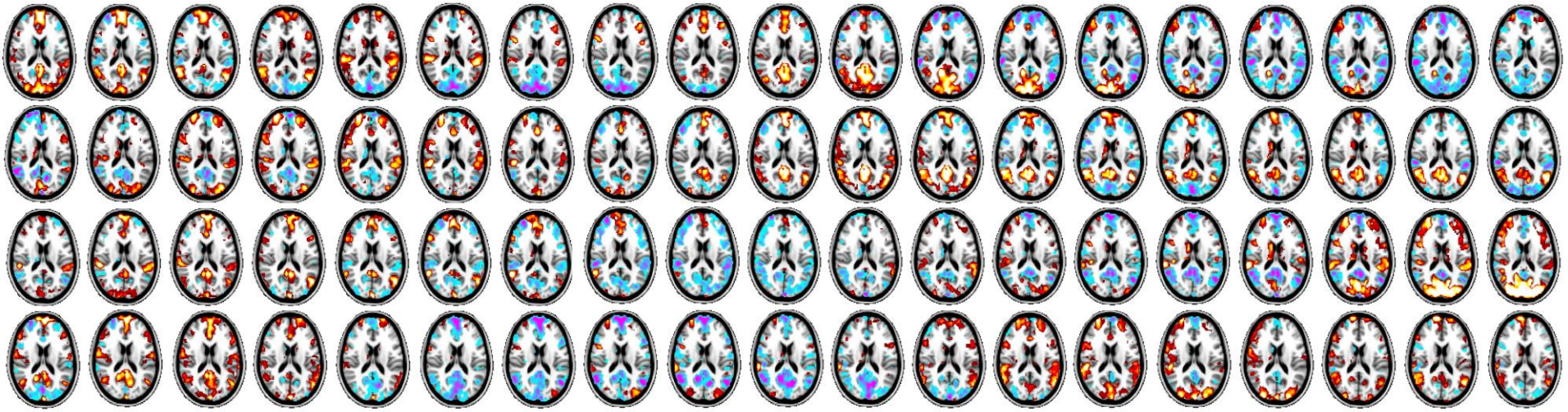
Clasificación de estados cerebrales usando neuroimágenes funcionales

Clase 4:

Mas sobre *machine learning*



Enzo Tagliazucchi (tagliazucchi.enzo@googlemail.com)

universidad de buenos aires - exactas
departamento de física

- Primera clase: introducción a las neuroimágenes funcionales + demostración práctica de preprocesado de datos funcionales.

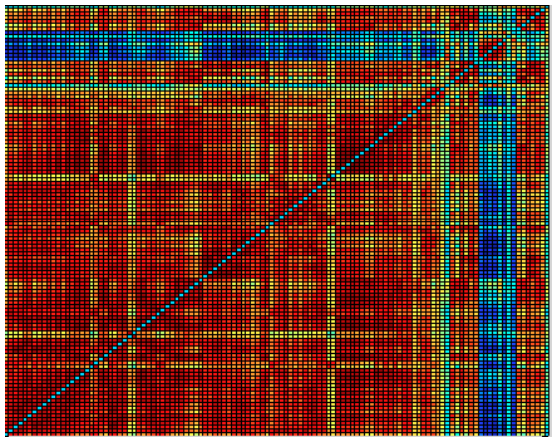
- Segunda clase: introducción a la idea de conectividad cerebral (funcional y anatómica) + demostración sobre obtención de conectividad funcional (Python)

- Tercera clase: introducción básica a conceptos de machine learning + desarrollo de un clasificador para distinguir vigilia de sueño profundo (Python + scikit-learn)

- Cuarta clase: más métodos de machine learning (feature selection, mapeo de relevancia de features, clasificadores multi clase) cómo medir significancia estadística de clasificadores, etc)

