Selección de Modelo y Regularización

Fernando Lozano

Universidad de los Andes

19 de septiembre de 2022

Minimización de error y complejidad

Minimización de error y complejidad

- Datos $(x, y) \sim \mathcal{D}$

Minimización de error y complejidad

- Datos $(x, y) \sim \mathcal{D}$
- clase de hipótesis $\mathcal{H}$ (p.ej. Redes Neuronales con arquitectura dada).

Minimización de error y complejidad

- Datos $(x, y) \sim \mathcal{D}$
- clase de hipótesis $\mathcal{H}$ (p.ej. Redes Neuronales con arquitectura dada).
- Meta:

Minimización de error y complejidad

- Datos $(x, y) \sim \mathcal{D}$
- clase de hipótesis $\mathcal{H}$ (p.ej. Redes Neuronales con arquitectura dada).
- Meta: encontrar $h \in \mathcal{H}$ que minimiza error

$$e(h) = \mathbf{P}_{\mathcal{D}}\left[h(x) \neq y\right]$$

Minimización de error y complejidad

- Datos $(x, y) \sim \mathcal{D}$
- clase de hipótesis $\mathcal{H}$ (p.ej. Redes Neuronales con arquitectura dada).
- Meta: encontrar $h \in \mathcal{H}$ que minimiza error

$$e(h) = \mathbf{P}_{\mathcal{D}}\left[h(x) \neq y\right]$$

- Intuición: Hipótesis $h \in \mathcal{H}$ que minimiza error en los datos sobre suficientes datos, produce un error $e(h)$ pequeño.

Minimización de error y complejidad

- Datos $(x, y) \sim \mathcal{D}$
- clase de hipótesis $\mathcal{H}$ (p.ej. Redes Neuronales con arquitectura dada).
- Meta: encontrar $h \in \mathcal{H}$ que minimiza error

$$e(h) = \mathbf{P}_{\mathcal{D}}\left[h(x) \neq y\right]$$

- Intuición: Hipótesis $h \in \mathcal{H}$ que minimiza error en los datos sobre suficientes datos, produce un error $e(h)$ pequeño.
- Queremos balancear la complejidad de $\mathcal{H}$ con el ajuste de $h \in \mathcal{H}$ a los datos de entrenamiento:

Minimización de error y complejidad

- Datos $(x, y) \sim \mathcal{D}$
- clase de hipótesis $\mathcal{H}$ (p.ej. Redes Neuronales con arquitectura dada).
- Meta: encontrar $h \in \mathcal{H}$ que minimiza error

$$e(h) = \mathbf{P}_{\mathcal{D}}\left[h(x) \neq y\right]$$

- Intuición: Hipótesis $h \in \mathcal{H}$ que minimiza error en los datos sobre suficientes datos, produce un error $e(h)$ pequeño.
- Queremos balancear la complejidad de $\mathcal{H}$ con el ajuste de $h \in \mathcal{H}$ a los datos de entrenamiento:
  - $\mathcal{H}$ muy simple puede no contener una buena aproximación a la función que queremos aprender

Minimización de error y complejidad

- Datos $(x, y) \sim \mathcal{D}$
- clase de hipótesis $\mathcal{H}$ (p.ej. Redes Neuronales con arquitectura dada).
- Meta: encontrar $h \in \mathcal{H}$ que minimiza error

$$e(h) = \mathbf{P}_{\mathcal{D}}\left[h(x) \neq y\right]$$

- Intuición: Hipótesis $h \in \mathcal{H}$ que minimiza error en los datos sobre suficientes datos, produce un error $e(h)$ pequeño.
- Queremos balancear la complejidad de $\mathcal{H}$ con el ajuste de $h \in \mathcal{H}$ a los datos de entrenamiento:
  - $\mathcal{H}$ muy simple puede no contener una buena aproximación a la función que queremos aprender
  - $\mathcal{H}$ muy compleja puede ajustarse bien a los datos pero predecir pobremente.

Minimización de error y complejidad

- Datos $(x, y) \sim \mathcal{D}$
- clase de hipótesis $\mathcal{H}$ (p.ej. Redes Neuronales con arquitectura dada).
- Meta: encontrar $h \in \mathcal{H}$ que minimiza error

$$e(h) = \mathbf{P}_{\mathcal{D}}\left[h(x) \neq y\right]$$

- Intuición: Hipótesis $h \in \mathcal{H}$ que minimiza error en los datos sobre suficientes datos, produce un error $e(h)$ pequeño.
- Queremos balancear la complejidad de $\mathcal{H}$ con el ajuste de $h \in \mathcal{H}$ a los datos de entrenamiento:
  - $\mathcal{H}$ muy simple puede no contener una buena aproximación a la función que queremos aprender
  - $\mathcal{H}$ muy compleja puede ajustarse bien a los datos pero predecir pobremente.
- Crítico cuando:

Minimización de error y complejidad

- Datos $(x, y) \sim \mathcal{D}$
- clase de hipótesis $\mathcal{H}$ (p.ej. Redes Neuronales con arquitectura dada).
- Meta: encontrar $h \in \mathcal{H}$ que minimiza error

$$e(h) = \mathbf{P}_{\mathcal{D}} \left[h(x) \neq y \right]$$

- Intuición: Hipótesis $h \in \mathcal{H}$ que minimiza error en los datos sobre suficientes datos, produce un error $e(h)$ pequeño.
- Queremos balancear la complejidad de $\mathcal{H}$ con el ajuste de $h \in \mathcal{H}$ a los datos de entrenamiento:
  - $\mathcal{H}$ muy simple puede no contener una buena aproximación a la función que queremos aprender
  - $\mathcal{H}$ muy compleja puede ajustarse bien a los datos pero predecir pobremente.
- Crítico cuando:
  - Número de datos es pequeño.

Minimización de error y complejidad

- Datos $(x, y) \sim \mathcal{D}$
- clase de hipótesis $\mathcal{H}$ (p.ej. Redes Neuronales con arquitectura dada).
- Meta: encontrar $h \in \mathcal{H}$ que minimiza error

$$e(h) = \mathbf{P}_{\mathcal{D}}\left[h(x) \neq y\right]$$

- Intuición: Hipótesis $h \in \mathcal{H}$ que minimiza error en los datos sobre suficientes datos, produce un error $e(h)$ pequeño.
- Queremos balancear la complejidad de $\mathcal{H}$ con el ajuste de $h \in \mathcal{H}$ a los datos de entrenamiento:
  - $\mathcal{H}$ muy simple puede no contener una buena aproximación a la función que queremos aprender
  - $\mathcal{H}$ muy compleja puede ajustarse bien a los datos pero predecir pobremente.
- Crítico cuando:
  - Número de datos es pequeño.
  - Datos ruidosos.

Algoritmos de selección de modelo

- Complejidad de la clase de modelos es una variable a determinar por el algoritmo de aprendizaje.

Algoritmos de selección de modelo

- Complejidad de la clase de modelos es una variable a determinar por el algoritmo de aprendizaje.
- Considere la secuencia anidada de clases de hipótesis:

$$\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \cdots \mathcal{H}_d \subseteq \cdots$$

Algoritmos de selección de modelo

- Complejidad de la clase de modelos es una variable a determinar por el algoritmo de aprendizaje.
- Considere la secuencia anidada de clases de hipótesis:

$$\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \cdots \mathcal{H}_d \subseteq \cdots$$

- Selección de modelo procede en dos pasos:

Algoritmos de selección de modelo

- Complejidad de la clase de modelos es una variable a determinar por el algoritmo de aprendizaje.
- Considere la secuencia anidada de clases de hipótesis:

$$\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \cdots \mathcal{H}_d \subseteq \cdots$$

- Selección de modelo procede en dos pasos:
  1. Seleccione una función candidata h_i de cada clase $\mathcal{H}_i$ (usualmente minimizando criterio de error empírico en $\mathcal{H}$).

Algoritmos de selección de modelo

- Complejidad de la clase de modelos es una variable a determinar por el algoritmo de aprendizaje.
- Considere la secuencia anidada de clases de hipótesis:

$$\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \cdots \mathcal{H}_d \subseteq \cdots$$

- Selección de modelo procede en dos pasos:
  1. Seleccione una función candidata h_i de cada clase $\mathcal{H}_i$ (usualmente minimizando criterio de error empírico en $\mathcal{H}$).
  2. Use algún criterio para seleccionar $h \in \{h_1, h_2, \ldots, h_d, \ldots\}$ tal que $e(h)$ sea pequeño.

Validación cruzada

Validación cruzada

- Estimación directa de $e(h_i)$

Validación cruzada

- Estimación directa de $e(h_i)$
  1. Datos S ise dividen en subconjuntos S_{train} and S_{test}, con $|S_{train}| = (1-\gamma)|S|$ y $|S_{test}| = \gamma|S|$, $\gamma \in (0,1)$.

Validación cruzada

- Estimación directa de $e(h_i)$
  1. Datos S ise dividen en subconjuntos S_{train} and S_{test}, con $|S_{train}| = (1-\gamma)|S|$ y $|S_{test}| = \gamma|S|$, $\gamma \in (0,1)$.
  2. Se halla hipótesis candidata en $h_d \in \mathcal{H}_d$ minimizando error empírico (o función sustituta) enS_{train}.

