Pontificia Universidad Católica de Chile
Escuela de Ingeniería, Depto. de Ciencia de la Computación
IIC-2613 Inteligencia Artificial

29 Noviembre, 2016.

Examen

Nombre:_____

a. Tiempo: 8:35 a 11:35.

b. Está permitido usar todo tipo de material personal.

c. **Sólo** debe responder 8 de las 9 preguntas, con la siguiente restricción: la pregunta 1 NO es opcional y debe ser contestada por todos los alumnos.

d. Conteste en el espacio proporcionado para cada pregunta. Use el reverso de las hojas. Para facilitar el proceso de corrección, **conteste cada pregunta en hojas independientes**. **No olvide poner nombre a todas las hojas**.

e. A excepción de cuando el profesor o los ayudantes indiquen, cualquier tipo de comunicación con otra persona que habite este planeta será considerada como caso de copia. Connecciones espirituales o con otros mundos son bienvenidas.

Pregunta	Puntaje Máximo	Puntaje Obtenido
P1	14	
P2	12	
P3	12	
P4	12	
P5	12	
P6	12	
P7	12	
P8	12	
P9	12	
	TOTAL	
	NOTA FINAL	

1. (*14 puntos*) **Responda indicando si la expresión es verdadera o falsa, o respondiendo directamente la consulta respectiva. En todo los casos fundamente brevemente su respuesta.**

1.1. El conjunto de fórmulas $\{\forall x \forall y\, x = y, P(a) \wedge \neg P(b)\}$ es inconsistente.

Solución: *Verdadero. La primera fórmula obliga a tener un dominio con un solo elemento, mientras que la segunda requiere un dominio de al menos dos elementos para satisfacerse.*

1.2. Sea φ una oración arbitraria y Σ un conjunto de oraciones arbitrario, y sea $Skolem[\varphi, \Sigma]$ la versión *skolemizada* de φ con respecto a un conjunto de fórmulas Σ. Entonces $\{\varphi\} \models Skolem[\varphi, \Sigma]$.

Solución: *Falso. Para verlo basta con tomar* $\Sigma = \{\}$ *y* $\varphi = \exists x\, P(x)$

1.3. Sea c una constante arbitraria. Entonces Prolog responde `true` a la consulta $p(c)$ con este programa:

```
p(X) :- q(X).
q(a).
```

si y solo si $\{Q(x) \rightarrow P(x), Q(a)\} \models P(c)$.

Solución: *Verdadero si interpretamos* $Q(x) \rightarrow P(x)$ *como una oración cuantificada universalmente. Cuando* $c = a$ *tenemos que* $P(a)$ *es consecuencia lógica (y la consulta entrega* `true`*). Cuando* $c \neq a$ $P(c)$ *no es consecuencia lógica y la consulta no entrega* `true`*.*

1.4. La tercera respuesta entregada por el intérprete de Prolog para la consulta `p(X)` sobre el programa de abajo, es `p(2)`.

```
p(0).
p(N) :- p(N-1).
```

Solución: *Falso. Ante la consulta, Prolog entrega una respuesta* `p(0)` *(dada por* `X = 0`*) y luego entra en un loop infinito*

1.5. Sea P un problema de búsqueda y h una heurística admisible para P. Al resolver P con A* y $2h$ como heurística, no es posible encontrar una solución óptima.

Solución: *Falso. Sí es posible. Hay muchos ejemplos sencillos que se pueden construir con espacios de búsqueda pequeños; por ejemplo uno con solo dos estados. (Se espera que el alumno dé uno o justifique correctamente por qué).*

1.6. El algoritmo de búsqueda de profundización iterativa (también conocido como Iterative Deepening Search o Iterative Deepening Depth-First Search) puede necesitar expandir menos nodos que Breadth-First Search al resolver el mismo problema de búsqueda, incluso cuando ambos algoritmos generan nodos en el mismo orden.

Solución: *Falso. El algoritmo de profundización iterativa expande al menos tantos nodos como BFS (y generalmente muchos más).*

1.7. El sobreajuste o overfitting se caracteriza por aumentar las capacidades inductivas de generalización de un algoritmo de aprendizaje de máquina.

Solución: *Falso, justamente lo contrario, el sobreajuste se manifiesta cuando el modelo pierde capacidad de generalización y se centra en modelar efectivamente sólo el set de entrenamiento.*

1.8. Para red neuronal del tipo feedforward con una capa oculta, el algoritmo de backpropagation garantiza alcanzar un 100% de exactitud en la clasificación para cualquier set de training que sea linealmente separable (asuma suficiente tiempo de entrenamiento y un coeficiente de aprendizaje pequeño).

