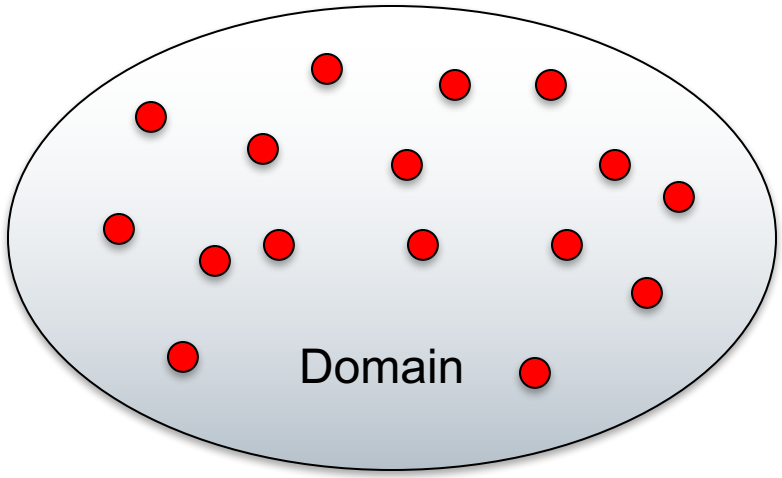
Introduction to Machine Learning

Machine Learning: Classification

- **Consider a domain of objects that we wish to classify**
  - In our lab, the domain is Titanic passengers



The red dots represent objects to be classified

Binary Classification

- **Only two classes exist**
- **Some domains and binary classification tasks:**
  - Images of Faces:
    - ✓ classify as male or female, old or young, etc.
  - Texts of Emails:
    - ✓ classify as “spam” or “ham”, “personal” or “business-related”, etc.
  - Financial records:
    - ✓ classify as ”likely” or “unlikely” to default on a loan
  - Medical measurements:
    - ✓ classify patient as “needs ICU” or “doesn’t need ICU”

Multi-class Classification

- **More than two classes exist**
- **Some domains and multi-class classification tasks:**
  - Images of Faces:
    - ✓ classify as young_male, young_female, old_male, old_female
  - Images of digits:
    - ✓ classify as the numbers 0 to 9
  - Traffic-related data:
    - ✓ classify commute time as short, average, or long

Extracting Features from the Objects

- **The objects to clasify are often too complex to process directly**
- **Solution:**
  - Process the objects to extract a set of features
  - Group the features together into a feature vector
  - We will refer to a feature vector as an instance
  - We will refer to the set of all feature vectors as the instance space

The Training Set

- The training set is a set of ordered pairs

$$\mathcal{T} = \{\underline{\mathbf{x}}_i, l(\underline{\mathbf{x}}_i)\}$$

- Here $\underline{x}_i$ is an instance (feature vector) and *l( )* is the true labelling function (its output is the correct class)
- The training set is samples of the true classification function
- The function we learn should approximate the true function

$$\hat{l}(\underline{\mathbf{x}}) \approx l(\underline{\mathbf{x}})$$

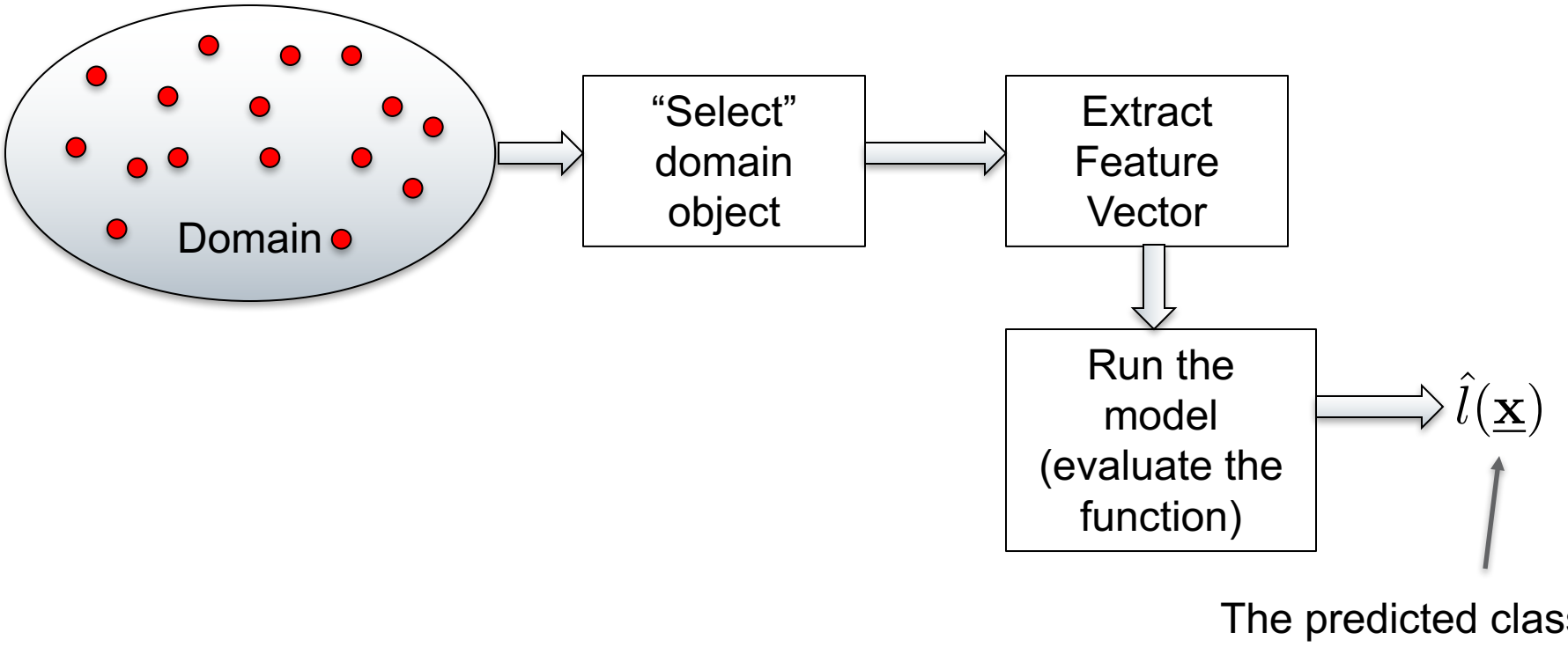
Supervised Learning of a Classification Task