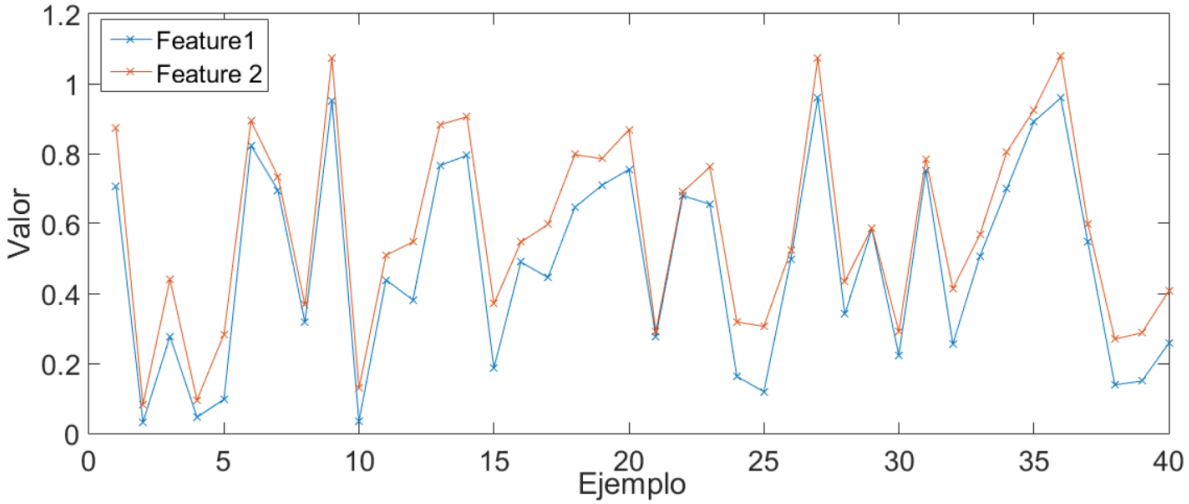
- Quinta clase: temas pendientes + charla general sobre investigación actual en conciencia + preguntas + evaluación

Seleccion de features



Espacio de *features*: matriz de correlaciones entre N regiones

$M \sim N^2$ *features*



Includir a *feature* 1 y 2 no aporta mas informacion extra.
Ademas puede haber features irrelevantes para la clasificacion

Busqueda exhaustiva:

3 features, 8-1=7 posibilidades (1,1,1),(0,1,1),(1,0,1),(1,1,0),(0,0,1),(0,1,0),(1,0,0)

En general, hay $2^M - 1$ posibilidades → prohibitivo para M grande!

Seleccion recursiva (*forward*):

1. Evaluo la *performance* (e.g. AUC) incluyendo un solo *feature*:

   (1,0,0),(0,1,0),(0,0,1)

2. Selecciono el *feature* que da la mayor *performance* (digamos el primero) y evaluo la *performance* del clasificador agregando un *feature* mas

   (1,1,0),(1,0,1)

3. Agrego el *feature* que aumenta mas la *performance* (digamos el primero) y evaluo la *performance* agregando un *feature* mas

   (1,1,1)

4. Elijo la combinacion que me da la mayor *performance*.

En total evalue 3+2+1=6 posibilidades en vez de 7 (busqueda exhaustiva).
Generalmente son:

$$\sum_{i=1}^{M} i = \frac{M(M-1)}{2} \sim M^2 \ll 2^M$$

Seleccion recursiva (*backward*):

1. Evaluo la *performance* (e.g. AUC) incluyendo todos los *features*:

   (1,1,1)

2. Evaluo la *performance* del clasificador removiendo un *feature*:

   (1,1,0),(1,0,1),(0,1,1)

3. Remuevo el *feature* que aumenta mas la *performance* (digamos el primero) y evaluo la *performance* removiendo un *feature* mas

   (1,0,0),(0,1,0)

4. Elijo la combinacion que me da la mayor *performance*.

En total evalue 3+2+1=6 posibilidades en vez de 7 (busqueda exhaustiva).
Generalmente son:

$$\sum_{i=1}^{M} i = \frac{M(M-1)}{2} \sim M^2 \ll 2^M$$

Observacion (importante):

Si utilice una tecnica de este tipo para elegir los *features* y evalue la *performance* usando *cross-validation*, entonces necesito un *test set* aparte para estimar la *performance* final

(porque la *performance* usando *cross-validation* es el resultado de una optimizacion)

Filtros univariados:

La cantidad de fiteos del modelo necesarias para implementar un filtro recursivo es $\sim M^2$ que puede ser un numero muy grande para M grande.