Solución: *Falso, la solución inicial esta dada por pesos aleatorios y el algoritmo de backpropagation puede quedar atrapado en óptimos locales, lo cual no permite garantizar encontrar la solución óptima.*

1.9. Para toda expresión de lógica proposicional siempre es posible construir un árbol de decisión que, usando las mismas variables booleanas, permita representar la misma función lógica.

Solución: *Verdadero, un árbol de decisión puede modelar cualquier disjunción de conjunciones, lo cual a su vez puede modelar cualquier función de lógica proposicional.*

1.10. En un clasificador de Naive Bayes, asumiendo que el supuesto de Naive Bayes es verdadero, la hipótesis seleccionada por el clasificador es la que alcanza el mejor rendimiento en el set de entrenamiento.

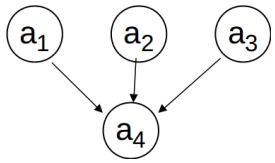
Solución: *Falso, el clasificador Naive Bayes entrega la hipótesis MAP (máxima verosimilitud), es decir, la hipótesis que tiene la mayor probabilidad de ser la correcta, y por ende entregar el mejor rendimiento. Esto es la maximización de una probabilidad (hipótesis más probable), lo cual no garantiza que en la práctica la hipótesis encontrada alcance el mejor rendimiento en el set de entrenamiento.*

1.11. En un clasificador de Naive Bayes, asumiendo que el supuesto de Naive Bayes es verdadero, la hipótesis seleccionada por el clasificador es la que alcanza el mejor rendimiento en el set de test.

Solución: *Falso, ningún algoritmo de aprendizaje de máquina puede garantizar el mejor rendimiento en datos que nunca ha visto.*

1.12. Un red de Bayes con 4 atributos binarios corresponde a la siguiente factorización de la probabilidad conjunta: $p(a_1, \ldots, a_4) = p(a_1)p(a_2)p(a_3)p(a_4|a_1, a_2, a_3)$. La red resultante tiene 4 nodos y 4 conexiones (links).

Solución: *Falso, la figura muestra el modelo gráfico de la red con 4 nodos y 3 conexiones.*



1.13. Para la red anterior, el número de parámetros es 11.

Solución: *Falso, tenemos los parámetros de $p(a_1)$,$p(a_2)$ y $p(a_3)$, cada una de estas funciones con 2 parámetros (atributos binarios). Además, se necesita estimar $p(a_4|a_1, a_2, a_3)$, lo cual implica 16 parámetros (a_4 es binaria y hay 8 posibles condicionantes). Por tanto, el total de parámetros es 22.*

1.14. En un árbol de decisión que obtenga un rendimiento de 100% en el set de entrenamiento, no es necesario podar el árbol resultante.

Solución: *Falso, el obtener 100% de efectividad en set de entrenamiento puede ser un sintoma de problemas de sobreajuste, por tanto, podar el árbol puede ser algo necesario.*

2. (*12 puntos*)

a. Sea $\mathcal{P}$ la familia de problemas de búsqueda en donde los costos de las acciones son uniformes (es decir, son todos iguales). Muestre que existen infinitos problemas en $\mathcal{P}$ en donde Breadth-First Search funciona *significativamente mejor*—en términos del número de estados que se expanden—que A* usado con $h = 0$.

Solución: *BFS retorna al momento de generar un estado objetivo (es decir, justo antes de agregar un estado objetivo a la frontera de búsqueda). A* en cambio, retorna al momento de extraer un nodo de Open. Así, por ejemplo, es sencillo ver que cualquier espacio de búsqueda con factor de ramificación al menos 2, A* (con $h = 0$) deberá expandir muchos más nodos que BFS (la diferencia de estados expandidos es de hecho exponencial en la profundidad de la solución).*

b. Use resolución para demostrar que $\{\exists y \forall x\, P(x, y)\} \models \forall x \exists y\, P(x, y)$.

Solución: *Generamos el conjunto de fórmulas* $\{\exists y \forall x\, P(x, y), \exists x \forall y\, \neg P(x, y)\}$. *Luego de skolemizar tenemos* $\{P(x, C_1), \neg P(C_2, y)\}$, *y luego para resolver debemos usar la sustitución* $\theta = \{x/C_2, y/C_1\}$, *desde donde se obtiene la cláusula vacía.*

3. (*12 puntos*)

a. Escriba el predicado Prolog `contenida_en(L1,L2)` que se satisface cuando cada elemento de la lista `L1` está en la lista `L2`.