Supervised learning for classification is using a labeled training set to learn a function for mapping instances to classes

- The function learned is also referred to as a model
- The function learned predicts the class of instances that are not in the training set

Diagram of system's use after training

- **An object from the domain is "selected" by the problem at hand**
  - Example: an image arrives from a security camera



Assessing Performance

- **Create a contingency matrix**
  - Use the matrix to estimate useful probabilities

			Predicted class outputs			
		class 1	class 2	class 3		Total in test set
	class 1	80	10	8		98
Test set inputs	class 2	5	90	6		101
	class 3	1	3	75		79

  - Note: the highlighted values represent correct classifications

Assessing Performance cont.

- **Accuracy**
  - Estimates the probability

$$P(\hat{l}(\underline{\mathbf{x}}) = l(\underline{\mathbf{x}}))$$

    - ✓ Add the diagonal entries of matrix and divide by the sum of all entries

- **Error rate**
  - Estimates the probability

$$P(\hat{l}(\underline{\mathbf{x}}) \neq l(\underline{\mathbf{x}}))$$

    - ✓ One minus the accuracy

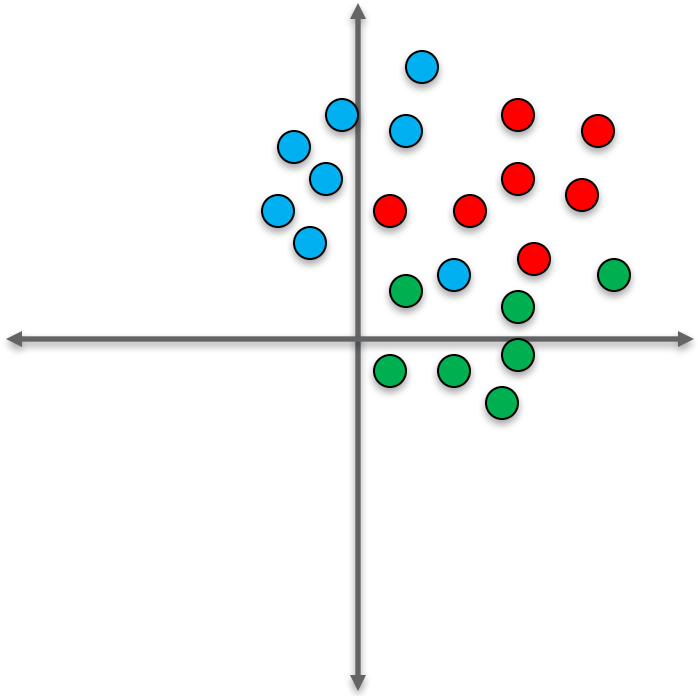
Assessing Performance cont.

- **Terminology to describe a model's performance *with respect to a given class*:**
  - True positive: A *correctly* "accepted" instance of the class
  - False positive: An *incorrectly* "accepted" non-instance of the class
  - True negative: A *correctly* "rejected" non-instance of the class
    - ✓ However, the rejected instance may or may not be correctly classified with respect to the remaining classes
  - False negative: An *incorrectly* "rejected" instance of the class

Note: these values can be different for different classes

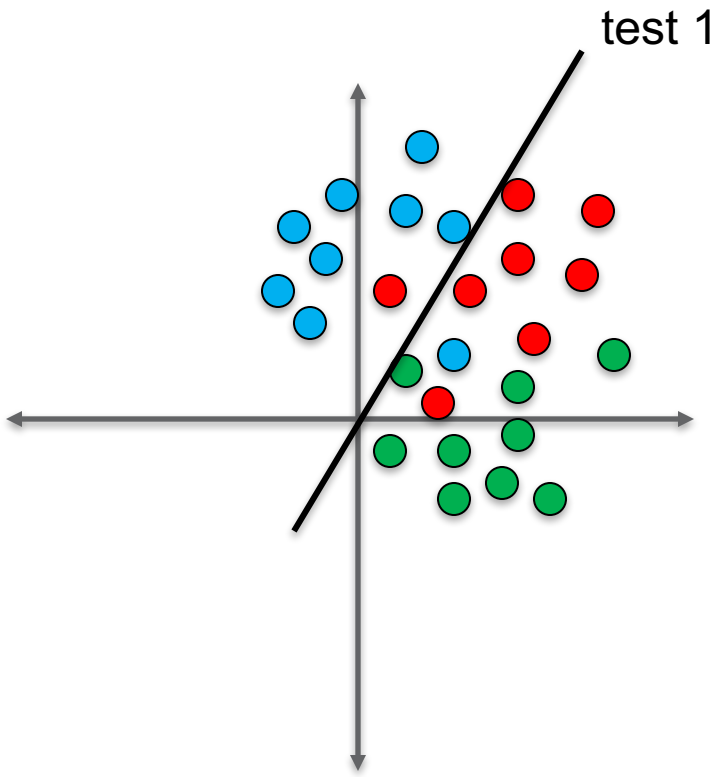
Visualizing performance

- **Assume the feature vector is 2D and there are 3 classes.**
  - the points below represent instances in the instance space (red: class 1, green: class 2, blue: class 3)



