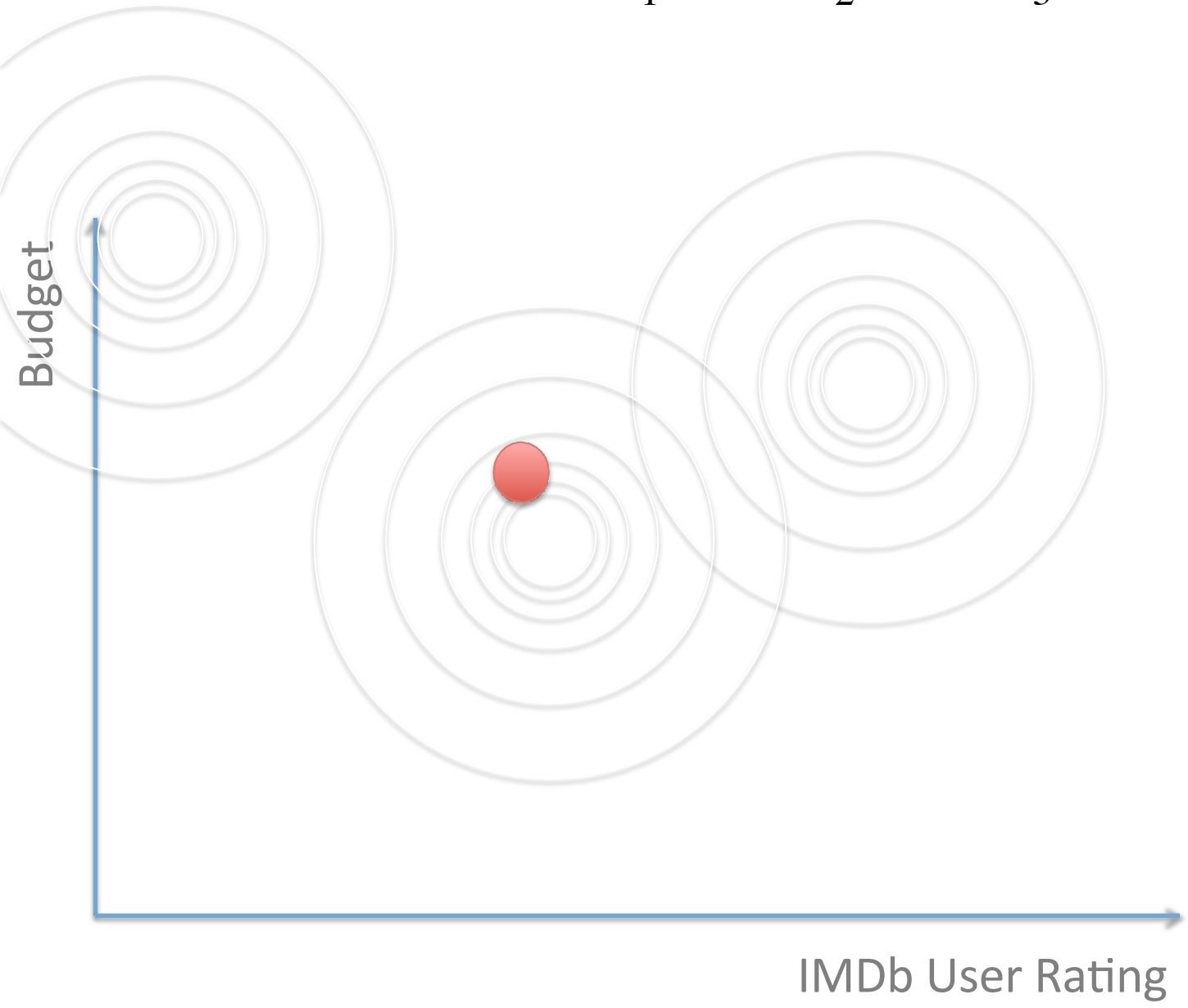
$$0 + 0.7a_1 + 0.9a_2 - 0.6a_3$$



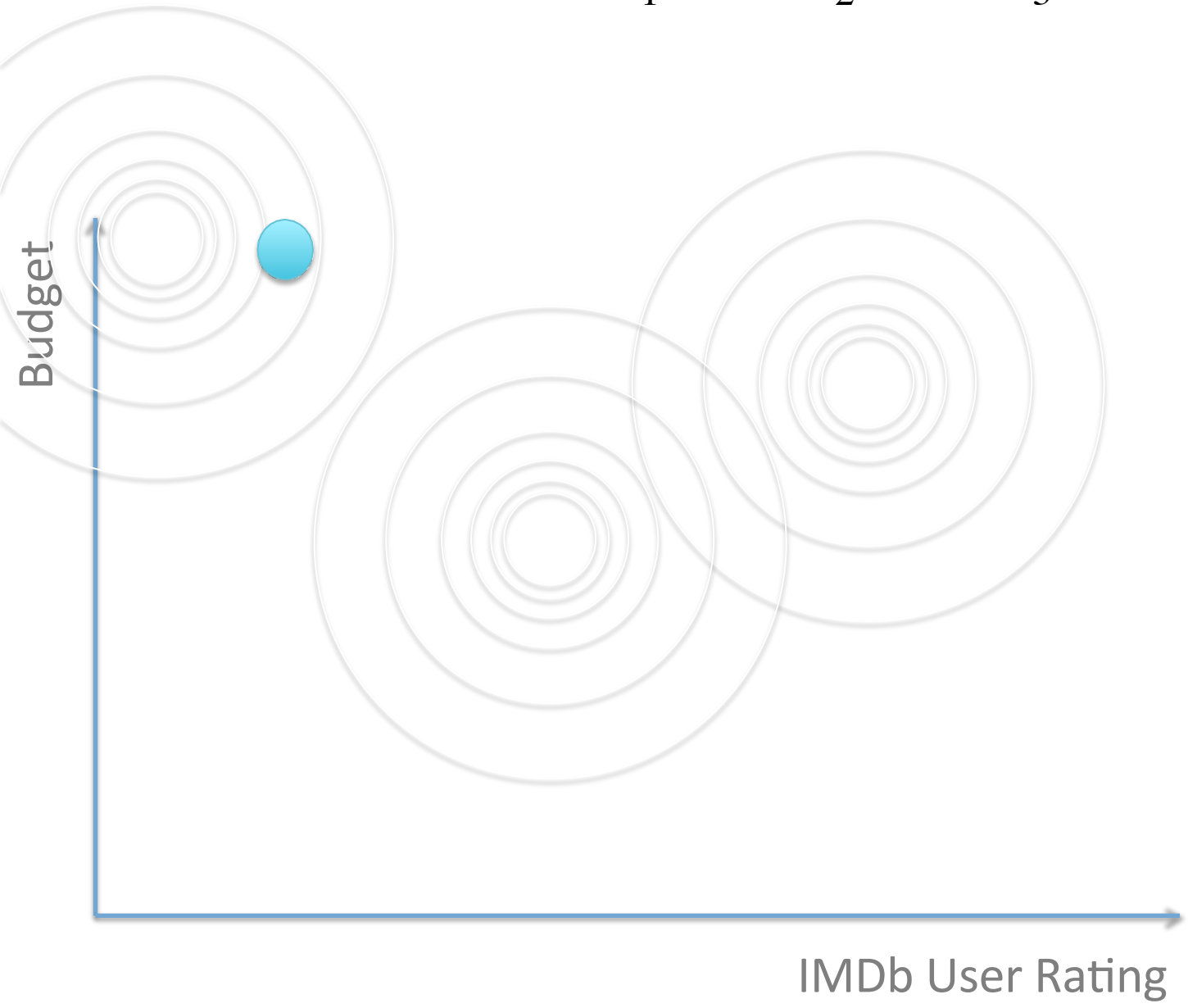
$$0 + 0.7a_1 + 0.9a_2 - 0.6a_3$$



