Supervised tasks

Università di Pisa

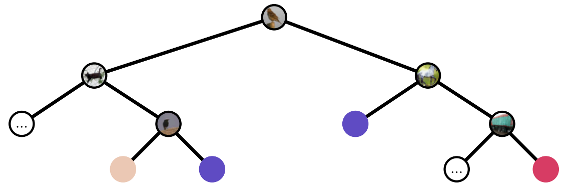
Four approaches to classification

Università di Pisa

By any means, not the only ones!

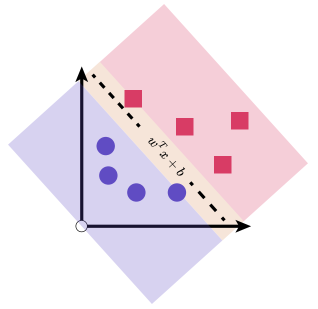
Trees

Can I learn a space partition that separates the data according to its label?



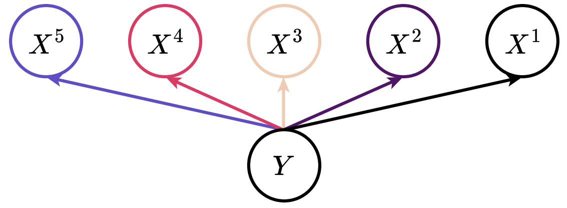
Kernel

Can I predict the label through its similarity with the data?



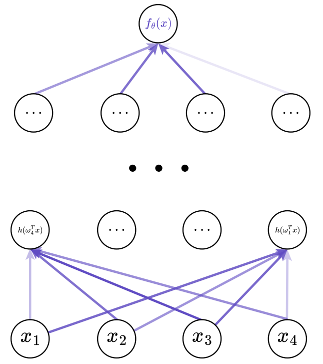
Bayesian

Can I estimate the probability of the label, condition on the data?



Neural

Can I predict through computational graphs?

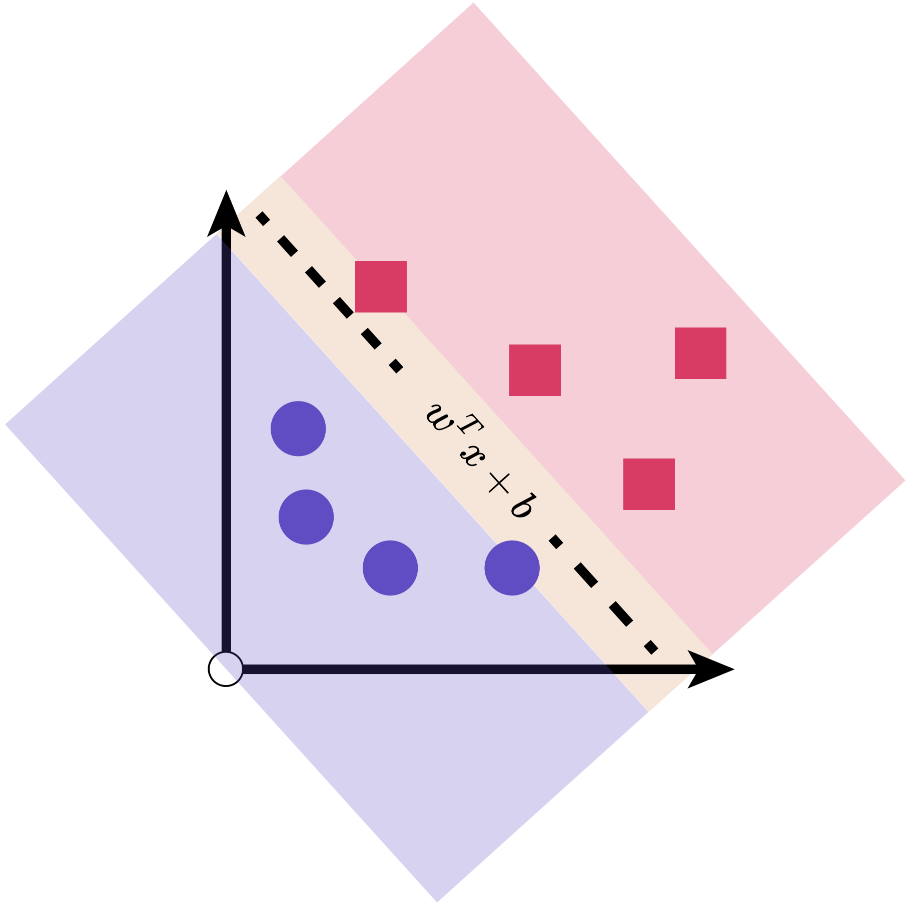


Kernel approaches

Beyond linearity

Kernel approaches: SVMs

The very same family of models we have seen for outlier detection, but moved from a one-class to a binary setting. The set of outliers $X^{\notin}$ is now given: it is one of the classes!



The *margin* (in beige) centered on the hyperplane separates the two classes: we wish to maximize this!

Support Vector Machines

Università di Pisa

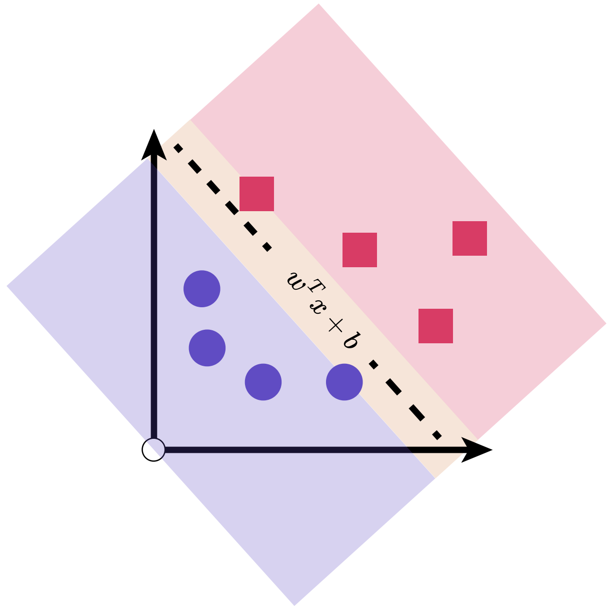
Let us define a hyperplane $w^T x + b = 0$ separating $X^{\in}$ and $X^{\notin}$, for which we have

$$\begin{cases} w^T x + b \geq +1 & \text{for } x \in X^{\in} \\ w^T x + b \leq -1 & \text{for } x \in X^{\notin} \end{cases}$$

Instances in the margin (called *support* instances/vectors) solve this for $w^T x + b = \pm 1$.

