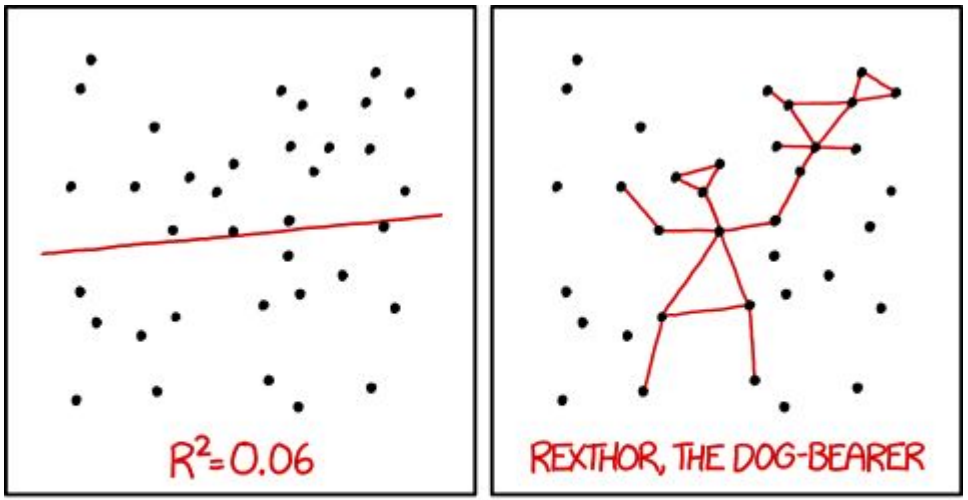
Advanced Regression

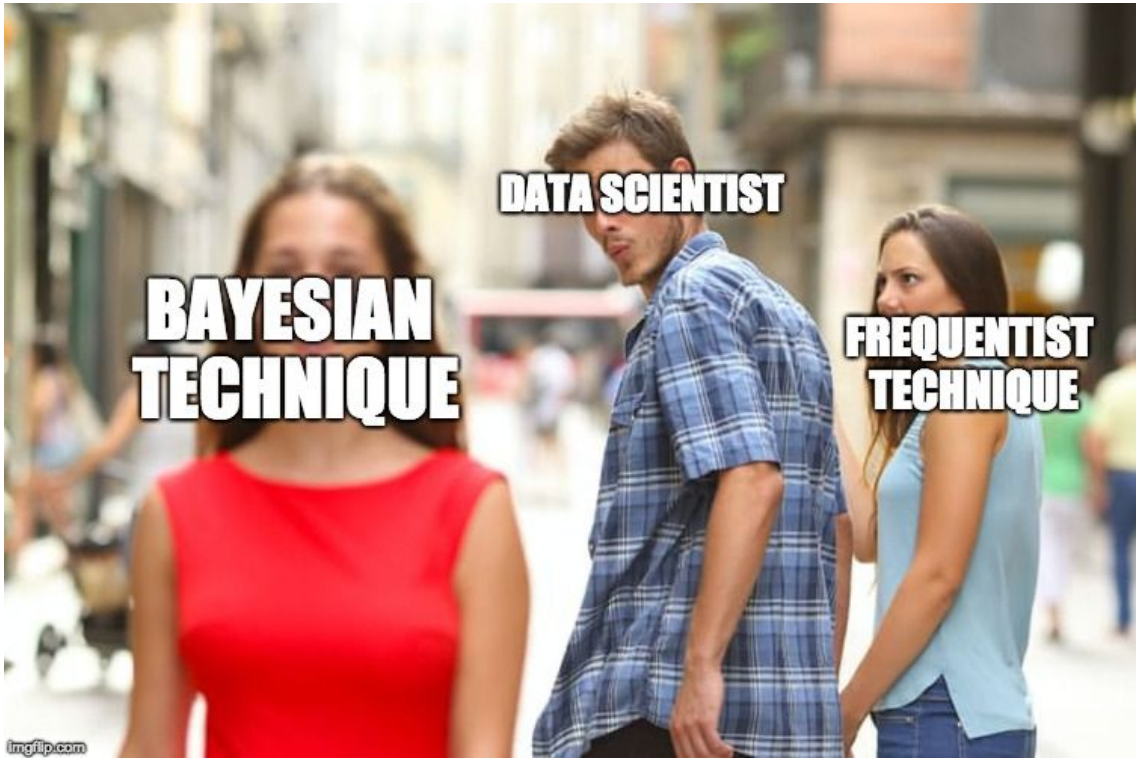
Probabilistic interpretation of LR. Classification algorithms for regression