Solución: *Solución:*

```
contenida_en(L1,L2) :- \+ (member(X,L1),\+ member(X,L2)).
```

b. Escriba un predicado `p` tal que, cuando se hace la consulta `p(X)` al intérprete se obtengan la siguiente secuencia infinita de respuestas.

```
X = 0 ;
X = s(0,0) ;
X = s(1,0) ;
X = s(2,0) ;
X = s(0,s(0,0)) ;
X = s(1,s(0,0)) ;
X = s(2,s(0,0)) ;
X = s(0,s(1,0)) ;
X = s(1,s(1,0)) ;
X = s(2,s(1,0)) ;
X = s(0,s(2,0)) ;
...
```

Solución: *Solución:*

```
p(0).
p(s(X,S)) :- p(S),member(X,[0,1,2]).
```

4. (*12 puntos*)
Minimax es un algoritmo extremadamente sencillo de programar. De hecho, este es un pseudocódigo (en la versión que se detiene en las hojas).

```
Minimax(nodo,d):  # recibe un nodo y una profundidad d
                  # retorna una tupla valor, indice, donde valor
                  # valor

   if es_hoja(nodo):  # aca retorna 1 o -1 dependiendo si gano yo u oponente
      if gano_yo(nodo):
          return 1,-1
      elif gano_el(nodo):
          return -1,-1
      else:
          return 0,-1   # empate

   if d%2 == 0:     # escojo si hago min o max
      funcion = max
      valor = -1000   # valor almacena el maximo (-infinito)
   else:
      funcion = min
      valor = 1000   # valor almacena el minimo (+infinito)

   for i in range(numero_jugadas_posibles(nodo)):
      hijo  = sucesor(i,nodo)  # hijo tiene al sucesor i-esimo de nodo
      valor_hijo, jugada = Minimax(hijo,d+1)
      valor_nuevo = funcion(valor,valor_hijo)
      if valor_nuevo != valor:
         valor = valor_nuevo
         mejor_jugada = i

   return valor,mejor_jugada
```

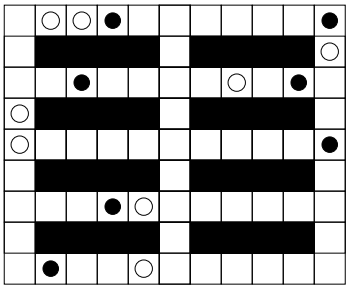
Una desventaja de este algoritmo es que, cuando se sabe ganador, es decir, cuando el llamado a `Minimax(nodo,0)` retorna una tupla en donde la primera componente tiene valor 1, el algoritmo simplemente retorna "cualquier jugada" de aquellas que garantizan que el juego se ganará en el futuro.
A veces esto se considera indeseable: uno quisiera retornar aquella jugada que *"liquidará al oponente lo más rápido posible"*. Así, queremos que el algoritmo retorne una jugada que garantice que el juego va a "durar menos" en vez de "durar más".
Muestre que esto es fácil de hacer definiendo un segundo criterio a considerar en la minimización y maximización. Modifique el pseudocódigo de arriba para lograrlo o describa, en detalle, cómo se debiera modificar.

Solución: *Además de un valor, ahora queremos que cada nodo tenga asociada una profundidad. Para un nodo n, dicha profundidad corresponde, intuitivamente, a qué tan abajo de n está el nodo*

más cercano que "justifica" el valor de n. Entonces para modificar el código hacemos lo siguiente.

- *Cuando encontramos una hoja retornamos profundidad 0.*
- *Bajo un nodo MAX n, si $n_1, \ldots, n_k$ son aquellos con el máximo valor, v, y $p_1, \ldots, p_k$ son las profundidades asociadas, entonces retornamos el valor v y la profundidad* $1 + \min\{p_1, \ldots, p_k\}$.
- *Bajo un nodo MIN n, si $n_1, \ldots, n_k$ son aquellos con el mínimo valor, v, y $p_1, \ldots, p_k$ son las profundidades asociadas, entonces retornamos el valor v y la profundidad* $1 + \max\{p_1, \ldots, p_k\}$ *(si suponemos que nuestro adversario querrá maximizar la duración del juego),* $1 +$ *avg* $\{p_1, \ldots, p_k\}$ *(si suponemos que nuestro adversario no está preocupado de la profundidad) o* $1 + \min\{p_1, \ldots, p_k\}$*, si suponemos que nuestro adversario tambien quiere minimizar la duración del juego.*

5. (*12 puntos*) Considere problema de mover N agentes en una habitación con pasillos como la que se muestra en la figura. Específicamente, el problema consiste en llevar a cada agente desde cierta posición inicial hasta cierta posición final. Los agentes se pueden mover en paralelo. Suponga, además, que una solución es mejor que otra cuando se minimiza el tiempo que toma llevar a todos los agentes a su posición final.



a. Observando que este problema se puede resolver usando A* con "acciones paralelas", diga qué heurística se puede usar y muestre que el factor de ramificación es exponencial en N.