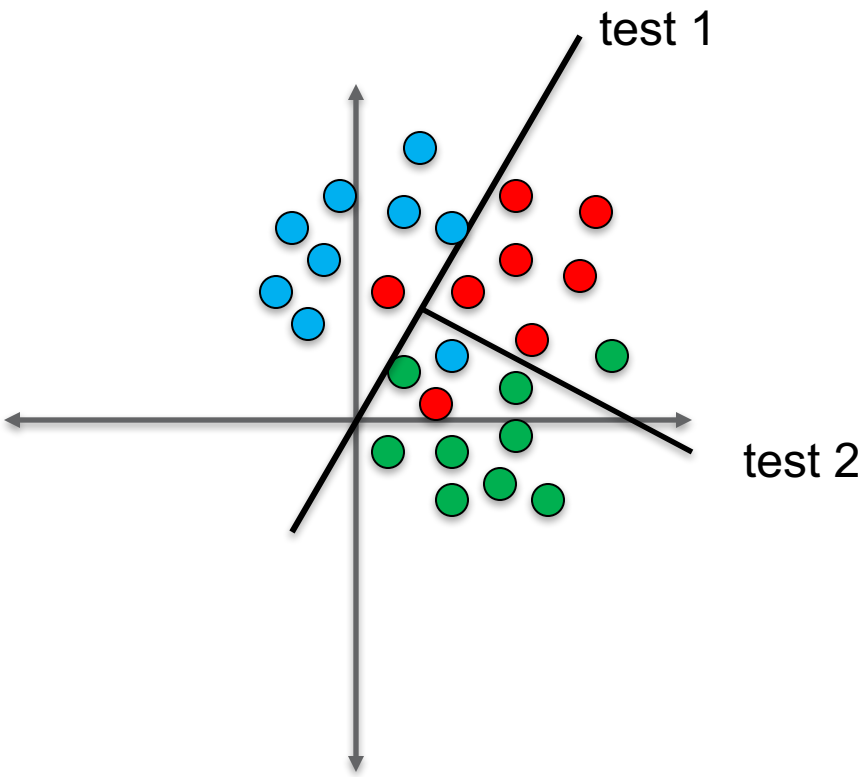
Separating the points

- **One approach: learn lines (planes) that separate points**
  - First, separate one class form the rest



Separating the points

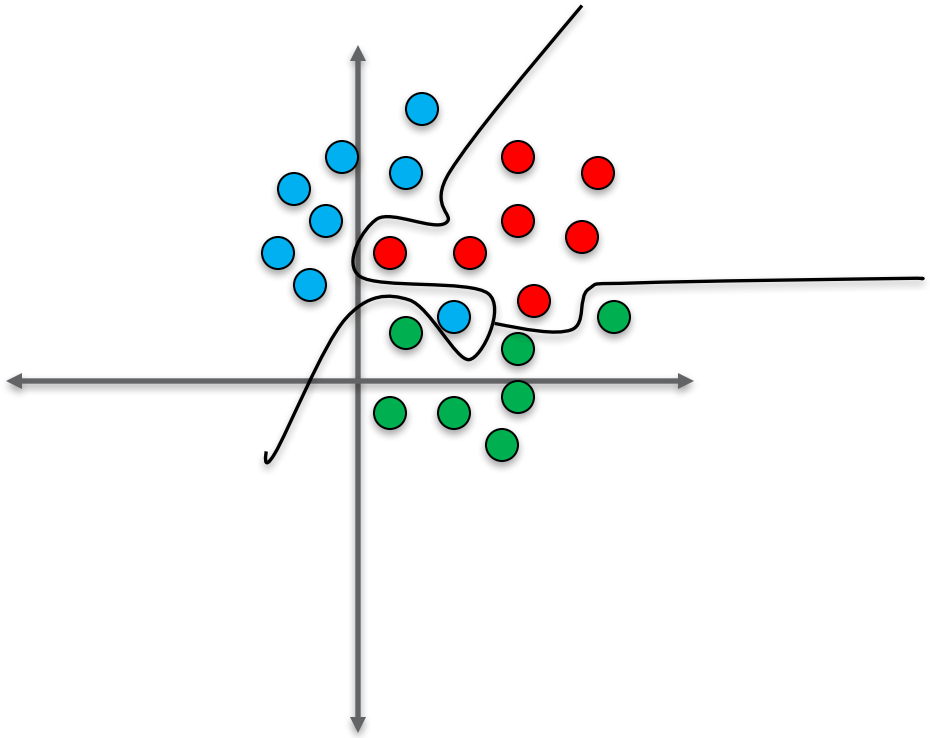
- **Second: separate the other two classes**