We can compact the two into

$$y^{\cdot}\left(w^T x + b\right) + 1 \geq 0$$



Instances and a separating hyperplane $w^T x + b = 0$. The two half-planes in red and blue are defined by $w^T x + b \geq +1$ and $w^T x + b \leq +1$, respectively.

Support Vector Machines

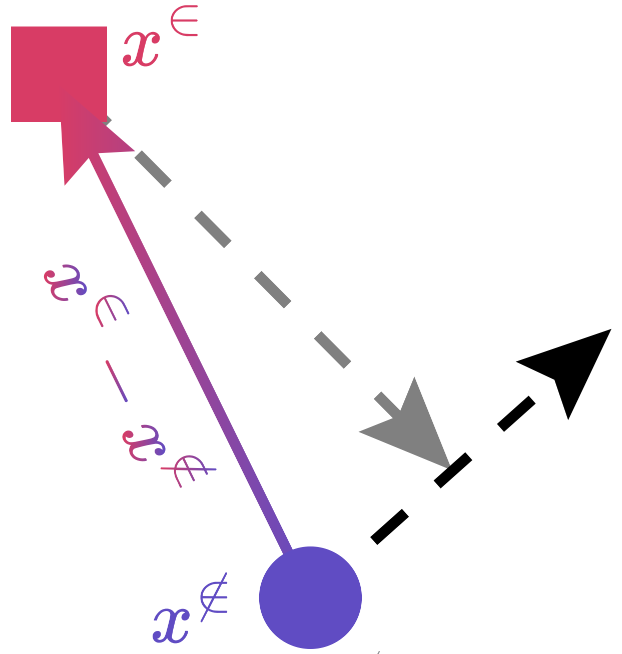
Geometrically, it is the projection of *margin* points onto a direction orthogonal to the margin:

$$(\hat{x}^{\in} - \hat{x}^{\notin}) \cdot \frac{w}{|| \, w \, ||},$$

which we can solve as

$$\frac{w \cdot \hat{x}^{\in} - w \cdot \hat{x}^{\notin}}{|| \, w \, ||} = \frac{(-b+1) - (-b-1)}{|| \, w \, ||} = $$

$$= \frac{2}{|| \, w \, ||}$$



Two instances $x^{\in}$ (red square), $x^{\notin}$ (blue circle), their difference $x^{\in} - x^{\notin}$ (in blue-to-red gradient), and a vector orthogonal to the margin (in black). The width of the margin is then the projection of the difference on such vector.

Support Vector Machines

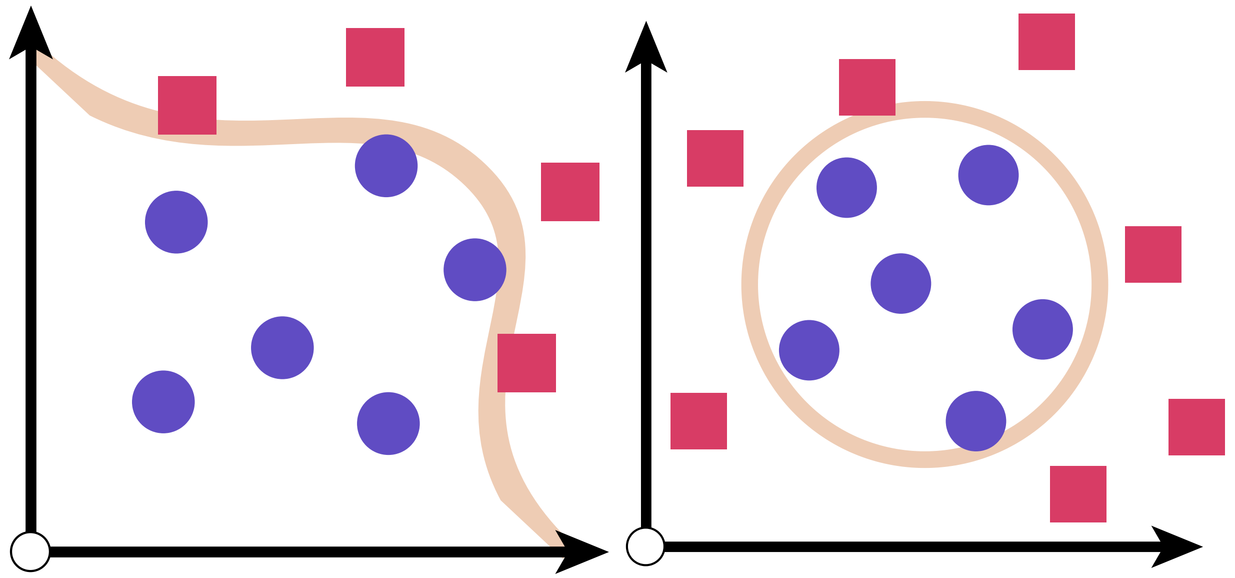
Solving analytically, we have that

1. The defining hyperplane w is a linear combination of the instances!
2. Some (hopefully many) instances have a zero coefficient λ_i, the others define (*support*) the hyperplane
3. The optimization takes the form $\Sigma_{i=1}^{n} \lambda_i - \frac{1}{2} \Sigma_{i=1}^{n} \Sigma_{j=1}^{n} \lambda_i \lambda_j y_i y_j \underbrace{x_i \cdot x_j}_{\textit{dot product!}}$

For a more in-depth derivation, see MIT OpencorseWare, 16. Learning: Support Vector Machines

Tackling linearity: the Kernel trick

We map data from the input space to Φ through a kernel function.