Contents

1. Bayesian explanation of regularized regression
2. Classification algorithms for regression



Bayesian regression



Classical probabilistic view on linear regression

Consider we have n points **Y** drawn i.i.d. from the normal distribution. The probability of those points being drawn defines the likelihood function which is just a multiplication of their densities in every points.

$$\mathcal{L}(\mu|y) = \prod_{i=1}^{n} P_Y(y_i|\mu,\sigma^2) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_i-\mu)^2}{2\sigma^2}} \quad (1)$$

A good estimate of mean maximizes that likelihood

$$\begin{aligned}
\hat{\mu} = \arg\max_{\mu} \mathcal{L}(\mu|y) &= \arg\max_{\mu} \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_i-\mu)^2}{2\sigma^2}} \\
&= \arg\max_{\mu} \log\left(\prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_i-\mu)^2}{2\sigma^2}}\right) \\
&= \arg\max_{\mu} \sum_{i=1}^{n} \log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) + \log\left(e^{-\frac{(y_i-\mu)^2}{2\sigma^2}}\right) \\
&= \arg\max_{\mu} \sum_{i=1}^{n} \log\left(e^{-\frac{(y_i-\mu)^2}{2\sigma^2}}\right) \\
&= \arg\max_{\mu} \sum_{i=1}^{n} -\frac{(y_i-\mu)^2}{2\sigma^2} \\
&= \arg\min_{\mu} \sum_{i=1}^{n} (y_i-\mu)^2 \quad (2)
\end{aligned}$$

Classical probabilistic view on linear regression

- Assume our mean is a function of predictors X

$$E(y|\mathbf{x}) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

- Thus our target is distributed according to

$$y \sim N(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p, \sigma^2)$$

- Estimating regression parameters given (2)

$$\beta = \arg\min_{\beta} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p))^2$$

$$= \arg\min_{\beta} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

A probabilistic interpretation of regularization

- Using a Bayes theorem we can estimate the **probability distribution** of the parameters θ given the data we observe **Y**

$$P(\theta|y) = \frac{P(y|\theta)P(\theta)}{P(y)}$$

$$\text{posterior} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}$$

- That gives us an opportunity to set the a prior distribution of model parameters
- Compare that with the classical method where we instead try to find the best parameters to maximize the **likelihood of parameters** given the data

A probabilistic interpretation of regularization

- We maximize the posterior probability estimate using Bayes theorem (Maximum A Posteriori estimation)

$$\begin{aligned}\hat{\theta}_{\mathbf{MAP}} &= \arg\max_{\theta} P(\theta|y) \\ &= \arg\max_{\theta} \frac{P(y|\theta)P(\theta)}{P(y)} \\ &= \arg\max_{\theta} P(y|\theta)P(\theta) \\ &= \arg\max_{\theta} \log(P(y|\theta)P(\theta)) \\ &= \arg\max_{\theta} \log P(y|\theta) + \log P(\theta)\end{aligned}$$