Validación cruzada

- Estimación directa de $e(h_i)$
  1. Datos S ise dividen en subconjuntos S_{train} and S_{test}, con $|S_{train}| = (1-\gamma)|S|$ y $|S_{test}| = \gamma|S|$, $\gamma \in (0,1)$.
  2. Se halla hipótesis candidata en $h_d \in \mathcal{H}_d$ minimizando error empírico (o función sustituta) enS_{train}.
  3. Se selecciona la hipótesis candidata h_d con el menor error empírico en S_{test}:

$$h_{d^\star} = \arg\min_{\{h_1,h_2,\ldots,\}} \hat{e}_{S_{test}}(h_d)$$

- Usando cotas de Chernoff, sabemos que para estimar $e(h)$ con precisión ε y confianza $1-\delta$ requerimos

$$|S_{test}| \geq$$

- Usando cotas de Chernoff, sabemos que para estimar $e(h)$ con precisión ε y confianza $1 - \delta$ requerimos

$$|S_{test}| \geq \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta}$$

- Usando cotas de Chernoff, sabemos que para estimar $e(h)$ con precisión ε y confianza $1-\delta$ requerimos

$$|S_{test}| \geq \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta}$$

- En la práctica es posible que no tengamos suficientes datos.

- Usando cotas de Chernoff, sabemos que para estimar $e(h)$ con precisión ε y confianza $1-\delta$ requerimos

$$|S_{test}| \geq \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta}$$

- En la práctica es posible que no tengamos suficientes datos.
- Selección de γ?

- Usando cotas de Chernoff, sabemos que para estimar $e(h)$ con precisión ε y confianza $1-\delta$ requerimos

$$|S_{test}| \geq \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta}$$

- En la práctica es posible que no tengamos suficientes datos.
- Selección de γ?
    - Muy pequeño $\Rightarrow$

- Usando cotas de Chernoff, sabemos que para estimar $e(h)$ con precisión ε y confianza $1 - \delta$ requerimos

$$|S_{test}| \geq \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta}$$

- En la práctica es posible que no tengamos suficientes datos.
- Selección de γ?
  - Muy pequeño $\Rightarrow$ estimación pobre de $e(h)$.

- Usando cotas de Chernoff, sabemos que para estimar $e(h)$ con precisión ε y confianza $1-\delta$ requerimos

$$|S_{test}| \geq \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta}$$

- En la práctica es posible que no tengamos suficientes datos.
- Selección de γ?
  - Muy pequeño $\Rightarrow$ estimación pobre de $e(h)$.
  - Muy grande $\Rightarrow$

- Usando cotas de Chernoff, sabemos que para estimar $e(h)$ con precisión ε y confianza $1-\delta$ requerimos

$$|S_{test}| \geq \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta}$$

- En la práctica es posible que no tengamos suficientes datos.
- Selección de γ?
  - Muy pequeño $\Rightarrow$ estimación pobre de $e(h)$.
  - Muy grande $\Rightarrow$ aprendizaje pobre.

- Usando cotas de Chernoff, sabemos que para estimar $e(h)$ con precisión ε y confianza $1 - \delta$ requerimos

$$|S_{test}| \geq \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta}$$

- En la práctica es posible que no tengamos suficientes datos.
- Selección de γ?
  - Muy pequeño $\Rightarrow$ estimación pobre de $e(h)$.
  - Muy grande $\Rightarrow$ aprendizaje pobre.
  - Típicamente $\gamma \approx 0{,}1$.

- Usando cotas de Chernoff, sabemos que para estimar $e(h)$ con precisión ε y confianza $1-\delta$ requerimos

$$|S_{test}| \geq \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta}$$

- En la práctica es posible que no tengamos suficientes datos.
- Selección de γ?
  - Muy pequeño $\Rightarrow$ estimación pobre de $e(h)$.
  - Muy grande $\Rightarrow$ aprendizaje pobre.
  - Típicamente $\gamma \approx 0{,}1$.
- Estimativo de $e(h_d)$ es usualmente ruidoso.

- Usando cotas de Chernoff, sabemos que para estimar $e(h)$ con precisión ε y confianza $1-\delta$ requerimos

$$|S_{test}| \geq \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta}$$

- En la práctica es posible que no tengamos suficientes datos.
- Selección de γ?
  - Muy pequeño $\Rightarrow$ estimación pobre de $e(h)$.
  - Muy grande $\Rightarrow$ aprendizaje pobre.
  - Típicamente $\gamma \approx 0{,}1$.
- Estimativo de $e(h_d)$ es usualmente ruidoso.
- En la práctica se usa validación cruzada k-múltiple.

Validación Cruzada k-múltiple

- Idea es suavizar estimativo de $e(h)$

Validación Cruzada k-múltiple

- Idea es suavizar estimativo de $e(h)$
- Para una clase $\mathcal{H}$:

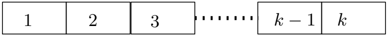
Validación Cruzada k-múltiple

- Idea es suavizar estimativo de $e(h)$
- Para una clase $\mathcal{H}$:
    1. S se divide en $S_1, S_2, \ldots, S_k$.

1	2	3	$\cdots\cdots$	$k-1$	k

Validación Cruzada k-múltiple

- Idea es suavizar estimativo de $e(h)$
- Para una clase $\mathcal{H}$:
  1. S se divide en $S_1, S_2, \ldots, S_k$.

  2. Para cada $i = 1, 2, \ldots, k$

Validación Cruzada k-múltiple

- Idea es suavizar estimativo de $e(h)$
- Para una clase $\mathcal{H}$:
  1. S se divide en $S_1, S_2, \ldots, S_k$.

1	2	3	⋯⋯	$k-1$	k

  2. Para cada $i = 1, 2, \ldots, k$
     1. Se halla h_i minimizando error empírico en $\bigcup_{j \neq i} S_j$

Validación Cruzada k-múltiple

- Idea es suavizar estimativo de $e(h)$
- Para una clase $\mathcal{H}$:
  1. S se divide en $S_1, S_2, \ldots, S_k$.

1	2	3	⋯	$k-1$	k

  2. Para cada $i = 1, 2, \ldots, k$
     1. Se halla h_i minimizando error empírico en $\bigcup_{j \neq i} S_j$
     2. Se estima error calculando error empírico $\hat{e}_{S_i}(h_i)$

Validación Cruzada k-múltiple

- Idea es suavizar estimativo de $e(h)$
- Para una clase $\mathcal{H}$:
  1. S se divide en $S_1, S_2, \ldots, S_k$.

1	2	3	·······	$k-1$	k

  2. Para cada $i = 1, 2, \ldots, k$
     1. Se halla h_i minimizando error empírico en $\bigcup_{j \neq i} S_j$
     2. Se estima error calculando error empírico $\hat{e}_{S_i}(h_i)$
     3. Se promedian valores obtenidos:

$$\hat{e}(h_d) = \frac{1}{k} \sum_{i=1}^{k} \hat{e}_{S_i}(h_i)$$

Validación Cruzada k-múltiple

- Idea es suavizar estimativo de $e(h)$
- Para una clase $\mathcal{H}$:
  1. S se divide en $S_1, S_2, \ldots, S_k$.

1	2	3	$\cdots$	$k-1$	k

  2. Para cada $i = 1, 2, \ldots, k$
     1. Se halla h_i minimizando error empírico en $\bigcup_{j \neq i} S_j$
     2. Se estima error calculando error empírico $\hat{e}_{S_i}(h_i)$
     3. Se promedian valores obtenidos:

$$\hat{e}(h_d) = \frac{1}{k} \sum_{i=1}^{k} \hat{e}_{S_i}(h_i)$$

- Para $d^\star$ que corresponde al menor valor de $\hat{e}(h_d)$, se halla h minimizando error empírico en $\mathcal{H}_{d^\star}$ en S.

Validación cruzada k-múltiple

Validación cruzada k-múltiple

- Procedimiento es costoso computacionalmente.

Validación cruzada k-múltiple

- Procedimiento es costoso computacionalmente.
- Usado ampliamente en la práctica.

Validación cruzada k-múltiple

- Procedimiento es costoso computacionalmente.
- Usado ampliamente en la práctica.
- Carece de soporte teórico, es un problema abierto importante.

Validación cruzada k-múltiple

- Procedimiento es costoso computacionalmente.
- Usado ampliamente en la práctica.
- Carece de soporte teórico, es un problema abierto importante.
- Errores no son v.a. normales, no son independientes.

Regularización

- Problemas mal condicionados (ill posed).

Regularización

- Problemas mal condicionados (ill posed).
  - Inestabilidad de la solución.

Regularización

- Problemas mal condicionados (ill posed).
  - Inestabilidad de la solución.
  - Inestabilidad numérica.

Regularización

- Problemas mal condicionados (ill posed).
    - Inestabilidad de la solución.
    - Inestabilidad numérica.
    - Algoritmos de solución.

Regularización

- Problemas mal condicionados (ill posed).
  - Inestabilidad de la solución.
  - Inestabilidad numérica.
  - Algoritmos de solución.
- Introducir suposiciones adicionales que estabilizan la solución.

Regularización

- Problemas mal condicionados (ill posed).
  - Inestabilidad de la solución.
  - Inestabilidad numérica.
  - Algoritmos de solución.
- Introducir suposiciones adicionales que estabilizan la solución.
- En Machine Learning:

Regularización

- Problemas mal condicionados (ill posed).
  - Inestabilidad de la solución.
  - Inestabilidad numérica.
  - Algoritmos de solución.
- Introducir suposiciones adicionales que estabilizan la solución.
- En Machine Learning:
  - Inestabilidad con respecto a los datos (overfitting).

Regularización

- Problemas mal condicionados (ill posed).
  - Inestabilidad de la solución.
  - Inestabilidad numérica.
  - Algoritmos de solución.
- Introducir suposiciones adicionales que estabilizan la solución.
- En Machine Learning:
  - Inestabilidad con respecto a los datos (overfitting).
  - Introducir conocimiento previo con respecto al modelo.