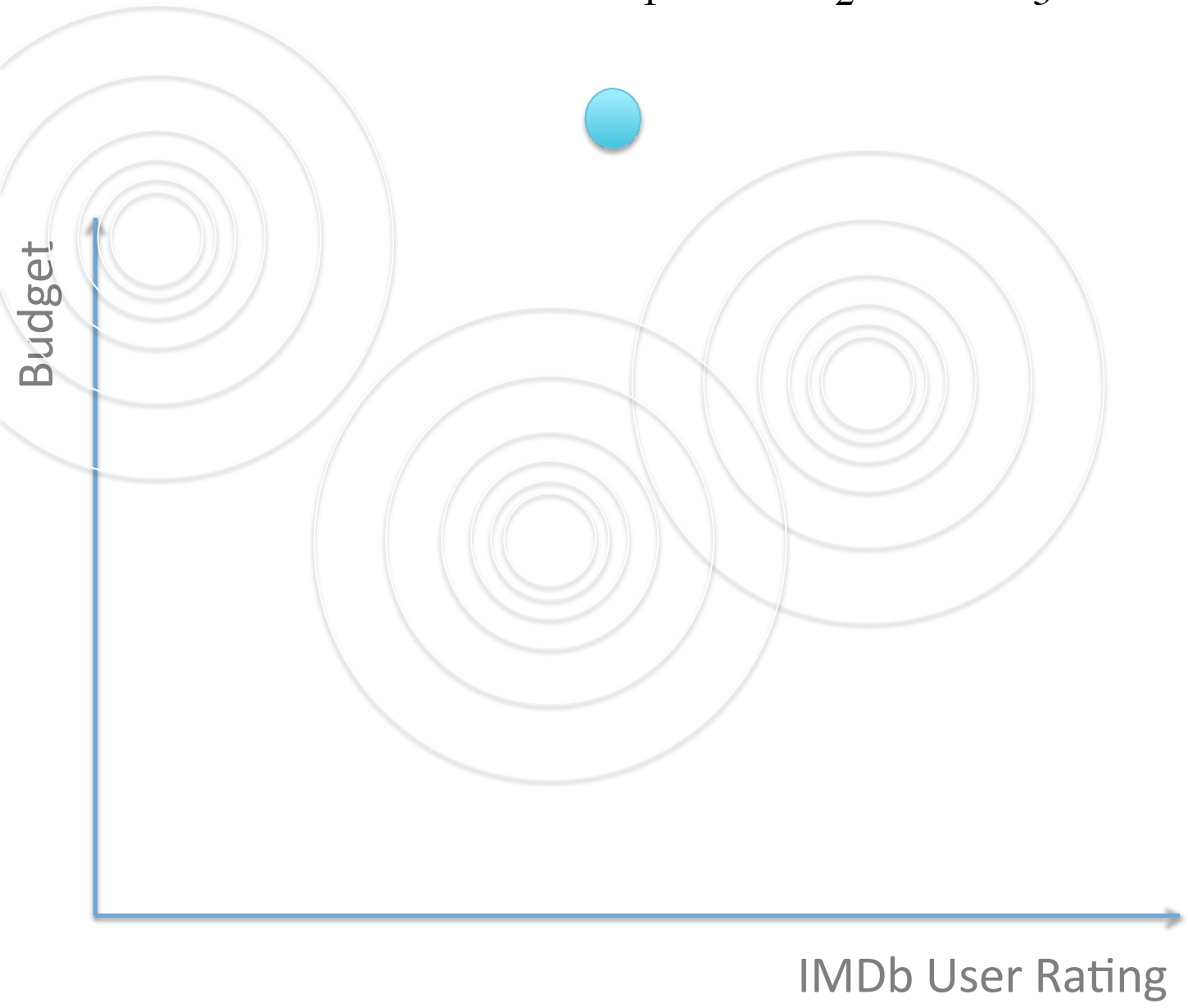
$$0 + 0.7a_1 + 0.9a_2 - 0.6a_3$$



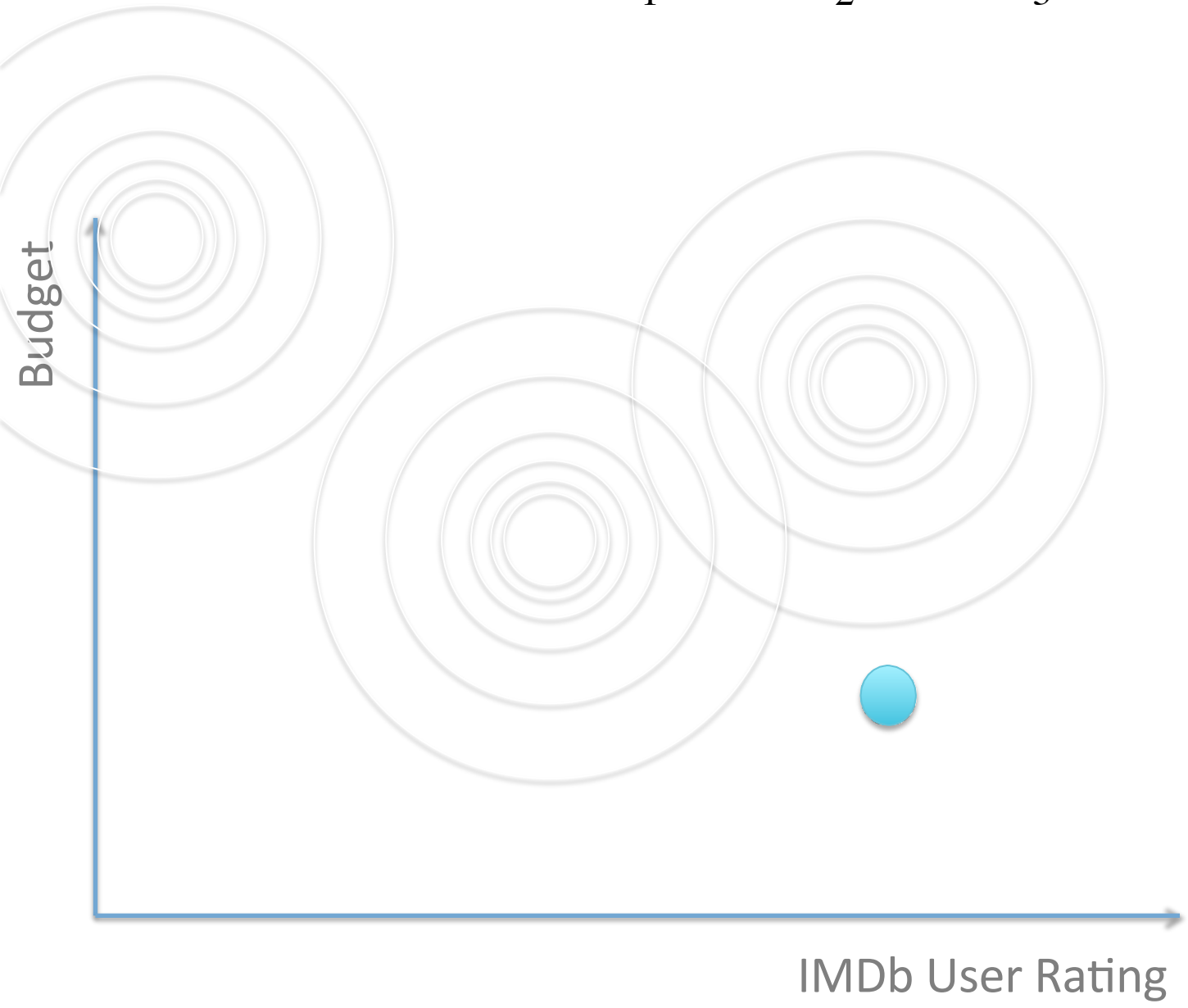
$$0 + 0.7a_1 + 0.9a_2 - 0.6a_3$$