- Compare that to MLE estimate

$$\hat{\theta}_{\mathbf{MLE}} = \arg\max_{\theta} \log P(y|\theta)$$

A probabilistic interpretation of L2 regularization

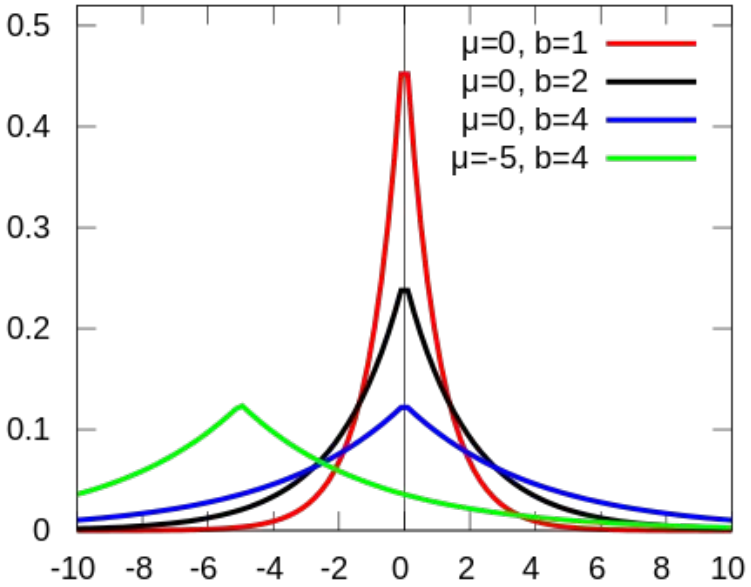
- Assume our model parameters zero-mean normally distributed with ***t^2*** variance (prior knowledge)

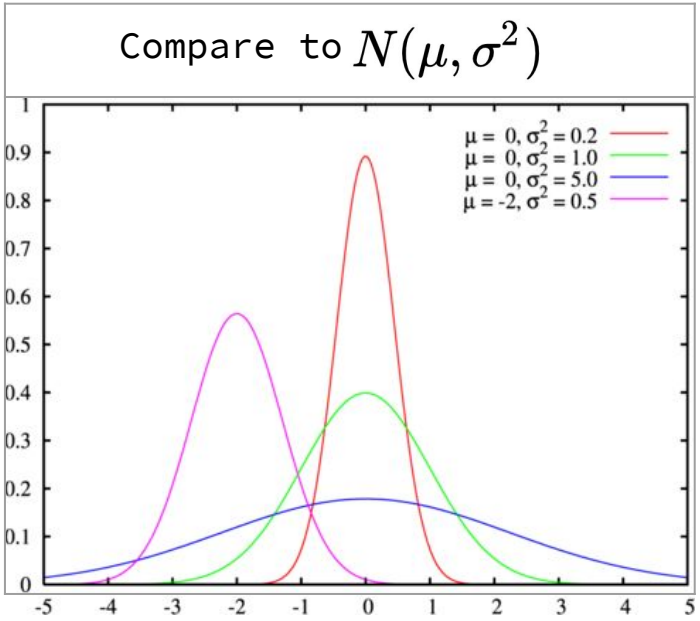
$$\begin{aligned}
&\arg\max_{\beta}\Big[\log\prod_{i=1}^{n}\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(y_i-(\beta_0+\beta_1 x_{i,1}+\ldots+\beta_p x_{i,p}))^2}{2\sigma^2}}+\log\prod_{j=0}^{p}\frac{1}{\tau\sqrt{2\pi}}e^{-\frac{\beta_j^2}{2\tau^2}}\Big]\\
&=\arg\max_{\beta}\Big[-\sum_{i=1}^{n}\frac{(y_i-(\beta_0+\beta_1 x_{i,1}+\ldots+\beta_p x_{i,p}))^2}{2\sigma^2}-\sum_{j=0}^{p}\frac{\beta_j^2}{2\tau^2}\Big]\\
&=\arg\min_{\beta}\frac{1}{2\sigma^2}\Big[\sum_{i=1}^{n}(y_i-(\beta_0+\beta_1 x_{i,1}+\ldots+\beta_p x_{i,p}))^2+\frac{\sigma^2}{\tau^2}\sum_{j=0}^{p}\beta_j^2\Big]\\
&=\arg\min_{\beta}\Big[\sum_{i=1}^{n}(y_i-(\beta_0+\beta_1 x_{i,1}+\ldots+\beta_p x_{i,p}))^2+\lambda\sum_{j=0}^{p}\beta_j^2\Big]
\end{aligned}$$

- Small variance ***t^2*** (large **λ**) leads to reduction of coefficients. If we have a large variance (small **λ**) the coefficients are not affected much.

