

Ensembling Method

Bias-Variance Decomposition

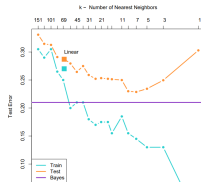
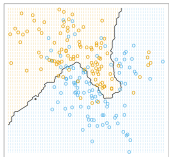
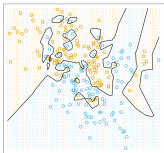
Quando facciamo delle stime, abbiamo una media $E[y]=y$.

Ogni singola misura può spostarsi da questa media, è abbastanza normale, ma la media deve essere quella. Se mi discosto da quella media, ho un BIAS. Non è una varianza attenzione!

Varianza: quanto mi sposto dalla media, es: media = 10, varianza = 3, i valori vanno da 7 a 13.

Bias: media = 10, ma io ottengo media = 14, quindi sto distante dalla media.

- ▶ Recall that overly simple models underfit the data, and overly complex models overfit.



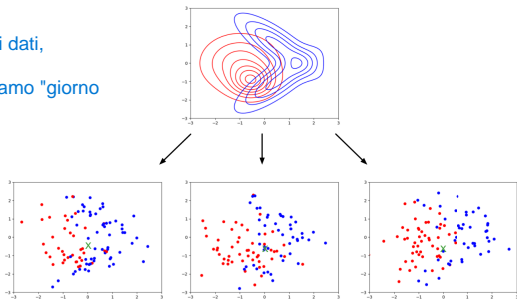
- ▶ We can quantify this effect in terms of the **bias/variance decomposition**.
- ▶ Bias and variance of what?

Bias-Variance Decomposition: Basic Setup

< valore riga "i" associato alla colonna $t(i)$ >

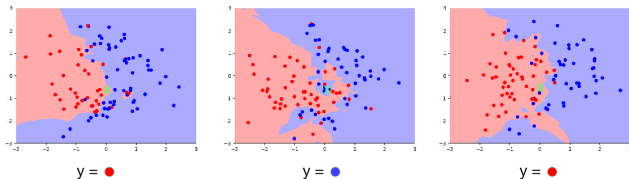
- ▶ Suppose the training set consists of pair $(\mathbf{x}^{(i)}, t^{(i)})$ sampled independent and identically distributed (i.i.d.) from a single data generating distribution P_{sample} .
- ▶ Pick a fixed query point \mathbf{x} (the green \mathbf{x} in the figure)
- ▶ Consider an experiment where we sample lot training sets independently from P_{sample}

qui sto partizionando i dati,
indipendenti,
ad esempio partizioniamo "giorno
per giorno".



Bias-Variance Decomposition: Basic Setup

- ▶ Let's run our learning algorithm on each training set, and compute its prediction y at the query point x .
- ▶ We can view y as a random variable, where the randomness comes from the choice of training set.
- ▶ The classification accuracy is determined by the distribution of y .

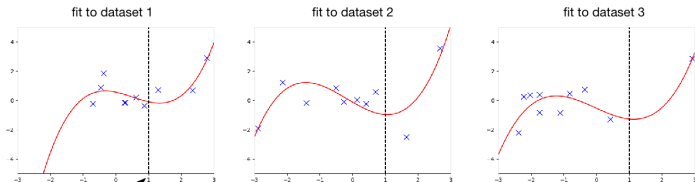


ora, su questi training set, spezzati giorno per giorno, eseguo l'algoritmo di learning. Quindi sto semplicemente applicando quello che ho fatto fino ad ora, ma separando i dati che sono raggruppati secondo qualche criterio. Per ogni gruppo faccio una predizione, che sarà diversa di gruppo in gruppo probabilmente.

Bias-Variance Decomposition: Basic Setup

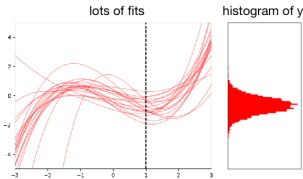
Here is the analogous for regression

qui abbiamo dei grafici rossi che fittano i vari dataset.



query location

fissiamo una certa " x ", e vediamo il valore che assume $y(x)$ in ogni dataset.



come vediamo, mettendo insieme sovrapposti tutti i dataset, abbiamo delle y diverse per la ' x ' cercata, e quindi y è una variabile aleatoria (x è fissa), con tutti gli aspetti di una variabile aleatoria, di conseguenza

Since y is a random variable, we can talk about its expectation, variance, etc.