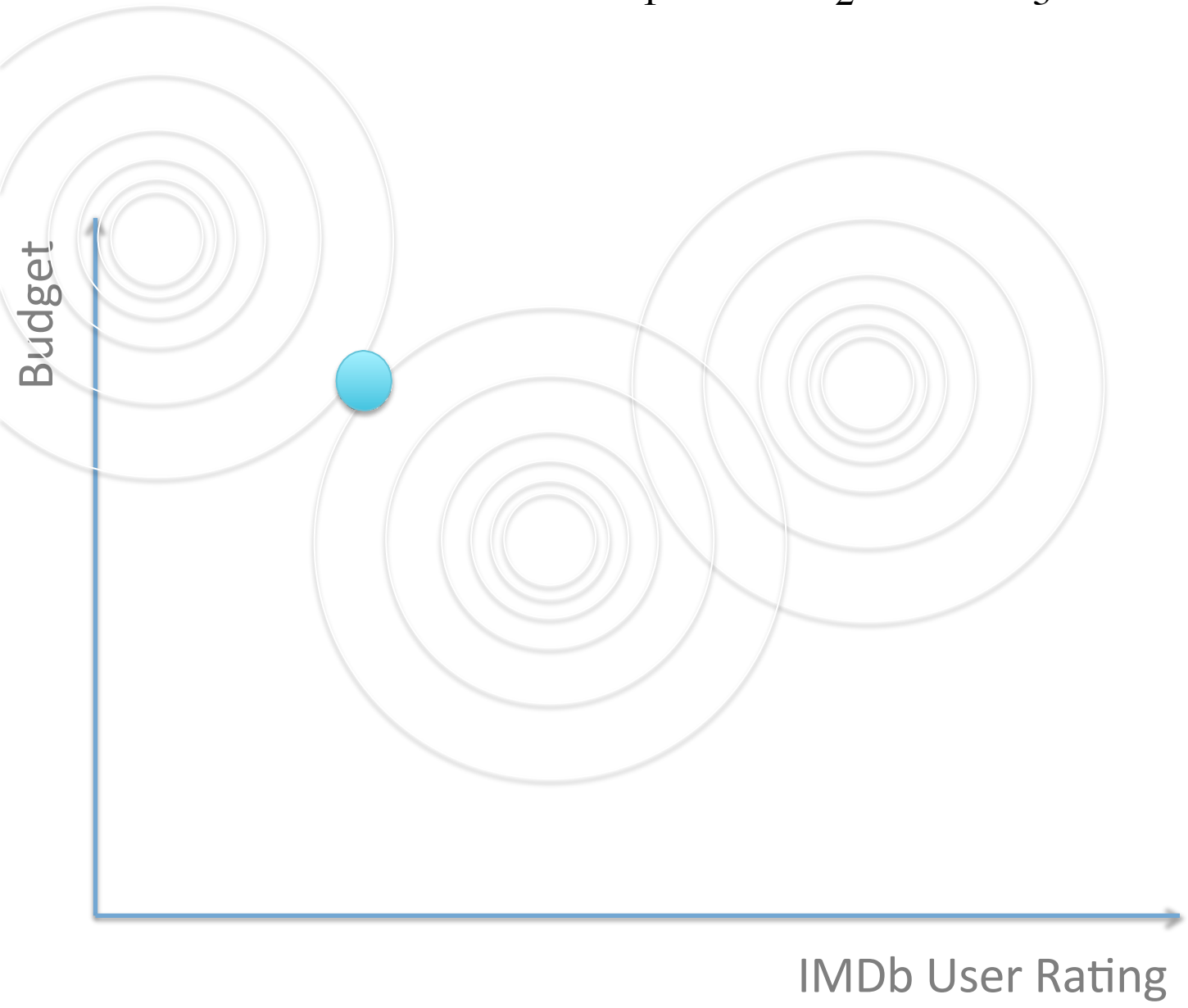
$$0 + 0.7a_1 + 0.9a_2 - 0.6a_3$$



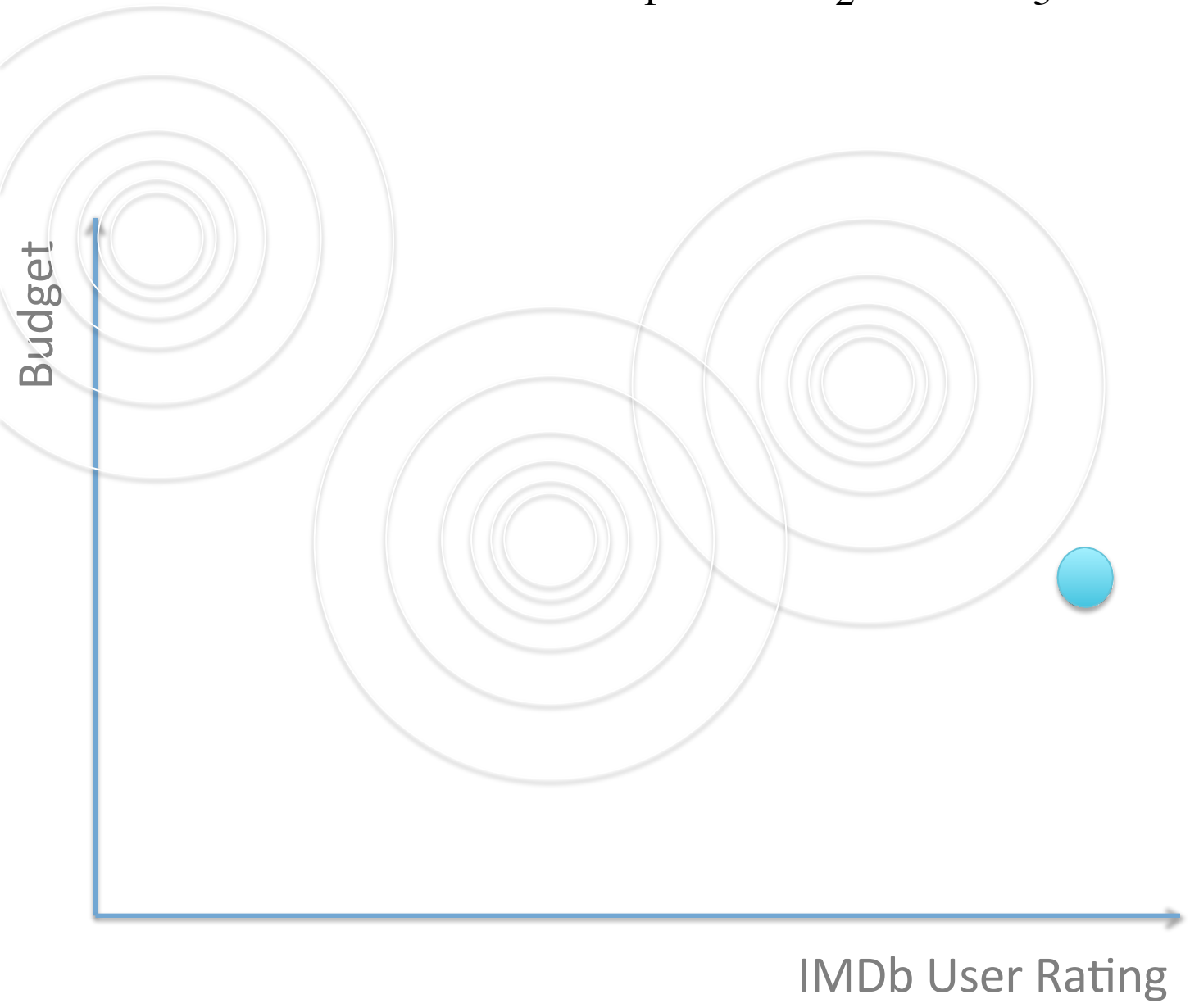
$$0 + 0.7a_1 + 0.9a_2 - 0.6a_3$$