Kernel SVM: the margin can take nonlinear form.

Kernel	Linear	Radial basis	Polynomial
Formulation	$x^T y$	$exp(-\frac{\|\| x - y \|\|^2}{2\sigma^2})$	$(x^T y + c)^d$

Support Vector Machines: theoretical pointers

Capacity

SVMs have a straightforward interpretation of capacity:

- number of support vectors
- complexity of the kernel

Variance

Variance of SVMs is low when no kernel is used, since they offer a closed-form solution of the problem. It may increase with the complexity of the kernel used.

Support Vector Machines: regularization

UNIVERSITÀ DI PISA

Support regularization

SVMs offer a form of $L1$ regularization aimed at decreasing capacity by reducing the magnitude of support vectors.

The performance measure of a $L1$-regularized SVM is defined as:

$$\Sigma_{i=1}^{n}\lambda_i - \frac{1}{2}\Sigma_{i=1}^{n}\Sigma_{j=1}^{n}\lambda_i\lambda_j y_i y_j x_i \cdot x_j + \underbrace{\overbrace{\beta}^{weight} \mid\mid \lambda \mid\mid}_{regularization}$$

Support Vector Machines: regularization

Soft regularization
SVMs offers a form of regularization aimed at increasing variance by allowing for some incorrectly classified instances. That is, of the margin defined by the support vector may be violated from time to time.

The margin constraint thus become

$$y^{\cdot}(w^T x + b) + 1 \geq 0 \rightarrow y^{\cdot}(w^T x + b) + (1 - \underbrace{\xi}_{margin\ violation}) \geq 0,$$

where ξ plays the role of a *slack* variable, cutting some slack to the strictness of the margin.

Bayes

Back to the probability world.

Probabilities

Given a discrete random variable X describing a phenomenon, we have a probability $\Pr(X = x_i), x \in dom(X)$ that the event x_i occurs. For Pr to be a proper probability function it must hold that:

- **[Bounds]** For any event $x_i \in dom(X), \Pr(X = x_i) \in [0, 1]$
- **[Certainty].** The certain (x_T) and impossible event (x_F) have $\Pr(x_T) = 1, \Pr(x_F) = 0$
- **[Disjunction].** For any two events x_i, x_j, it holds $\Pr(x_i \vee x_j) = \Pr(x_i) + \Pr(x_j) - \Pr(x_i \wedge x_j)$

Multiple variables

Marginalization

When *jointly* considering multiple discrete variables with joint distribution $\Pr(X^i, X^j)$, we can reduce to one variable through *marginalization*, i.e.,

$$\Pr(X^i) = \sum_{x_j \in dom(X^j)} \Pr(X^i, X^j = x_j),$$

which, in simple terms, marginalizes over all possible events of X^j, thus factoring it out of the distribution.

Multiple variables

UNIVERSITÀ DI PISA

Conditioning

When *jointly* considering multiple discrete variables with joint distribution $\Pr(X^i, X^j)$, we can estimate the probability $\Pr(X^i \mid X^j)$ of X^i given X^j. This is a special case of joint distribution: of all the joint events wherein an event x_j occurs, only the ones in which also x_i occurs:

$$\Pr(X^i \mid X^j) = \frac{\Pr(X^i, X^j)}{\Pr(X_j)}, \text{ and symmetrically, } \Pr(X^j \mid X^i) = \frac{\Pr(X^i, X^j)}{\Pr(X_i)}.$$

Note: $\Pr(X^i, X^j) = \Pr(X^j, X^i)$.

From conditioning to Bayes

Conditioning allows us to define a powerful rule for probability, relating two variables and their conditioning. Back to the conditioning rule, let us multiply both sides by of $\Pr(X^j \mid X^i)$ by $\Pr(X^i)$ to have $\Pr(X^i, X^j) = \Pr(X^j \mid X^i) \Pr(X^i)$. By plugging this back in $\Pr(X^i \mid X^j)$ we have

$$\Pr(X^i \mid X^j) = \frac{\Pr(X^i, X^j)}{\Pr(X_j)} = \frac{\Pr(X^j \mid X^i) \Pr(X^i)}{\Pr(X_j)}. \quad [\textit{Bayes rule}]$$

The **Bayes rule** relates the conditional probabilities of two random variables. It states that, whenever we have a conditioning probability in one direction, e.g., $X^i \mid X^j$, we can flip the direction.

Note: $\Pr(X^i, X^j) = \Pr(X^j, X^i)$.

Bayes and generative models

The Bayes rule can be interpreted as a generative model: given a random variable θ defining a generative process, and another variable X defining some observations of the process, it allows us to define the process from the observations. Thus, it is often described as:

$$\Pr(X \mid \theta) = \frac{\Pr(\theta \mid X) \Pr(X)}{\Pr(\theta)},$$

where θ parameterizes a generative model of X, and X is an observation of a phenomenon. Here, we can identify three components:

- *prior* the prior belief $\Pr(\theta)$ about the generative model;
- *likelihood* the likelihood $\Pr(X \mid \theta)$ of the data given the model;
- *posterior* the posterior probability $\Pr(\theta \mid X)$ of the model, given the data.