Solución: *Una heurística admisible es el máximo de la distancia entre cada agente y su posición final. El factor de ramificación es exponencial. Suponiendo que los agentes separados, cada agente tiene 2 acciones posibles, por lo tanto existen* 2^N *acciones paralelas.*

b. ¿Cómo es posible resolver este problema usando A* con un factor de ramificación *lineal* en N? Escriba un argumento que convenza a un experto que su enfoque encuentra soluciones óptimas. **Solución:** *Es posible usar A* con múltiples agentes usando un sistema de turnos.*

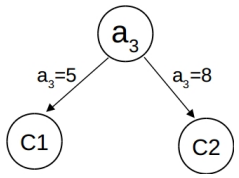
Entonces damos la oportunidad a cada agente para que se mueva, uno después del otro. La idea es que luego que los N *turnos están completos aquellas* N *acciones serán interpretadas como una sola. Para que esto funcione, sin embargo, después que los* N *turnos están completos [falta algo de detalle, pero lo dejo hasta acá por ahora...]*

6. (*12 puntos*)

a. (*4 pts*) Dibuje el árbol de decisión resultante para el siguiente set de datos de entrenamiento:

Atributo-1	Atributo-2	Atributo-3	Atributo-4	Clase
4	-4	5	25	1
8	-5	8	25	2
5	-45	5	25	1
17	-45	8	25	2
30	-45	5	25	1

Solución: *Este problema se puede resolver por simple inspección de los datos. El atributo 3 divide en forma perfecta los registros según el valor de la clase, por tanto, el árbol resultante es:*

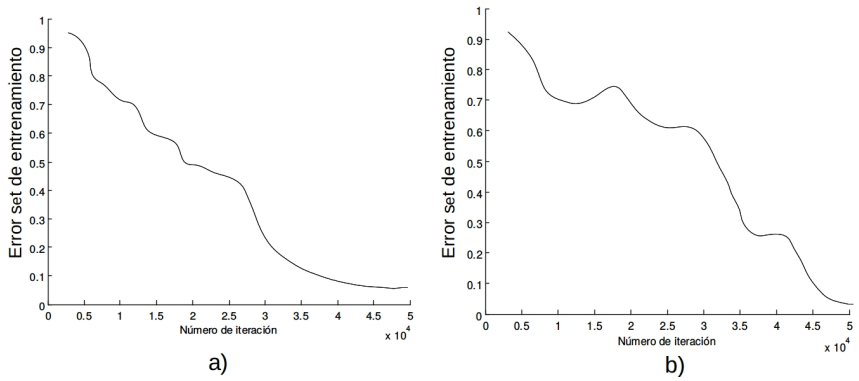


b. (*4 pts*) Dibuje el árbol de decisión resultante para el siguiente set de datos de entrenamiento:

Atributo-1	Atributo-2	Atributo-3	Atributo-4	Clase
3	4	5	6	1
3	4	5	6	1
3	4	5	6	1
3	4	5	6	1
3	4	5	6	1