  - "Support Vector Machines" (SVMs) are used for separating two classes

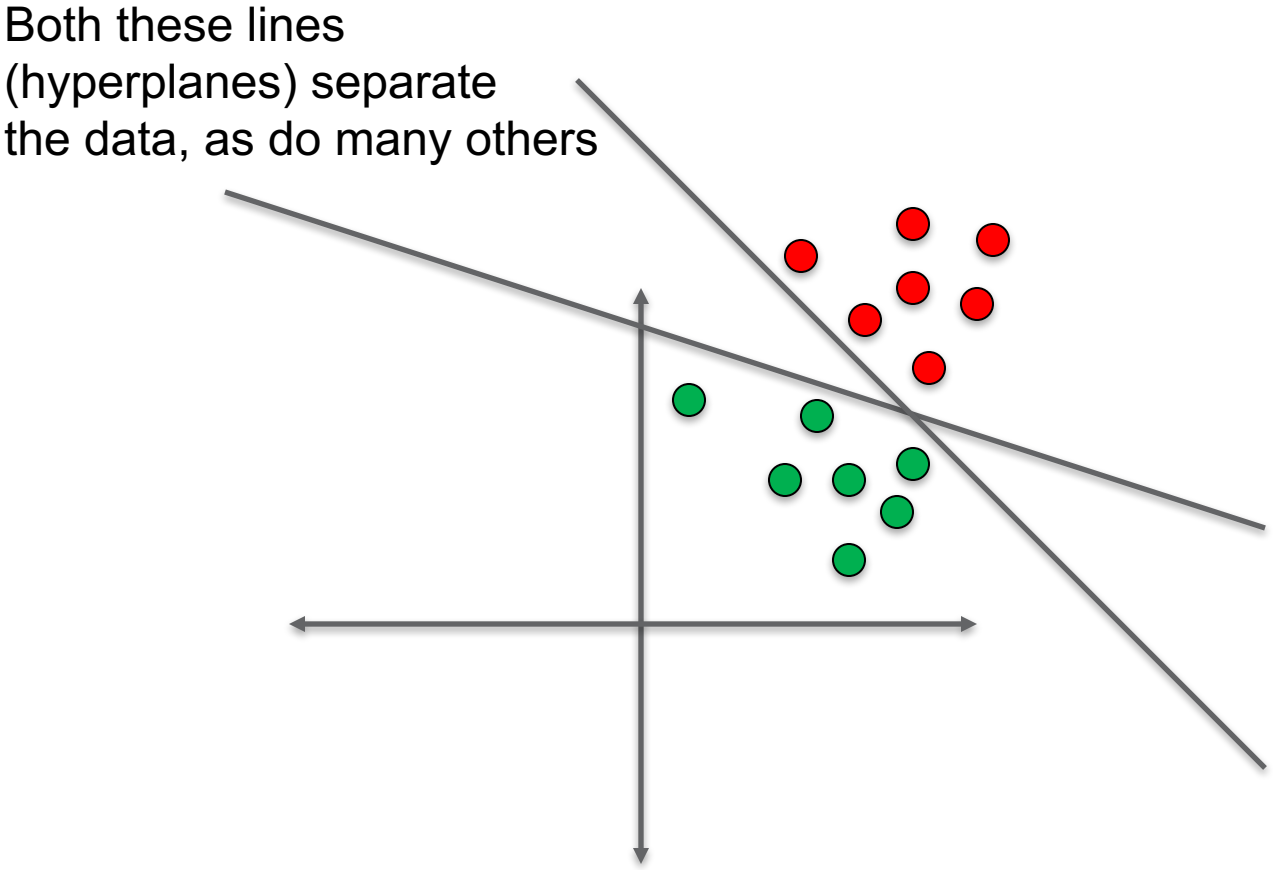
Overtraining

- **Danger: adapting your function too strongly to your training set**
  - Trying too hard to solve the outlier problem may yield a bad solution that doesn't generalize



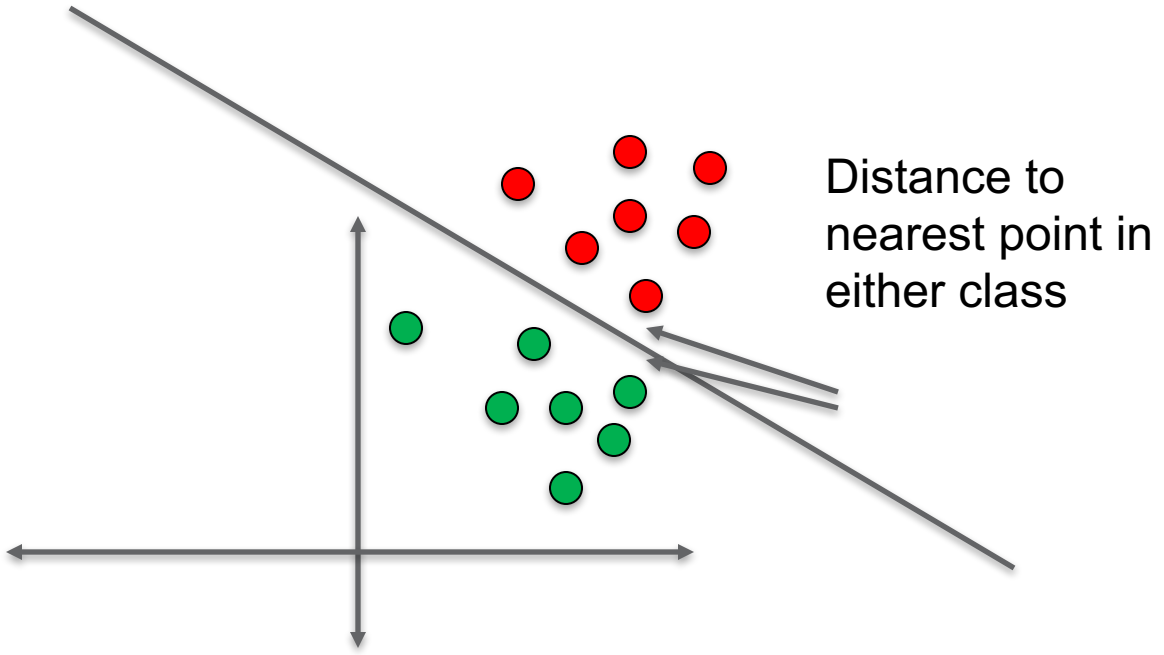
What if there are many separating hyperplanes?

- **When the separating hyperplane is not unique, which one should we choose?**



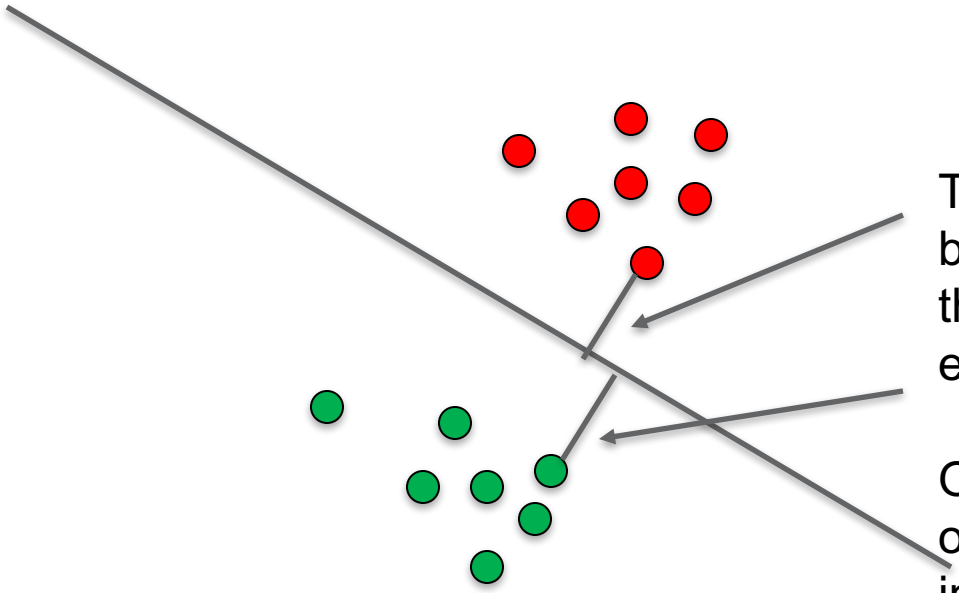
Solution: pick the one that "maximizes the margin"