Regularización

- Problemas mal condicionados (ill posed).
  - Inestabilidad de la solución.
  - Inestabilidad numérica.
  - Algoritmos de solución.
- Introducir suposiciones adicionales que estabilizan la solución.
- En Machine Learning:
  - Inestabilidad con respecto a los datos (overfitting).
  - Introducir conocimiento previo con respecto al modelo.
  - Restringe la complejidad de la clase de modelos.

Regularización

- Problemas mal condicionados (ill posed).
  - Inestabilidad de la solución.
  - Inestabilidad numérica.
  - Algoritmos de solución.
- Introducir suposiciones adicionales que estabilizan la solución.
- En Machine Learning:
  - Inestabilidad con respecto a los datos (overfitting).
  - Introducir conocimiento previo con respecto al modelo.
  - Restringe la complejidad de la clase de modelos.

Error Penalizado

Error Penalizado

- Modelo con parámetros $\mathbf{w}$, y función de error $E(\mathbf{w})$.

Error Penalizado

- Modelo con parámetros $\mathbf{w}$, y función de error $E(\mathbf{w})$.
- Error (riesgo) regularizado:

Error Penalizado

- Modelo con parámetros $\mathbf{w}$, y función de error $E(\mathbf{w})$.
- Error (riesgo) regularizado:

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda R(\mathbf{w})$$

Error Penalizado

- Modelo con parámetros $\mathbf{w}$, y función de error $E(\mathbf{w})$.
- Error (riesgo) regularizado:

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda R(\mathbf{w})$$

donde:

Error Penalizado

- Modelo con parámetros $\mathbf{w}$, y función de error $E(\mathbf{w})$.
- Error (riesgo) regularizado:

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda R(\mathbf{w})$$

  donde:
  - $R(\mathbf{w})$ es término de regularización que penaliza modelos complejos.

Error Penalizado

- Modelo con parámetros $\mathbf{w}$, y función de error $E(\mathbf{w})$.
- Error (riesgo) regularizado:

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda R(\mathbf{w})$$

donde:

  - $R(\mathbf{w})$ es término de regularización que penaliza modelos complejos.
  - $\lambda \geq 0$ es la constante de regularización (selección de modelo).

Error Penalizado

- Modelo con parámetros $\mathbf{w}$, y función de error $E(\mathbf{w})$.
- Error (riesgo) regularizado:

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda R(\mathbf{w})$$

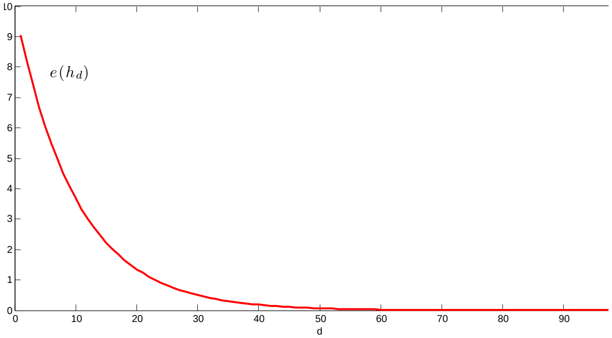
  donde:
  - $R(\mathbf{w})$ es término de regularización que penaliza modelos complejos.
  - $\lambda \geq 0$ es la constante de regularización (selección de modelo).
- Balance entre ajuste a los datos y complejidad del modelo.

Error Penalizado

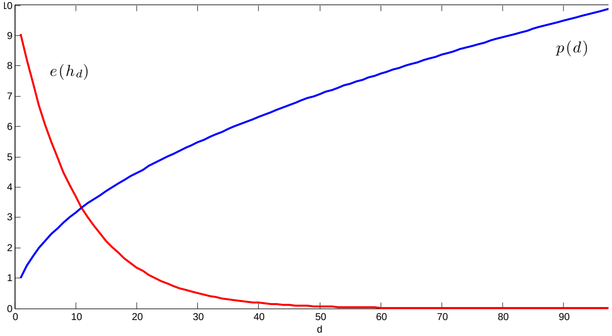
- Modelo con parámetros $\mathbf{w}$, y función de error $E(\mathbf{w})$.
- Error (riesgo) regularizado:

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda R(\mathbf{w})$$

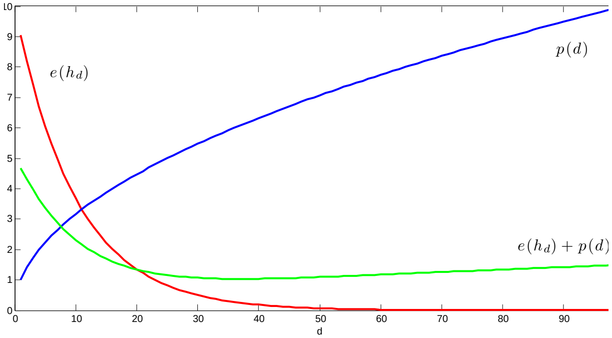
  donde:
  - $R(\mathbf{w})$ es término de regularización que penaliza modelos complejos.
  - $\lambda \geq 0$ es la constante de regularización (selección de modelo).
- Balance entre ajuste a los datos y complejidad del modelo.
- Reducir varianza, introduciendo sesgo.



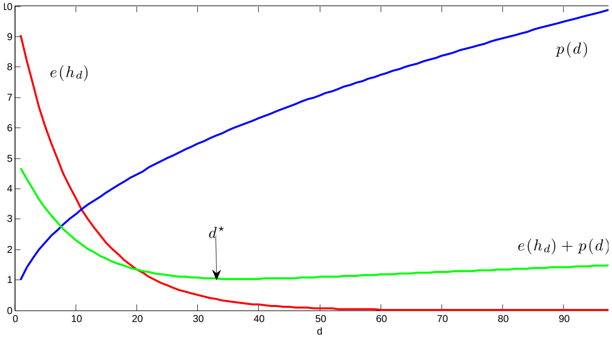
$e(h_d)$
d



$e(h_d)$
$p(d)$
d



10
9
8
7
6
5
4
3
2
1
0
0
10
20
30
40
50
60
70
80
90
d
$e(h_d)$
$p(d)$
$e(h_d) + p(d)$


10
9
8
7
6
5
4
3
2
1
0
0
10
20
30
40
50
60
70
80
90
d
$e(h_d)$
$p(d)$
$d^\star$
$e(h_d) + p(d)$

Regularización convexa

Regularización convexa

- Considere el problema de optimización:

$$\min_{\mathbf{x}} f(\mathbf{x}) + \lambda g(\mathbf{x})$$

donde f y g son funciones estrictamente convexas.

Regularización convexa

- Considere el problema de optimización:

$$\min_{\mathbf{x}} f(\mathbf{x}) + \lambda g(\mathbf{x})$$

  donde f y g son funciones estrictamente convexas.
- Para cada $\lambda > 0$, existe t_λ, tal que el siguiente problema de optimización es equivalente:

$$\begin{aligned} \text{mín} \quad & f(\mathbf{x}) \\ \text{sujeto a} \quad & g(\mathbf{x}) \leq t_\lambda \end{aligned}$$

- $\mathbf{x}^* = \arg\min_{\mathbf{x}} \left[f(\mathbf{x}) + \lambda g(\mathbf{x}) \right]$

- $\mathbf{x}^* = \arg\min_{\mathbf{x}} \left[f(\mathbf{x}) + \lambda g(\mathbf{x})\right]$
- $(\mathbf{x}, \lambda)$ satisfacen KKT para el problema:

$$\begin{aligned} \text{mín} \quad & f(\mathbf{x}) \\ \text{sujeto a} \quad & g(\mathbf{x}) \leq {\color{red} g(\mathbf{x}^*)} \end{aligned}$$

- $\mathbf{x}^* = \arg\min_{\mathbf{x}} \left[f(\mathbf{x}) + \lambda g(\mathbf{x}) \right]$
- $(\mathbf{x}, \lambda)$ satisfacen KKT para el problema:

$$\begin{array}{rl} \text{mín} & f(\mathbf{x}) \\ \text{sujeto a} & g(\mathbf{x}) \leq g(\mathbf{x}^*) \end{array}$$

1. $g(\mathbf{x}^*) \leq g(\mathbf{x}^*)$

- $\mathbf{x}^* = \arg\min_{\mathbf{x}} [f(\mathbf{x}) + \lambda g(\mathbf{x})]$
- $(\mathbf{x}, \lambda)$ satisfacen KKT para el problema:

$$\begin{aligned} \text{mín} \quad & f(\mathbf{x}) \\ \text{sujeto a} \quad & g(\mathbf{x}) \leq \color{red}{g(\mathbf{x}^*)} \end{aligned}$$

1. $g(\mathbf{x}^*) \leq \color{red}{g(\mathbf{x}^*)}$
2. $\lambda > 0$

- $\mathbf{x}^* = \arg\min_{\mathbf{x}} \left[f(\mathbf{x}) + \lambda g(\mathbf{x}) \right]$
- $(\mathbf{x}, \lambda)$ satisfacen KKT para el problema:

$$\begin{aligned} \text{mín} \quad & f(\mathbf{x}) \\ \text{sujeto a} \quad & g(\mathbf{x}) \leq g(\mathbf{x}^*) \end{aligned}$$

1. $g(\mathbf{x}^*) \leq g(\mathbf{x}^*)$
2. $\lambda > 0$
3. $\mathbf{x}^* = \arg\min_{\mathbf{x}} L(\mathbf{x}, \lambda)$

- $\mathbf{x}^* = \arg\min_{\mathbf{x}} [f(\mathbf{x}) + \lambda g(\mathbf{x})]$
- $(\mathbf{x}, \lambda)$ satisfacen KKT para el problema:

$$\begin{aligned} \text{mín} \quad & f(\mathbf{x}) \\ \text{sujeto a} \quad & g(\mathbf{x}) \leq g(\mathbf{x}^*) \end{aligned}$$