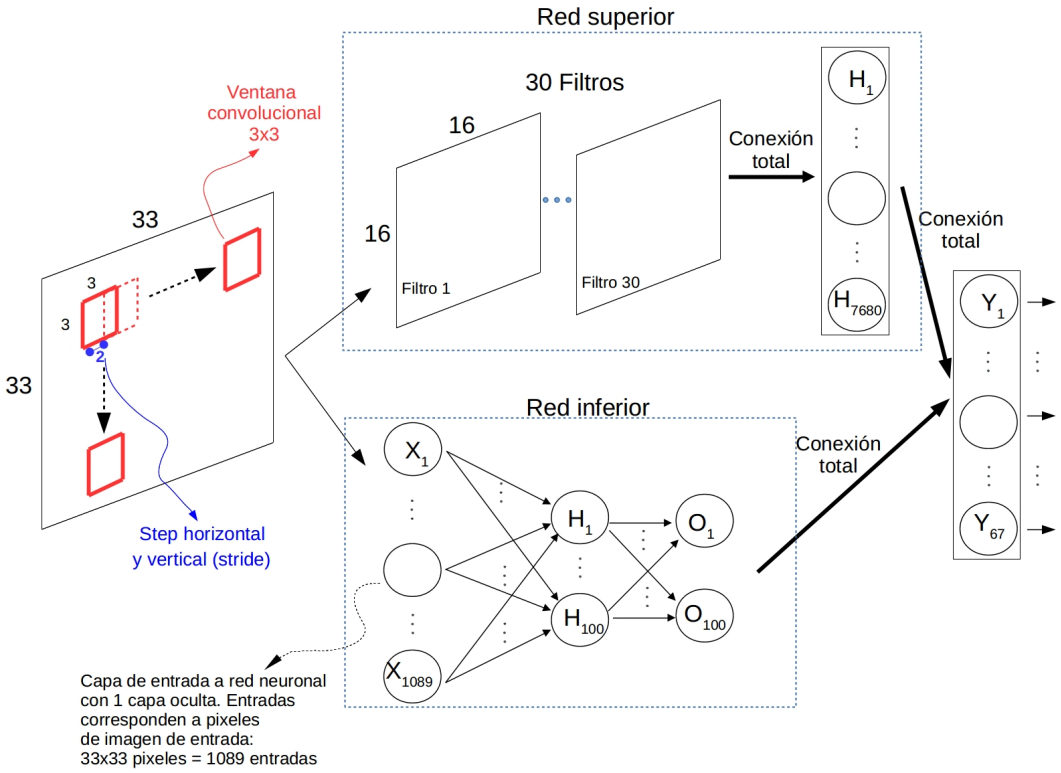
Solución: *Todos los registros tienen el mismo valor de la clase, por tanto, el árbol corresponde sólo a un nodo hoja indicando clase 1.*

c. (*4 pts***)** Las siguientes figuras muestran la evolución del error de un algoritmo de aprendizaje de máquina a medida que itera en su exploración del espacio de hipótesis. ¿Cuál de los **algoritmos de aprendizaje de máquina vistos en clase** y **técnica de entrenamiento** pueden haber generado cada curva?, justifique su respuesta.



Solución: *En clases el único algoritmo que iteraba para explorar el espacio de hipótesis era la red neuronal. Los otros algoritmos que revisamos resolvían directamente el problema de optimización planteado, de manera de encontrar la mejor hipótesis según la métrica de rendimiento utilizada. Según lo anterior:*
Curva a): En cada iteración, el error en el set de entrenamiento siempre disminuye, esto indica un posible uso de backpropagation utilizando el método batch de entrenamiento.
Curva b): El hecho que en algunas iteraciones el error en el set de entrenamiento aumente levemente indica un posible uso de backpropagation utilizando el método incremental de entrenamiento (gradiente estocástico).

7. (*12 puntos*) Se tiene la siguiente red neuronal que opera sobre imágenes de entrada de 33x33 pixeles.



En el esquema, dada la imagen de entrada de 33x33 pixeles:

- La parte superior corresponde a una red neuronal convolucional de 30 filtros, donde el mapa de activación de cada filtro es generado por la convolución de ventanas de 3x3 y un paso (stride) horizontal y vertical de 2 pixeles. El producto de cada convolución es luego procesado por una función de activación sigmoidal. **Obs:** note que no se aplica ningún paso de max-pooling.
- La parte inferior corresponde a una red tipo feed-forward de conexión total con 1 capa oculta, donde la capa oculta y la capa de salida tienen 100 neuronas cada una.
- Las salidas de las redes superior e inferior son luego concatenadas para pasar por un perceptron con función de activación sigmoidal y 67 neuronas como salida.

Indique:

a. (*4 pts*) ¿Cuál es el número total de neuronas de la red?. En su cuenta, para el caso de la red neuronal inferior de capa oculta, no considere como neuronas las 1089 entradas correspondientes a los pixeles de imagen de entrada (nodos X_1 a X_{1089}).

Solución:
Para la red superior se tiene:

- Cada filtro se compone de 16x16 neuronas, por simplicidad este dato estaba indicado en la figura, pero también se podía deducir del número de posiciones posible de la ventana de 3x3 usando un paso de 2 pixeles.
- Hay 30 filtros, por tanto, el número de neuronas en los pasos convolucionales es 16x16x30=7680 neuronas.
- Luego del paso convolucional hay un etapa de conexión total a otras 7680 neuronas.
- Por tanto, el número total de neuronas en la red superior es= 7680+7680=15360 neuronas.

Para la red inferior se tiene:

- 100 neuronas en capa oculta + 100 neuronas en capa de salida.
- Por tanto, el número total de neuronas en la red inferior es= 100+100=200 neuronas.