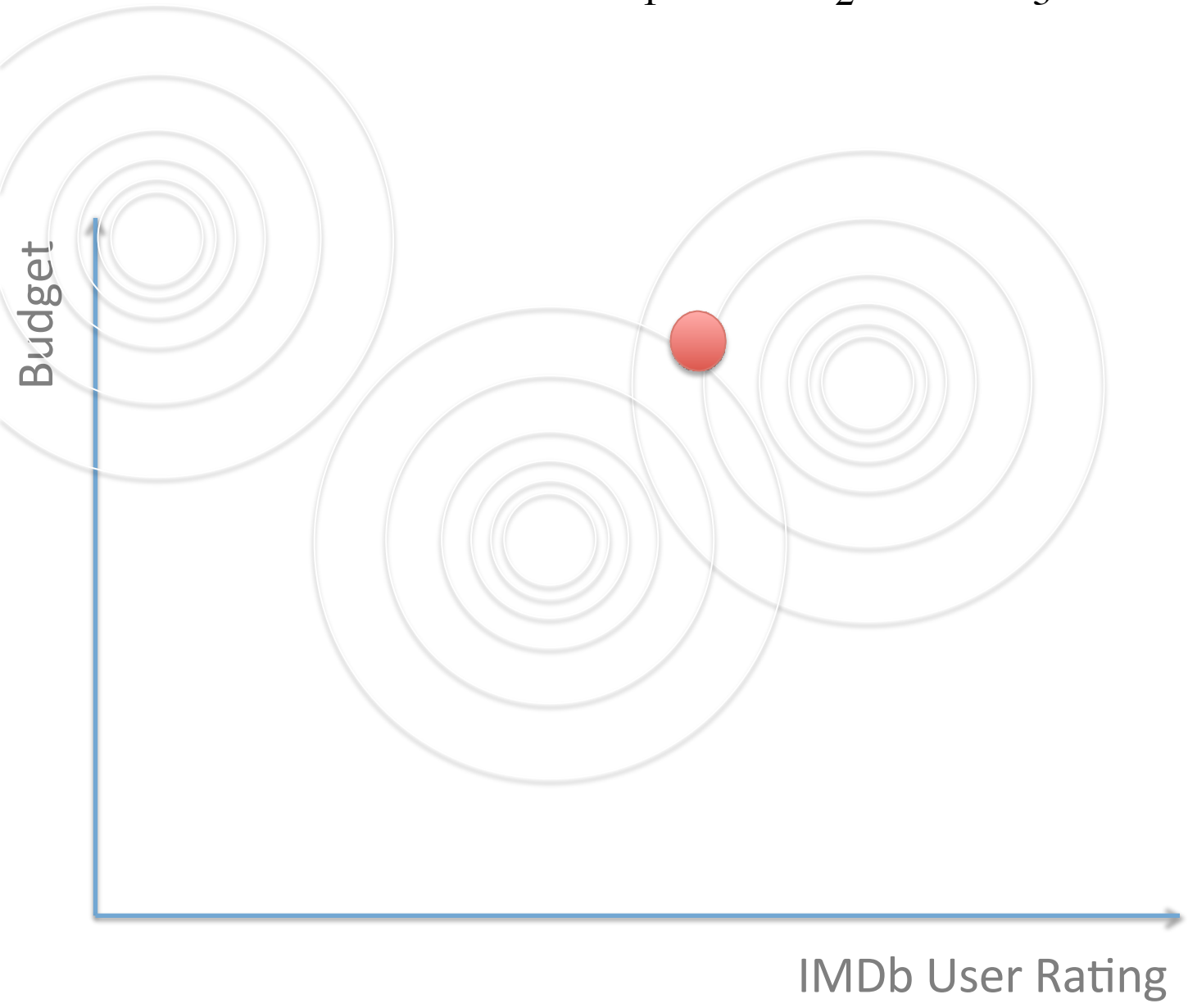
$$0 + 0.7a_1 + 0.9a_2 - 0.6a_3$$



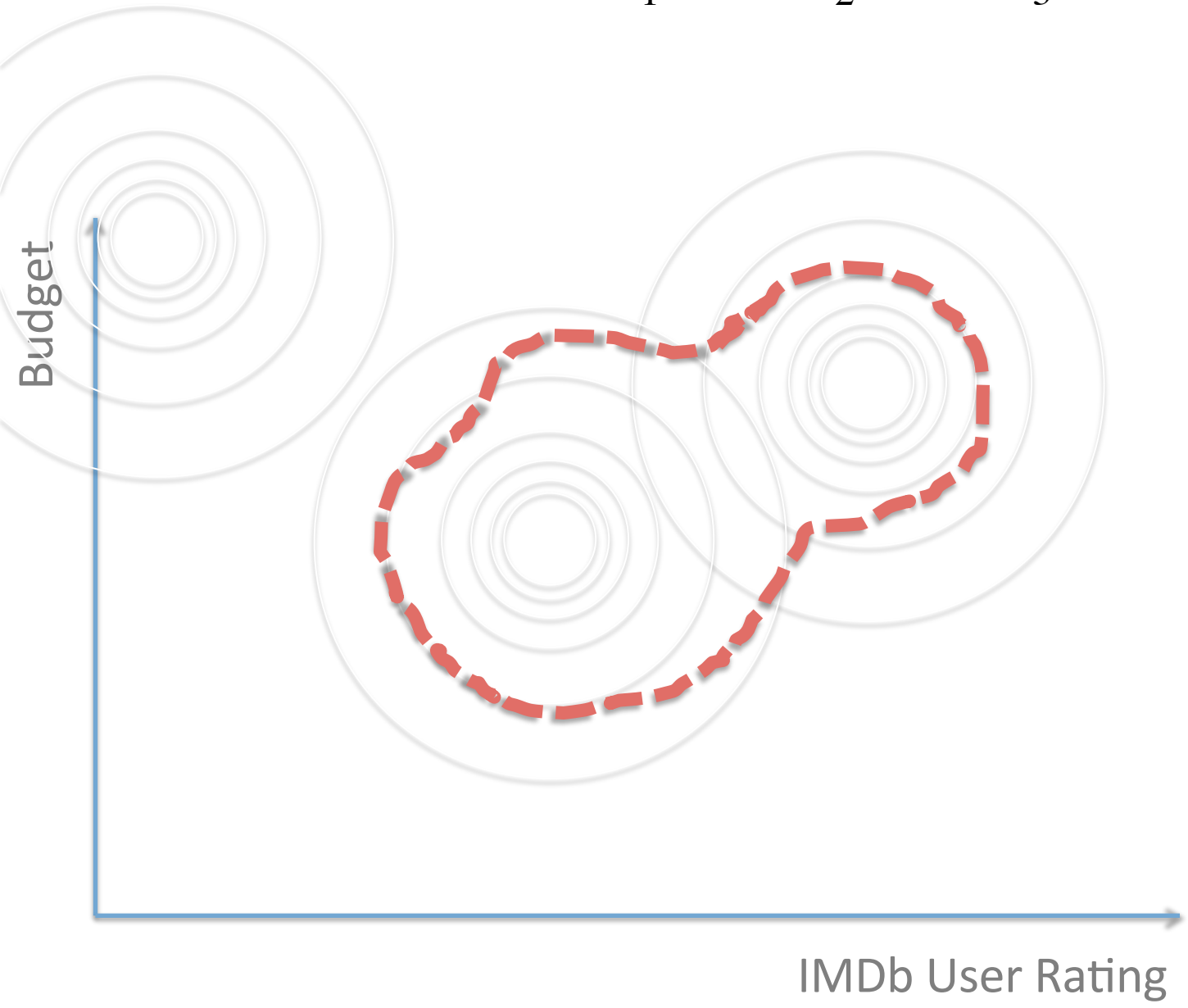
$$0 + 0.7a_1 + 0.9a_2 - 0.6a_3$$