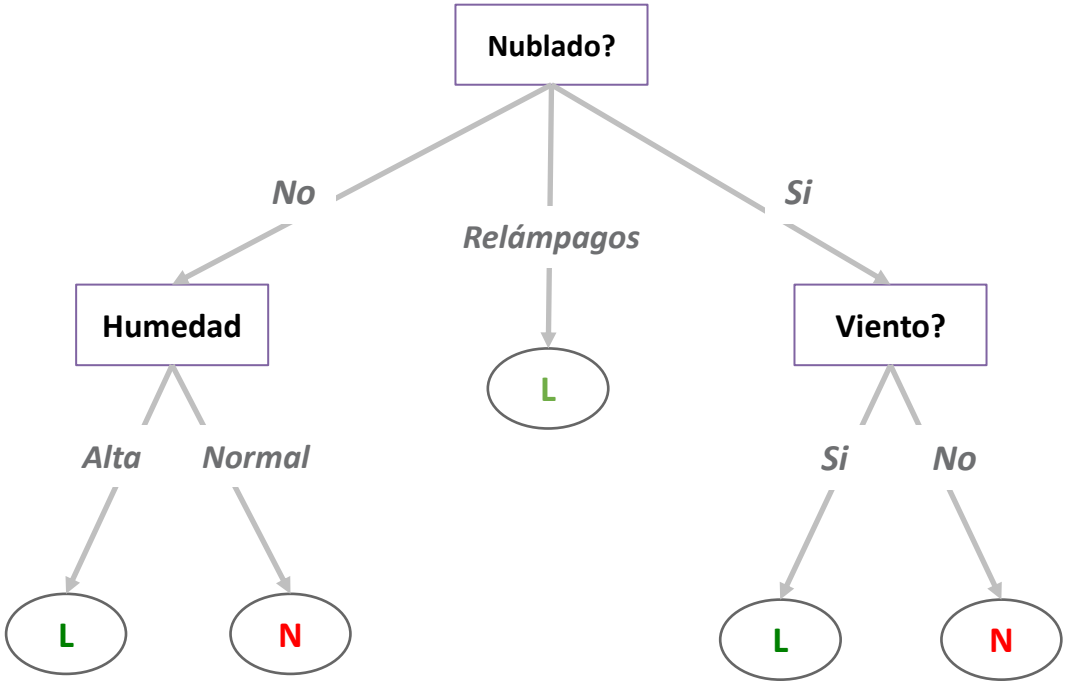
Para cada fold, puedo aplicar un criterio univariado dentro del training set para decidir si incluyo o no al feature, y luego seleccionar los mismos features en el test set para evaluar.

1. Reduzco la cantidad de features sin tener que entrenar mas veces el modelo

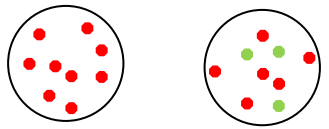
2. Para cada feature compute un numero (e.g. F-test, t-test, etc) que cuantifica que tan distinto es ese feature entre ambas clases (dentro del training set).

Luego me quedo con el X% de los features con el numero mas grande.