L'accuratezza dipende dalla distribuzione di " y ". Dall'istogramma vediamo che la forma è simil-gaussiana. Vogliamo: media = y , varianza $\rightarrow 0$, bias $\rightarrow 0$ (\rightarrow è tendente)

Bias-Variance Decomposition: Basic Setup

Abbiamo visto che una predizione dipende sicuramente dal dataset che gli forniamo. Se avessi tanti dataset otterrei predizioni più o meno diverse, e come visto sopra otterremmo anche una distribuzione.

- ▶ Recap the basic setup
 - ▶ Fix a query point \mathbf{x}
 - ▶ Repeat:
 - ▶ Sample a random training dataset $(\mathbf{x}^{(i)}, t^{(i)})$ from the data generating distribution P_{sample}
 - ▶ Run the learning algorithm on $(\mathbf{x}^{(i)}, t^{(i)})$ to get a prediction y at \mathbf{x}
 - ▶ Sample the true target from the conditional distribution $P(y | \mathbf{x})$
- ▶ This gives a distribution over the loss at \mathbf{x} , with expectation $E[\mathcal{L}(y; t) | \mathbf{x}]$.
- ▶ For each query point \mathbf{x} , the expected loss is different. We are interested in minimizing the expectation of this with respect to \mathbf{x} sampled according to P_{sample} .

Bias-Variance Decomposition

Matematicamente facciamo delle assunzioni:

$g(\mathbf{x})$ è deterministica

ϵ è indipendente, randomica di media 0.

- Assume that $t = g(\mathbf{x}) + \epsilon$
 - Noise ϵ is sampled from a normal distribution with mean 0 and variance σ^2 : $\epsilon \sim N(0, \sigma^2)$
 - Noise lower-bounds the performance we can achieve
- Recall we want to minimize the objective function

questa è la
funzione costo da
minimizzare
(Sarebbe la media
delle funzioni loss)

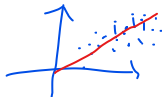
$$\mathcal{J}(\theta) = \frac{1}{N} \sum_{i=1}^N (t^{(i)} - f_{\mathbf{w}}(\mathbf{x}^{(i)}))$$

calcolata mediante
uno degli algoritmi
di ML visti.

- We can re-write this as the expected value of the squared error:

$$E(t - f_{\mathbf{w}}(\mathbf{x}))^2$$

le nostre "t" seguono una curva:



Bias-Variance Decomposition

aggiungo e sottraggo $g(x)$, in questo modo applico il quadrato di un binomio: $a^2 + b^2 + 2ab$, ponendo: $a = t - g(x)$ e $b = g(x) - f_w(x)$

$$\begin{aligned}E[(t - f_w(x))^2] &= E[(t - g(x) + g(x) - f_w(x))^2] \\&= E[(t - g(x))^2] + E[(g(x) - f_w(x))^2] \\&\quad + 2E[(g(x) - f_w(x))(t - g(x))] \\&= E[(t - g(x))^2] + E[(g(x) - f_w(x))^2] \\&\quad * + 2(E[g(x)f_w(x)] + E[tg(x)] - E[tf_w(x)] - E[g(x)^2])\end{aligned}$$

sviluppo qui il doppio prodotto

the last four terms cancel out...Therefore

$$\begin{aligned}E[(t - f_w(x))^2] &= E[(t - g(x))^2] + E[(g(x) - f_w(x))^2] \\&= \text{Var}[\epsilon] + E[(g(x) - f_w(x))^2]\end{aligned}$$

$\rightarrow t = \text{var}[t - g(x)] = \text{var}[g(x) + e - g(x)]$



Le due componenti in rosso ed in blu si semplificano, quindi il doppio prodotto è 0.

Nel caso rosso, se prendiamo il secondo elemento, si ha che

$E\{t * f(x)\} = E\{[g(x)+e] * f(x)\} = E[g(x)f(x)] + E[e*f(x)]$: il primo termine si semplifica con il primo pezzo in rosso, il secondo, cioè $E[e*f(x)] = E[e]*E[f(x)]$, perchè 'e' indipendente, ma $E[e] = 0$, quindi anche questo pezzo si toglie tutto. Similmente si risolve anche la parte blu.