A probabilistic interpretation of l1 regularization

- Laplace distribution with mean **μ** and diversity **b** defined by a probability density function $Laplace(\mu, b) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}}$





A probabilistic interpretation of l1 regularization

- Assume our model parameters zero-mean Laplace-distributed with diversity **b**

$$\begin{aligned}
&\arg\max_{\beta}\Big[\log\prod_{i=1}^{n}\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(y_i-(\beta_0+\beta_1 x_{i,1}+\ldots+\beta_p x_{i,p}))^2}{2\sigma^2}}+\log\prod_{j=0}^{p}\frac{1}{2b}e^{-\frac{|\beta_j|}{b}}\Big]\\
&=\arg\max_{\beta}\Big[-\sum_{i=1}^{n}\frac{(y_i-(\beta_0+\beta_1 x_{i,1}+\ldots+\beta_p x_{i,p}))^2}{2\sigma^2}-\sum_{j=0}^{p}\frac{|\beta_j|}{b}\Big]\\
&=\arg\min_{\beta}\frac{1}{2\sigma^2}\Big[\sum_{i=1}^{n}(y_i-(\beta_0+\beta_1 x_{i,1}+\ldots+\beta_p x_{i,p}))^2+\frac{2\sigma^2}{b}\sum_{j=0}^{p}|\beta_j|\Big]\\
&=\arg\min_{\beta}\Big[\sum_{i=1}^{n}(y_i-(\beta_0+\beta_1 x_{i,1}+\ldots+\beta_p x_{i,p}))^2+\lambda\sum_{j=0}^{p}|\beta_j|\Big]
\end{aligned}$$

- L1 regularization promotes sparsity in comparison with “just reducing the coefficients” in L2. That makes sense if you look at Laplacean density where there is a sharp increase in x=mean.

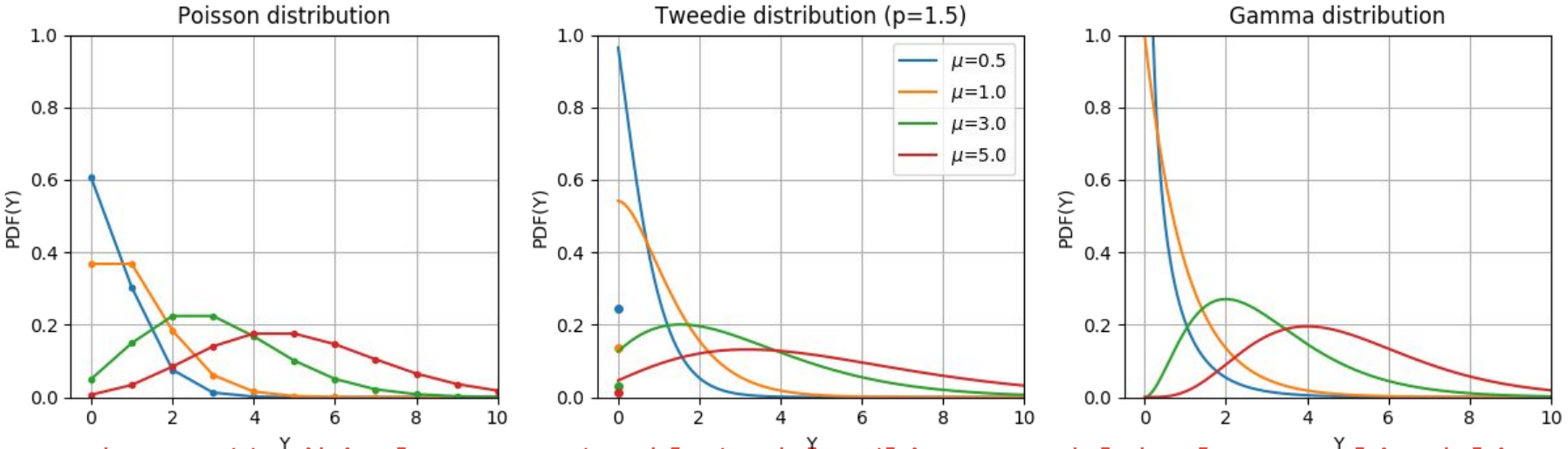
Why L1 zero out coefficients whereas L2 does not?