Relevancia de *features* (arboles de decision)



Cada vez que uso un *feature* para dividir en dos los ejemplos, termino con dos conjuntos de distinta pureza



Computo el aumento en pureza que obtengo cuando divido los ejemplos usando ese *feature* y eso me da una idea de su importancia

Para *Random Forests*: promedio la importancia sobre todos los arboles construidos

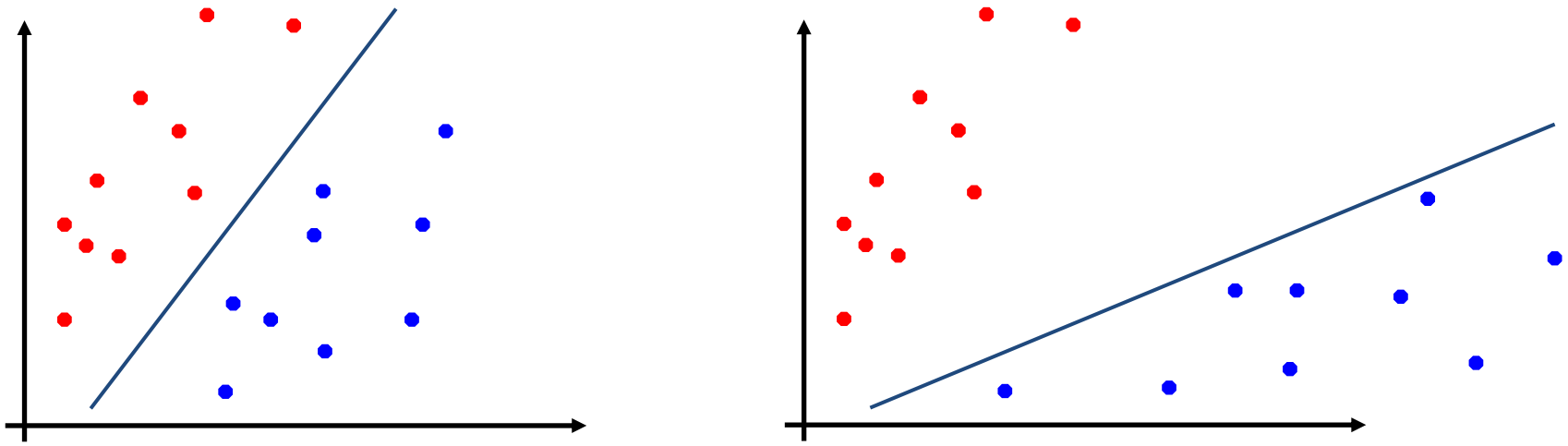
Problemas:

- Si dos *features* estan muy correlacionados entonces si primero divido los datos usando uno, cuando divido usando el otro voy a tener un aumento de pureza muy bajo → problemas
- Hay un sesgo a favor de los *features* con muchas valores discretos (no es problema en *features* continuos, como ser correlaciones lineales)

Normalizacion

En ciertos algoritmos, si dos *features* tienen un rango de valores muy distintos, puede resultar en parametros suboptimos

Support Vector Machine lineal



El feature con el rango de valores mas grande influencia mas el borde entre las clases!

Normalizacion:

- z-scores (restar el promedio y dividir por el desvio estandar)
- Escaleo min-max: lleva a cada feature al intervalo [0,1]

Significancia estadistica y permutacion de clases

Cross-validation divide los datos al azar entre entrenamiento y validacion

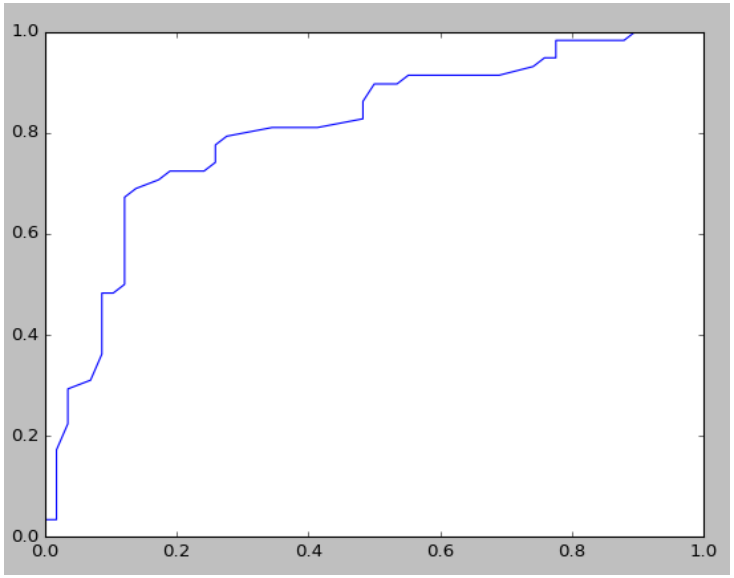
Si hago cross-validation una sola vez y obtengo AUC=0.55, puede que cuando lo repita varias veces tenga AUC=0.48, 0.45, 0.51, etc

