Combinación de Clasificadores

Parte 2: Random Forests, Gradient Boosting y Mezcla de Expertos

Reconocimiento de Patrones

Departamento de Procesamiento de Señales
Instituto de Ingeniería Eléctrica
Facultad de Ingeniería, UdelaR

2018

Bishop, cap. 14 – Hastie, Tibshirani & Friedman: cap. 10 (secs. 10.9 -10.12), cap. 15 (hasta 15.3 inclusive)

Agenda

Random Forests:

- Bagging con árboles de decisión;
- Para hacer los árboles menos correlacionados, cada árbol se construye sobre una selección aleatoria de atributos;
- Muy popular: en general, performance muy similar a boosting, pero más fácil de entrenar.

Gradient Boosting:

- Surge de la observación que boosting puede intepretarse como un algoritmo de optimización de una función de costo debidamente diseñada.
- Clasificador basado en regresión. Se va refinando iterativamente agregando modelos de regresión que ajustan a la dirección opuesta del gradiente del costo.

Mezcla de expertos:

- Combinación ponderada de clasificadores que deciden de forma independiente;
- Los asignación de los pesos (o nivel de confianza) de clasificadores varía con la muestra específica a clasificar (la expertise depende de la entrada).

Random Forests (Breiman, 2001)

Sobre los clasificadores base, vimos que:

- Conviene combinar clasificadores débiles (sesgo bajo, varianza alta);
- A mayor independencia entre clasificadores base, menor error de clasificación en la combinación.

Random Forests:

- Modificación de bagging clásico con árboles de decisión, mediante una técnica que reduce la correlación entre los árboles.
- Cómo: construyendo cada árbol sobre un subconjunto de características seleccionadas al azar.

Random Forests: algoritmo

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Random Forests: por qué funciona

Los árboles de decisión generados por bootstrap son idénticamente distribuidos, con sesgo b y varianza σ^2, pero no independientes:

- El sesgo del promedio también vale $b_{RF} = b$;
- Si ρ es el coeficiente de correlación de Pearson de un par de árboles, la varianza del promedio de B árboles vale

$$\sigma^2_{RF} = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

- Al aumentar B el segundo término se reduce, pero no el primero.

$\Rightarrow$ Para reducir la varianza de la combinación, Random Forests propone:

- Diseñar el conjunto de árboles para reducir ρ, cuidando no aumentar mucho σ^2;
- Hacer esto construyendo cada árbol sobre un subconjunto aleatorio de m características entre el total de p.
- ¿Cómo elegir m? Breiman recomienda $m = \lfloor\sqrt{p}\rfloor$; mejor elegir por CV o OOB.