Naive Bayes

UNIVERSITÀ DI PISA

By framing labels as values, we can simply look to learn a probabilistic model $\Pr(Y = y \mid X^1 = x_1, \ldots, X^m = x_m)$ for an instance $x = [x_1, \ldots, x_m]$ and label y. Computing over all the labels, we have probabilities for all possible labels, and can pick the best one. A Naive Bayes model implements a function

$$f(x) = arg \max_y \Pr(Y = y | X^1 = x_1, \ldots, X^m = x_m).$$

Note: When dealing with probabilities, convention is to use capital letters to indicate random variables, and lowercase letters to indicate values. Here, X^i does not indicate the feature matrix, rather the random variable associated to the i-th feature . Same holds for Y.

Bayes and factorization

UNIVERSITÀ DI PISA

Chain rule.

For a joint distribution $\Pr(X^1, \ldots, X^n)$ it holds

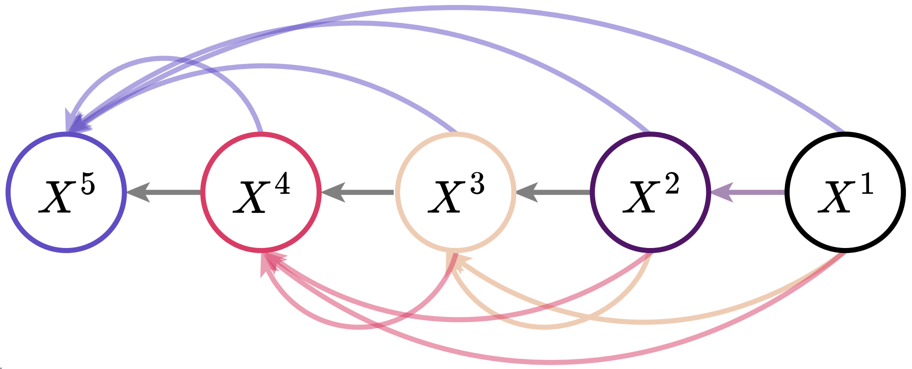
$$\Pr(X^1, \ldots, X^n) = \prod_{i=n}^{1} \Pr(X^i \mid X^{i-1}, \ldots X^1)$$

The chain rule states that we can model the relationship between any set of variables, but at a huge cost: a chain of conditional dependencies.

Bayes and factorization

Università di Pisa

The chain rule states that we can model the relationship between any set of variables, but at a huge cost: a chain of conditional dependencies.



Factorization of $\Pr(X^5, X^4, X^3, X^2, X^1)$ according to the chain rule: $\Pr(X^5, X^4, X^3, X^2, X^1) = \Pr(X^5 \mid X^4, X^3, X^2, X^1)\Pr(X^4 \mid X^3, X^2, X^1)\Pr(X^3 \mid X^2, X^1)\Pr(X^2 \mid X^1)\Pr(X^1)$. Every variable is dependent on a set of other variables.

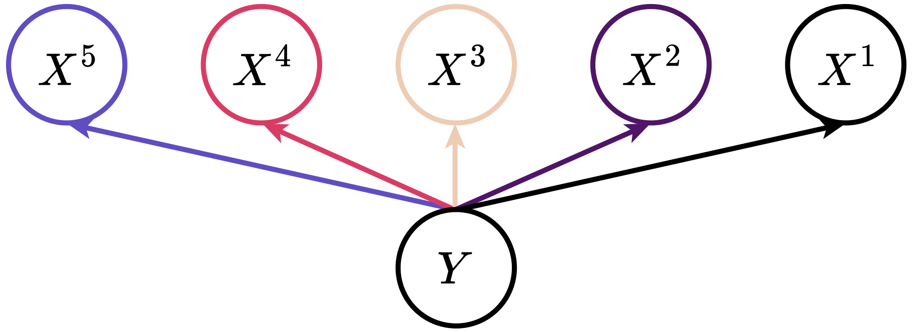
Independence in Naive Bayes

Università di Pisa

To tackle chain factorization, Naive Bayes *assumes independence* of the variables

Variable independence.

For two variables X^i, X^j, we say that X^i is independent of X^j if $\Pr(X^i) = \Pr(X^i \mid X^j)$.



A factorization of $\Pr(Y \mid X^1, \ldots, X^n)$ with independence assumption: the label Y is conditioned on the variables, for which no dependence is assumed.

Naive Bayes

A Naive Bayes model is of the form

$$f(x) = arg \max_{y \in dom(Y)} \prod_{i=1}^{n} \Pr(Y = y \mid X_i).$$

Leveraging Bayes' theorem, this amounts to $arg \max_{y \in dom(Y)} \prod_{i=1}^{n} \Pr(X_i \mid Y = y)$.

Which begs the question: how to estimate $\Pr(Y = y \mid X)$?

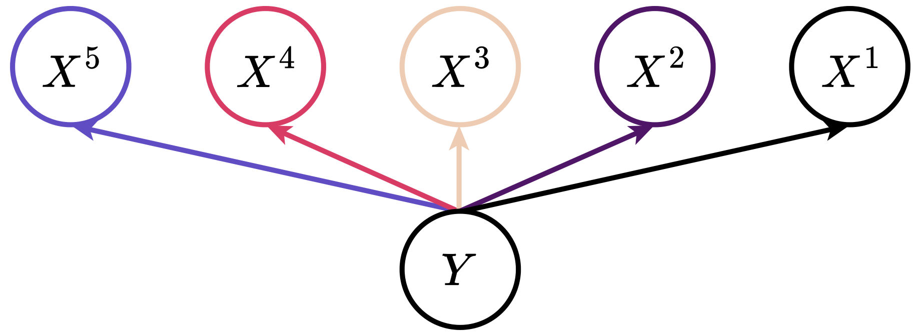
- Empirical count
- Density of a probability density function of a distribution fit on $Y \times X_i$

Naive Bayes

UNIVERSITÀ DI PISA

- Simple definition
- Extremely fast to compute

- Assumption of variable independence
- Limited by the distributional approach



A graphical view of the a Naive Bayes model: the label is a function of independent variables.