- **Choose the hyperplane that "maximizes the minimum distance" to any point in either set**
  - Twice this minimum distance is called the "margin"
  - Maximizing one quantity automatically maximizes the other



The SVM hyperplane

- **A zoomed in picture of the previous slide**

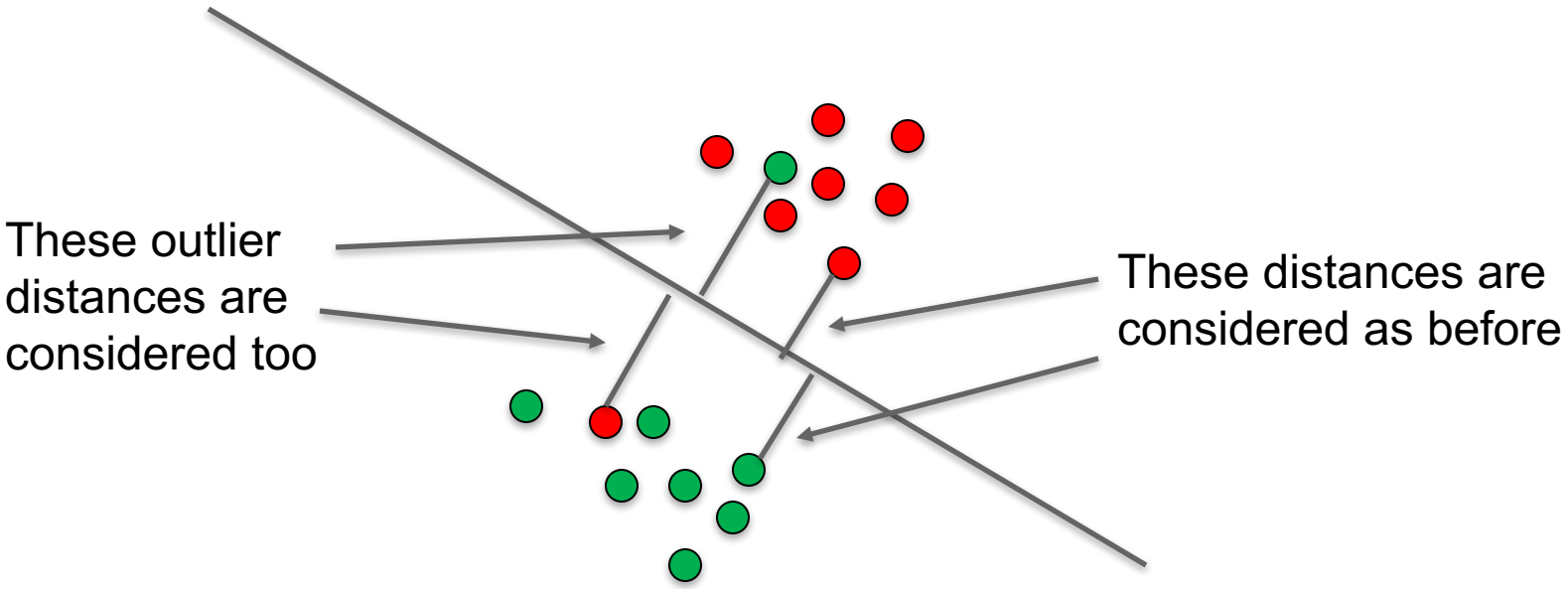


The best hyperplane will be equally distant from the nearest point in each cluster.

Otherwise, moving it one way or the other will increase the minimum distance

What if the data isn't linearly separable?

- **A different more complicated optimization is performed that takes into account "outliers"**



Using the data wisely

- **Given a labeled set of data, how should one proceed?**
  - Solution
    - ✓ Divide available data into a training set and a test set
      - – Divisions in range of 70% / 30% to 80% / 20% are common (training set percentage is listed first)
    - ✓ Train (i.e. learn the prediction function/model) using the training set
    - ✓ Assess the models performance using the test set

Note: sometimes the data is divided into three sets in a 70/15/15 type ratio. The new set is the "validation" set and is used to select a model (e.g. k-nearest neighbor vs SVM. You might train both models and pick the one that works the best on the validation set)

Linear Regression

Data Analytics

- **Analzying data sets to find useful correlations that allow predictions is a common real-world problem**
  - Economics: how is an increase or decrease in taxes likely to affect growth
  - Medicine: how will a decrease in weight and an increase in exercise affect life expectancy?
  - Data networking: how will streaming video traffic react to a major sporting event?
  - Materials science: how will changes in the carbon concentration of steel affect MTBF?

Intuitions regarding prediction

- **If you were told that someone was an American male and you had to predict their weight, what number would you choose?**
- **If you were told someone was an American male from Boulder what number would you choose?**
  - Does this additional information help you make a more accurate prediction?
- **Would a larger vector of "features" describing someone help you make an even better prediction?**
  - How would you use these features to form the prediction?