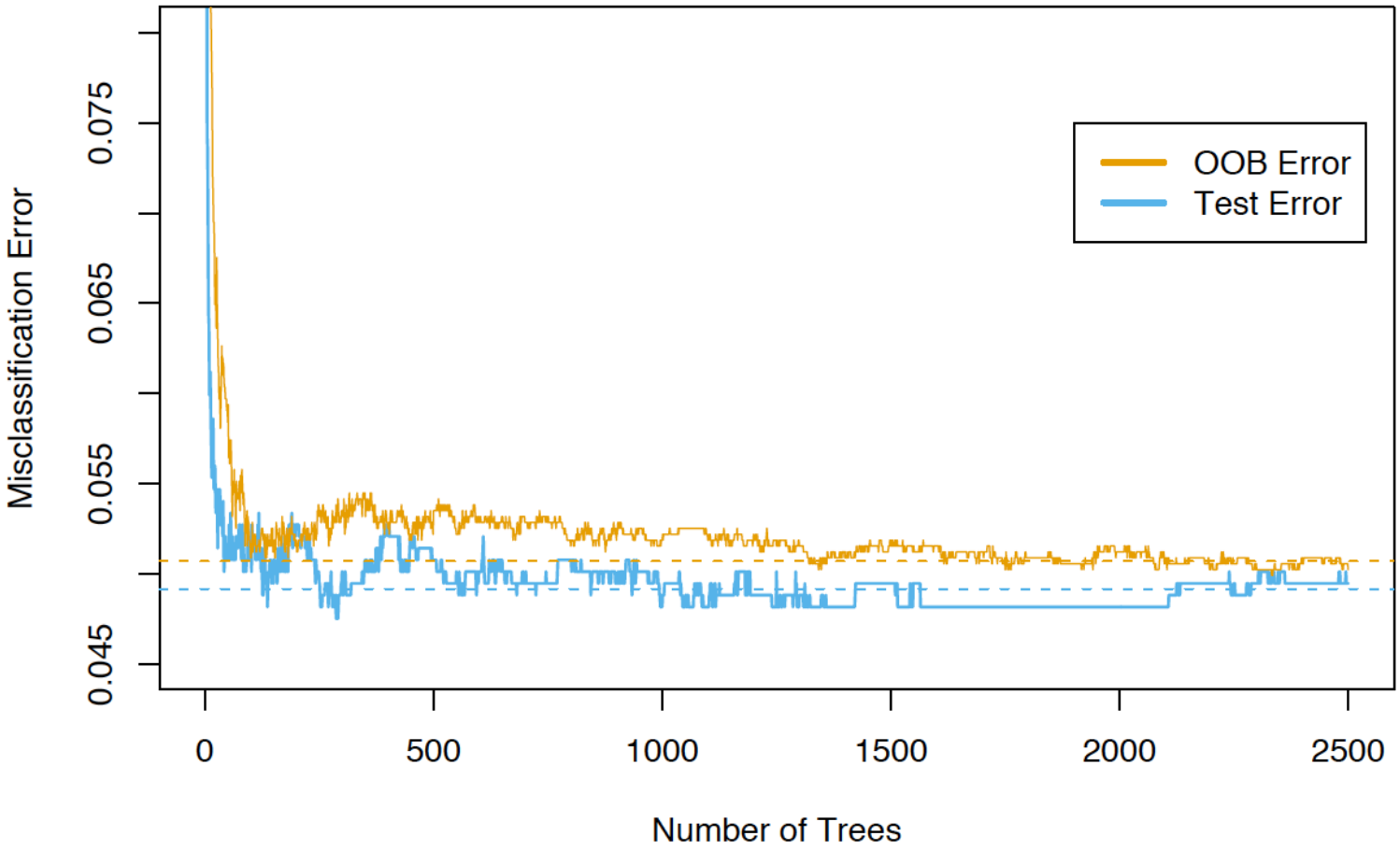
Estimación Out-of-bag (OOB)

Es un método para estimar el error de predicción en bagging, durante el entrenamiento, muy utilizado en Random Forests.

El error de predicción (o de generalización) se mide promediando los errores de predicción de las muestras OOB:

- Para cada muestra de entrenamiento $(\mathbf{x}_n, t_n)$, se construye su predictor RF promediando sólo los árboles de los muestreos bootstrap que no la contienen.
- El training se termina cuando el error OOB se estabiliza.
- Se puede ver que cuando B crece la estimación de error de generalización por OOB tiende a la de N-fold CV (N es el total de muestras de training).

Estimación Out-of-bag (OOB)



Error OOB estimado durante training, comparado con error en testing (dataset `spam`; Hastie, Tibshirani & Friedman).

Gradient Boosting (Friedman, 1999)

Gradient boosting nace de la observación siguiente (más intuitiva en regresión):

- Consideremos un predictor $y = f(\mathbf{x})$, y la función de costo cuadrática,

$$L(f) = \sum_{n=1}^{N} L(y_n, f(\mathbf{x}_n)) = \sum_{n=1}^{N} (y_n - f(\mathbf{x}_n))^2 .$$

- Supongamos que queremos optimizar f iterativamente, y llamemos $f^{(m)}$ el modelo en la iteración m. Para esto se busca un nuevo estimador h que refine el clasificador:

$$f^{(m+1)}(\mathbf{x}) = f^{(m)}(\mathbf{x}) + h(\mathbf{x}).$$

- Si h fuese perfecto, $f^{(m+1)}(\mathbf{x}) = f^{(m)}(\mathbf{x}) + h(\mathbf{x}) = y$, es decir $h(\mathbf{x}) = y - f^{(m)}(\mathbf{x})$.