Why L1 zero out coefficients whereas L2 does not?

- Laplace distribution (sharp in x=mean) vs Normal distribution (smooth)
- Intuitive understanding through Gradient Descent https://developers.google.com/machine-learning/crash-course/regularization-for-sparsity/l1-regularization
- Intuitive understanding through visualization in 2d case https://explained.ai/regularization/L1vsL2.html

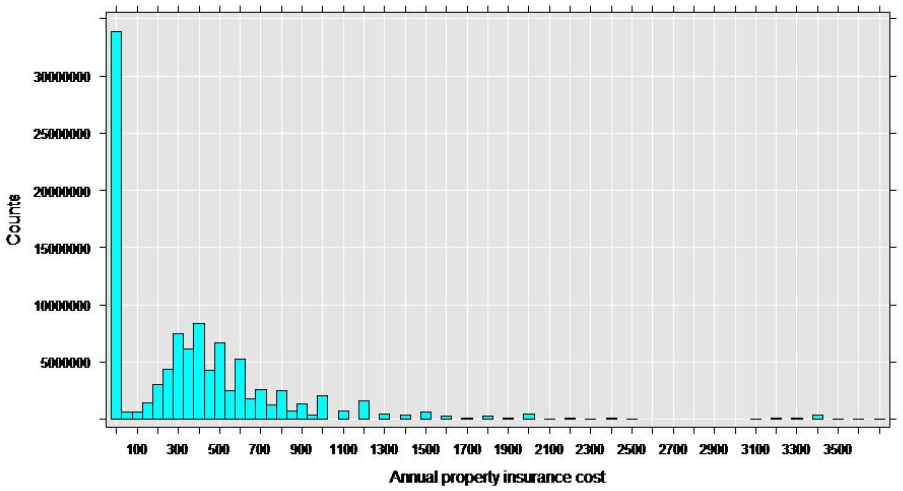
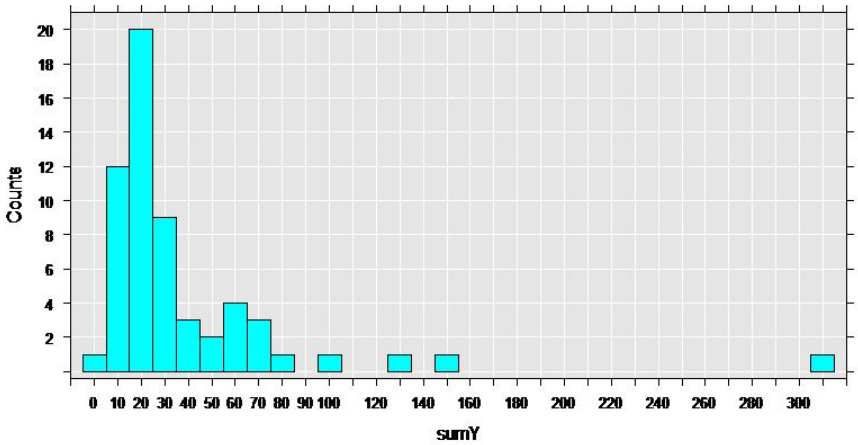
Generalized linear models

- What if we change the hypothesis $y \sim N(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p, \sigma^2)$