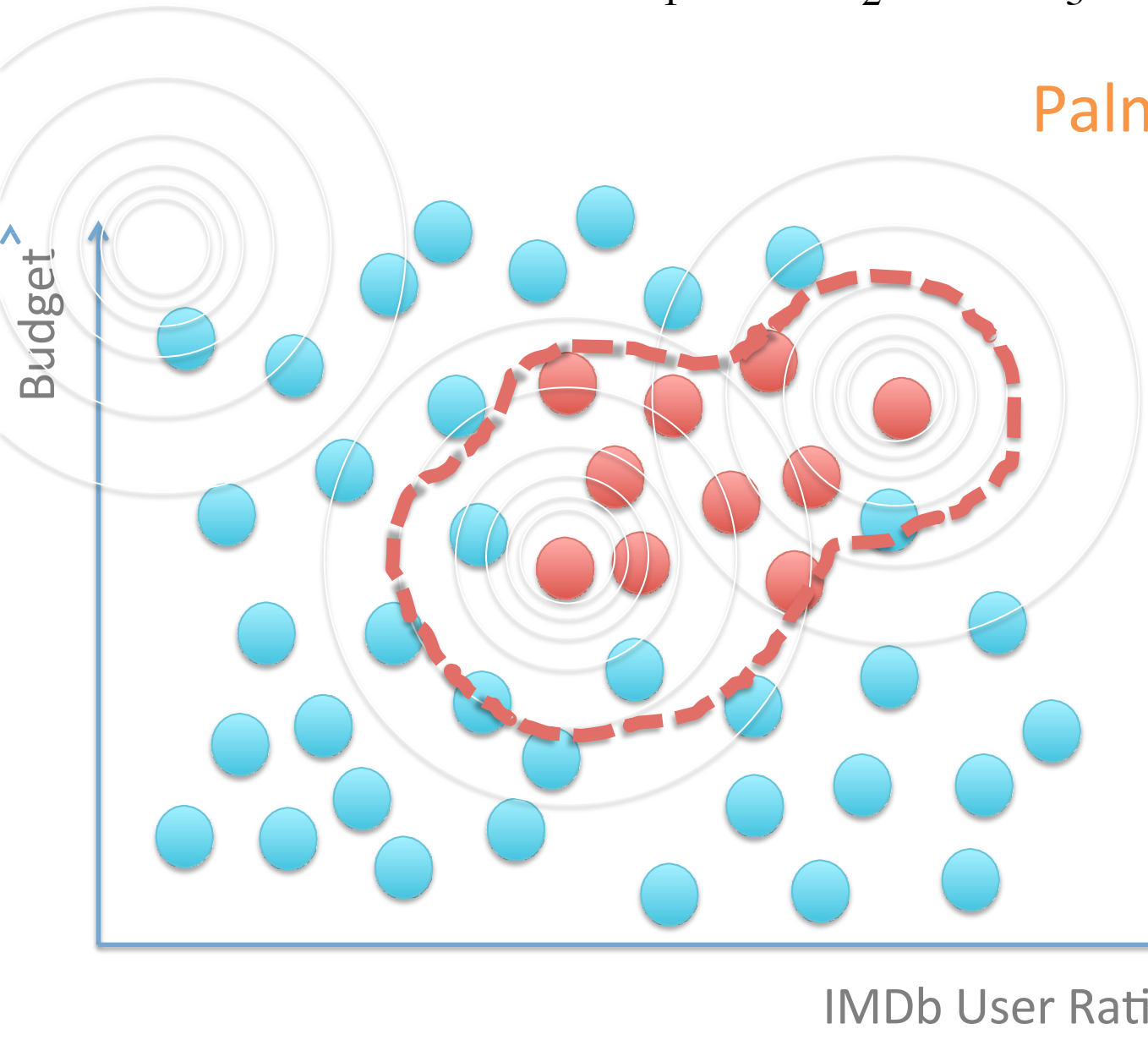
$$0 + 0.7a_1 + 0.9a_2 - 0.6a_3$$



$$0 + 0.7a_1 + 0.9a_2 - 0.6a_3$$



$$0 + 0.7a_1 + 0.9a_2 - 0.6a_3$$

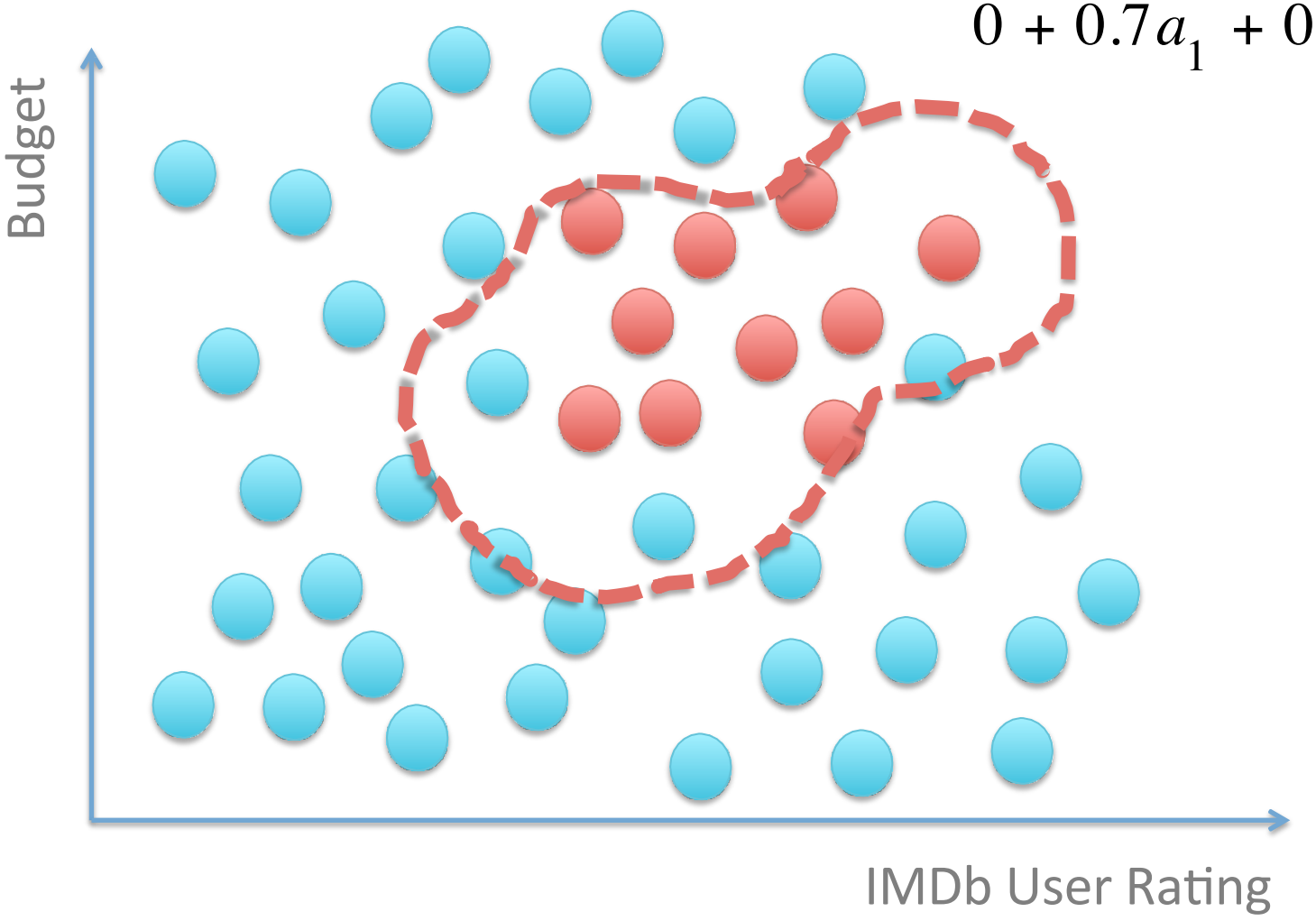