1. $g(\mathbf{x}^*) \leq g(\mathbf{x}^*)$
2. $\lambda > 0$
3. $\mathbf{x}^* = \arg\min_{\mathbf{x}} L(\mathbf{x}, \lambda) = \arg\min_{\mathbf{x}} [f(\mathbf{x}) + \lambda g(\mathbf{x}) - \lambda g(\mathbf{x}^*)]$

- $\mathbf{x}^* = \arg\min_{\mathbf{x}} \left[f(\mathbf{x}) + \lambda g(\mathbf{x})\right]$
- $(\mathbf{x}, \lambda)$ satisfacen KKT para el problema:

$$\begin{aligned} \text{mín} \quad & f(\mathbf{x}) \\ \text{sujeto a} \quad & g(\mathbf{x}) \leq g(\mathbf{x}^*) \end{aligned}$$

1. $g(\mathbf{x}^*) \leq g(\mathbf{x}^*)$
2. $\lambda > 0$
3. $\mathbf{x}^* = \arg\min_{\mathbf{x}} L(\mathbf{x}, \lambda) = \arg\min_{\mathbf{x}} \left[f(\mathbf{x}) + \lambda g(\mathbf{x}) - \lambda g(\mathbf{x}^*)\right]$
4. $\lambda > 0 \Rightarrow g(\mathbf{x}^*) = g(\mathbf{x}^*)$

- $\mathbf{x}^* = \arg\min_{\mathbf{x}} [f(\mathbf{x}) + \lambda g(\mathbf{x})]$
- $(\mathbf{x}, \lambda)$ satisfacen KKT para el problema:

$$\begin{aligned} \text{mín} \quad & f(\mathbf{x}) \\ \text{sujeto a} \quad & g(\mathbf{x}) \leq g(\mathbf{x}^*) \end{aligned}$$

1. $g(\mathbf{x}^*) \leq g(\mathbf{x}^*)$
2. $\lambda > 0$
3. $\mathbf{x}^* = \arg\min_{\mathbf{x}} L(\mathbf{x}, \lambda) = \arg\min_{\mathbf{x}} [f(\mathbf{x}) + \lambda g(\mathbf{x}) - \lambda g(\mathbf{x}^*)]$
4. $\lambda > 0 \Rightarrow g(\mathbf{x}^*) = g(\mathbf{x}^*)$

$\Rightarrow \mathbf{x}^*$es solución óptima del segundo problema.

Ejemplo

$$\min_{x} \quad (x-a)^2 + \lambda(x-b)^2$$

Ejemplo

$$\min_{x} \quad (x-a)^2 + \lambda(x-b)^2$$

$$x^* = \frac{a+\lambda b}{1+\lambda}$$

Ejemplo

$$\min_x \quad (x-a)^2 + \lambda(x-b)^2$$

$$x^* = \frac{a+\lambda b}{1+\lambda} \quad g(x^*) = \left(\frac{a-b}{1+\lambda}\right)^2$$

Ejemplo

$$\min_x \quad (x-a)^2 + \lambda(x-b)^2$$

$$x^* = \frac{a+\lambda b}{1+\lambda} \quad g(x^*) = \left(\frac{a-b}{1+\lambda}\right)^2$$

$$\begin{aligned} \text{mín} \quad & (x-a)^2 \\ \text{sujeto a} \quad & (x-b)^2 \leq \left(\frac{a-b}{1+\lambda}\right)^2 \end{aligned}$$

Regresión Lineal

$$
\begin{aligned}
E(\mathbf{w}) &= \sum_{i=1}^{n}(\mathbf{w}^T\mathbf{x}_i - y_i)^2 \\
&= \mathbf{w}^T\mathbf{H}\mathbf{w} - 2\mathbf{b}^T\mathbf{w} + c
\end{aligned}
$$

donde:

$$
\mathbf{H} = \sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T, \quad \mathbf{b} = \sum_{i=1}^{n}\mathbf{x}_i y_i, \quad c = \sum_{i=1}^{n} y_i
$$

Regresión de Cresta (Ridge Regression)

Regresión de Cresta (Ridge Regression)

- Datos centrados y escalados, $w_0 = \bar{y}_i$.

Regresión de Cresta (Ridge Regression)

- Datos centrados y escalados, $w_0 = \bar{y_i}$.
- Regularización con norma l_2:

$$E(\mathbf{w}) = \sum_{i=1}^{n}(\mathbf{w}^T\mathbf{x}_i - y_i)^2 + \lambda\|\mathbf{w}\|^2$$

Regresión de Cresta (Ridge Regression)

- Datos centrados y escalados, $w_0 = \bar{y_i}$.
- Regularización con norma l_2:

$$E(\mathbf{w}) = \sum_{i=1}^{n}(\mathbf{w}^T\mathbf{x}_i - y_i)^2 + \lambda\|\mathbf{w}\|^2$$

- Penaliza parámetros grandes.

Regresión de Cresta (Ridge Regression)

- Datos centrados y escalados, $w_0 = \bar{y}_i$.
- Regularización con norma l_2:

$$E(\mathbf{w}) = \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|^2$$

- Penaliza parámetros grandes.
- Con $\lambda \geq 0$, $E(\mathbf{w})$ es una función convexa.

Regresión de Cresta (Ridge Regression)

- Datos centrados y escalados, $w_0 = \bar{y_i}$.
- Regularización con norma l_2:

$$E(\mathbf{w}) = \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|^2$$

- Penaliza parámetros grandes.
- Con $\lambda \geq 0$, $E(\mathbf{w})$ es una función convexa.
- Método de encogimiento (shrinkage)

- Para $\lambda = 0$:

$$E(\mathbf{w}) = \mathbf{w}^T\mathbf{H}\mathbf{w} - 2\mathbf{b}^T\mathbf{w} + c$$

- Para $\lambda = 0$:

$$E(\mathbf{w}) = \mathbf{w}^T \mathbf{H}\mathbf{w} - 2\mathbf{b}^T \mathbf{w} + c$$

El mínimo $\hat{\mathbf{w}}$ satisface:

- Para $\lambda = 0$:

$$E(\mathbf{w}) = \mathbf{w}^T\mathbf{H}\mathbf{w} - 2\mathbf{b}^T\mathbf{w} + c$$

El mínimo $\hat{\mathbf{w}}$ satisface:

$$\mathbf{H}\hat{\mathbf{w}} - \mathbf{b} = 0$$

- Para $\lambda = 0$:
$$E(\mathbf{w}) = \mathbf{w}^T\mathbf{H}\mathbf{w} - 2\mathbf{b}^T\mathbf{w} + c$$
El mínimo $\hat{\mathbf{w}}$ satisface:
$$\mathbf{H}\hat{\mathbf{w}} - \mathbf{b} = 0$$

- Para $\lambda > 0$
$$\tilde{E}(\mathbf{w}) = \mathbf{w}^T(\mathbf{H} + \lambda\mathbf{I})\mathbf{w} - 2\mathbf{b}^T\mathbf{w} + c$$

- Para $\lambda = 0$:
$$E(\mathbf{w}) = \mathbf{w}^T\mathbf{H}\mathbf{w} - 2\mathbf{b}^T\mathbf{w} + c$$
El mínimo $\hat{\mathbf{w}}$ satisface:
$$\mathbf{H}\hat{\mathbf{w}} - \mathbf{b} = 0$$
- Para $\lambda > 0$
$$\tilde{E}(\mathbf{w}) = \mathbf{w}^T(\mathbf{H} + \lambda\mathbf{I})\mathbf{w} - 2\mathbf{b}^T\mathbf{w} + c$$
El mínimo $\tilde{\mathbf{w}}$ satisface:
$$(\mathbf{H} + \lambda\mathbf{I})\tilde{\mathbf{w}} - \mathbf{b} = 0$$

- Para $\lambda = 0$:

$$E(\mathbf{w}) = \mathbf{w}^T\mathbf{H}\mathbf{w} - 2\mathbf{b}^T\mathbf{w} + c$$

El mínimo $\hat{\mathbf{w}}$ satisface:

$$\mathbf{H}\hat{\mathbf{w}} - \mathbf{b} = 0$$

- Para $\lambda > 0$

$$\tilde{E}(\mathbf{w}) = \mathbf{w}^T(\mathbf{H} + \lambda\mathbf{I})\mathbf{w} - 2\mathbf{b}^T\mathbf{w} + c$$

El mínimo $\tilde{\mathbf{w}}$ satisface:

$$(\mathbf{H} + \lambda\mathbf{I})\tilde{\mathbf{w}} - \mathbf{b} = 0$$

- Si $\mathbf{H} > 0$,

$$\hat{\mathbf{w}} = \sum_j \hat{w}_j \mathbf{u}_j \qquad \tilde{\mathbf{w}} = \sum_j \tilde{w}_j \mathbf{u}_j$$

- Si $\mathbf{H} > 0$,

$$\hat{\mathbf{w}} = \sum_j \hat{w}_j \mathbf{u}_j \qquad \tilde{\mathbf{w}} = \sum_j \tilde{w}_j \mathbf{u}_j$$

donde $\mathbf{H}\mathbf{u}_i = \alpha_i \mathbf{u}_i$

- Si $\mathbf{H} > 0$,

$$\hat{\mathbf{w}} = \sum_j \hat{w}_j \mathbf{u}_j \qquad \tilde{\mathbf{w}} = \sum_j \tilde{w}_j \mathbf{u}_j$$

donde $\mathbf{H}\mathbf{u}_i = \alpha_i \mathbf{u}_i$ (componentes principales!)