Ensemble methods

Tackling variance, once more

Mitigating variance... or not

As highlighted by the bias-variance decomposition, learning algorithms can incur in a variance in terms of generalization. We have seen two strategies to mitigate this:

- Cross validation: reduce variance by increasing samples
- Regularization: reduce variance by reducing capacity

There exists a third way, somewhat related, approach: leverage variance.

Bagging

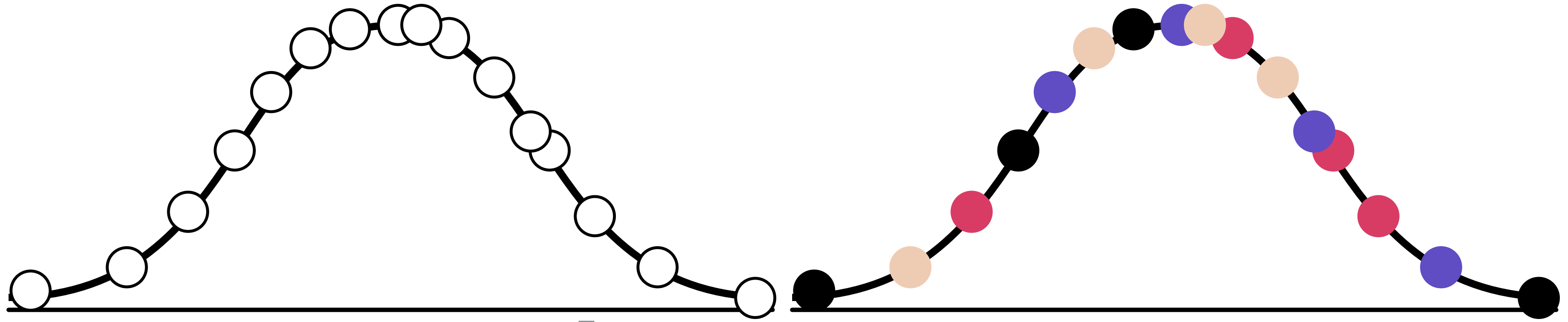
Bagging leverages learning algorithms with **moderate-to-high variance**: rather than learn a single model f_θ, it learns a set of models $f_1, \dots, f_T$, and combines them into one. A bagging model is of the form

$$f_\theta(x) = \sum_{i=1}^{T} \eta f_i(x), \qquad \eta \in \mathbb{R}.$$

Following the bias-variance, bagging algorithms aim to learn a model on separate samples of the distribution $\Pr^E$.

Assumption: models retain a low-enough bias to be accurate on their respective bags.

Bagging


On the left, data, sampled from the distribution $\Pr^E$. On the right, a visualization of bagging: samples are split in several bags (here, color-coded). Each bag is used to learn a different model, which will then be part of an ensemble.

Random Forests

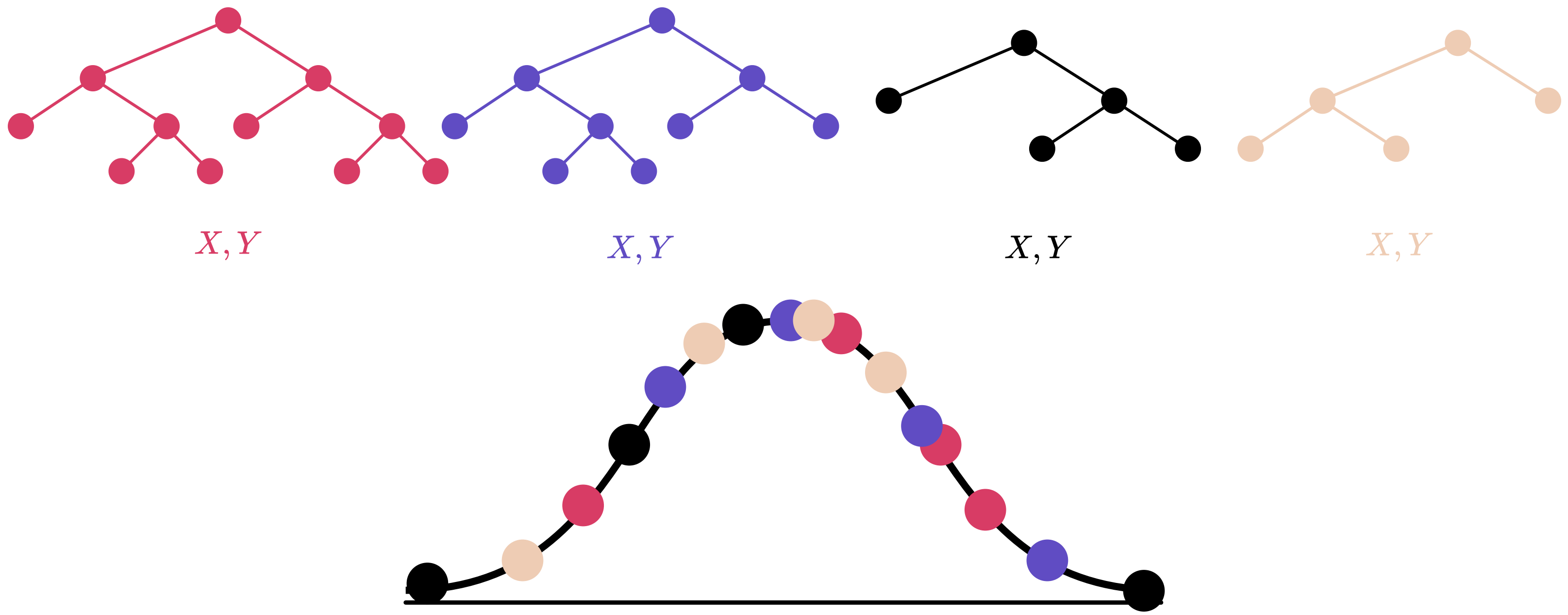
Decision Trees show a moderate variance, and thus are a perfect candidate. Random Forests sample a set of *bags* by sampling:

- Random instances, through stratification
- Random features within the bag

Thus creating learning sets heterogeneous in both instances and features. Bags are sampled with replacement!

