Machine Learning

2025 (ML-2024)

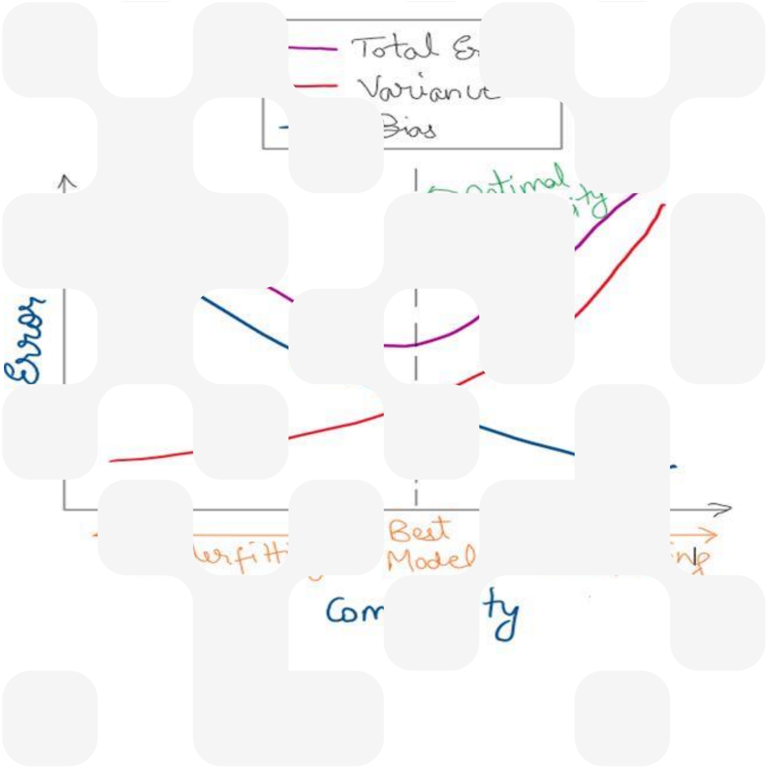
Lecture 05. Gradient Boosting and Suport Vector Machine

by Alexei Kornaev, Dr. Sc., Assoc. Prof., Robotics and CV, Innopolis University
Researcher at the RC for AI, National RC for Oncology n.a. NN Blokhin

Agenda

I. GRADIENT BOOSTING

II. SUPPORT VECTOR MACHINE (SVM)



Warm-up

Gradient boosting intuition

Consider a linear regression model $f = \left[x^{(i)}, \boldsymbol{\gamma}\right]$ parameterized with $\boldsymbol{\gamma}$ that maps each i-th input sample $x^{(i)}$ into the predictioin $F^{(i)}$ that should be close to the label $y^{(i)}$.

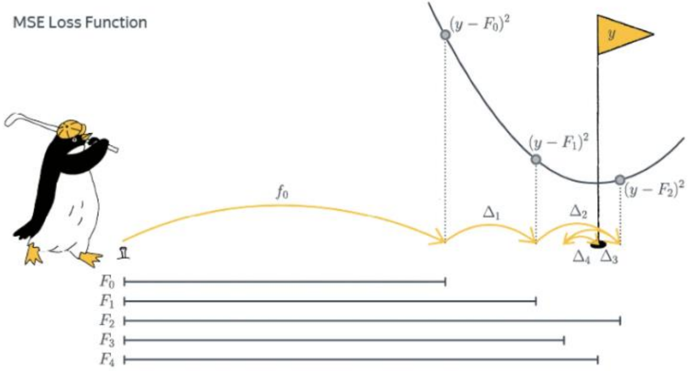
$$L(\boldsymbol{y}, \boldsymbol{\gamma}) = \frac{1}{2m} \sum_{i=1}^{m} \left(y^{(i)} - F^{(i)}\right)^2 \Rightarrow \min.$$

Consider a composition: $F_k = a_0 + a_1 + \cdots + a_k$.

Train $F_0 = a_0$ model (a simple decision tree) and check it's residual r_0.

Train $F_1 = a_0 + a_1$ model to reduce the residual r_0 and check the residual r_1.

…



For example, $m = 1$:

$F_0 = a_0, r_0 = y - F_0$.