- Si $\mathbf{H} > 0$,

$$\hat{\mathbf{w}} = \sum_j \hat{w}_j \mathbf{u}_j \qquad \tilde{\mathbf{w}} = \sum_j \tilde{w}_j \mathbf{u}_j$$

donde $\mathbf{H}\mathbf{u}_i = \alpha_i \mathbf{u}_i$ (componentes principales!)

- Reemplazando:

$$-\mathbf{b} + \sum_j \hat{w}_j \alpha_j \mathbf{u}_j = 0 \qquad -\mathbf{b} + \sum_j \tilde{w}_j \alpha_j \mathbf{u}_j + \lambda \sum_j \tilde{w}_j \mathbf{u}_j = 0$$

- Si $\mathbf{H} > 0$,

$$\hat{\mathbf{w}} = \sum_j \hat{w}_j \mathbf{u}_j \qquad \tilde{\mathbf{w}} = \sum_j \tilde{w}_j \mathbf{u}_j$$

donde $\mathbf{H}\mathbf{u}_i = \alpha_i \mathbf{u}_i$ (componentes principales!)

- Reemplazando:

$$-\mathbf{b} + \sum_j \hat{w}_j \alpha_j \mathbf{u}_j = 0 \qquad -\mathbf{b} + \sum_j \tilde{w}_j \alpha_j \mathbf{u}_j + \lambda \sum_j \tilde{w}_j \mathbf{u}_j = 0$$

- Por ortonormalidad de los $\mathbf{u}_i$:

$$\hat{w}_j \alpha_j = \tilde{w}_j \alpha_j + \lambda \tilde{w}_j$$

- Si $\mathbf{H} > 0$,

$$\hat{\mathbf{w}} = \sum_j \hat{w}_j \mathbf{u}_j \qquad \tilde{\mathbf{w}} = \sum_j \tilde{w}_j \mathbf{u}_j$$

  donde $\mathbf{H}\mathbf{u}_i = \alpha_i \mathbf{u}_i$ (componentes principales!)
- Reemplazando:

$$-\mathbf{b} + \sum_j \hat{w}_j \alpha_j \mathbf{u}_j = 0 \qquad -\mathbf{b} + \sum_j \tilde{w}_j \alpha_j \mathbf{u}_j + \lambda \sum_j \tilde{w}_j \mathbf{u}_j = 0$$

- Por ortonormalidad de los $\mathbf{u}_i$:

$$\hat{w}_j \alpha_j = \tilde{w}_j \alpha_j + \lambda \tilde{w}_j \Rightarrow \tilde{w}_j = \hat{w}_j \left(\frac{\alpha_j}{\alpha_j + \lambda} \right)$$

- Si $\mathbf{H} > 0$,

$$\hat{\mathbf{w}} = \sum_j \hat{w}_j \mathbf{u}_j \qquad \tilde{\mathbf{w}} = \sum_j \tilde{w}_j \mathbf{u}_j$$

donde $\mathbf{H}\mathbf{u}_i = \alpha_i \mathbf{u}_i$ (componentes principales!)

- Reemplazando:

$$-\mathbf{b} + \sum_j \hat{w}_j \alpha_j \mathbf{u}_j = 0 \qquad -\mathbf{b} + \sum_j \tilde{w}_j \alpha_j \mathbf{u}_j + \lambda \sum_j \tilde{w}_j \mathbf{u}_j = 0$$

- Por ortonormalidad de los $\mathbf{u}_i$:

$$\hat{w}_j \alpha_j = \tilde{w}_j \alpha_j + \lambda \tilde{w}_j \Rightarrow \tilde{w}_j = \hat{w}_j \left(\frac{\alpha_j}{\alpha_j + \lambda} \right)$$

$$\alpha_j \gg \lambda \Rightarrow \tilde{w}_j \approx \hat{w}_j$$

- Si $\mathbf{H} > 0$,

$$\hat{\mathbf{w}} = \sum_j \hat{w}_j \mathbf{u}_j \qquad \tilde{\mathbf{w}} = \sum_j \tilde{w}_j \mathbf{u}_j$$

  donde $\mathbf{H}\mathbf{u}_i = \alpha_i \mathbf{u}_i$ (componentes principales!)

- Reemplazando:

$$-\mathbf{b} + \sum_j \hat{w}_j \alpha_j \mathbf{u}_j = 0 \qquad -\mathbf{b} + \sum_j \tilde{w}_j \alpha_j \mathbf{u}_j + \lambda \sum_j \tilde{w}_j \mathbf{u}_j = 0$$

- Por ortonormalidad de los $\mathbf{u}_i$:

$$\hat{w}_j \alpha_j = \tilde{w}_j \alpha_j + \lambda \tilde{w}_j \Rightarrow \tilde{w}_j = \hat{w}_j \left(\frac{\alpha_j}{\alpha_j + \lambda} \right)$$

$$\alpha_j \gg \lambda \Rightarrow \tilde{w}_j \approx \hat{w}_j \qquad \alpha_j \ll \lambda \Rightarrow |\tilde{w}_j| \ll |\hat{w}_j|$$

El problema de optimización:

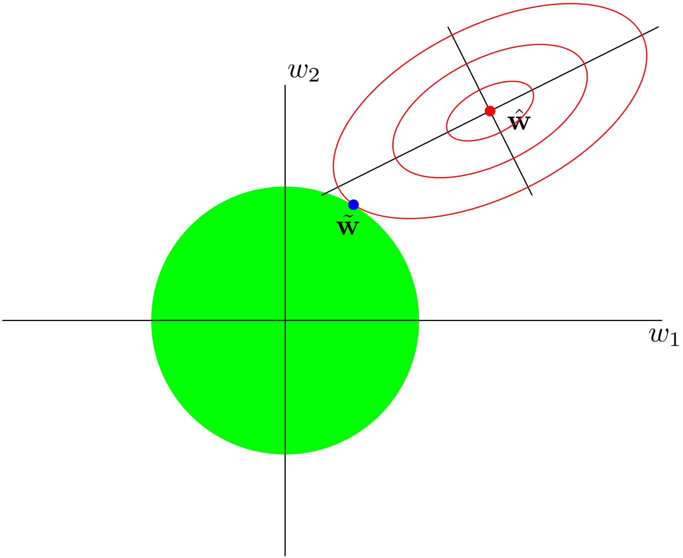
$$\min_{\mathbf{w}} \quad \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|^2$$

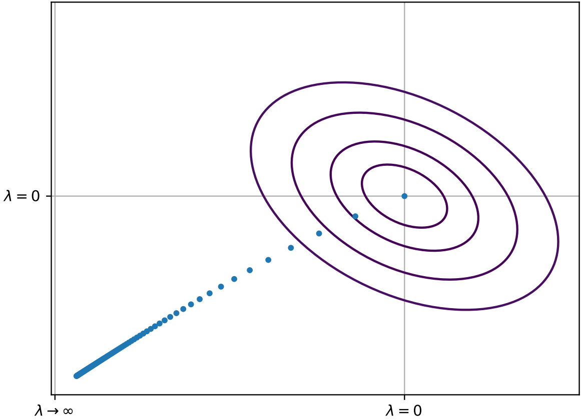
El problema de optimización:

$$\min_{\mathbf{w}} \quad \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|^2$$

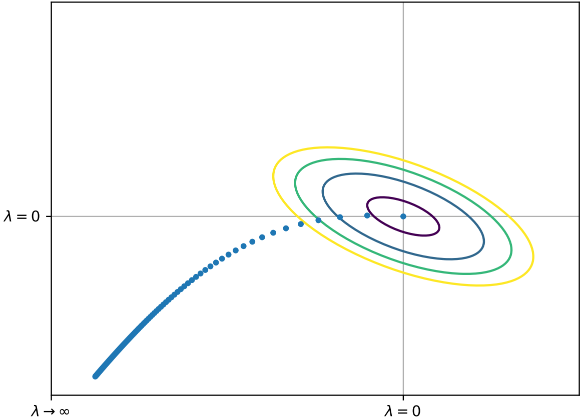
Se puede formular equivalentemente como:

$$\begin{aligned} \text{mín} \quad & \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 \\ \text{sujeto a} \quad & \|\mathbf{w}\|^2 \leq t \end{aligned}$$


w_2
$\hat{\mathbf{w}}$
$\tilde{\mathbf{w}}$
w_1



$\lambda = 0$
$\lambda \to \infty$
$\lambda = 0$


$\lambda = 0$
$\lambda \to \infty$
$\lambda = 0$

Regresión lineal con penalización l_1(LASSO)

Regresión lineal con penalización l_1(LASSO)

- Datos centrados y escalados, $w_0 = \bar{y}_i$.

Regresión lineal con penalización l_1(LASSO)

- Datos centrados y escalados, $w_0 = \bar{y}_i$.
- Regularización con norma l_1:

$$E(\mathbf{w}) = \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|_1$$

Regresión lineal con penalización l_1(LASSO)

- Datos centrados y escalados, $w_0 = \bar{y}_i$.
- Regularización con norma l_1:

$$
\begin{aligned}
E(\mathbf{w}) &= \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|_1 \\
&= \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \sum_{j} |w_j|
\end{aligned}
$$

Regresión lineal con penalización l_1(LASSO)

- Datos centrados y escalados, $w_0 = \bar{y}_i$.
- Regularización con norma l_1:

$$
\begin{aligned}
E(\mathbf{w}) &= \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|_1 \\
&= \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \sum_j |w_j|
\end{aligned}
$$

- Penaliza parámetros grandes.