Bagging

UNIVERSITÀ DI PISA



On the bottom, the bagged data, each bag indicated by a different color. On top, a set of learned trees, color-coded as the bag they have been learned from: the red tree has been learned on the red bag, and so forth. Note: Random Forest samples bags with replacement: an instance may be part of 0 up to K bags.

Random Forests

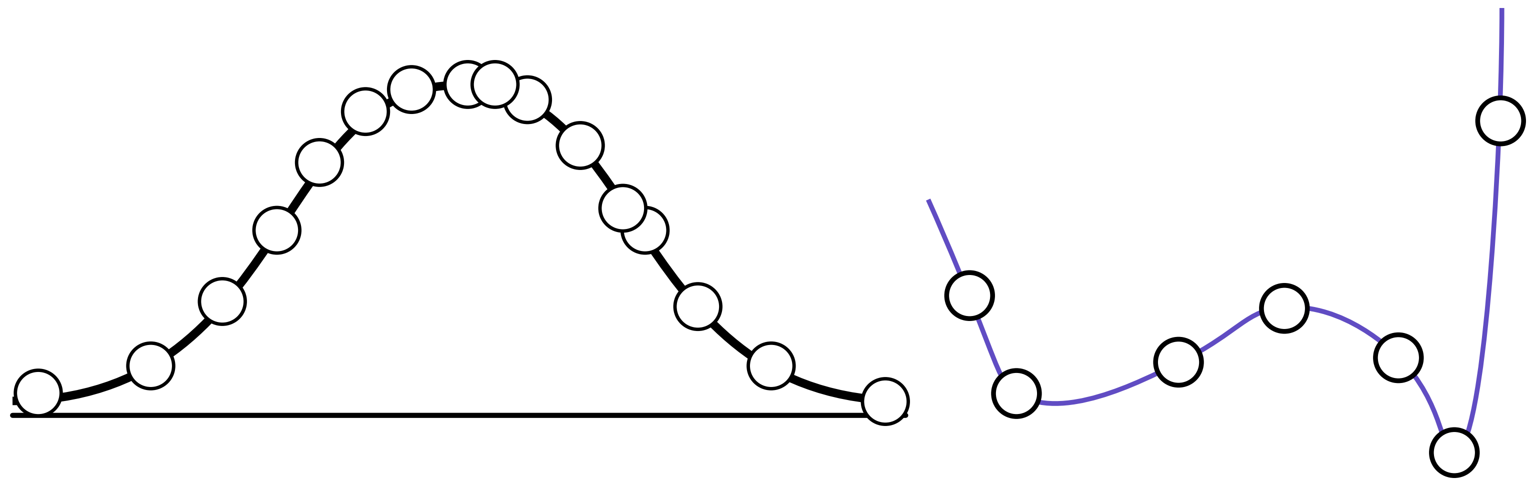
Università di Pisa

- Simple definition with high variance models
- Interpretable-ish results
- Fast learning

- Weak to high degrees of covariance
- Sampling *with replacement*: risk of high correlation
- Random bagging

Bagging... what?

How would *you* bag?



From bagging... to Boosting

With bagging models, we assume locality in the data distribution, and thus learn different models on different bags. Up to re-sampling, each instance is uniquely predicted by one model. If models have high-enough bias, then we are not gaining much from bagging data.

The *hard* bags of bagging are limited by the bias of each model: if none of them are good enough, their number does not matter, and dividing the task yields no advantage. In other words, **in a crowd of weak learners, there's no wisdom to be had**.

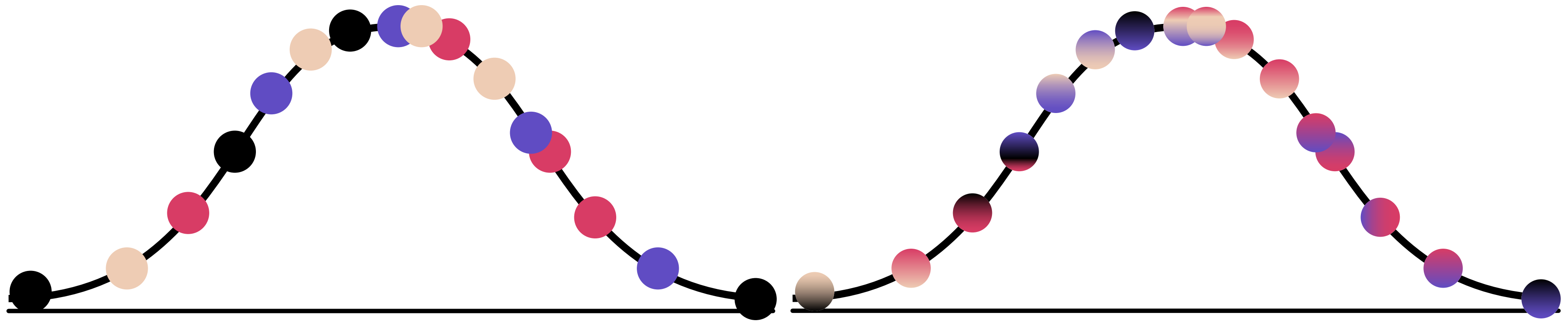
Why don't we bag the task instead?

Robert E. Schapire, The Strength of Weak Learnability.

Boosting

Università di Pisa

Boosting creates *fuzzy soft* bags, wherein instances are jointly predicted by all models, thus **decomposing the task**, rather than the data!



Bagging VS Boosting. In bagging, each bag is used to learn a model. In boosting, bags are *fuzzy*, and the task is decomposed so that each model predicts a subset of it.