Prediction with Several Variables

- **Assume a dependent variable, y, will be predicted using a vector of variables, $\underline{\mathbf{x}}$**
- **We have the collection of instance points**

$$\{x_{1i}, x_{2i}, \ldots, x_{Ni}; y_i\}$$

- **We can find an MSE predictor of the form**

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_M x_M$$

Easiest to use Matrices

- We form the linear matrix equation

$$\underline{\mathbf{y}} = A\underline{\beta}$$

- Where *A* and $\underline{\mathbf{y}}$ contain the known values as follows

$$\underline{\mathbf{y}} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} \qquad A = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1N} \\ 1 & x_{21} & x_{22} & \dots & x_{2N} \\ \vdots & & & & \\ 1 & x_{M1} & x_{M2} & \dots & x_{MN} \end{bmatrix} \qquad \underline{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_N \end{bmatrix}$$

Observations

- **The matrix A is M x (N+1)**
  - M is the number of observations we make
  - N is the number of predictor variables
  - The columns of *A* contain the measured predictor variable values
- **The vector $\underline{y}$ is the predicted variable values**
- **The β_0 term allows the prediction equation to have a constant (intercept term): the column of ones in *A* enables this**

Solution cases

- **If M < N+1**
  - The solution is under-determined
  - Not enough observations
- **If M = N+1**
  - The solution is exactly determined
- **If M > N+1**
  - The solution is over-determined
  - The equation can still have an exact solution, but only if there is no measurement noise and the system is perfectly linear

The typical case

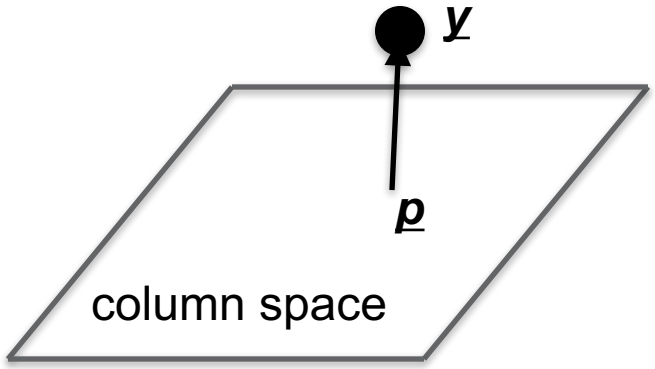
- **M > N+1**
  - The model is over-determined
  - It is usually desirable to collect lots of data
- **There is noise in the system and/or the system is not perfectly linear**
  - There is no exact solution
  - An approximate solution of some sort is needed

We derived the pseudo-inverse earlier in this class for solving the problem (SVD is the most general solution)

Reminder: Finding the solution

- **To find the least-squared error solution:**
  - Project $\underline{\boldsymbol{y}}$ onto the column space of A. Call this point $\underline{\boldsymbol{p}}$
  - Find the vector $\underline{\beta}$ such that $A\,\underline{\beta}$ equals $\underline{\boldsymbol{p}}$
    - ✓ Geometricallly it is clear that no other point in the column space can be closer to $\underline{\boldsymbol{y}}$ than $\underline{\boldsymbol{p}}$ (think Pythagorean theorem and the triangle inequality)

Note: the vector from $\underline{\boldsymbol{p}}$ to $\underline{\boldsymbol{y}}$ is orthogonal to every vector in the column space



Non-linear prediction

One type of non-linear prediction

- **Apply a non-linear function to each column of *A***
  - Take the logarithm of the column
  - Take *e* to the column's power
  - Raise the column to a power
  - etc.
- **Solve the matrix equation to get $\underline{\beta}$ as before**

Example

- **Logarithmic regression on one predictor variable**
  - Consider the following matrix where *x(1), x(2), ...x(M)* are the observations

$$A = \begin{bmatrix} 1 & \log(x(1)) \\ 1 & \log(x(2)) \\ \vdots & \vdots \\ 1 & \log(x(M)) \end{bmatrix}$$

Another Example

- **Polynomial regression on <u>one</u> variable**
  - Consider the following matrix where *x(1), x(2), ...x(M)* are the observations

$$A = \begin{bmatrix} 1 & x(1) & x^2(1) & x^3(1) & \dots & x^N(1) \\ 1 & x(2) & x^2(2) & x^3(2) & \dots & x^N(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x(M) & x^2(M) & x^3(3) & \dots & x^N(M) \end{bmatrix}$$

Example

- **Thresholded regression using multiple variables**
  - Compute the mean and standard deviation of each column of *A* separately (i.e. for each predictor variable)
  - Then for each row of each column
    - ✓ Set the measurement value to 1 if it exceeds the mean by some multiple of the column standard deviation; set it to 0 otherwise. In other words, compare to the threshold

$$T = \mu + \alpha\sigma$$

Assessing Regression Performance

Performance Measure

- **We need a way to compare the performance of two predictors**
- **We will mathematically define the following terms**
  - Total Sum of Squares (TSS)
  - Residual Sum of Square (RSS)
- **Definitions**

$$\text{TSS} = \sum_{i=1}^{M} (y_i - \mu_y)^2$$

Equals the variance if divided by *M*

$$\text{RSS} = \sum_{i=1}^{M} (y_i - \hat{y}_i)^2$$

Equals the error power if divided by *M*

R^2 and the Fraction of Variance Unexplained

- **We define the *Fraction of Variance Unexplained* as**

$$\text{FVU} = \frac{\text{RSS}}{\text{TSS}}$$

- **And define the Coefficient of Determination, R^2, via**

$$R^2 = 1 - \text{FVU} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

When RSS = 0, then R^2 = 1, we say all of the variance is explained
When RSS = TSS, then R^2 = 0, we say none of the variance is explained

Note: RSS <= TSS, since we can always pick $\hat{y} = \mu_y$

Interpretation

- **R^2 is the fraction of *y*'s variance that is "explained" by the prediction**
- **1 - R^2 is the fraction of the variance that is "unexplained" by the prediction**
  - Note that, by re-arranging the definition, we have

$$1 - R^2 = \frac{\text{RSS}}{\text{TSS}}$$

Further Interpretation

- **We can further define the Explained Sum of Squares as**

$$\mathrm{ESS} = \sum_{i=1}^{M} (\hat{y}_i - \mu_y)^2$$

- **In some situations, e.g. it can be proven for linear regression, we have**
  - TSS = ESS + RSS
  - In this case we have

$$R^2 = \frac{\mathrm{ESS}}{\mathrm{TSS}}$$

Additional Information

Assessing Classification Performance cont.