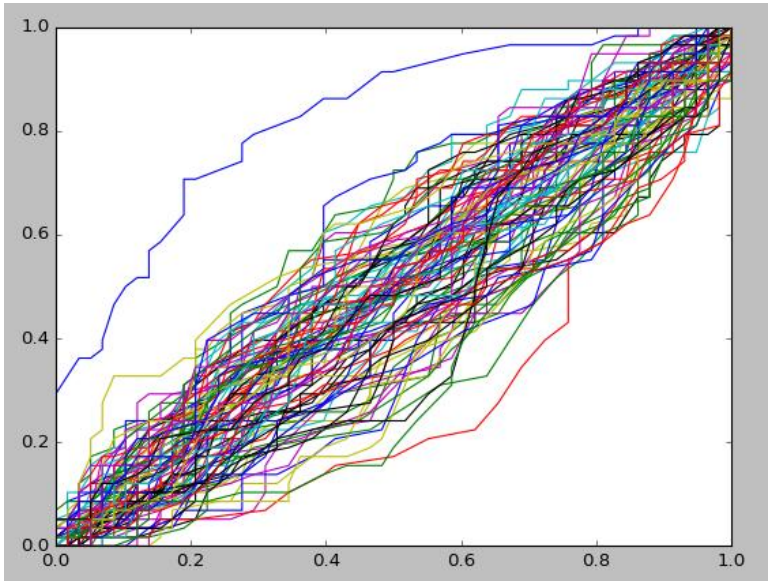
Es decir, hay una variabilidad intrinseca por el azar del proceso → como puedo saber si el clasificador es **realmente** mejor que tirar una moneda?

Solucion: estimar esa variabilidad y usarla para entender si el AUC de mi clasificador esta o no dentro del rango de esa variabilidad