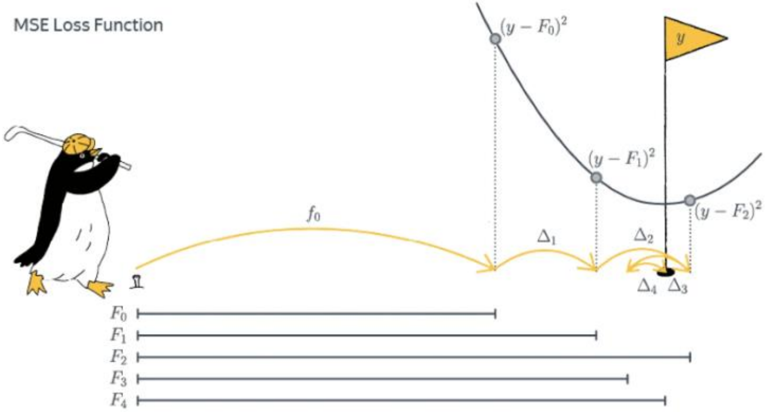
Suppose, $a_1 = -r_0$, then $F_1 = a_0 + a_1 = (y - r_0) - r_0 = y, r_1 = 0$.

Yandex Handbook on ML

Gradient boosting intuition

Consider a linear regression model $f = \left[x^{(i)}, \boldsymbol{\gamma}\right]$ parameterized with $\boldsymbol{\gamma}$ that maps each i-th input sample $x^{(i)}$ into the predictioin $F^{(i)}$ that should be close to the label $y^{(i)}$.

$$L(\boldsymbol{y}, \boldsymbol{\gamma}) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - F^{(i)}\right)^2 \Rightarrow \min.$$


MSE Loss Function

In a more general case, $m > 1$:

$$F_0 = \underset{\boldsymbol{\gamma}}{\operatorname{argmin}}\, L(\boldsymbol{y}, \boldsymbol{\gamma}),\ r_{i0} = y^{(i)} - F_0^{(i)}.$$

$$F_1 = F_0 + h_0 = F_0 + \underset{\boldsymbol{\gamma}}{\operatorname{argmin}}\, L(\boldsymbol{r}_0, \boldsymbol{\gamma}),\ r_{i1} = y^{(i)} - F_1^{(i)}.$$

…

$$F_k = F_{k-1} + h_k = F_{k-1} + \underset{\boldsymbol{\gamma}}{\operatorname{argmin}}\, L(\boldsymbol{r}_{k-1}, \boldsymbol{\gamma}),\ r_{ik} = y^{(i)} - F_k^{(i)}.$$

It can bee seen that the residual is the negative gradient of the loss : $-\left[\frac{\partial L}{\partial F_k}\right]_{F=F_k} = -\frac{1}{2m}\left[\frac{\partial}{\partial F_k}\left(\sum_{i=1}^{m}\left(y^{(i)} - F^{(i)}\right)^2\right)\right]_{F=F_k} = \boldsymbol{r}_k.$

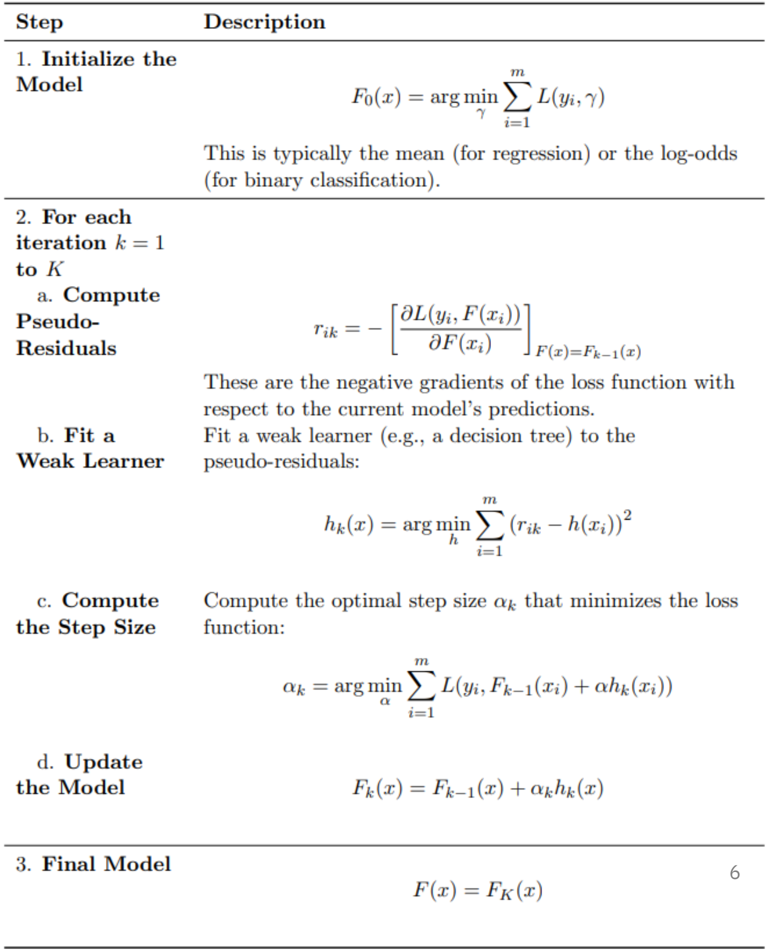
Yandex Handbook on ML

Gradient boosting algorithm

Explanation:

1. **Initialize the Model**: Start with a simple model (often a constant value).
2. **Iteratively Add Models**: At each iteration, add a new model that attempts to correct the errors made by the current ensemble. **Fit a Weak Learner**: Fit a weak learner (e.g., a decision tree) to the pseudo-residuals. **Compute the Step Size**: Compute the optimal step size that minimizes the loss function. **Update the Model**: Update the model by adding the new weak learner with the computed step size.
3. **Final Model**: The final model is the combination of all the weak learners.

DeepSeek speaks



Step	Description
1. **Initialize the Model**	$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{m} L(y_i, \gamma)$$ This is typically the mean (for regression) or the log-odds (for binary classification).
2. **For each iteration** $k = 1$ **to** K	
a. **Compute Pseudo-Residuals**	$$r_{ik} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{k-1}(x)}$$ These are the negative gradients of the loss function with respect to the current model's predictions.
b. **Fit a Weak Learner**	Fit a weak learner (e.g., a decision tree) to the pseudo-residuals: $$h_k(x) = \arg\min_{h} \sum_{i=1}^{m} (r_{ik} - h(x_i))^2$$
c. **Compute the Step Size**	Compute the optimal step size α_k that minimizes the loss function: $$\alpha_k = \arg\min_{\alpha} \sum_{i=1}^{m} L(y_i, F_{k-1}(x_i) + \alpha h_k(x_i))$$
d. **Update the Model**	$$F_k(x) = F_{k-1}(x) + \alpha_k h_k(x)$$
3. **Final Model**	$$F(x) = F_K(x)$$

Why Gradient Boosting rules?

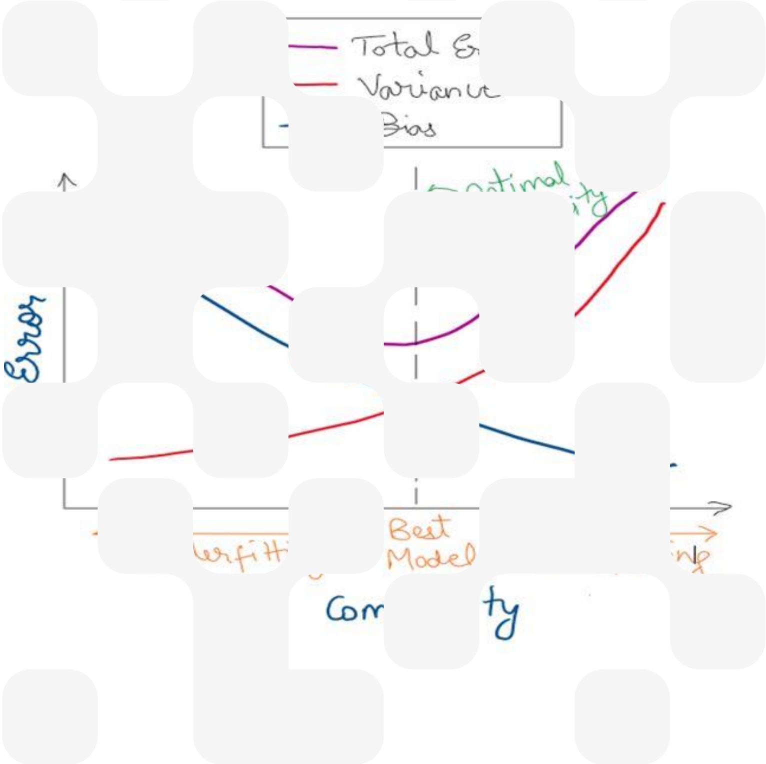
1. **High Predictive Accuracy.** GB is known for its ability to achieve high predictive accuracy. It does this by iteratively building an ensemble of weak learners (typically decision trees) and combining their predictions to create a strong learner. Each new tree is trained to correct the errors made by the previous trees, leading to a model that can capture complex patterns in the data.
2. **Flexibility.** GB can be applied to a wide range of loss functions, making it flexible for different types of problems. Any differentiable loss function can be used.
3. **Handles Various Data Types.** GB can handle numerical features, categorical features, missing values.
4. **Feature Importance.** GB provides a measure of feature importance, which helps in understanding which features are most influential in making predictions. This is useful for interpretability and feature selection.
5. **Regularization Techniques.** GB includes several regularization techniques to prevent overfitting. **Shrinkage**: A learning rate (or shrinkage factor) is applied to each tree's contribution reducing overfitting. **Subsampling**: Stochastic GB involves training each tree on a random subset of the data, which introduces randomness and reduces overfitting. **Tree Constraints**: Limiting the depth of the trees or the number of leaves can prevent the model from becoming too complex.
6. **Scalability**. Modern implementations of Gradient Boosting, such as XGBoost, LightGBM, and CatBoost, are highly optimized and can handle large datasets efficiently.
7. **Interpretability.** GB models can provide insights through feature importance and partial dependence plots.