- **True positive rate (sensitivity/recall) *for a given class L***
  - Estimates the probability

$$P(\hat{l}(\underline{\mathbf{x}}) = l(\underline{\mathbf{x}}) \mid \underline{\mathbf{x}} \in \mathcal{L})$$

- **True negative rate (specificity) *for a given class L***
  - Estimates the probability

$$P(\hat{l}(\underline{\mathbf{x}}) \neq l(\underline{\mathbf{x}}) \mid \underline{\mathbf{x}} \notin \mathcal{L})$$

- **Precision (confidence) *for a given class L***
  - Estimates the probability

$$P(\underline{\mathbf{x}} \in \mathcal{L} \mid \hat{l}(\underline{\mathbf{x}}) \in \mathcal{L})$$

k-nearest neighbors classification

- **Given a "distance" measure, the following algorithm is sometimes practical**
  - To classify a new instance **x**
    - ✓ Compute the distance between **x** and each element of the training set
    - ✓ Determine its k nearest neighbors in the training set (k can be 1)
    - ✓ Assign to **x** the majority class of its k-nearest neighbors

Linear Regression with One Predictor (The Details)

Can express the prediction error cost function in vector notation

- **Assume we have a set of N measurements, x_i**
- **We can collect them together into a vector**
  - $\underline{\mathbf{x}} = (x_1, x_2, \ldots, x_N)$
  - The length of $\underline{\mathbf{x}}$, when squared, is its "energy"
    - ✓ This is an analogy to an electrical signal, where the x_i are samples of a voltage
- **Notation: for a constant, μ, we'll denote the length *N* vector with μ as <u>every</u> component as <u>μ</u>**

Other useful concepts: variance and standard deviation

- **The average error energy per component is the error power (recall: power = energy /time). We call this the *variance***

$$\sigma^2 = \frac{1}{N} \left| \underline{\mathbf{x}} - \underline{\mu} \right|^2$$

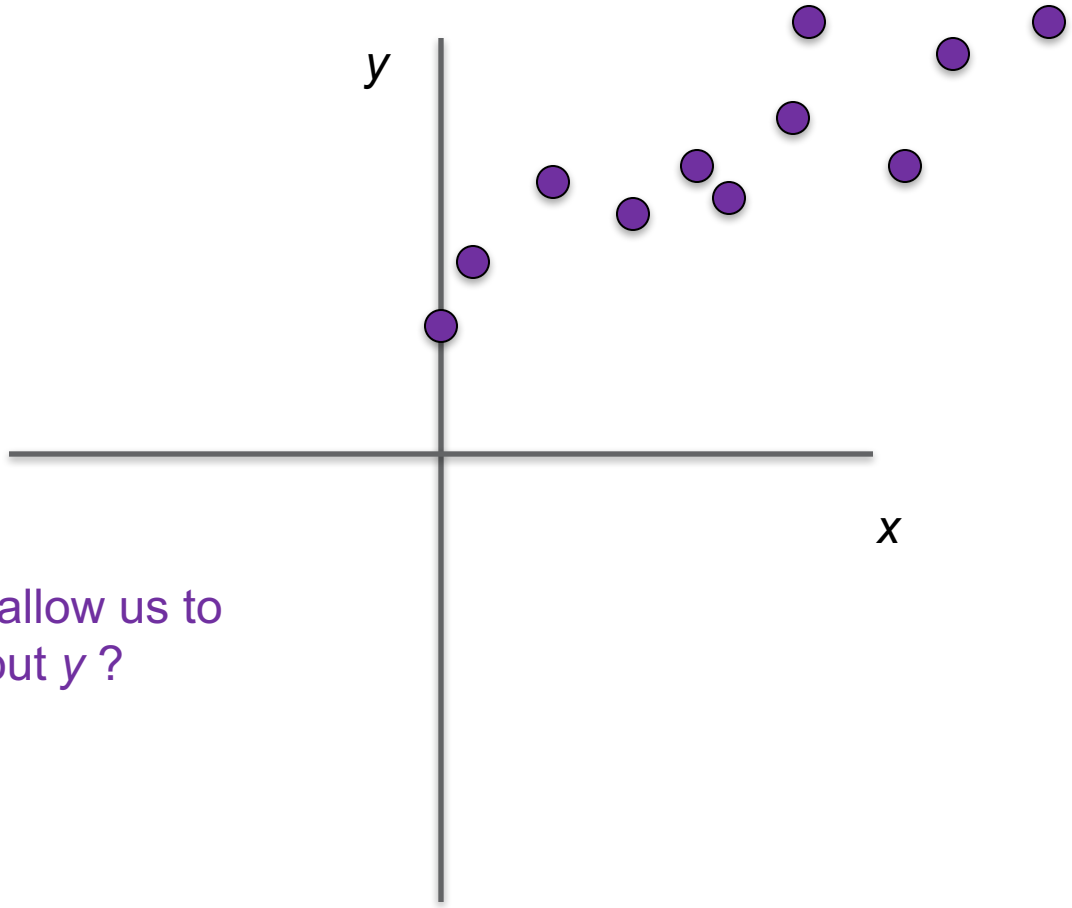
The vertical bars mean vector length; $\underline{\mu}$ is the mean vector for $\underline{x}$

- **The square root of this power is the *standard deviation***

$$\sigma = \frac{1}{\sqrt{N}} \left| \underline{\mathbf{x}} - \underline{\mu} \right|$$

Scatter plot of data points

- **Consider a set of data measurements**



Does knowing *x* allow us to say anything about *y* ?