Bias-Variance Decomposition

$$\begin{aligned}E[(t - f_{\mathbf{w}}(\mathbf{x}))^2] &= \text{Var}[\epsilon] + E[(g(\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x}))^2] \\&= \text{Var}[\epsilon] + E[(g(\mathbf{x}) - E[f_{\mathbf{w}}(\mathbf{x})] + E[f_{\mathbf{w}}(\mathbf{x})] - f_{\mathbf{w}}(\mathbf{x}))^2] \\&= \text{Var}[\epsilon] + E[(g(\mathbf{x}) - E[f_{\mathbf{w}}(\mathbf{x})])^2] + E[(E[f_{\mathbf{w}}(\mathbf{x})] - f_{\mathbf{w}}(\mathbf{x}))^2] \\&\quad + 2E[(E[f_{\mathbf{w}}(\mathbf{x})] - f_{\mathbf{w}}(\mathbf{x}))(g(\mathbf{x}) - E[f_{\mathbf{w}}(\mathbf{x})])] \\&= \text{Var}[\epsilon] + E[(g(\mathbf{x}) - E[f_{\mathbf{w}}(\mathbf{x})])^2] + E[(E[f_{\mathbf{w}}(\mathbf{x})] - f_{\mathbf{w}}(\mathbf{x}))^2] \\&\quad + 2(E[g(\mathbf{x})E[f_{\mathbf{w}}(\mathbf{x})]] - E[E[f_{\mathbf{w}}(\mathbf{x})]^2] - E[g(\mathbf{x})f_{\mathbf{w}}(\mathbf{x})] + E[f_{\mathbf{w}}(\mathbf{x})E[f_{\mathbf{w}}(\mathbf{x})]])\end{aligned}$$

...by similar arguments as before we get

$$E[(t - f_{\mathbf{w}}(\mathbf{x}))^2] = \text{Var}[\epsilon] + E[(g(\mathbf{x}) - E[f_{\mathbf{w}}(\mathbf{x})])^2] + E[(E[f_{\mathbf{w}}(\mathbf{x})] - f_{\mathbf{w}}(\mathbf{x}))^2]$$

$$E[(t - f_{\mathbf{w}}(\mathbf{x}))^2] = \text{bias}(f_{\mathbf{w}}(\mathbf{x}))^2 + \text{var}(f_{\mathbf{w}}(\mathbf{x})) + \sigma^2$$

 rappresenta cose
"non modellabili"

Bias-Variance Decomposition

$$E[(t - f_{\mathbf{w}}(\mathbf{x}))^2] = \text{bias}(f_{\mathbf{w}}(\mathbf{x}))^2 + \text{var}(f_{\mathbf{w}}(\mathbf{x})) + \sigma^2$$

- ▶ We just split the expected loss into three terms:
 - ▶ **bias**: how wrong the expected prediction is (corresponds to **underfitting**) indice di un modello troppo semplice
 - ▶ **variance**: the amount of variability in the predictions (corresponds to **overfitting**) indice di un modello buono sul training, pessimo nel testing non migliorabile, in quanto non predicibile
 - ▶ **Bayes error**: the inherent **unpredictability** of the targets
- ▶ Even though this analysis only applies to squared error, we often loosely use "bias" and "variance" as synonyms for "underfitting" and "overfitting".

Bagging: Motivation

(meglio leggere da slide 12 a slide 15, e poi tornare qui)

- ▶ Suppose we could somehow **sample m independent training** sets from P_{sample} .
- ▶ We could then compute the prediction y_i based on each one, and take the **average $y = \frac{1}{m} \sum_{i=1}^m y_i$** . nb: qui la media viene chiamata "y", non $E[y]$, sotto capiamo perchè!
- ▶ How does this affect the three terms of the expected loss?
 - ▶ **Bayes error: unchanged**, since we have **no control over it**
 - ▶ **Bias: unchanged**, since the **averaged prediction** has **the same expectation**

cioè, y è la media delle y_i ,
quindi $E[y] = E[E[y_i]] = E[y_i]$
in quanto fare la media di una media è ridondante

$$E[y] = E \left[\frac{1}{m} \sum_{i=1}^m y_i \right] = E[y_i]$$

- ▶ **Variance: reduced**, since we are averaging over independent samples

portando fuori $1/m$ dalla funzione varianza, cioè σ^2 , questo lo devo portare fuori al quadrato

$$Var[y] = Var \left[\frac{1}{m} \sum_{i=1}^m y_i \right] = \frac{1}{m^2} \sum_{i=1}^m Var[y_i] = \frac{1}{m} Var[y_i]$$