Palme d'Or Winners at Cannes

Palme d'Or Winners at Cannes

Fitting the intricate boundary with the kernel

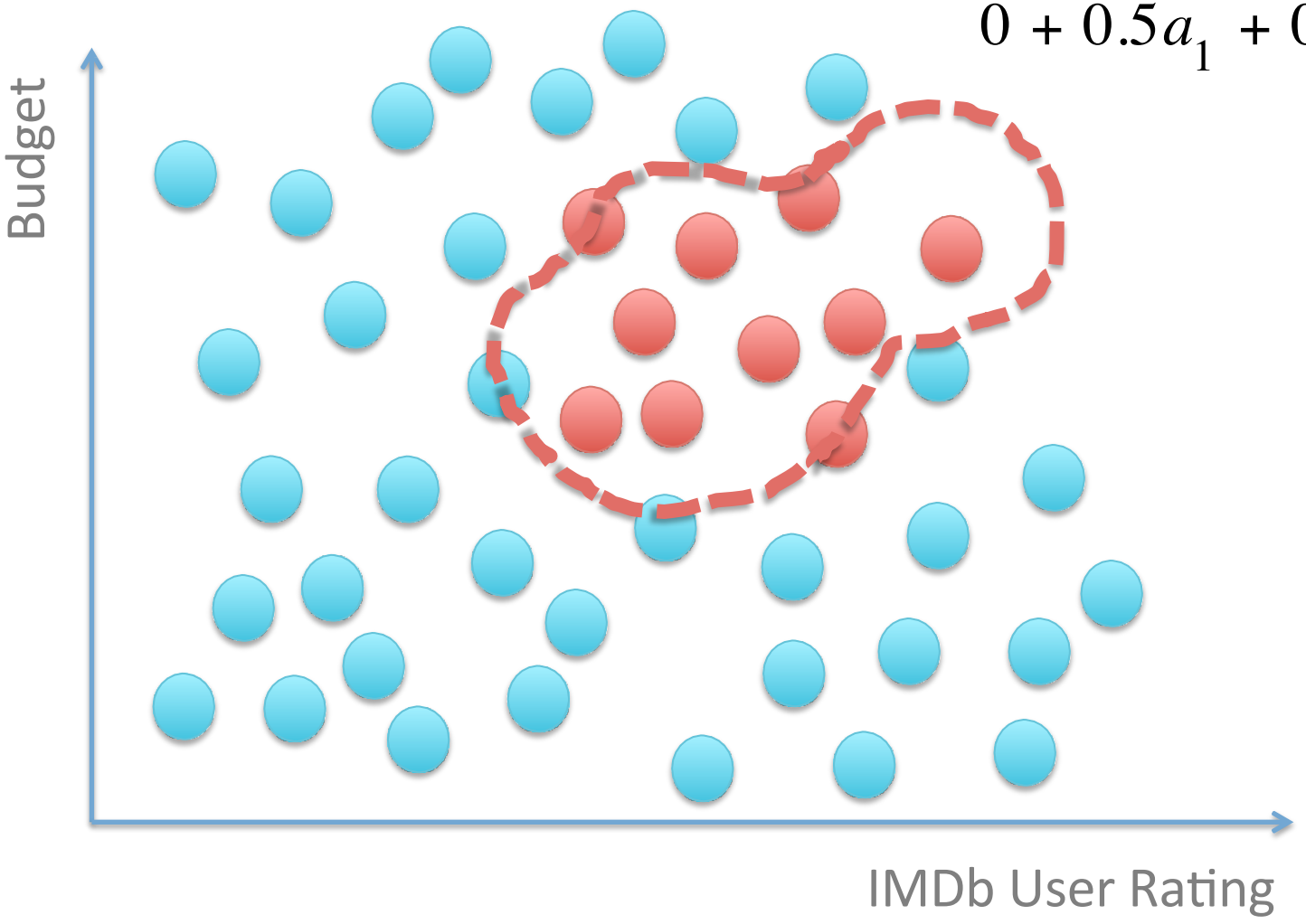