- Entonces, para refinar f iterativamente, se busca que h se ajuste lo mejor posible al residuo $y - f_m(\mathbf{x})$. Observando que $y - f(\mathbf{x}) = -\frac{\partial}{\partial f}\left\{(y - f(\mathbf{x}))^2\right\}$, gradient boosting propone refinar f mediante un descenso por gradiente,

$$f^{(m+1)} = f^{(m)} - \eta \frac{\partial L}{\partial f}(f^{(m)}).$$

Gradient Boosting para clasificación

- N muestras $(\mathbf{x}_n, \mathbf{t}_n)$, K clases, $\mathbf{t}_n = (t_{n1}, \ldots, t_{nK})$, con $t_{nk} = \mathbf{1}_{\{\mathbf{x}_n \text{ de clase } k\}}$.

- Costo: entropía cruzada,

$$L(\mathbf{t}_n, f_1(\mathbf{x}_n), \ldots, f_K(\mathbf{x}_n)) = -\sum_{k=1}^{K} t_{nk} \log p_k(\mathbf{x}_n),$$

  con $p_k(\mathbf{x}) = \frac{e^{f_k(\mathbf{x})}}{\sum_{l=1}^{K} e^{f_l(\mathbf{x})}}$ (softmax).

- Tenemos:

$$\nabla_{f_k} L(\mathbf{t}_n, f_1(\mathbf{x}_n), \ldots, f_K(\mathbf{x}_n)) = t_{nk} - p_k(\mathbf{x}_n),$$

  (cuantifica el error de clasificación: si $\mathbf{x}_n$ es de clase K, $p_k(\mathbf{x}_n)$ debería estar cerca de 1).

Gradient Boosting: algoritmo para clasificación

Algorithm Gradient Boosting para K clases

Ajustar un modelo de regresión inicial $f_k^0(\mathbf{x})$ para cada clase $k = 1, \ldots, K$.
for $m = 1$ to M **do**
  for $k = 1$ to K **do**
    for $n = 1$ to N **do**
      Calcular $p_k^{(m)}(\mathbf{x}_n) = \frac{e^{f_k^{(m)}(\mathbf{x}_n)}}{\sum_{l=1}^{K} e^{f_l^{(m)}(\mathbf{x})}}$.
      Calcular los residuos

$$r_{nmk} = -\nabla_{f_k^{(m)}} L(\mathbf{t}_n, f_1^{(m)}(\mathbf{x}_n), \ldots, f_K^{(m)}(\mathbf{x}_n)) = -(t_{nk} - p_k^{(m)}(\mathbf{x}_n)).$$

    end for
    Ajustar un modelo de regresión $h_{mk}(\mathbf{x})$ al conjunto $\{\mathbf{x}_n, r_{nmk}\}_{n=1}^{N}$.
    Actualizar f_k: $f_k^{(m)} = f_k^{(m-1)} + \eta_{mk} h_{mk}$.
  end for
end for
Return: modelos de regresión $f_1^{(M)}, f_2^{(M)}, \ldots, f_K^{(M)}$.

Clasificación: En testing, $\mathbf{x}$ se asigna a la clase $k^* = \text{argmax}_{k \in \{1,\ldots,K\}} p_k^M(\mathbf{x})$.

Gradient Boosting: observaciones

- Para los f_k se utilizan modelos de predicción débiles, típicamente árboles de decisión de igual profundidad J.
- Cantidad de iteraciones M: cada iteración reduce el costo L sobre los datos de entrenamiento. Si M es demasiado grande hay sobre-ajuste.
- J y M son hiper-parámetros, se ajusta en validación.
- Elegir M chico es análogo a hacer *early stopping* en redes.
- Regularización por *shrinkage*: escalar la contribución de cada regresor por un factor $\nu \in (0, 1]$. Se pueden combinar ν chicos con M grandes.

