**Capitolo: Riorganizzazione degli Appunti del 29/11**

**1. Cross Validation (CV)**

La **Cross Validation** (CV) è una tecnica fondamentale per valutare le prestazioni di un modello e scegliere i suoi iperparametri, riducendo il rischio di overfitting e bilanciando bias e varianza.

**1.1 Problema principale:**

* Difficoltà nel scegliere la complessità ottimale di un modello.
* La scelta dipende dal numero di parametri e iperparametri dell'algoritmo.

**1.2 Principio della Cross Validation:**

* Una parte dei dati di training viene riservata come **validation set** per valutare le prestazioni del modello.
* Permette di simulare il comportamento del modello su dati non osservati.

**1.3 Divisione dei dati:**

* **Training set:** Dati utilizzati per addestrare il modello.
* **Validation set:** Dati utilizzati per testare le predizioni.

**1.4 Iteratività:**

* La CV viene ripetuta alternando il ruolo dei dati nei set di training e validation.
* Il modello finale è quello con le migliori prestazioni medie su tutte le iterazioni.

**1.5 Collegamenti:**

* **Overfitting:** La CV aiuta a evitare l'adattamento eccessivo ai dati di training.
* **Bias-Variance Tradeoff:** Aiuta a scegliere un modello bilanciato.
* **Errori di predizione:** La CV valuta metriche come l'errore quadratico medio (MSE).

**2. k-Fold Cross Validation**

La **k-Fold Cross Validation** è una variante della CV che suddivide il dataset in k sottoinsiemi (fold).

**2.1 Procedura:**

1. Suddividere casualmente i dati in k fold.
2. Addestrare il modello su k-1 fold e validarlo sul rimanente.
3. Ripetere il processo per ciascun fold.
4. Calcolare le prestazioni medie su tutti i fold.

**2.2 Vantaggi:**

* **Riuso dei dati:** Ogni esempio viene utilizzato sia per il training che per la validazione.
* **Riduzione della varianza:** Le metriche medie su più partizioni forniscono stime più affidabili.

**3. Regolarizzazione**

La **regolarizzazione** controlla la complessità del modello aggiungendo un termine di penalizzazione alla funzione di errore.

**3.1 Formula:**

Errore totale = Errore sui dati + λ × Termine di penalizzazione

* **λ (lambda):** Parametro che controlla la forza della penalizzazione.
* Penalizza modelli troppo complessi per favorire la generalizzazione.

**3.2 Tipi di regolarizzazione:**

1. **Ridge Regression (L2):** Penalizza la somma dei quadrati dei parametri.
2. **Lasso Regression (L1):** Penalizza i valori assoluti dei parametri, favorendo la selezione delle feature.

**3.3 Vantaggi:**

* Riduce il rischio di overfitting.
* Migliora la generalizzazione su dati non osservati.

**4. Tecniche per Prevenire il Sovradattamento**

1. **Regolarizzazione:** Aggiunta di un termine di penalizzazione.
2. **Cross Validation:** Valutazione su diverse partizioni dei dati.
3. **Early Stopping:** Interruzione del training quando le prestazioni sul validation set peggiorano.
4. **Riduzione della complessità:** Limitare la profondità degli alberi o il numero di parametri.
5. **Aggiunta di dati o Data Augmentation:** Generare nuovi dati per migliorare la generalizzazione.
6. **Riduzione delle feature:** Selezionare solo le feature più rilevanti.

**5. Metodi Basati su Kernel**

I **metodi basati su kernel** proiettano i dati in uno spazio ad alta dimensione per risolvere problemi non lineari.

**5.1 Tipi di kernel:**

1. **Lineare:** Adatto a dati separabili linearmente.
2. **Polinomiale:** Cattura relazioni non lineari.
3. **RBF (Gaussian):** Misura la somiglianza tra punti.
4. **Sigmoid:** Simula l'attivazione delle reti neurali.

**5.2 Applicazioni:**

* **SVM (Support Vector Machines):**
  + Separano le classi massimizzando il margine.
  + L'iperpiano separatore è calcolato nello spazio trasformato dal kernel.

**6. Ensemble Learning**

L'**ensemble learning** combina le predizioni di più modelli per migliorare le prestazioni.

**6.1 Tecniche principali:**

1. **Bagging:**
   * Riduce la varianza combinando modelli addestrati su sottoinsiemi casuali.
   * Esempio: **Random Forest**.
2. **Boosting:**
   * Riduce il bias addestrando modelli sequenzialmente per correggere errori residui.
   * Esempio: **Gradient Boosting**.
3. **Stacking:**
   * Combina modelli diversi utilizzando un meta-modello.

**7. Random Forest**

La **Random Forest** è un esempio di bagging applicato agli alberi di decisione.

**Caratteristiche principali:**

* Ogni albero è addestrato su un sottoinsieme casuale di dati.
* Le predizioni finali sono ottenute tramite:
  + **Media:** Per problemi di regressione.
  + **Voto:** Per problemi di classificazione.

**8. Functional Gradient Boosting**

Il **Functional Gradient Boosting** addestra modelli sequenzialmente per correggere errori residui.

**Procedura:**

1. Ogni iterazione aggiorna la predizione aggiungendo un contributo proporzionale al gradiente della funzione di perdita.
2. Minimizza iterativamente la funzione di perdita.

**9. Confronto tra Tecniche di Ensemble**

| **Tecnica** | **Obiettivo** | **Addestramento** | **Modelli base** | **Finalizzazione** |
| --- | --- | --- | --- | --- |
| Bagging | Ridurre la varianza | Parallelamente | Indipendenti | Media o voto |
| Boosting | Ridurre il bias | Incrementale | Sequenziali | Combinazione ponderata |
| Stacking | Combinare modelli | Parallelamente | Indipendenti | Meta-modello |

**Quando usare:**

* **Bagging:** Dataset ampi, modelli instabili (es. alberi di decisione).
* **Boosting:** Problemi complessi, alto bias.
* **Stacking:** Necessità di combinare modelli diversi con risorse computazionali elevate.

**10. Considerazioni Finali**

Le tecniche di validazione e regolarizzazione, insieme ai metodi basati su kernel ed ensemble learning, sono strumenti fondamentali per creare modelli robusti e generalizzabili. La scelta della tecnica più adatta dipende dalla natura del problema, dalla complessità dei dati e dalle risorse disponibili.