Se ho indipendenza tra le varianze

Bagging: The Idea

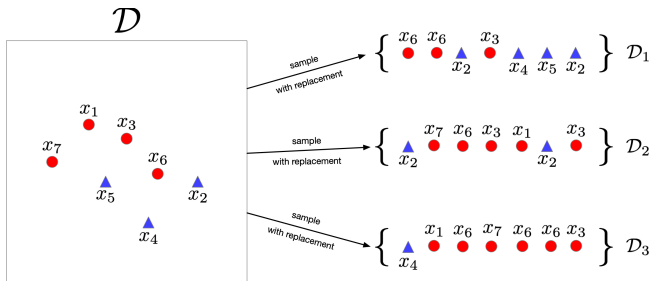
Ho un dataset, con cui ci faccio training e validation. Spesso ho pochi dati, come posso avere più dati da il sample che ho? L'idea è crearne altri sample a partire da quell'unico dataset che ho.

- ▶ In practice, the sampling distribution P_{sample} is often finite or expensive to sample from. (quindi i set li assumiamo "diversi", però non è che siano proprio indipendenti!)
- ▶ So training separate models on **independently sampled** datasets is very wasteful of data!
- ▶ **Solution:** given training set \mathcal{D} , use the empirical distribution $P_{\mathcal{D}}$ as a proxy for P_{sample} . This is called **bootstrap aggregation**, or **bagging**.
 - ▶ Take a single dataset \mathcal{D} with n examples.
 - ▶ Generate **m new datasets** ("resamples" or "bootstrap samples"), each by **sampling n training examples from \mathcal{D} , with replacement**.
 - ▶ Average the predictions of models trained on each of these datasets.
- ▶ The bootstrap is one of the most important ideas in all of statistics!
 - ▶ Intuition: As $|\mathcal{D}| \rightarrow \infty$, we have $P_{\mathcal{D}} \rightarrow P_{sample}$.

Più \mathcal{D} è grande, più il dataset è buono, perchè posso catturare più info se l'insieme di partenza è grande.

Bagging

Qui abbiamo dei dati in D , creo dei sottoinsiemi, pescando con reintroduzione.
I dati partono sempre da D , quindi potrei avere più copie di un certo x_i in D_1 , D_2 o con D_3 .
In questi sottoinsiemi posso fare ciò che normalmente faccio sempre, media, varianza etc.

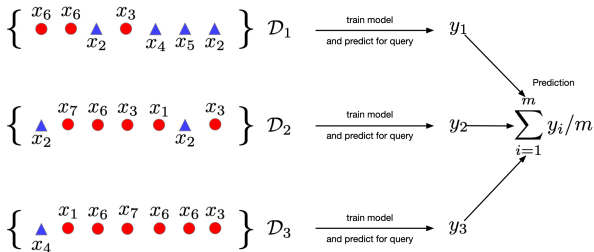


Fare così statisticamente è meglio che non farlo proprio!
(idealmente, pescando a caso, ma con stessi dati, riesco a muovermi bene).

In this example $n = 7, m = 3$

Bagging

Qui, per ogni sottoinsieme, faccio una predizione per una certa query, e poi devo mettere "insieme" queste predizioni, per cui faccio una media non pesata (cioè ogni dataset ha stessa "validità").



quindi stiamo dicendo che la predizione y "generale" = $\text{media}[y_i]$

predicting on a query point x

Bagging: Effect on Hypothesis Space

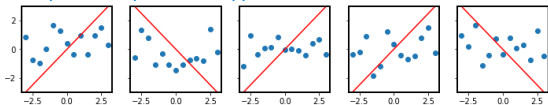
- ▶ We saw that in case of squared error, bagging does not affect bias.
- ▶ But it can change the hypothesis space
- ▶ Illustrative example:

- ▶ $x \sim \mathcal{U}(-3, 3), t \sim \mathcal{N}(0, 1)$

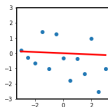
- ▶ $\mathcal{H} = \{wx | w \in \{-1, 1\}\}$

- ▶ Sampled datasets & fitted hypotheses:

Stiamo dicendo che la tecnica di Bagging può alterare lo spazio delle ipotesi. Supponiamo che lo spazio di partenza sia da -3 a +3, ed io prendo come spazio delle ipotesi solo due valori: -1, +1, quindi ogni predizione sarà rappresentata da una pendenza pari a "+1" oppure "-1".



- ▶ Ensembled hypotheses (mean over 1000 samples):



Se poi metto insieme questi risultati, e faccio una media di queste curve, ottengo una retta di pendenza vicino allo 0, fuori dalle ipotesi iniziali!

- ▶ The ensemble hypothesis is not in the original hypothesis space!

Bagging for Binary Classification

Se contesto probabilistico, quindi con probabilità COMPRESA TRA 0 ED 1, l'ago della valutazione diventa da 0.5
Se supero 0.5, allora l'esito è 1. Se non supero, l'esito è 0.

- ▶ If our classifiers output **real-valued probabilities**, $z_i \in [0, 1]$, then we can **average the predictions before thresholding**:

$$y_{bagged} = \mathbb{I}(z_{bagged} > 0,5) = \mathbb{I}\left(\sum_{i=1}^m \frac{z_i}{m} > 0,5\right)$$

- ▶ If our classifiers **output binary decisions**, $y_i \in \{0; 1\}$, then we can **average the predictions before thresholding**:

Se la classificazione genera valori "0" o "1",
vedo la media delle probabilità come si comporta,
se >0.5 o meno.

$$y_{bagged} = \mathbb{I}\left(\sum_{i=1}^m \frac{y_i}{m} > 0,5\right)$$

This is the same as **taking a majority vote**.

- ▶ A bagged classifier can be stronger than the average underlying model.

Bagging: Effect of Correlation

Prima abbiamo fatto ragionamenti DANDO PER SCONTATO L'INDIPENDENZA.
In realtà è più facile NON averla che averla. Come ci comportiamo in caso di indipendenza?

- ▶ Problem: the datasets are not independent, so we don't get the $1/m$ variance reduction.
 - ▶ it is possible to show that if the correlation ρ , then

$$\text{Var}[y] = \text{Var} \left[\frac{1}{m} \sum_{i=1}^m y_i \right] = \frac{1}{m} (1 - \rho) \sigma^2 + \rho \sigma^2$$

- ▶ It can be advantageous to introduce additional variability into your algorithm, as long as it reduces the correlation between samples.
 - ▶ Can help to use average over multiple algorithms, or multiple configurations of the same algorithm.

Se ρ è 0, quindi correlazione = 0, quindi indipendenza, torno al caso principale. Se invece è 1, allora tutti i dati gli stessi dati, quindi ho totale dipendenza, e quindi NON MIGLIORO LA VARIANZA.

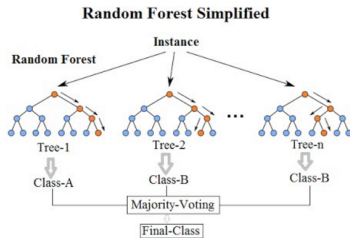
Ora, abbiamo visto bagging, dove lo applico? potenzialmente è sfruttabile da ogni algoritmo di machine learning, tra cui RANDOM FOREST, che vediamo ora:

Random Forests

metto insieme alberi decisionali con tecniche di bagging.
Quindi genero dataset diversi (bagging) e per ogni dataset creo un albero (ad esempio con ID3). Insieme formano Random Forest.

- ▶ **Random forests** = **bagged decision trees**, with one extra trick to decorrelate the predictions
 - ▶ When choosing each node of the decision tree, choose a random set of d input features, and only consider splits on those features

ID3 potrebbe però creare alberi piuttosto simili, questo avviene se può scegliere tra tutte le feature, in quanto sceglie la migliore, ed è probabile che soprattutto le feature iniziali siano le migliori per più dataset. La soluzione è fornire ad ogni albero un sottoinsieme di feature su cui lavorare, così differenzio.