Regresión lineal con penalización l_1(LASSO)

- Datos centrados y escalados, $w_0 = \bar{y}_i$.
- Regularización con norma l_1:

$$
\begin{aligned}
E(\mathbf{w}) &= \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|_1 \\
&= \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \sum_j |w_j|
\end{aligned}
$$

- Penaliza parámetros grandes.
- Método de encogimiento (shrinkage)

Regresión lineal con penalización l_1(LASSO)

- Datos centrados y escalados, $w_0 = \bar{y}_i$.
- Regularización con norma l_1:

$$
\begin{aligned}
E(\mathbf{w}) &= \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|_1 \\
&= \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \sum_{j} |w_j|
\end{aligned}
$$

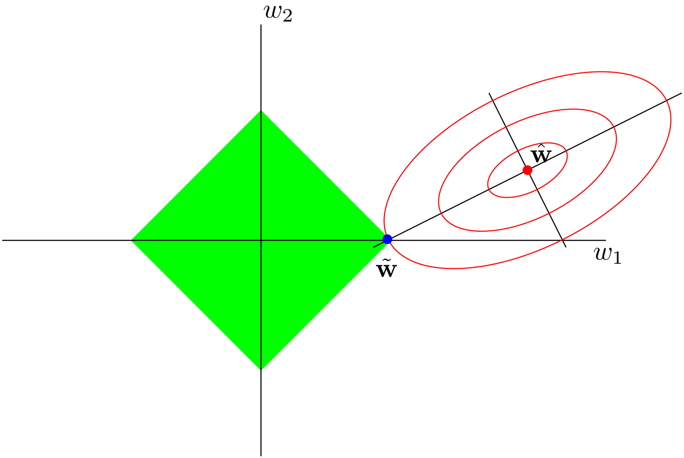
- Penaliza parámetros grandes.
- Método de encogimiento (shrinkage)
- Con $\lambda \geq 0$, $E(\mathbf{w})$ es una función convexa, pero no diferenciable.

Regresión lineal con penalización l_1(LASSO)

- Datos centrados y escalados, $w_0 = \bar{y}_i$.
- Regularización con norma l_1:

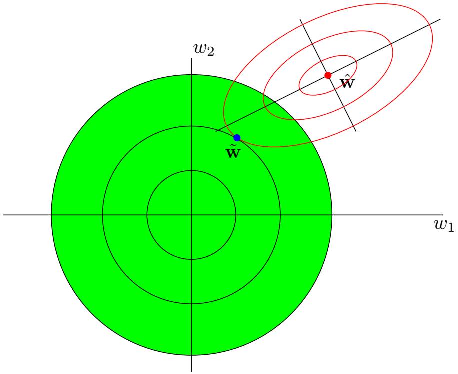
$$E(\mathbf{w}) = \sum_{i=1}^{n}(\mathbf{w}^T\mathbf{x}_i - y_i)^2 + \lambda\|\mathbf{w}\|_1$$
$$= \sum_{i=1}^{n}(\mathbf{w}^T\mathbf{x}_i - y_i)^2 + \lambda\sum_{j}|w_j|$$

- Penaliza parámetros grandes.
- Método de encogimiento (shrinkage)
- Con $\lambda \geq 0$, $E(\mathbf{w})$ es una función convexa, pero no diferenciable.
- Modelos con coeficientes dispersos

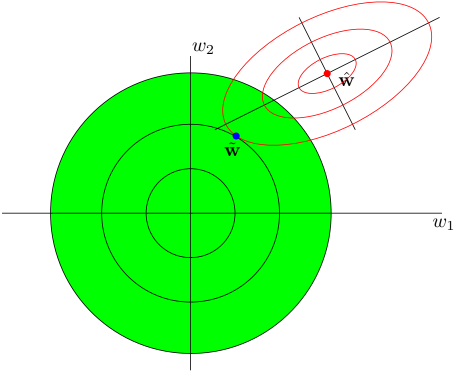

w_2
$\hat{\mathbf{w}}$
$\tilde{\mathbf{w}}$
w_1

Regularización y selección de modelo

Regularización y selección de modelo

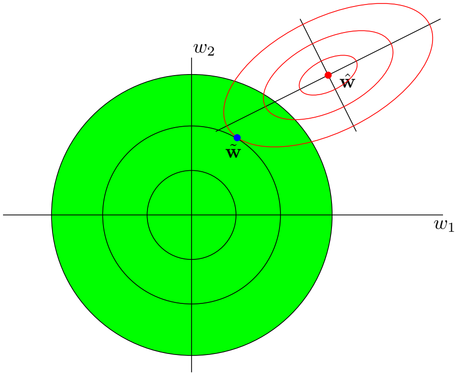


Regularización y selección de modelo



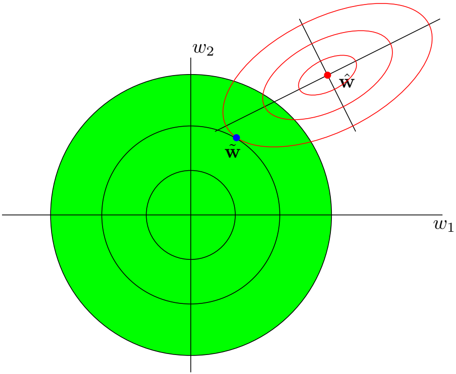
- Regularización restringe el conjunto de modelos que se pueden ajustar a los datos.

Regularización y selección de modelo



- Regularización restringe el conjunto de modelos que se pueden ajustar a los datos.
- Seleccionar $\lambda \longrightarrow$ selección de modelo

Regularización y selección de modelo



- Regularización restringe el conjunto de modelos que se pueden ajustar a los datos.
- Seleccionar $\lambda \longrightarrow$ selección de modelo
- Validación cruzada.

Weight Decay

Weight Decay

- Regularización con norma l_2 en redes neuronales:

Weight Decay

- Regularización con norma l_2 en redes neuronales:

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda \|\mathbf{w}\|^2$$

donde $\mathbf{w}$ es el vector que contiene todos los pesos de la red.

Weight Decay

- Regularización con norma l_2 en redes neuronales:

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda \|\mathbf{w}\|^2$$

  donde $\mathbf{w}$ es el vector que contiene todos los pesos de la red.
- $E(\mathbf{w})$ no es una función convexa!!

Weight Decay

- Regularización con norma l_2 en redes neuronales:

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda \|\mathbf{w}\|^2$$

  donde $\mathbf{w}$ es el vector que contiene todos los pesos de la red.
- $E(\mathbf{w})$ no es una función convexa!!
- Heurística:

Weight Decay

- Regularización con norma l_2 en redes neuronales:

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda \|\mathbf{w}\|^2$$

  donde $\mathbf{w}$ es el vector que contiene todos los pesos de la red.
- $E(\mathbf{w})$ no es una función convexa!!
- Heurística:
  - Pesos grandes resultan en curvatura grande de la función.

Weight Decay

- Regularización con norma l_2 en redes neuronales:

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda \|\mathbf{w}\|^2$$

  donde $\mathbf{w}$ es el vector que contiene todos los pesos de la red.
- $E(\mathbf{w})$ no es una función convexa!!
- Heurística:
  - Pesos grandes resultan en curvatura grande de la función.
  - Pesos pequeños producen funciones aproximadamente lineales.

Weight Decay

- Regularización con norma l_2 en redes neuronales:

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda\|\mathbf{w}\|^2$$

  donde $\mathbf{w}$ es el vector que contiene todos los pesos de la red.
- $E(\mathbf{w})$ no es una función convexa!!
- Heurística:
  - Pesos grandes resultan en curvatura grande de la función.
  - Pesos pequeños producen funciones aproximadamente lineales.
  - Corresponde a región de la sigmoide aproximadamente lineal.

- El gradiente modificado:

$$\nabla \tilde{E} = \nabla E + 2\lambda \mathbf{w}$$

- El gradiente modificado:

$$\nabla \tilde{E} = \nabla E + 2\lambda \mathbf{w}$$

- entrenando por descenso de gradiente, en el límite de tiempo continuo (ignorando E):

$$\frac{\partial \mathbf{w}}{\partial \tau} = -2\eta\lambda \mathbf{w}$$

- El gradiente modificado:

$$\nabla \tilde{E} = \nabla E + 2\lambda \mathbf{w}$$

- entrenando por descenso de gradiente, en el límite de tiempo continuo (ignorando E):

$$\frac{\partial \mathbf{w}}{\partial \tau} = -2\eta\lambda \mathbf{w}$$

- Solucionando:

$$\mathbf{w}(\tau) \approx \mathbf{w}(0) \exp(-2\eta\lambda\tau)$$

- El gradiente modificado:

$$\nabla \tilde{E} = \nabla E + 2\lambda \mathbf{w}$$

- entrenando por descenso de gradiente, en el límite de tiempo continuo (ignorando E):

$$\frac{\partial \mathbf{w}}{\partial \tau} = -2\eta\lambda \mathbf{w}$$

- Solucionando:

$$\mathbf{w}(\tau) \approx \mathbf{w}(0) \exp(-2\eta\lambda\tau)$$

- Es decir, los pesos decaen exponencialmente hacia cero.

- El gradiente modificado:

$$\nabla \tilde{E} = \nabla E + 2\lambda \mathbf{w}$$

- entrenando por descenso de gradiente, en el límite de tiempo continuo (ignorando E):

$$\frac{\partial \mathbf{w}}{\partial \tau} = -2\eta\lambda \mathbf{w}$$

- Solucionando:

$$\mathbf{w}(\tau) \approx \mathbf{w}(0) \exp(-2\eta\lambda\tau)$$

- Es decir, los pesos decaen exponencialmente hacia cero.
- El algoritmo de aprendizaje para la función de error modificada es una simple extensión de backpropagation.