La salida de la red agrega otras 67 neuronas, por tanto, el número total es: 15360+200+67=15627 neuronas.

b. (*4 pts*) ¿Cuál es el número total de parámetros de la red?. Por simplicidad, considere que no se incluye en cada neurona una entrada constante de bias.

Solución:
Para la red superior se tiene:

- Cada filtro convolucional tiene 3x3=9 parámetros (pesos de la red).
- Hay 30 filtros, por tanto, el número de parámetros de la etapa convolucional 3x3x30=270 parámetros.
- Luego del paso convolucional hay un etapa de conexión total entre las 7680 neuronas convolucionales y otras 7680 neuronas. Por tanto, se agregan 7680x7680 parámetros.
- Por tanto, el número total de parámetros de la red superior:7680x7680+270.

Para la red inferior se tiene:

- 1089 entradas que se conectan con 100 neuronas en capa oculta, lo cual implica 1089x100 parámetros.

- Las 100 neuronas en capa oculta se conectan con 100 neuronas en capa de salida, lo cual implica 100x100 parámetros.
- Por tanto, el número total de parámetros de la red inferior es :1089x100+100x100.

La salida de la red superior 7680 neuronas se concatenan con las 100 salidas de la red inferior para conectarse con 67 neuronas en capa de salida. Esto implica (7680+100)*67 nuevos parámetros. Por tanto, el número total de parámetros de la red en la figura es: 7680x7680+270 + 1089x100+100x100 + (7680+100)*67 parámetros.

c. (*4 pts***)** ¿Qué tipo de información codifican respectivamente las partes superior e inferior de la red?, justifique brevemente su respuesta.

Solución:
Las neuronas en la red superior pertenecen a mapas convolucionales, donde cada filtro ve un área local de la imagen de entrada, ventanas de 3x3 pixeles. Por tanto, estos filtros aprenden a reconocer patrones locales de la imagen de entrada, como bordes o otro tipo de patrón local.
Por su parte, las neuronas en la red inferior reciben información de todos los pixeles de la imagen (mapa holístico), por tanto, aprenden a reconocer patrones generales presentes en la entrada, como texturas globales.

En resumen, la red superior codifica información local, mientras que la red inferior codifica información global.

8. (*12 puntos*)

a. (*6 pts*) Considere la formulación vista en clases para el clasificador binario SVM utilizando variables de slag ξ_k.

$$\underset{w_1,w_0,\xi_k}{\operatorname{argmin}} \ \frac{1}{2}||w_1||^2 + C\sum_{k=1}^{K}\xi_k; \text{ with } \xi_k = |z_k - g(x_k)|$$
$$\text{subject to: } z_k(w_1\phi(x_k) + w_0) \geq 1 - \xi_k, \ \xi_k \geq 0, \ k = 1 \ldots N.$$

Considere también el error de este clasificador utilizando la estrategia de evaluación de rendimiento denominada Leave-One-Out (LOO). Esta estrategia consiste en entrenar el clasificador N veces, usando N-1 instancias para el entrenamiento y la instancia restante para evaluar, de ahí el nombre leave-one-out. Una condición importante es que en cada evaluación se deja afuera una instancia distinta para evaluar. Después de este proceso de N entrenamientos y evaluaciones, la evaluación final del rendimiento del clasificador queda dada por el promedio de las N evaluaciones.

Demuestre que para un problema de clasificación binaria, donde las variables de slag siempre cumplen $\xi_{k'} < 1, \forall k'$ que no es un vector de soporte, y el valor de la constante C asociada a las slags es fijo, se cumple:

$$Error_{LOO} \leq \frac{|SVs|}{N}$$

donde $Error_{LOO}$ corresponde al error del clasificador SVM utilizando la estrategia LOO y $|SVs|$ indica el número de vectores de soporte que se obtiene al entrenar el clasificador usando las N instancias de entrenamiento.

AYUDA: Recuerde que la solución proporcionada por un clasificador SVM (hiperplano separador) queda totalmente determinada por los vectores de soporte.