Gradient Boosting: observaciones

Stochastic Gradient Boosting:

- Codificación propuesta por Friedman, motivada por la técnica de bagging.
- En cada iteración m, se muestrean al azar sin reposición un conjunto $\mathcal{S}_m$ de $f \times N$ muestras, $0 < f < 1$. Los $f_k^{(m)}$ se ajustan sobre $\mathcal{S}_m$.
- Tiene un efecto de regularización, y acelera dramáticamente el training.
- Como en bagging, permite estimar el error de generalización por OOB.

Comparación: Bagging vs. RF vs. GB

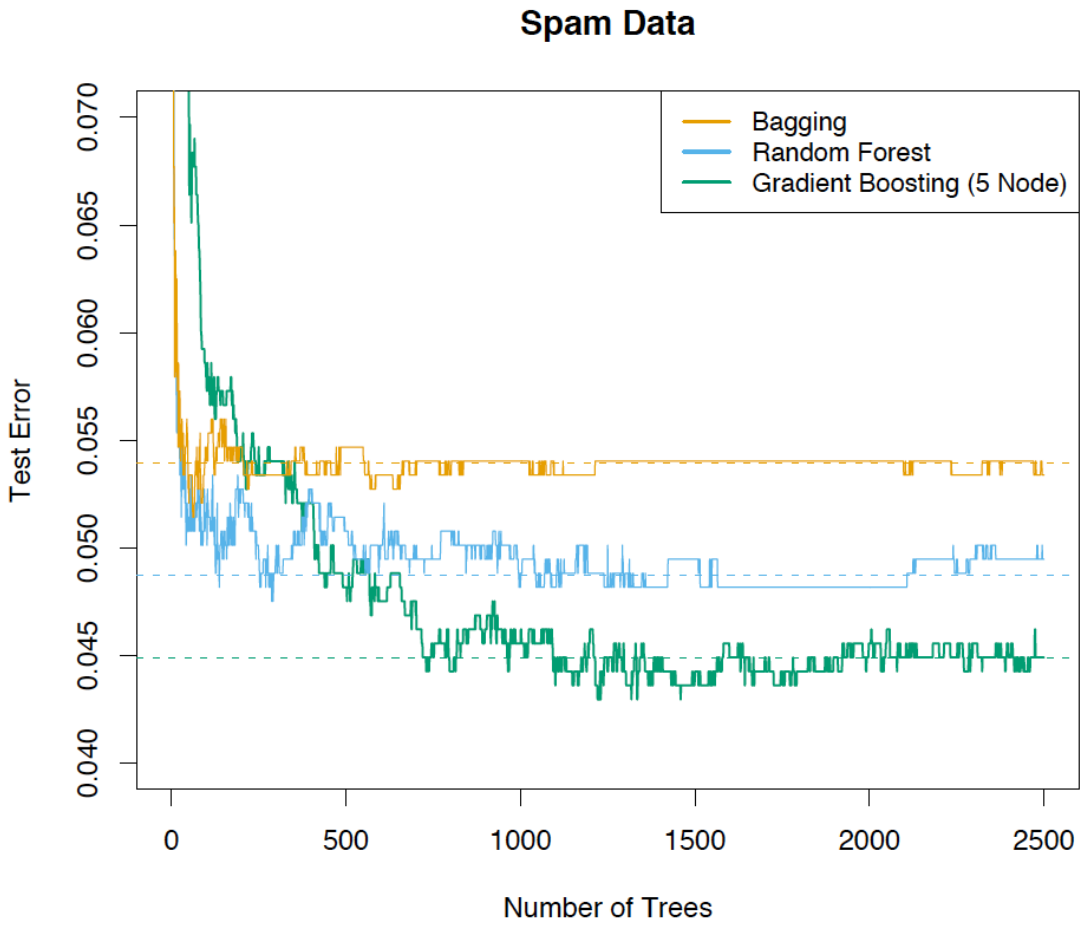


FIGURE 15.1. *Bagging, random forest, and gradient boosting, applied to the spam data. For boosting, 5-node trees were used, and the number of trees were chosen by 10-fold cross-validation (2500 trees). Each "step" in the figure corresponds to a change in a single misclassification (in a test set of 1536).*