Boosting: a general formulation

Like bagging, we have a model of the form

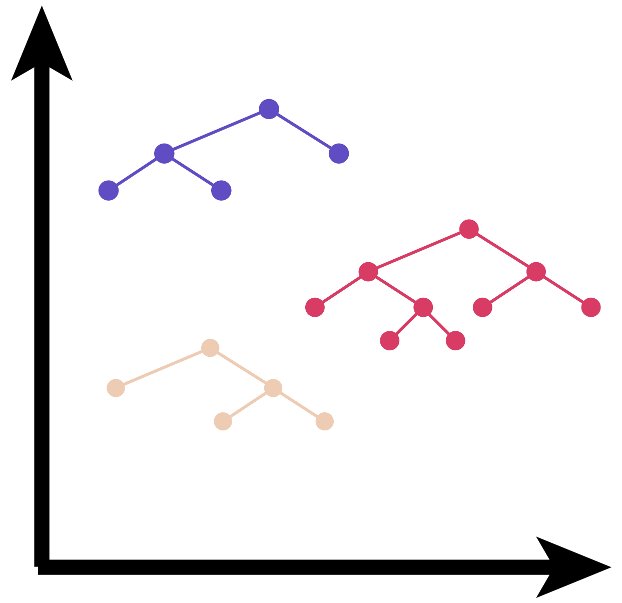
$$f^T(x) = \sum_{i=1}^{T} \eta_i f_i(x), \qquad \eta_i \in \mathbb{R},$$

which is learned iteratively: first f^1, then f^2, etc. Sums and multiplications by scalar: looks like a job for linear algebra!

Boosting and the functional (linear) space

Since we are operating with finite datasets, we can interpret functions as vectors $\vec{f}_i = [f_i(x_1), \ldots, f_i(x_n)]$ in a vector space. Thus, we can interpret operations on functions as operations on vectors:

- $f + g = \vec{f} + \vec{g}$
- $\eta f = \eta \vec{f}$
- $f \cdot g = \vec{f} \cdot \vec{g}$



A model space of decision trees: in this space, summing allows us to generate novel decision trees, while inner products allows us to compute their alignment.

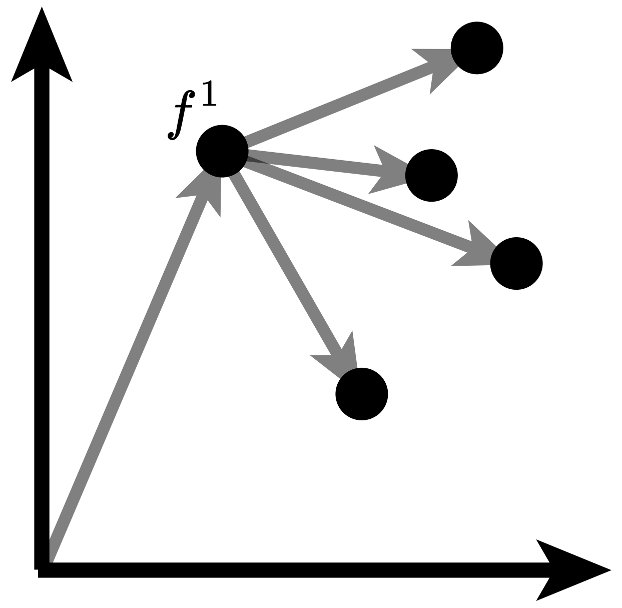
Boosting: exploring the functional space

Università di Pisa

Boosting models are learned iteratively, each additional model f^{t+1} looking to reduce a predefined loss

$$L^{t+1} = l(Y_i, f^t(x) + \eta_{t+1} f_{t+1}).$$

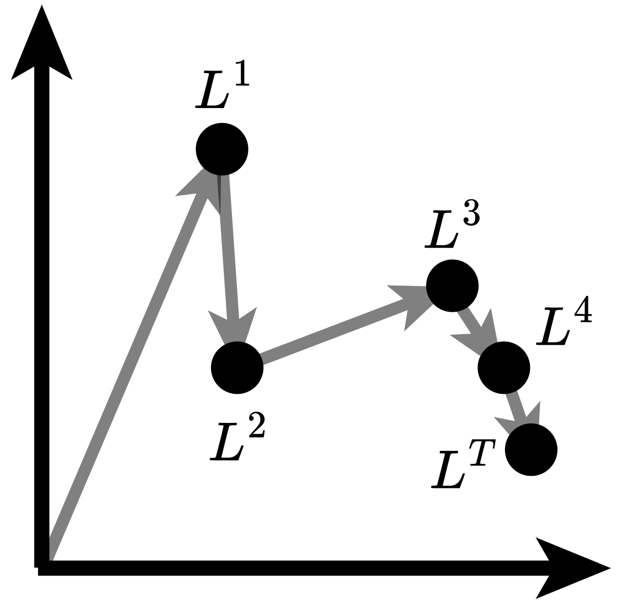
The additional function f_{t+1} is one of possibly many (possibly infinite) directions we can take in functional space. We want to pick the direction minimizing loss! What direction minimizes loss?



f_1 and some (of possibly infinite) directions f_2.

Exploring optimization: the gradient

The one **minimizing** L^{t+1}! To know such direction, we rely on calculus, i.e., we compute the gradient of L^{t+1} w.r.t. f^t : $\nabla_{f^T} L^{t+1}$. The negated gradient is the direction of minimization.