- ▶ Random forests are probably the best black-box machine learning algorithm - they often work well with no tuning whatsoever.
 - ▶ one of the most widely used algorithms in Kaggle competitions

Bagging Summary

riduce overfitting, perchè l'overfitting lo ho con modello semplice, spesso dato dalla presenza di pochi dati, e quindi col bagging è come se lavorassi su più dati.

- ▶ Bagging reduces overfitting by averaging predictions.
- ▶ Used in most competition winners
 - ▶ Even if a single model is great, a small ensemble usually helps.
- ▶ Limitations:
 - ▶ Does not reduce bias in case of squared error.
 - ▶ There is still correlation between classifiers.
 - ▶ Random forest solution: Add more randomness.
 - ▶ Naive mixture (all members weighted equally).
 - ▶ If members are very different (e.g., different algorithms, different data sources, etc.), we can often obtain better results by using a principled approach to weighted ensembling.
- ▶ Boosting, up next, can be viewed as an approach to weighted ensembling that strongly decorrelates ensemble members.

Boosting

Col boosting implementiamo il concetto di peso sui training set.

▶ Boosting

- ▶ Train classifiers sequentially, each time focusing on training examples that the previous ones got wrong.
- ▶ The shifting focus strongly decorrelates their predictions.
- ▶ To focus on specific examples, boosting uses a **weighted training set**

Weighted Training Set

questi rappresentano gli errori, di cui faccio la media

- ▶ The misclassification rate $\frac{1}{N} \sum_{n=1}^N \mathbb{I}[h(x^{(n)}) \neq t^{(n)}]$ weights each training set equally
- ▶ **Key Idea:** we can learn a classifier using different costs (aka weights) for examples.
 - ▶ classifier "tries harder" on examples with higher cost

- ▶ Change cost function:

L'idea è ritentare più volte sulle cose che sbaglio, finché non risultano ben classificate

$$\frac{1}{N} \sum_{n=1}^N \mathbb{I}[h(x^{(n)}) \neq t^{(n)}] \quad \text{becomes} \quad \sum_{n=1}^N w^{(n)} \mathbb{I}[h(x^{(n)}) \neq t^{(n)}]$$

- ▶ Usually require $w^{(n)} > 0$ and $\sum_{n=1}^N w^{(n)} = 1$

La somma dei pesi deve fare "1".

AdaBoost [Freund & Shapire, 1997]

- ▶ A meta-learning algorithm with great theoretical and empirical performance
- ▶ Turns a base learner, i.e., a weak learner/classifier into a high performance classifier
- ▶ Creates an ensemble of weak learner by repeatedly emphasizing misspredicted instances

Si parte da un classificatore "scarso", e si cerca di massimizzare le sue prestazioni.

- ▶ We can now describe the AdaBoost algorithm.
- ▶ Given a base classifier, the key steps of **AdaBoost** are:
 1. At each iteration, re-weight the training samples by assigning larger weights to samples (i.e., data points) that were classified incorrectly.
 2. Train a new base classifier based on the re-weighted samples.
 3. Add it to the ensemble of classifiers with an appropriate weight.
 4. Repeat the process many times.
- ▶ Requirements for base classifier:
 - ▶ Needs to minimize weighted error.
 - ▶ Ensemble may get very large, so base classifier should be fast. It turns out that any so-called **weak learner/classifier** suffices.

Individually, weak learners may have high bias (underfit). By making each classifier focus on previous mistakes, AdaBoost **reduces bias**.

Weak Learner/classifier

Un algoritmo "debole" è così definito se fa meglio di un classificatore random.
In caso di classificazione binaria, basterebbe che il weak learner faccia la giusta classificazione con probabilità del 51%

- ▶ (Informal) Weak learner is a learning algorithm that outputs a hypothesis (e.g., a classifier) that performs slightly better than chance, e.g., it predicts the correct label with probability 0.51 in binary label case.
- ▶ We are interested in weak learners that are computationally efficient.
 - ▶ Decision trees
 - ▶ Even simpler: **Decision Stump**: A decision tree with a single split

Un esempio di "weak learner" è DECISION STUMP, ovvero albero di decisione con un singolo split, è come se partizionassi in base ad un solo attributo.