DeepSeek speaks

Agenda

I. GRADIENT BOOSTING

II. SUPPORT VECTOR MACHINE (SVM)



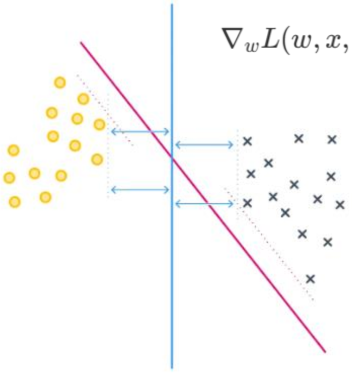
Support Vector Machine intuition

SVM is a linear method which aims to find the hyperplane that maximizes the *margin* between the classes. The margin is the distance between the hyperplane and the nearest data points from each class, known as support vectors. The goal is to find the hyperplane that provides the best generalization to unseen data.

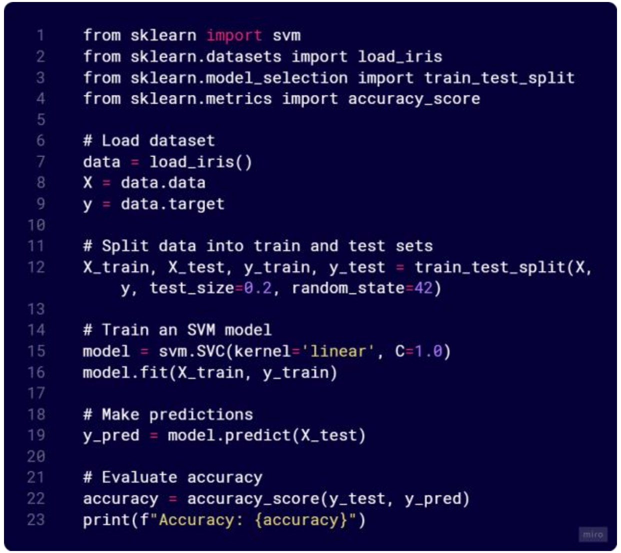
$$F(M) = \max(0, 1 - M)$$

$$L(w, x, y) = \lambda \| |w| \|_2^2 + \sum_i \max(0, 1 - y_i \langle w, x_i \rangle)$$

$$\nabla_w L(w, x, y) = 2\lambda w + \sum_i \begin{cases} 0, & 1 - y_i \langle w, x_i \rangle \leq 0 \\ -y_i x_i, & 1 - y_i \langle w, x_i \rangle > 0 \end{cases}$$



Идеальный классификатор
с маленьким отступом



```python
from sklearn import svm
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load dataset
data = load_iris()
X = data.data
y = data.target

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X,
    y, test_size=0.2, random_state=42)

# Train an SVM model
model = svm.SVC(kernel='linear', C=1.0)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

Yandex Handbook on ML

DeepSeek writes a code

SVM: kernel trick

SVM: advantages and disadvantages

- Effective in High-Dimensional Spaces: Works well even when the number of dimensions is greater than the number of samples.
- Memory Efficient: Uses only a subset of the training points (support vectors) in the decision function.
- Versatile: Can be used for both classification and regression tasks.

- Sensitive to Noise: A few misclassified points can significantly affect the hyperplane.
- Computationally Expensive: Training can be slow for large datasets.
- Requires Tuning: Hyperparameters like the kernel type, kernel parameters, and regularization parameter CC need to be tuned.

Thank you for your attention!

a.kornaev@innopolis.ru, @avkornaev

Notes