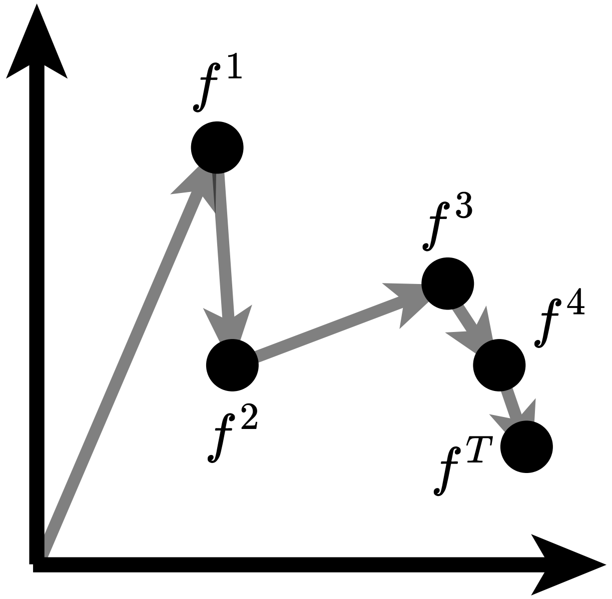
Boosting and iterative optimization: we construct the model one loss improvement at a time, exploring the loss space.

Exploring optimization: the gradient

Università di Pisa

Note: the space of models $\mathcal{F}$ to which f_{t+1} belongs may not be continuous, thus we can't necessarily directly define f_{t+1}, so we choose the closest match, i.e., a f_{t+1} maximizing its similarity:

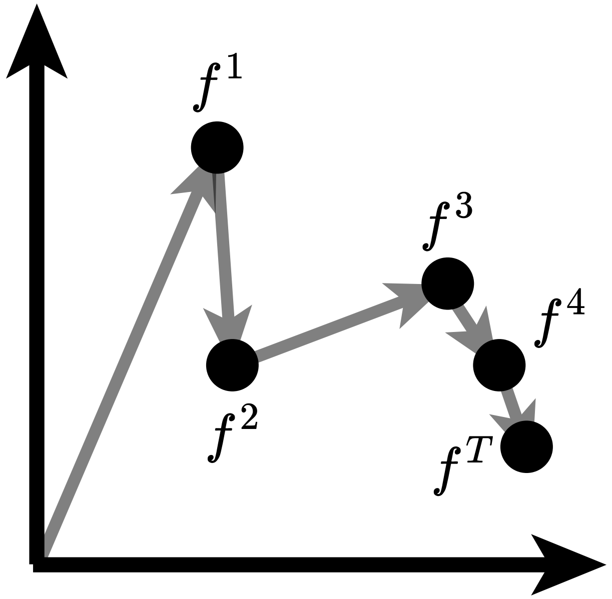
$$f_{t+1} = arg \max_{f \in \mathcal{F}} f \cdot -\nabla_{f^T} L^{t+1}.$$



Boosting and iterative optimization: each model is added to the boosting model, thus moving across the model space.

The gradient in boosting: residuals

In boosting models, gradient of the loss gives us a search direction, but also a *residual*: practically, since each model is additive, gradients define directions as much as *residuals* we want to optimize. Thus, we fit models on datasets $(X, -\nabla L^t)$.



Boosting and iterative optimization: each model is added to the boosting model, thus moving across the model space.

Boosting: a most generic algorithm

Knowing how to learn a model, we can inductively define any boosting algorithm:

1. Initialize f_1
2. Find optimal direction $-\nabla_{f_t} L^t$
3. Find admissible direction f_t
4. Find learning step η_t
5. Learn f^t
6. Go to 2.

Boosting: a most generic algorithm

Knowing how to learn a model, we can inductively define any boosting algorithm:

1. Initialize f_1
2. Find optimal direction $-\nabla_{f_t} L^t$
3. Find admissible direction f_t
4. Find learning step η_t
5. Learn f^t
6. Go to 2.

Notes

- The function space could be anything: the space of Decision Trees (Gradient-Boosted Trees), of Logistic Regression (Logitboost), etc.
- Regularization is usually applied to f_t: allows to have weak learners, which helps with decreasing overfit

Boosting and its many flavors

- **Adaboost.** Uses an exponential loss, adapts η with a closed form search
- **Gradient-Boosted Trees.** Leverage Decision Trees as models
- **XGBoost.** Leverages trees, and uses a more robust loss approximation through the Hessian, rather than the gradient

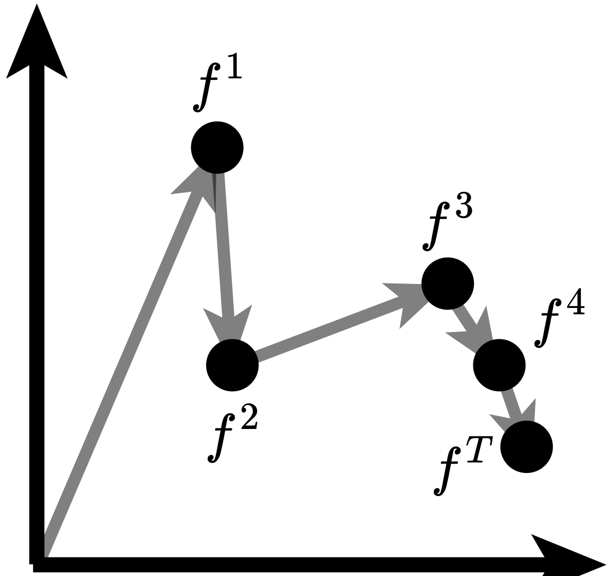
Boosting and bagging: regularization

Regularization is performed directly on the weak learners (to make sure we keep variance high), but can be applied post-hoc too through pruning!

Boosting

- Model-agnostic
- Given some relatively likely theoretical assumptions, has extremely low bias
- Unlikely to overfit
- Computationally quick

- Largely uninterpretable



Boosting: we iteratively refine the model f^T by exploring the function and loss space.

References

	Source
Bayes	David Barber. Bayesian Reasoning and Machine Learning. Sections 1.1-1.3
Support Vector Machines	Support Vector Machines
Bias, Variance	Deep Learning. I. Goodfellow, Y. Bengio, A. Courville. Sections 5.4
Boosting, Bagging	Deep Learning. I. Goodfellow, Y. Bengio, A. Courville. Sections 7.11
Random Forests	Random Forests