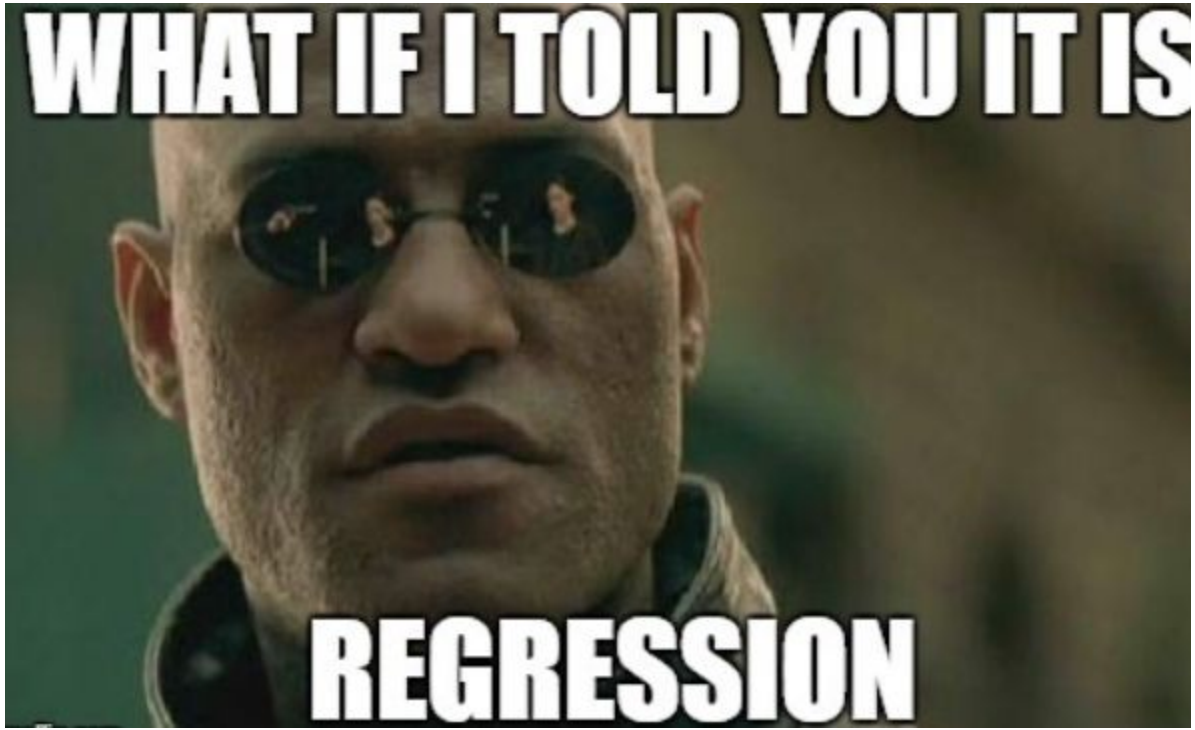


- https://scikit-learn.org/stable/modules/linear_model.html#generalized-linear-regression

Real world examples

Insurance cost (Tweedie distribution)	Number of calls arriving in a call center per hour (Poisson distribution)
	

Classification algorithms for regression



KNN regressor

How to calculate continuous variable for KNN?

KNN REGRESSOR

How to calculate continuous variable for KNN?

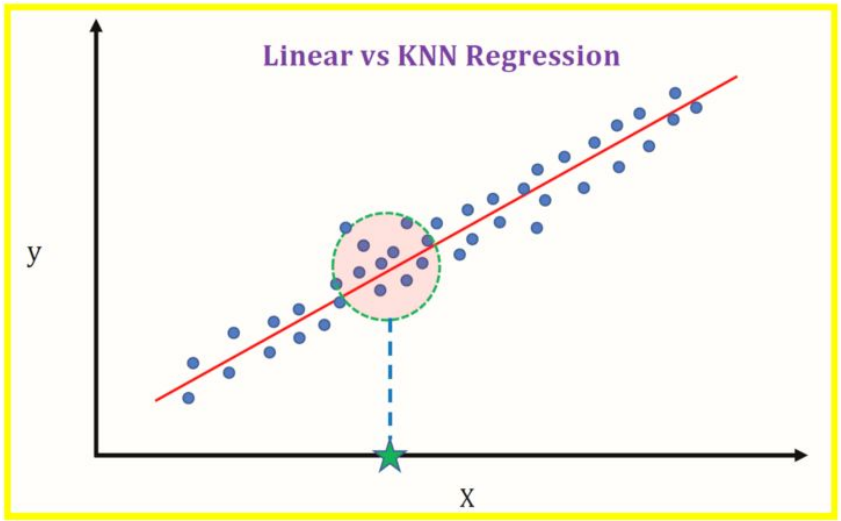
- Intuitive - each object in training has known target value
- We have k neighbors for prediction - let's average their target value!
- We can use distancing

Pros:

- Simple, not many changes from Classifier

Cons:

- All the cons of KNN



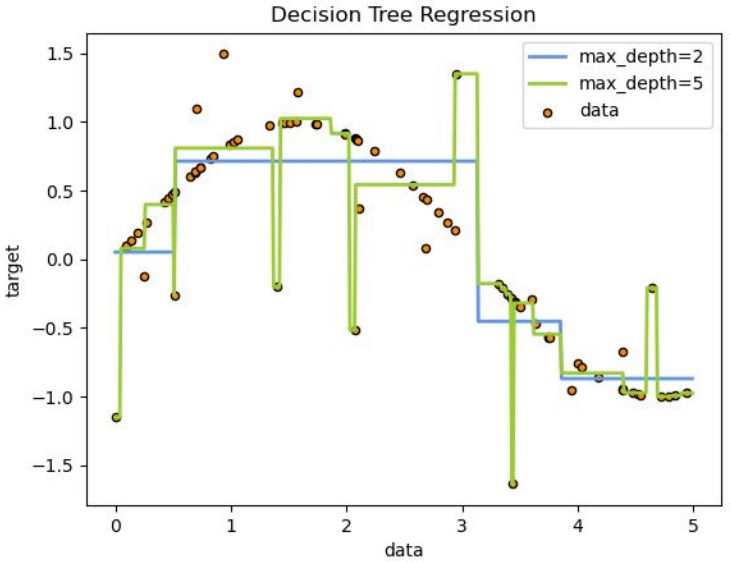
Decision tree regressor

How we can change decision tree to solve regression tasks?

Decision tree regressor

How we can change decision tree to solve regression tasks?

- Every leaf now contains the set of objects. Their average is the prediction we are looking for.
- We have to use other, continuous measures of information gain:
  - Variance (standard deviation)



Pros:

- Simplicity and interpretability of DT

Cons:

- Limited set of predicted values

Random forest regressor

How we can change random forest to solve regression task?

Random forest regressor

How we can change random forest to solve regression task?

- Nothing has changed, just take decision tree regressor as basic learner and average the result across estimators

Pros & cons:

- Everything is the same as in random forest classifier

Support Vector Machine

How this is going to work?

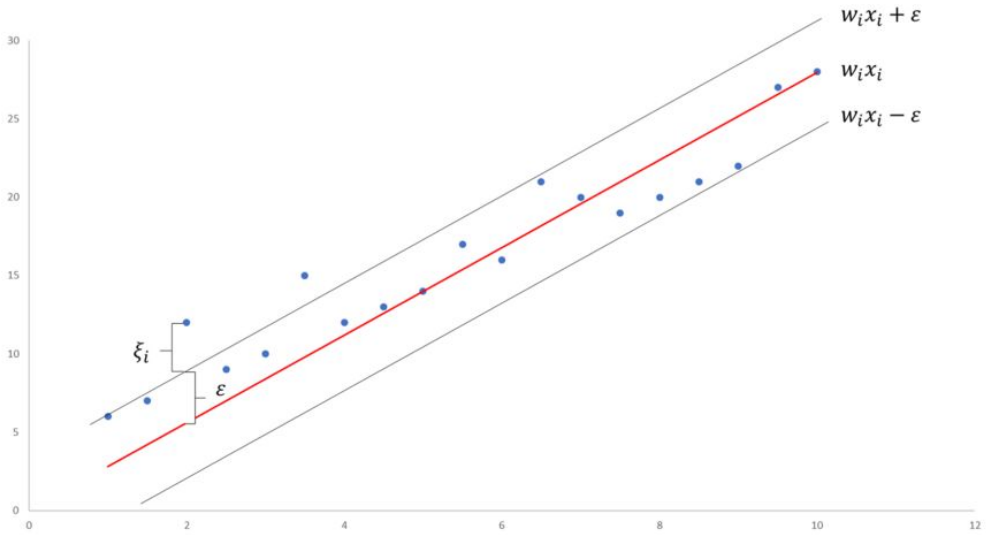
Support Vector Machine

How this is going to work?

- Reversing the SVM task: we create the plane, as narrow as possible, which includes as many points as it can inside:

Minimize $\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}|\xi_i|$

Constrain $|y_i - w_i x_i| \leq \varepsilon + |\xi_i|$



Gradient boosting

How do we use GB for regression tasks?