Mezcla de Expertos

Preliminares: mezcla de modelos de regresión logística

- $K = 2$ clases, M clasificadores de regresión logística.
- $y_m = \sigma(\mathbf{w}_m^T \mathbf{x})$, $\boldsymbol{\theta} = \{\mathbf{w}_m, \pi_m\}_{m=1}^M$ parámetros de la mezcla.
- Densidad del target para la mezcla:

$$p(\mathbf{x}, t; \boldsymbol{\theta}) = \sum_{m=1}^{M} \pi_m y_m^t (1 - y_m)^{1-t}.$$

- N muestras independientes, $(\mathbf{x}_n, t_n)$, $t_n \in \{0, 1\}$, $\mathbf{t} = (t_1, t_2, \ldots, t_N)$. $y_{nm} = \sigma(\mathbf{w}_m^T \mathbf{x}_n)$,
- Función de verosimilitud de los datos para la mezcla:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n, t_1, \ldots, t_n; \boldsymbol{\theta}) = \prod_{n=1}^{N} \sum_{m=1}^{M} \pi_m y_{nm}^{t_n} (1 - y_{nm})^{1-t_n}. \qquad (*)$$

Queremos encontrar $\boldsymbol{\theta}$ que mejor explica los datos $\Rightarrow$ debemos maximizar $(*)$.
¿Cómo?

Mezcla de Expertos: preliminares, mezcla de modelos logísticos

Recordemos **EM** para una mezcla de densidades, con N muestras i.i.d.:

- Modelamos las V.A. latentes como $Z_n = m$ si $\mathbf{x}_n$ proviene de la componente m.
- *Expectation:*

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^i) = \sum_{n=1}^{N} \sum_{m=1}^{M} P(Z_n = m|\mathbf{x}_n; \boldsymbol{\theta}^i)(\log p(\mathbf{x}_n, t_n|Z_n = m; \boldsymbol{\theta}) + \log \pi_m),$$

con

$$P(Z_n = m|\mathbf{x}_n; \boldsymbol{\theta}^i) = \frac{\pi_m y_{nm}^{t_n}(1 - y_{nm})^{1-t_n}}{\sum_{l=1}^{M} \pi_l y_{nl}^{t_n}(1 - y_{nl})^{1-t_n}},$$

$$\log p(\mathbf{x}_n, t_n|Z_n = m; \boldsymbol{\theta}) = t_n \log y_{nm} + (1 - t_n) \log(1 - t_n),$$

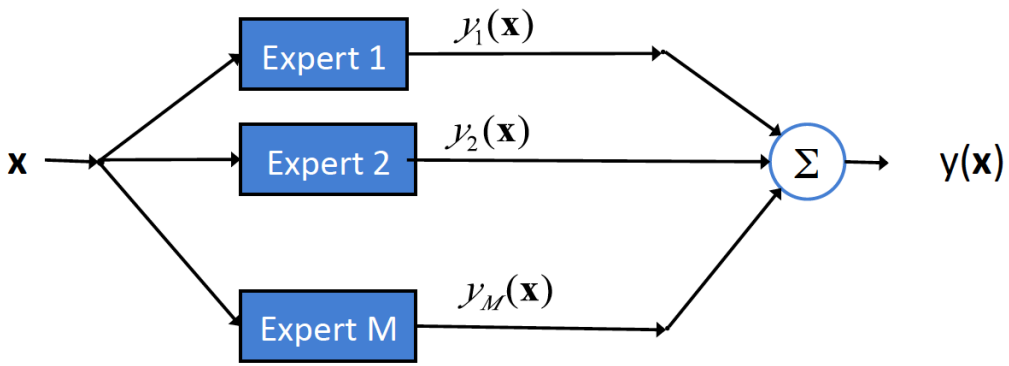
$$\pi_m = \frac{1}{N} \sum_{n=1}^{N} P(Z_n = m|\mathbf{x}_n; \boldsymbol{\theta}^i).$$

