07 - EVALUATING LEARNING ALGORITHMS

Supponiamo di aver implementato un algoritmo di learning ma quando lo testiamo su un nuovo dataset, fa grossi errori di predizione.
Ci sono varie cose che potrei provare cambiando l'algoritmo o gli esempi.

Machine learning diagnostic

Tecniche per fare delle scelte sensate per migliorare un algoritmo che richiedono tempo.

Evaluating a hypothesis

Si cerca di minimizzare il training error e, una volta ottenuti buoni risultati, si va verificare come l'algoritmo si comporta con dei nuovi dati andando a valutare la funzione di costo sul dataset di dev/test.
Tutto dipende dalla funzione di costo che abbiamo in base al tipo di algoritmo.

Lavorando sul dataset, eseguo un'operazione di model selection.

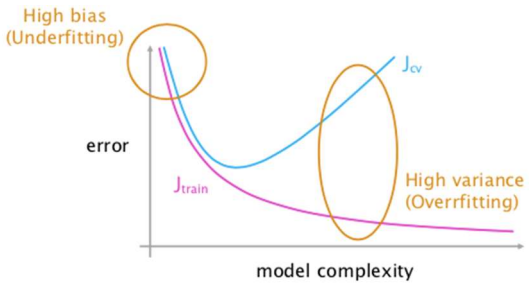
Model selection

Si cerca di affinare il modello facendo dei cambiamenti, configurandolo su tutti i suoi parametri (livelli, unit, activation function)
Dati diversi modelli bisogna "fittarli", quindi **minimizzare sempre la funzione di costo sul training set**, calcolare il cross-validation error, scegliere il modello con l'errore più basso e valutare come generalizzi vedendo il comportamento sul test set.

Se qualcosa non funziona, devo rendermente conto.

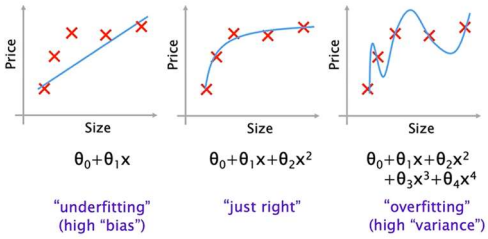
Diagnosing high bias/variance

Comparazione tra il training error e il cross-validation error con il crescere della complessità del modello.



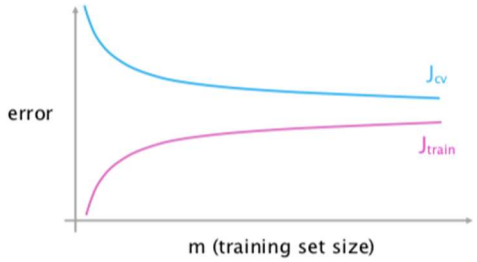
Con un modello molto semplice si ha un **alto bias** in quanto la rete non riesce a modellare sufficientemente il problema. → **<u>underfitting</u>**
Man mano che aumenta la complessità del modello si può avere **alta varianza** → **<u>overfitting</u>**.

<u>L'errore di cross-validation infatti lo dimostra con il gap.</u>



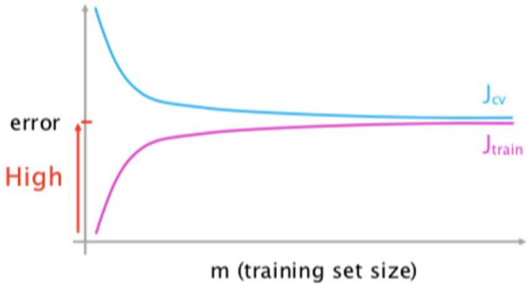
Learning Curves

Si analizza cosa succede in **dipendenza della dimensione del training set**.
È come se artificiosamente partissi da 1 esempio nel dataset per aggiungerne man mano.
All'inizio l'ipotesi da imparare è semplice dato il singolo esempio quindi il training error è basso.
Man mano l'errore per imparare l'ipotesi cresce con i dati fino a raggiungere una "stabilità" asintotica.
L'opposto accade per l'errore di cross-validation.



Jtrain: errore di training cresce
Jcv: errore di cross-validation che decrementa man mano che aumento i dati di training

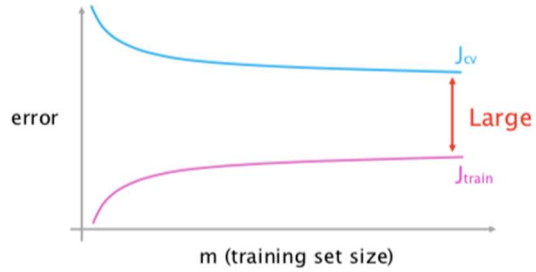
In una situazione di alto bias



Gli errori raggiungono l'asintoticità ad un valore alto perché il modello è troppo semplice e aggiungere dati e basta potrebbe non servire.

- Non cè abbastanza variabilità → bias

In una situazione di alta varianza



C'è un gap largo con l'errore di cross-validation → alta varianza (probabilmente ho over-fittato i dati)

Gli errori sembrano poter migliorare aumentando i dati rispetto a quello che non sembra con alto bias → il problema sembra poter essere affrontabile aumentando i dati.