- El gradiente modificado:

$$\nabla \tilde{E} = \nabla E + 2\lambda \mathbf{w}$$

- entrenando por descenso de gradiente, en el límite de tiempo continuo (ignorando E):

$$\frac{\partial \mathbf{w}}{\partial \tau} = -2\eta\lambda \mathbf{w}$$

- Solucionando:

$$\mathbf{w}(\tau) \approx \mathbf{w}(0) \exp(-2\eta\lambda\tau)$$

- Es decir, los pesos decaen exponencialmente hacia cero.
- El algoritmo de aprendizaje para la función de error modificada es una simple extensión de backpropagation.
- Restar $2\eta\lambda w_i$ cada vez que el peso w_i es modificado.

Eliminación de pesos

Eliminación de pesos

- Primo cercano de weight decay.

Eliminación de pesos

- Primo cercano de weight decay.
- La función de error en este caso es:

$$E(\tilde{\mathbf{w}}) = E(\mathbf{w}) + \lambda \sum_i \left(\frac{w_i^2/c^2}{1 + w_i^2/c^2} \right)$$

Eliminación de pesos

- Primo cercano de weight decay.
- La función de error en este caso es:

$$E(\tilde{\mathbf{w}}) = E(\mathbf{w}) + \lambda \sum_i \left(\frac{w_i^2/c^2}{1 + w_i^2/c^2} \right)$$

- Cuando $w_i \gg c$, se tiene $\frac{w_i^2/c^2}{1+w_i^2/c^2} \approx 1$, y el término regularizador cuenta el número de pesos.

Eliminación de pesos

- Primo cercano de weight decay.
- La función de error en este caso es:

$$E(\tilde{\mathbf{w}}) = E(\mathbf{w}) + \lambda \sum_i \left(\frac{w_i^2/c^2}{1 + w_i^2/c^2} \right)$$

- Cuando $w_i \gg c$, se tiene $\frac{w_i^2/c^2}{1+w_i^2/c^2} \approx 1$, y el término regularizador cuenta el número de pesos.
- Cuando $w_i \ll c$, se tiene $\frac{w_i^2/c^2}{1+w_i^2/c^2} \propto w_i^2$ y tenemos weight decay.

Eliminación de pesos

- Primo cercano de weight decay.
- La función de error en este caso es:

$$\tilde{E(\mathbf{w})} = E(\mathbf{w}) + \lambda \sum_i \left(\frac{w_i^2/c^2}{1 + w_i^2/c^2} \right)$$

- Cuando $w_i \gg c$, se tiene $\frac{w_i^2/c^2}{1+w_i^2/c^2} \approx 1$, y el término regularizador cuenta el número de pesos.
- Cuando $w_i \ll c$, se tiene $\frac{w_i^2/c^2}{1+w_i^2/c^2} \propto w_i^2$ y tenemos weight decay.
- Seleccionando c apropiadamente, podemos forzar a la red a buscar soluciones con unos pocos pesos grandes, o muchos pesos pequeños.

Técnicas de prunning

Técnicas de prunning

- Método de encogimiento

Técnicas de prunning

- Método de encogimiento
- Se comienza con una red suficientemente grande.

Técnicas de prunning

- Método de encogimiento
- Se comienza con una red suficientemente grande.
- Se entrena esta red.

Técnicas de prunning

- Método de encogimiento
- Se comienza con una red suficientemente grande.
- Se entrena esta red.
- Se remueven pesos y/o nodos.

Técnicas de prunning

- Método de encogimiento
- Se comienza con una red suficientemente grande.
- Se entrena esta red.
- Se remueven pesos y/o nodos.

Estrategia

- Eliminar pesos, pero manteniendo capacidad funcional para resolver el problema.

Estrategia

- Eliminar pesos, pero manteniendo capacidad funcional para resolver el problema.
  - Mejor generalización.

Estrategia

- Eliminar pesos, pero manteniendo capacidad funcional para resolver el problema.
  - Mejor generalización.
  - Entrenamiento más fácil.

Estrategia

- Eliminar pesos, pero manteniendo capacidad funcional para resolver el problema.
  - Mejor generalización.
  - Entrenamiento más fácil.
- Aproximación más simple:

Estrategia

- Eliminar pesos, pero manteniendo capacidad funcional para resolver el problema.
  - Mejor generalización.
  - Entrenamiento más fácil.
- Aproximación más simple:
  1. Entrenar.

Estrategia

- Eliminar pesos, pero manteniendo capacidad funcional para resolver el problema.
  - Mejor generalización.
  - Entrenamiento más fácil.
- Aproximación más simple:
  1. Entrenar.
  2. Eliminar pesos pequeños (comparar con umbral).

Estrategia

- Eliminar pesos, pero manteniendo capacidad funcional para resolver el problema.
  - Mejor generalización.
  - Entrenamiento más fácil.
- Aproximación más simple:
  1. Entrenar.
  2. Eliminar pesos pequeños (comparar con umbral).
- Problema : pesos pequeños pueden tener efecto grande en la función de error.

Saliency

- Medida de cuánto afecta el remover un peso dado a la función de error.

Saliency

- Medida de cuánto afecta el remover un peso dado a la función de error.
- Cerca al mínimo:

$$E \approx \delta\mathbf{w}^T \mathbf{H} \delta\mathbf{w}$$

donde $(\mathbf{H})_{ij} = \frac{\partial^2 E}{\partial w_i w_j}$.

Saliency

- Medida de cuánto afecta el remover un peso dado a la función de error.
- Cerca al mínimo:

$$E \approx \delta\mathbf{w}^T \mathbf{H} \delta\mathbf{w}$$

  donde $(\mathbf{H})_{ij} = \frac{\partial^2 E}{\partial w_i w_j}$.
- Asumiendo **H** diagonal:

Saliency

- Medida de cuánto afecta el remover un peso dado a la función de error.
- Cerca al mínimo:
$$E \approx \delta\mathbf{w}^T\mathbf{H}\delta\mathbf{w}$$
donde $(\mathbf{H})_{ij} = \frac{\partial^2 E}{\partial w_i w_j}$.
- Asumiendo **H** diagonal:
$$E \approx \sum_i \delta w_i^2 (\mathbf{H})_{ii}$$

Saliency

- Medida de cuánto afecta el remover un peso dado a la función de error.
- Cerca al mínimo:

$$E \approx \delta \mathbf{w}^T \mathbf{H} \delta \mathbf{w}$$

  donde $(\mathbf{H})_{ij} = \frac{\partial^2 E}{\partial w_i w_j}$.
- Asumiendo **H** diagonal:

$$E \approx \sum_i \delta w_i^2 (\mathbf{H})_{ii}$$

- Es decir

$$w_i \to 0 \quad \Rightarrow \quad \delta E = (\mathbf{H})_{ii} w_i^2$$

Optimal Brain Damage

1. Entrenar red relativamente grande.

Optimal Brain Damage

1. Entrenar red relativamente grande.
2. Entrenar hasta cierto criterio.

Optimal Brain Damage

1. Entrenar red relativamente grande.
2. Entrenar hasta cierto criterio.
3. Calcular $(\mathbf{H})_{ii}$.

Optimal Brain Damage

1. Entrenar red relativamente grande.
2. Entrenar hasta cierto criterio.
3. Calcular $(\mathbf{H})_{ii}$.
4. Ordenar los pesos de acuerdo a $(\mathbf{H})_{ii} w_i^2$

Optimal Brain Damage

1. Entrenar red relativamente grande.
2. Entrenar hasta cierto criterio.
3. Calcular $(\mathbf{H})_{ii}$.
4. Ordenar los pesos de acuerdo a $(\mathbf{H})_{ii} w_i^2$
5. Eliminar pesos con $(\mathbf{H})_{ii} w_i^2$ pequeño.

Optimal Brain Damage

1. Entrenar red relativamente grande.
2. Entrenar hasta cierto criterio.
3. Calcular $(\mathbf{H})_{ii}$.
4. Ordenar los pesos de acuerdo a $(\mathbf{H})_{ii}w_i^2$
5. Eliminar pesos con $(\mathbf{H})_{ii}w_i^2$ pequeño.
6. Volver a entrenar.

Optimal Brain Surgeon

- No asume $\mathbf{H}$ diagonal.

Optimal Brain Surgeon

- No asume **H** diagonal.
- Cómo afecta borrar w_i a los otros pesos?

Optimal Brain Surgeon

- No asume $\mathbf{H}$ diagonal.
- Cómo afecta borrar w_i a los otros pesos?
- Hallar $\delta\mathbf{w}$ que minimiza δE sujeto a $w_i = 0$.