- *Maximization:* se resuelve por descenso por gradiente, Newton o IRLS ($\frac{\partial Q(\boldsymbol{\theta}, \boldsymbol{\theta}^i)}{\partial \mathbf{w}_m}$ no admite forma cerrada, a diferencia de la mezcla de Gaussianas).

Mezcla de Expertos: diferencia con otras estrategias de combinación

Boosting, bagging, mezcla de densidades:

- Los expertos cooperan para predecir la salida.

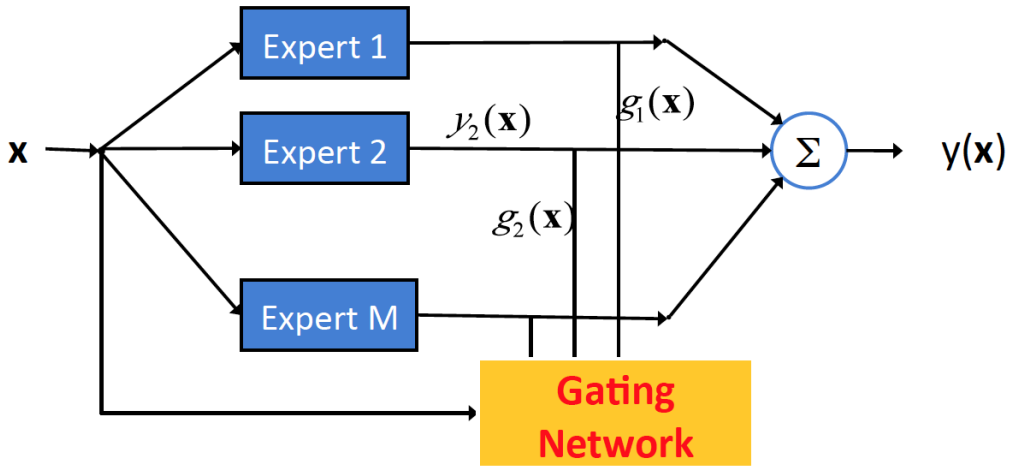


- Cada experto tiene un peso o confiabilidad π_m constante (en las mezclas sería el prior), independiente de qué dato esté clasificando: $y(\mathbf{x}) = \sum_{m=1}^{M} \pi_m y_m(\mathbf{x})$.
- Podemos pensar en π_m como el expertise general: de hecho $\pi_m = \frac{1}{N} \sum_{n=1}^{N} P(Z_n = m|\mathbf{x}_n; \boldsymbol{\theta}^i)$ es el promedio sobre todo el training set.

Mezcla de Expertos: diferencia con otras estrategias de combinación

Mezcla de expertos:

- El peso de cada experto, su expertise, depende del dato $\mathbf{x}$ a clasificar.
- Este peso es controlado por las *gating networks*, que fomentan la especialización (conocimiento especializado) en la cooperación: $y(\mathbf{x}) = \sum_{m=1}^{M} \pi_m(\mathbf{x}) y_m(\mathbf{x})$.



- Las *gating functions* deben verificar las condiciones de mezcla usuales: $\forall\, \mathbf{x}:\ \pi_m(\mathbf{x}) \in [0,1],\ \sum_{m=1}^{M} \pi_m(\mathbf{x}) = 1.$
- Es usual usar $\pi_m(\mathbf{x}) = \frac{e^{z_m(\mathbf{x})}}{\sum_{l=1}^{M} e^{z_l(\mathbf{x})}}$ (softmax).
- Se puede optimizar para todos los y_m y z_m mediante EM con IRLS.