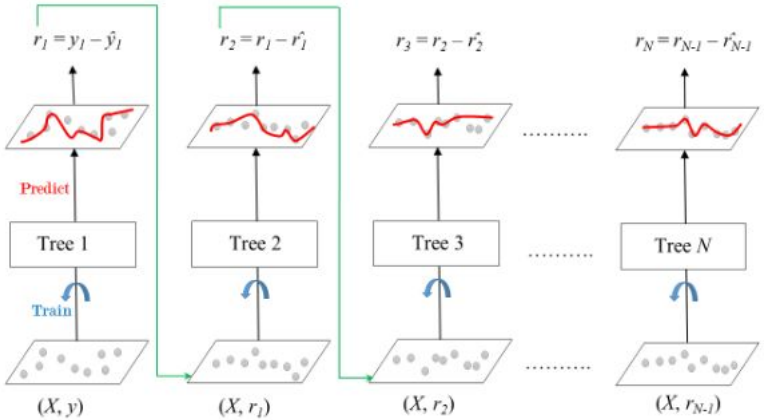
Gradient boosting

How do we use GB for regression tasks?

- Every new learner is fitted on error gradient with respect to ensemble of previous learners
- That means we fit every new tree on residuals from previous step

$$L_{\text{MSE}} = \frac{1}{2}(y - F(x))^2$$

$$h_m(x) = -\frac{\partial L_{\text{MSE}}}{\partial F} = y - F(x).$$



Advanced Hyperparameter Tuning

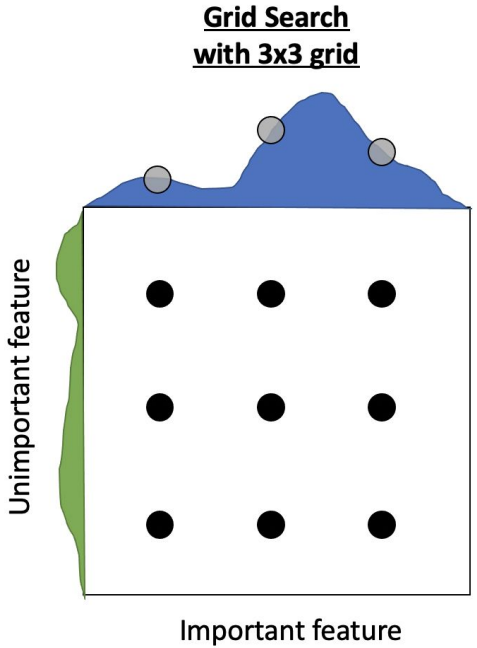
Advanced Hyperparameter Tuning

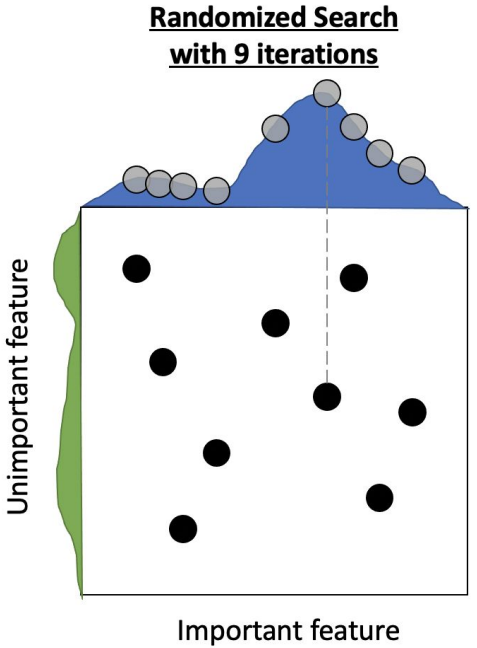
Which technics do you already know?

Advanced Hyperparameter Tuning

Which technics do you already know?

- Blind pick
- Grid Search
- Random Search





Advanced Hyperparameter Tuning

- HyperOpt http://hyperopt.github.io/hyperopt/. The idea behind can be explained through bayesian optimization