Remap the vectors

- **We will first remap each vector as follows**

$$\underline{\mathbf{w}} = \frac{1}{\sigma_x}(\underline{\mathbf{x}} - \underline{\mu_{\mathbf{x}}})$$

$$\underline{\mathbf{z}} = \frac{1}{\sigma_y}(\underline{\mathbf{y}} - \underline{\mu_{\mathbf{y}}})$$

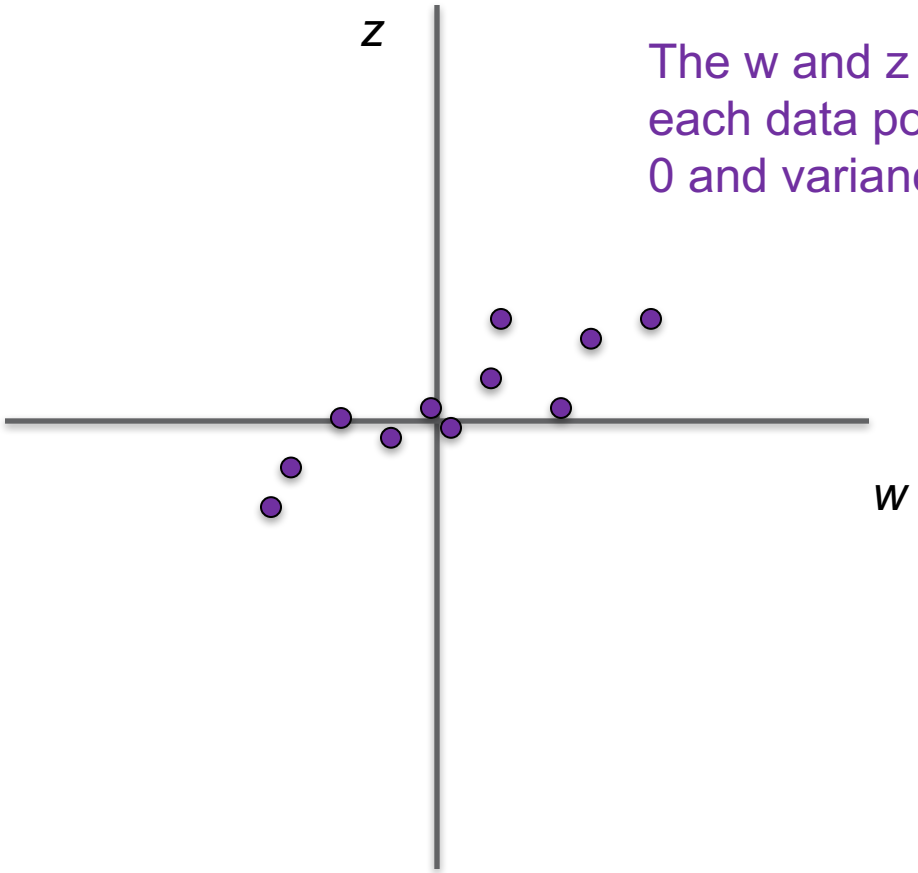
Note that both of these vectors are now unit vectors

- **Notice that these mappings are invertible. For example, given a z_i value, we can always get back to its corresponding y_i value via**

$$y_i = \sigma_y z_i + \mu_y$$

Plot of remapped data points

- **Data points after remapping**



The w and z components of each data point now have mean 0 and variance 1

Linear regression

- **We seek a line, defined by the scalar constants *a* and *b*, that minimizes the cost (i.e. error power)**

$$C(a,b) = | \underline{\mathbf{z}} - (a\underline{\mathbf{w}} + \underline{b}) |^2$$

- **We can write this as a dot product**

$$C(a,b) = (\underline{\mathbf{z}} - (a\underline{\mathbf{w}} + \underline{b})) \cdot (\underline{\mathbf{z}} - (a\underline{\mathbf{w}} + \underline{b}))$$

After simplifying

- **Expanding the dot product, grouping terms, and simplifying yields**

$$C(a,b) = N - 2a\underline{\mathbf{w}} \cdot \underline{\mathbf{z}} + a^2 N + Nb^2$$

To get this result, convince yourself that the following equations are true (why? because the components of *w* and *z* are both zero mean)

$$\underline{\mathbf{b}} \cdot \underline{\mathbf{w}} = 0 \text{ and } \underline{\mathbf{b}} \cdot \underline{\mathbf{z}} = 0$$

Solution to the minimization

$$C(a,b) = \underbrace{N - 2a\underline{\mathbf{w}} \cdot \underline{\mathbf{z}} + a^2 N}_{} + Nb^2$$

This is a parabola in terms of *a* and has its vertex (minimum) at

$$a = \frac{\underline{\mathbf{w}} \cdot \underline{\mathbf{z}}}{N}$$

$$= \frac{\underline{\mathbf{w}}}{\sqrt{N}} \cdot \frac{\underline{\mathbf{z}}}{\sqrt{N}}$$

$$= \cos\theta$$

This is the only term that depends on *b* and it is minimum for

$$b = 0$$

The correlation coefficient

- **The value *a* is the correlation coefficient. It is the cosine of the angle between the vectors *w* and *z* !!**
  - Traditional to use ρ for the correlation coefficient

The Covariance

- **For prediction with a single variable, in terms of the original vectors, substitution yields**

$$\rho = \frac{\frac{1}{N}(\underline{\mathbf{x}} - \underline{\mu}_x) \cdot (\underline{\mathbf{y}} - \underline{\mu}_y)}{\sigma_x \sigma_y}$$

$$= \frac{\mathrm{cov}(\underline{\mathbf{x}}, \underline{\mathbf{y}})}{\sigma_x \sigma_y}$$

The numerator is defined to be the covariance

The regression line

- **We have just shown that the MSE regression line for predicting *z* in terms of *w* is**

$$z = \rho w$$

This is the projection of the vector **w** onto **z**

- **Or in terms of the original data**

$$y = \beta x + (\mu_y - \beta \mu_x) \text{ where } \beta = \rho \frac{\sigma_y}{\sigma_x}$$

Linear prediction procedure

- **Obtain a set of data points**
- **Compute the means and standard deviations for the *(x,y)* data points**
- **Compute the correlation coefficient**
- **Given a new point, *x*, the predicted value for *y* is**

$$y = \beta x + (\mu_y - \beta \mu_x) \text{ where } \beta = \rho \frac{\sigma_y}{\sigma_x}$$