Optimal Brain Surgeon

- No asume **H** diagonal.
- Cómo afecta borrar w_i a los otros pesos?
- Hallar $\delta\mathbf{w}$ que minimiza δE sujeto a $w_i = 0$.

$$\begin{aligned} \text{mín} \quad & \frac{1}{2}\delta\mathbf{w}^T\mathbf{H}\delta\mathbf{w} \\ \text{sujeto a} \quad & \mathbf{e}_i^T w_i + w_i = 0 \end{aligned}$$

Optimal Brain Surgeon

- No asume $\mathbf{H}$ diagonal.
- Cómo afecta borrar w_i a los otros pesos?
- Hallar $\delta\mathbf{w}$ que minimiza δE sujeto a $w_i = 0$.

$$\begin{aligned} \text{mín} \quad & \frac{1}{2}\delta\mathbf{w}^T\mathbf{H}\delta\mathbf{w} \\ \text{sujeto a} \quad & \mathbf{e}_i^T w_i + w_i = 0 \end{aligned}$$

- El lagrangiano:

$$L(\delta w, \lambda) =$$

Optimal Brain Surgeon

- No asume $\mathbf{H}$ diagonal.
- Cómo afecta borrar w_i a los otros pesos?
- Hallar $\delta\mathbf{w}$ que minimiza δE sujeto a $w_i = 0$.

$$
\begin{aligned}
\min \quad & \frac{1}{2}\delta\mathbf{w}^T\mathbf{H}\delta\mathbf{w} \\
\text{sujeto a} \quad & \mathbf{e}_i^T w_i + w_i = 0
\end{aligned}
$$

- El lagrangiano:

$$
L(\delta w, \lambda) = \frac{1}{2}\delta w^T\mathbf{H}\delta w +
$$

Optimal Brain Surgeon

- No asume **H** diagonal.
- Cómo afecta borrar w_i a los otros pesos?
- Hallar $\delta\mathbf{w}$ que minimiza δE sujeto a $w_i = 0$.

$$\begin{array}{rl} \text{mín} & \frac{1}{2}\delta\mathbf{w}^T\mathbf{H}\delta\mathbf{w} \\ \text{sujeto a} & \mathbf{e}_i^T w_i + w_i = 0 \end{array}$$

- El lagrangiano:

$$L(\delta w, \lambda) = \frac{1}{2}\delta w^T\mathbf{H}\delta w + \lambda(\mathbf{e}_i^T w_i + w_i)$$

- Derivando tenemos:

$$\nabla L_w = \mathbf{H}\delta\mathbf{w} + \lambda\mathbf{e}_i = 0$$
$$\nabla L_\lambda = \mathbf{e}_i^T\delta\mathbf{w} + w_i = 0$$

- Derivando tenemos:

$$\nabla L_w = \mathbf{H}\delta\mathbf{w} + \lambda\mathbf{e}_i = 0$$
$$\nabla L_\lambda = \mathbf{e}_i^T\delta\mathbf{w} + w_i = 0$$

- Obtenemos $\lambda = -\frac{w_i}{\mathbf{e}_i^T\mathbf{H}^{-1}\mathbf{e}_i}$ y $\delta\mathbf{w} = \frac{-w_i\mathbf{H}^{-1}\mathbf{e}_i}{[\mathbf{H}^{-1}]_{ii}}$

- Derivando tenemos:

$$\nabla L_w = \mathbf{H}\delta\mathbf{w} + \lambda\mathbf{e}_i = 0$$
$$\nabla L_\lambda = \mathbf{e}_i^T\delta\mathbf{w} + w_i = 0$$

- Obtenemos $\lambda = -\frac{w_i}{\mathbf{e}_i^T\mathbf{H}^{-1}\mathbf{e}_i}$ y $\delta\mathbf{w} = \frac{-w_i\mathbf{H}^{-1}\mathbf{e}_i}{[\mathbf{H}^{-1}]_{ii}}$
- Reemplazando en δE obtenemos el saliency:

$$L_i = \frac{w_i^2}{2(\mathbf{H}^{-1})_{ii}}$$

- Cerca al mínimo $\mathbf{H} \approx \mathbf{J}^T\mathbf{J} \Rightarrow$ se puede aproximar $\mathbf{H}^{-1}$ invirtiendo $\mathbf{J}^T\mathbf{J} + \epsilon\mathbf{I}$ para ϵ pequeño, usando fórmula de inversión de Sherman-Morrison-Woodbury.

- Cerca al mínimo $\mathbf{H} \approx \mathbf{J}^T\mathbf{J} \Rightarrow$ se puede aproximar $\mathbf{H}^{-1}$ invirtiendo $\mathbf{J}^T\mathbf{J} + \epsilon\mathbf{I}$ para ϵ pequeño, usando fórmula de inversión de Sherman-Morrison-Woodbury.

  $$\mathbf{A}_1 = \mathbf{A}_0 + \mathbf{XRY}$$

- Cerca al mínimo $\mathbf{H} \approx \mathbf{J}^T\mathbf{J} \Rightarrow$ se puede aproximar $\mathbf{H}^{-1}$ invirtiendo $\mathbf{J}^T\mathbf{J} + \epsilon\mathbf{I}$ para ϵ pequeño, usando fórmula de inversión de Sherman-Morrison-Woodbury.

$$\mathbf{A}_1 = \mathbf{A}_0 + \mathbf{XRY} \Rightarrow \mathbf{A}_1^{-1} = \mathbf{A}_0^{-1} - \mathbf{A}_0^{-1}\mathbf{X}(\mathbf{Y}\mathbf{A}_0^{-1}\mathbf{X} + \mathbf{R}^{-1})^{-1}\mathbf{Y}\mathbf{A}_0^{-1}$$

- Cerca al mínimo $\mathbf{H} \approx \mathbf{J}^T\mathbf{J} \Rightarrow$ se puede aproximar $\mathbf{H}^{-1}$ invirtiendo $\mathbf{J}^T\mathbf{J} + \epsilon\mathbf{I}$ para ϵ pequeño, usando fórmula de inversión de Sherman-Morrison-Woodbury.

$$\mathbf{A}_1 = \mathbf{A}_0 + \mathbf{XRY} \Rightarrow \mathbf{A}_1^{-1} = \mathbf{A}_0^{-1} - \mathbf{A}_0^{-1}\mathbf{X}(\mathbf{Y}\mathbf{A}_0^{-1}\mathbf{X} + \mathbf{R}^{-1})^{-1}\mathbf{Y}\mathbf{A}_0^{-1}$$

- En este caso:

- Cerca al mínimo $\mathbf{H} \approx \mathbf{J}^T\mathbf{J} \Rightarrow$ se puede aproximar $\mathbf{H}^{-1}$ invirtiendo $\mathbf{J}^T\mathbf{J} + \epsilon\mathbf{I}$ para ϵ pequeño, usando fórmula de inversión de Sherman-Morrison-Woodbury.

$$\mathbf{A}_1 = \mathbf{A}_0 + \mathbf{XRY} \Rightarrow \mathbf{A}_1^{-1} = \mathbf{A}_0^{-1} - \mathbf{A}_0^{-1}\mathbf{X}(\mathbf{YA}_0^{-1}\mathbf{X} + \mathbf{R}^{-1})^{-1}\mathbf{Y}\mathbf{A}_0^{-1}$$

- En este caso:

$$\left(\epsilon\mathbf{I} + \mathbf{g}\mathbf{g}^T\right)^{-1}$$

- Cerca al mínimo $\mathbf{H} \approx \mathbf{J}^T\mathbf{J} \Rightarrow$ se puede aproximar $\mathbf{H}^{-1}$ invirtiendo $\mathbf{J}^T\mathbf{J} + \epsilon\mathbf{I}$ para ϵ pequeño, usando fórmula de inversión de Sherman-Morrison-Woodbury.

$$\mathbf{A}_1 = \mathbf{A}_0 + \mathbf{XRY} \Rightarrow \mathbf{A}_1^{-1} = \mathbf{A}_0^{-1} - \mathbf{A}_0^{-1}\mathbf{X}(\mathbf{Y}\mathbf{A}_0^{-1}\mathbf{X} + \mathbf{R}^{-1})^{-1}\mathbf{Y}\mathbf{A}_0^{-1}$$

- En este caso:

$$\left(\epsilon\mathbf{I} + \mathbf{g}\mathbf{g}^T\right)^{-1} = \frac{1}{\epsilon}\left(\mathbf{I} - \frac{1}{\|\mathbf{g}\|^2 + \epsilon}\mathbf{g}\mathbf{g}^T\right)$$

- Cerca al mínimo $\mathbf{H} \approx \mathbf{J}^T\mathbf{J} \Rightarrow$ se puede aproximar $\mathbf{H}^{-1}$ invirtiendo $\mathbf{J}^T\mathbf{J} + \epsilon\mathbf{I}$ para ϵ pequeño, usando fórmula de inversión de Sherman-Morrison-Woodbury.

$$\mathbf{A}_1 = \mathbf{A}_0 + \mathbf{XRY} \Rightarrow \mathbf{A}_1^{-1} = \mathbf{A}_0^{-1} - \mathbf{A}_0^{-1}\mathbf{X}(\mathbf{Y}\mathbf{A}_0^{-1}\mathbf{X} + \mathbf{R}^{-1})^{-1}\mathbf{Y}\mathbf{A}_0^{-1}$$

- En este caso:

$$\left(\epsilon\mathbf{I} + \mathbf{g}\mathbf{g}^T\right)^{-1} = \frac{1}{\epsilon}\left(\mathbf{I} - \frac{1}{\|\mathbf{g}\|^2 + \epsilon}\mathbf{g}\mathbf{g}^T\right)$$

- $\mathbf{g}$: Backprop.

Optimal Brain Surgeon

1. Entrenar red razonablemente grande hasta error mínimo.

Optimal Brain Surgeon

1. Entrenar red razonablemente grande hasta error mínimo.
2. Calcular $\mathbf{H}^{-1}$.

Optimal Brain Surgeon

1. Entrenar red razonablemente grande hasta error mínimo.
2. Calcular $\mathbf{H}^{-1}$.
3. Encontrar el peso w_i que de el saliency más pequeño.

Optimal Brain Surgeon

1. Entrenar red razonablemente grande hasta error mínimo.
2. Calcular $\mathbf{H}^{-1}$.
3. Encontrar el peso w_i que de el saliency más pequeño.
4. Actualizar todos los pesos con el $\delta\mathbf{w}$ correspondiente.

Optimal Brain Surgeon

1. Entrenar red razonablemente grande hasta error mínimo.
2. Calcular $\mathbf{H}^{-1}$.
3. Encontrar el peso w_i que de el saliency más pequeño.
4. Actualizar todos los pesos con el $\delta\mathbf{w}$ correspondiente.
5. Volver al paso 2 (hasta que no se puedan remover más pesos con incremento significativo del error).