$$0 + 0.7a_1 + 0.9a_2 - 0.6a_3$$



Palme d'Or Winners at Cannes

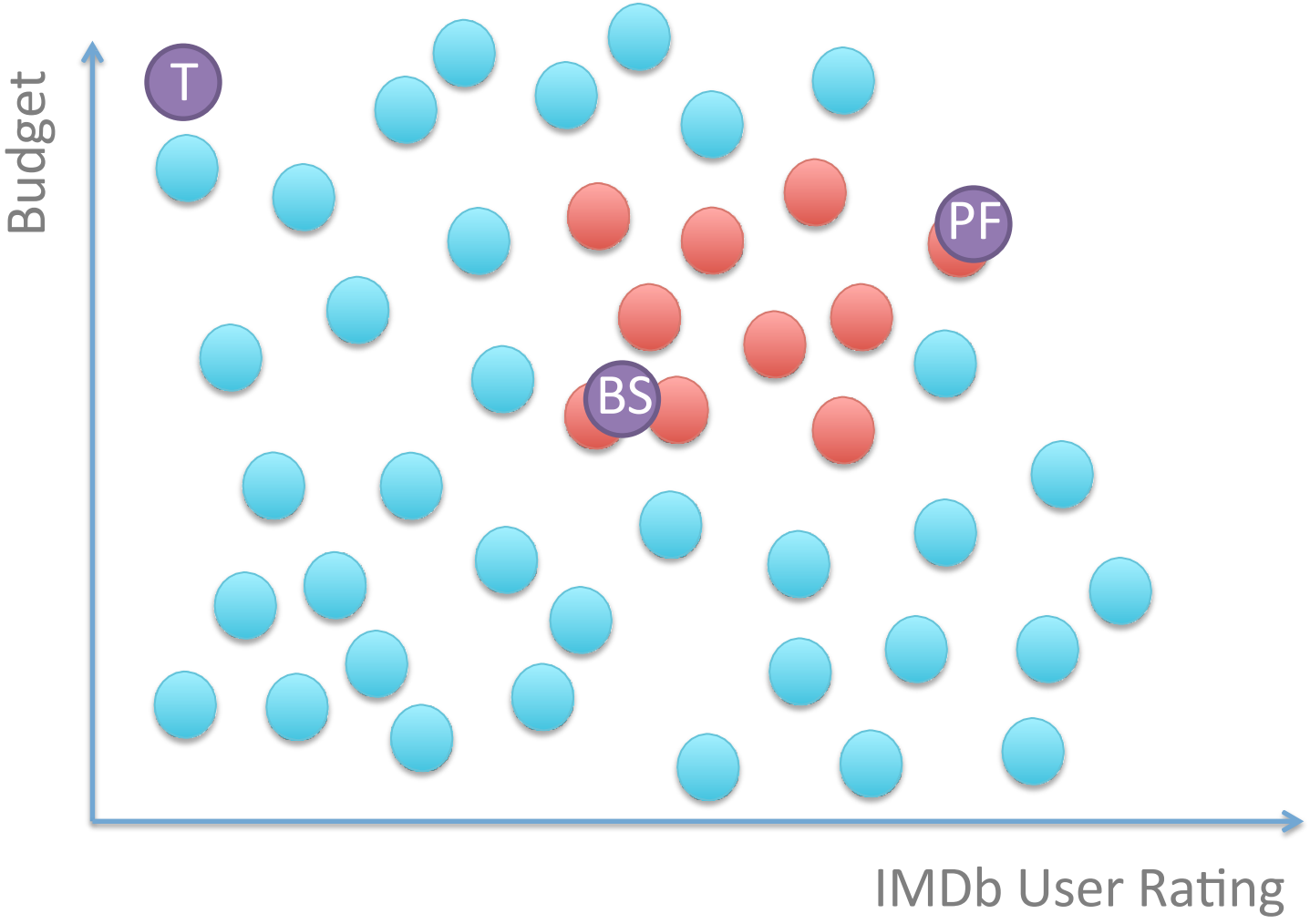
Fitting the intricate boundary with the kernel