Solución: *La clave de la demostración es lo que se indica en la "ayuda", es decir, que el hiperplano separador queda totalmente determinado por los vectores de soporte. Una posible demostración sigue la siguiente argumentación:*

- **a.** Sea D_N el set completo de N ejemplos de entrenamiento; F^* la solución utilizando todos los datos de entrenamiento; y SVM^* el set de vectores de soporte que determinan las posición del hiperplano F^*.
- **b.** Al remover de D_N un ejemplo $x_{k'} \notin SVM^*$, es decir, un ejemplo de entramiento que no está en el set de vectores de soporte que determina F^*, y usar el set resultante para entrenar un SVM, la solución encontrada también es F^*. Esto pues al agregar x'_k la solución no cambia ($x_{k'} \notin SVM^*$).

c. Adicionalmente, la condición $\xi_{k'} < 1, \forall k'$, garantiza que estos puntos fueron bien clasificados. Por ende, los únicos casos de posibles errores, son los entrenamientos donde LOO dejó fuera un vector de soporte, lo que implica que el número máximo de errores es $|SVs|$.

d. De las 2 condiciones anteriores se tiene que el error máximo de LOO es acotado superiormente por $|SVs|/N$, i.e.:

$$Error_{LOO} \leq \frac{|SVs|}{N}$$

b. (*6 pts*) En el contexto del algoritmo de clasificación de Naive Bayes, se tiene un problema de clasificación de 10 clases y 20 atributos a_i, $i \in [1 \dots 20]$. Un estudio inicial revela que el supuesto de Naive Bayes es válido y el número de datos de entrenamiento disponible es adecuado.

Adicionalmente se descubre que es posible obtener una mejor estimación de la probabilidad apriori de cada clase objetivo c_k, $k \in [1 \dots 10]$, si independiente se aplica un segundo modelo del tipo Naive Bayes para su estimación. Este segundo modelo también utiliza el supuesto de Naive Bayes, y construye una estimación en base a un set de datos de 10 atributos b_j, $j \in [1 \dots 10]$. En sus ecuaciones de inferencia, este segundo modelo de Naive Bayes estima el valor de la probabilidad apriori de cada clase utilizando una distribución uniforme (i.e., probabilidad apriori de $c_k = p(c_k) = 1/K$).
Indique las ecuaciones relevantes que utiliza el primer modelo para obtener la clasificación de la siguiente instancia de test: $(\hat{a}_1, \dots, \hat{a}_{20}, \hat{b}_1, \dots, \hat{b}_{10})$.

Solución:
Para el problema original las ecuaciones relevantes son:

$$c* = \underset{c_k}{\operatorname{argmax}} \hat{P}(c_k) \prod_{i=1}^{20} P(a_i|c_k), \quad k \in [1 \dots 10]$$

El problema secundario nos permite obtener una estimación del prior $\hat{P}(c_k)$, la ecuación relevante es:

$$\hat{P}(c_k) = \frac{1}{K} \prod_{j=1}^{10} P(b_j|c_k)$$

9. (*12 puntos*) Se tiene un problema de clasificación en que el set de datos de entrenamiento consiste de imágenes de 256x256 pixeles, en que cada imagen tiene además asociada una leyenda textual que proporciona información adicional. La siguiente figura muestra un ejemplo de una posible instancia de entrenamiento.

Figura 1. Estudiantes de IA muy concentrados tomando el examen final.

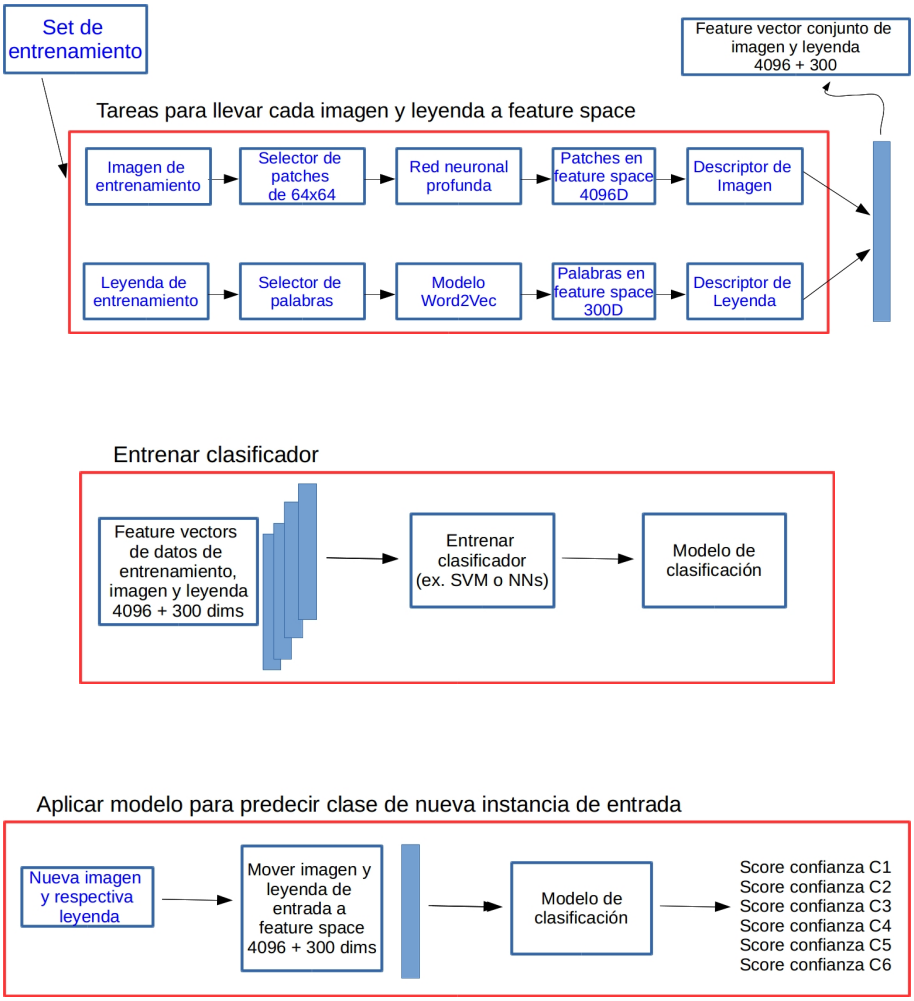
Al igual que en las tareas 2 y 3, asuma que además del set de entrenamiento, cuenta con lo siguiente:

- Una red neuronal profunda, que dada una imágen de entrada de 64x64 pixeles permite obtener un vector de características de 4096 dimensiones (visual feature space).
- Un modelo Word2Vec pre-entrenado con los textos del set de entrenamiento, el cual permite mapear cada posible palabra de entrada a un vector de características de 300 dimensiones (textual feature space).

Las imágenes y leyendas del set de entrenamiento provienen de 6 posibles dominios: Educación, Deportes, Política, Economía, Turísmo y Otro. El objetivo es obtener un clasificador que permita predecir el rótulo de nuevas imágenes de entrada con su respectiva leyendas, las cuales no son parte del set de entrenamiento (set de test).

a. (*6 pts*) Realice un diagrama que muestre los principales pasos de una posible solución al problema planteado.

Solución: Las figuras muestran el diseño de una posible solución, hay diversas variantes posibles que también serán consideradas como correctas. En general, se tendrá como guía el trabajo que realizaron en las tareas 2 y 3 del curso.

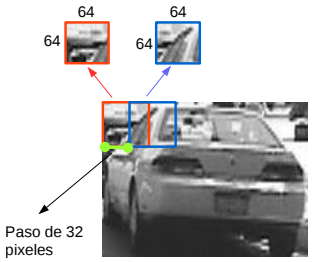


b. (*6 pts*) Describa brevemente la función realizada por cada uno de los pasos indicados en su solución. Sea breve y preciso.

Solución:

1. Tareas para llevar cada imagen y leyenda a feature space

- La primera gran tarea es llevar los datos de entrenamiento a feature space. En el caso de las imágenes, tal como en la tarea 2, se utiliza el modelo de deep learning. Para esto se seleccionan patches de 64x64, mediante un barrido horizontal y vertical de la imagen, tal como muestra la siguiente:



- Cada uno de estos patches es luego transformado al espacio de 4096D mediante la red neuronal profunda.
- Luego, se obtiene el descriptor de cada imagen integrando la información de sus patches, por ejemplo, usando max-pooling.
- En el caso del texto, cada palabra de la leyenda es transformada a feature space usando el modelo Word2Vec. Los vectores de cada palabra son luego integrados usando max-pooling para obtener el descriptor de la leyenda.
- Finalmente, el descriptor de cada ejemplo de entrenamiento queda dado por la concatenación de los vectores de características de la imagen y su respectiva leyenda.

2. Entrenar clasificador

- Los vectores de características de todos los ejemplos de entrenamiento son usados para entrenar el modelo de clasificación. Por ejemplo, una máquina de vectores de soporte o una red neuronal.
- El resultado de esta etapa es un modelo de clasificación que puede ser usado para clasificar nuevas instancias.

3. Aplicar modelo para predecir clase de nueva instancia de entrada

- Para clasificar nuevas instancias, el primer paso es mover la instancia de entrada al espacio de características, utilizando los pasos descritos en el punto 1.
- La instancia en espacio de características es ingresada al modelo para obtener el score de clasificación de las 6 clases posibles: Educación, Deportes, Política, Economía, Turísmo y Otro.