Errore con le classi skewed (sbilanciate)

Supponiamo di tornare all'esempio del classificatore del tumore maligno o benigno. Abbiamo usato una regressione logistica impostando una soglia pari a 0.5
Troviamo che il modello ha un errore dell'1% di cross-validation/test set
Il modello sembra poter effettuare nel 99% dei casi delle diagnosi corrette. Però se solo lo 0.5% dei pazienti ha il cancro avere l'errore dell'1% non sembra essere molto buono.
Questo tipo di classe si chiama skewed perché è sbilanciata. Una metrica che si può utilizzare non è semplicemente la funzione di costo ma **ci si può affidare ai precision/recall.**

Precision/Recall

Valutiamo le prestazioni del classificatore tramite la seguente tabella

		Actual value: 1	Actual value: 0
Predicted value	1	True positive	False positive
	0	False negative	True negative

Precision: dati tutti i pazienti a cui ho predetto il cancro, quanti ce l'hanno effettivamente?

$$\text{Precision} = \frac{\text{True pos}}{\text{\#pred. positive}} = \frac{\text{True pos}}{\text{True pos} + \text{False pos}}$$

Recall: dati tutti i pazienti che hanno il cancro, a quanti era stato predetto correttamente?

$$\text{Recall} = \frac{\text{True pos}}{\text{\#actual positive}} = \frac{\text{True pos}}{\text{True pos} + \text{False neg}}$$

Nota: classe 1, quella più rara.

Trading off precision and recall

Supponiamo di voler predire il cancro **solo se siamo veramente sicuri** quindi aumento il threshold alzando la precision, ma abbassando il recall.
Supponiamo di voler evitare di **non predirre troppi casi di cancro** (evitare false negative) quindi abbasso la soglia alzando il recall, ma abbassando la precisione.

Quindi come mettere a confronto la precision e la recall?

F1 (o F) score → media armonica

Immaginiamo di aver provato diversi algoritmi trovando precision e recall di ognuno

$$F_1 \text{ score} = \frac{P \times R}{P+R}$$

	Precision (P)	Recall (R)	Average	F_1 Score
Algorithm 1	0.5	0.4	0.45	0.444
Algorithm 2	0.7	0.1	0.4	0.175
Algorithm 3	0.02	1	0.51	0.039

*F_1 score = (P*R) / (P+R) è una media armonica e fa in modo che il risultato complessivo non possa essere buono se P o R è molto basso.*

Però ci possono anche **essere casi in cui basta la semplice media aritmetica**:

Algorithm	US	China	India	Other	Average
A	3%	7%	5%	9%	6%
B	5%	6%	5%	10%	6.5%
C	2%	3%	4%	5%	3.5%
D	5%	8%	7%	2%	5.25%
E	4%	5%	2%	4%	3.75%
F	7%	11%	8%	12%	9.5%

Ottimizzare / soddisfare le metriche

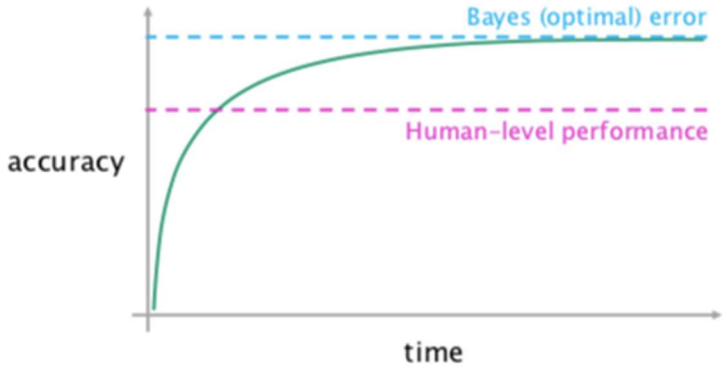
Siamo interessati alle performance di un classificatore in base a due metriche

Algorithm	Accuracy	Running time
A	90%	80ms
B	92%	95ms
C	95%	1,500ms

Una possibile metrica per unirle può essere: accuracy + 0.5 * running time
Lo scopo per la nuova metrica è di massimizzare l'accuracy (otpimizing) mantenendo il running time <= 100 ms (satisficing)

Quindi vogliamo ottimizzare una metrica e soddisfare un'altra. In generale al più si può ottimizzare una metrica soddisfacendo le altre.

Human-level performance



→ livello di accuratezza di un umano

In generale sapere quale risultato raggiungerebbe un essere umano aiuta a capire quali dati prendere e su cosa concentrarmi.

Immaginiamo un task di classificazione binario (cani o gatti)

Human error (≈ Bayes error)	1%	7.5%
	Avoidable bias	
Training error	8%	8%
		Variance
Cross-val./dev. error	11%	11%
	Focus on bias	Focus on variance

Se l'errore umano è confrontabile con quello di training ci si deve concentrare sulla varianza altrimenti se si ha gap tra questi due c'è alto bias.

Ortogonalizzazione

- Chain of assumptions (objectives) in ML
  - Fits training set well on cost function
  - Fits cross-validation/dev. set well on cost function
  - Fits test set well on cost function
  - Performs well in real world

Voglio trovare degli strumenti che servono a soddisfare le cose precedenti senza che qualcuno interferisca con l'altro quindi in maniera ortogonale.

- Si può cambiare modello
- Si possono cambiare iperparametri

Nota: quello che si cerca di fare è che le modifiche non si inflenzino tra loro ma per ora è un dilemma. Se abbasso bias, rischio di alzare varianza