$$0 + 0.5a_1 + 0.6a_2 - 0.8a_3$$



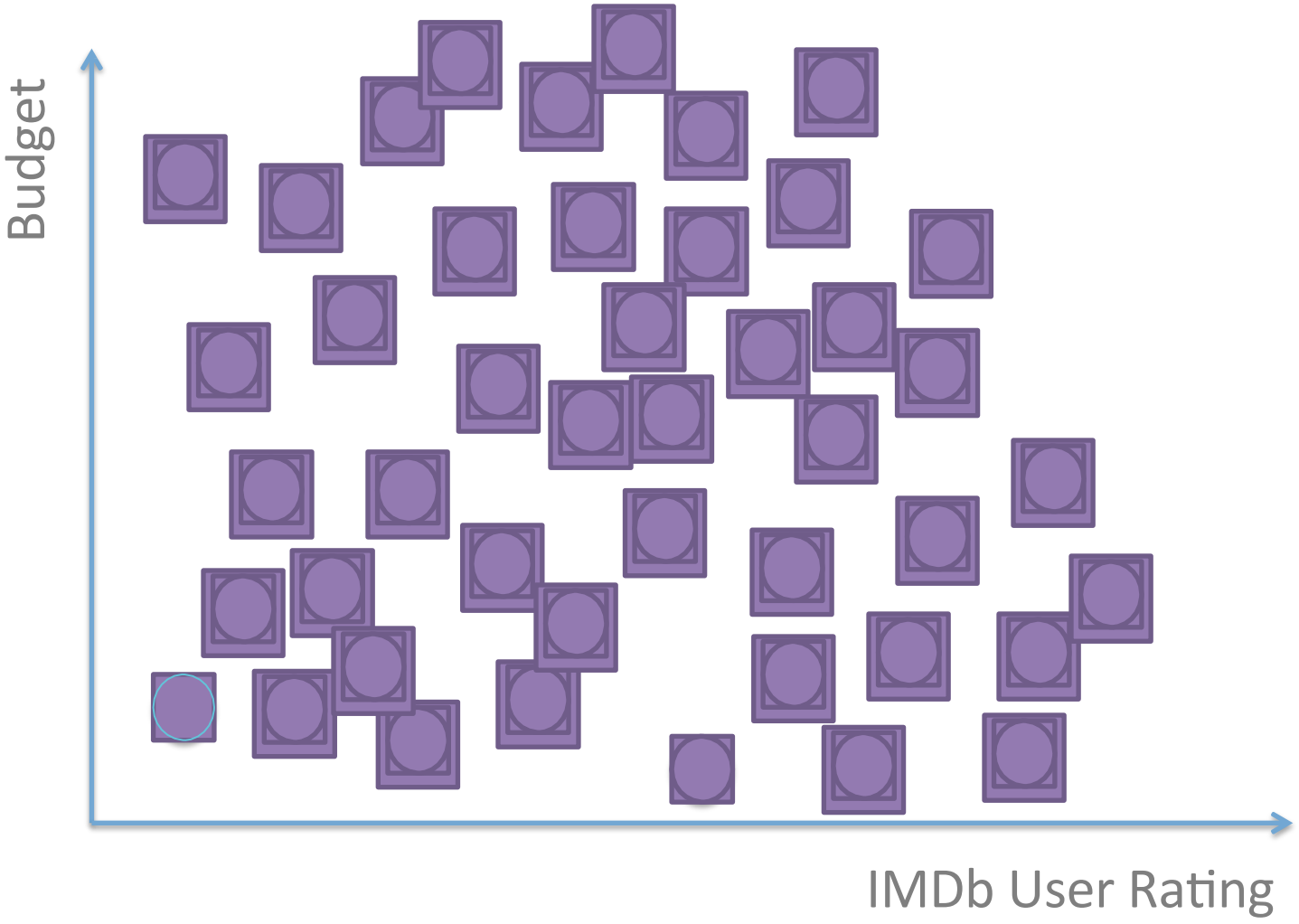
Ok, but how do I choose my feature movies?

What is my Pulp Fiction, Black Swan and Transformers?



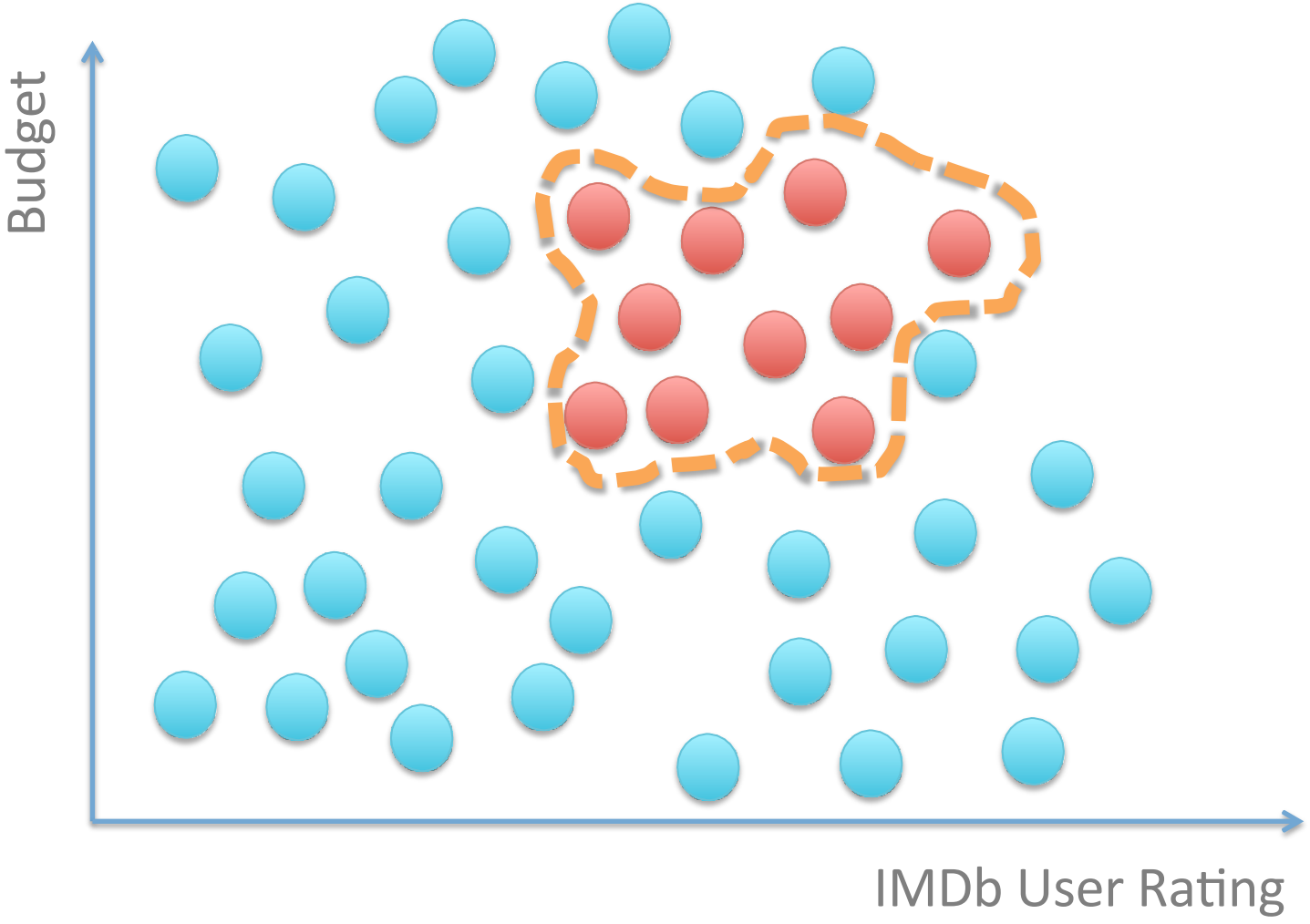
ALL OF THE TRAINING POINTS

Every movie I know is a feature itself



Palme d'Or Winners at Cannes

SVM: Fitting the intricate boundary with RBF kernel



Omg too many points! What do?

Approximate the feature map

```
from sklearn.kernel_approximation import Nystroem
```

Use a small random sample of points as new features

```
from sklearn.kernel_approximation import Nystroem
```

Use a small random sample of points as new features

```
from sklearn.kernel_approximation import RBFSampler
```

Try to mathematically approximate transformation with Monte Carlo sampling (faster, less accurate)

When to use SVM, Logistic Regression, etc.

A lot of features, but small data
(10K features, 1000 training examples)

Best you can do is a simple model, go for Logistic Regression or LinearSVC

When to use SVM, Logistic Regression, etc.

Few features, decent data
(5-100 features, 10K training examples)

Go for a Radial Basis Function Kernel SVC, rock the hell out of it.

When to use SVM, Logistic Regression, etc.

Few features, lots of data
(5-100 features, 100K training examples)

With that much data, you can extract more from weaker features. Add more features. Use Logistic Regression or LinearSVC.

(RBF kernel SVC is too slow with large data)