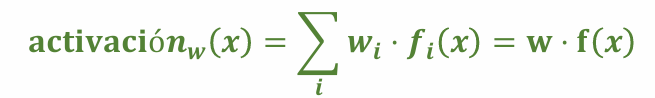
**Aprendizaje supervisado**

**Clasificadores**

* Las entradas son valores de una característica
* A cada característica se le asigna un peso
* La suma es la activación

****

**Pesos**

* Clasificación binaria Multiplicar el vector de las características con el vector de los pesos > 0 o < 0 (umbral)
* Aprendizaje a partir de los ejemplos ajustar/adecuar los pesos
* Producto escalar w\*f

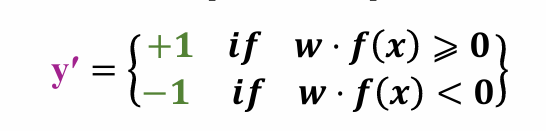
**Fronteras de decisión**

**Decisión binaria**

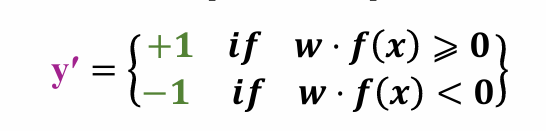
* En el espacio generado por los vectores de características
  + Los ejemplos son puntos
  + Cualquier vector de pesos será un hiperplano
  + Un lado se corresponde con Y = +1
  + El otro se corresponde con Y = -1

**Aprendizaje Perceptrón Binario**

* y la clase correcta,y’ la predecida
  + Inicializar pesos en 0 (único vector de pesos)
  + Por cada instancia de entrenamiento
    - Clasificar empleando los pesos actuales

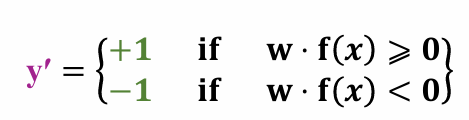
****

* + - Si es correcto (y’=y) no cambiar
    - Si es incorrecto ajustar (actualizar) el vector de pesos
      * Restar si la clase y (la real) es-1
      * Sumar si la clase y (la real) es +1
    - Entonces si y´ ha sido 1 y y es -1 w=w-f(x)
    - Entonces si y’ ha sido -1 y y es 1 w=w+f(x)

****

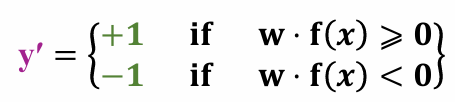
**Aprendizaje: Producto escalar de vectores y su significado**

* Producto escalar de 2 vectores Distancia del coseno
* Proyección de un vector sobre otro Coseno
* Por cada instancia de entrenamiento
  + Clasificar empleando los pesos actuales

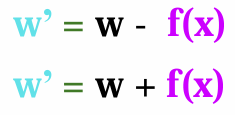


**Aprendizaje: Perceptrón binario**

* Inicializar pesos = 0
* Por cada instancia de entrenamiento
  + Clasificar empleando los pesos actuales

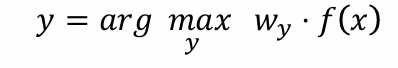
****

* + Si es correcto (y’=y) no cambiar
  + Si es incorrecto ajustar (actualizar) el vector de pesos sumando o restando el vector de las características
  + Restar si la clase y (real) es-1
  + Sumar en caso contrario



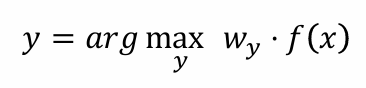
**Fronteras en decisiones multiclase**

* En clasificación multiclase:
  + Un vector de pesos por cada clase: Wy
  + Score (activación) de la clase y: Wy\*f(x)
  + Predicción con mayor score gana



**Aprendizaje Perceptron multiclase**

* Inicializar los pesos = 0
* Por cada ejemplo
  + Clasificar empleando los pesos actuales

****

* + Si es correcto No cambiar
  + Si es incorrecto
    - Decrementar el score de la incorrecta -> Wm’=Wm -f(x)
    - Aumentar el de la correcta -> Wc=Wc+f(x)

**Propiedades del perceptrón**

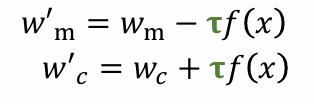
* **Separabilidad** Existen vectores de pesos tal que los ejemplos del entrenamiento queden correctamente clasificados
* **Convergencia** Si los ejemplos del conjunto de entrenamiento son separables, el perceptrón convergerá
* **Limite de error** El numero máximo de errores asociados al margen o grado de separabilidad

**Problemas con el perceptrón**

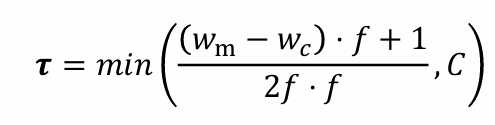
* **Ruido** Si los datos no son separables, pesos basura
  + Obtener medias de los vectores de pesos puede ayudar (averaged perceptron)
  + Generalización mediocre Encuentra una solución separable
  + Sobreajuste Los resultados en el test suelen ser peores

**Arreglando el perceptrón MIRA**

* ¿Otra solución? Ajustar la actualización de pesos para mitigar estos efectos adversos
* MIRA (Margin Infused Relaxed Algorithm) elegir una actualización que arregle la errónea clasificación del ejemplo actual de entrenamiento…. Pero que minimice el cambio sobre w



* En la practica hacer actualizaciones demasiado generosas no es apropiado
  + El ejemplo suele ser incorrectamente etiquetado
  + Solución capar el valor máximo de t (tau) con alguna constante c



* + Normalmente converge antes que el perceptrón
  + Normalemente es mejor,especialmente frente a casos ruidosos