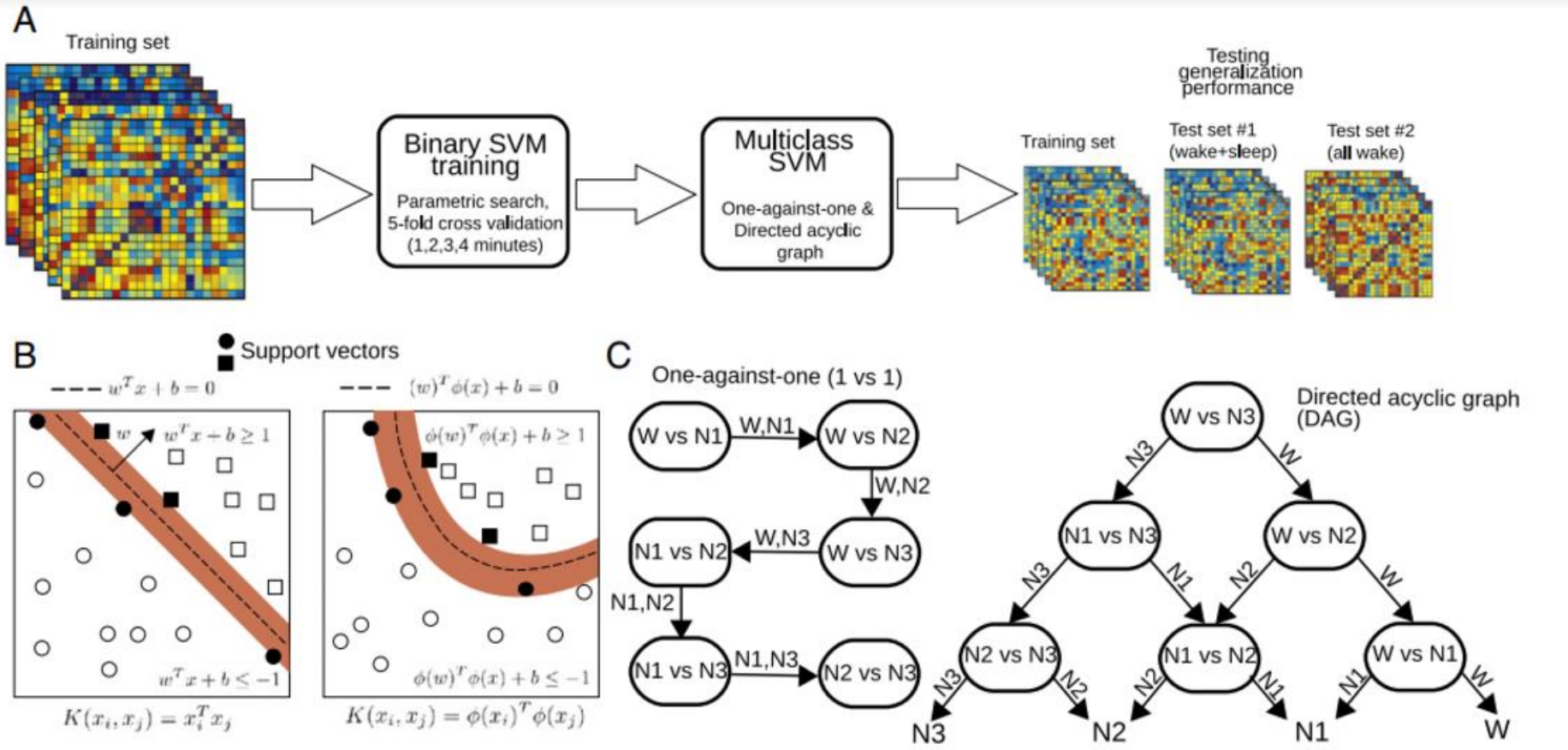
Entrenar y evaluar un clasificador muchas veces permutando la asignacion de clases al azar para tener una idea de cual es la variabilidad de "tirar una moneda" y contar la cantidad de veces que la AUC de los clasificadores permutados da mayor que la del clasificador con los datos real

… dividir por la cantidad de permutaciones y obtener una probabilidad (p-valor empirico)


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

Clasificadores multiclase


A
Training set
Binary SVM training
Parametric search, 5-fold cross validation (1,2,3,4 minutes)
Multiclass SVM
One-against-one & Directed acyclic graph
Testing generalization performance
Training set
Test set #1 (wake+sleep)
Test set #2 (all wake)
B
Support vectors
$w^T x + b = 0$
w
$w^T x - b \geq 1$
$w^T x + b \leq -1$
$K(x_i, x_j) = x_i^T x_j$
$(w)^T \phi(x) + b = 0$
$\phi(w)^T \phi(x) + b \geq 1$
$\phi(w)^T \phi(x) + b \leq -1$
$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$
C
One-against-one (1 vs 1)
W vs N1
W,N1
W vs N2
W,N2
N1 vs N2
W,N3
W vs N3
N1,N2
N1 vs N3
N1,N3
N2 vs N3
Directed acyclic graph (DAG)
W vs N3
N3
W
N1 vs N3
W vs N2
N3
N1
N2
W
N2 vs N3
N1 vs N2
W vs N1
N3
N2
N2
N1
N1
W
N3
N2
N1
W