STK-INF3000/4000

WEEK 11

ENSEMBLE METHODS

FIELD TRIP:

REGRESSION ALGORITHMS
FOR ANOMALY DETECTION
IN TIME SERIES DATA

TIME SERIES DATA

- Given $x_i = x(t_i), i = 1, \ldots, n$.
  - Quantities measured at a given *time*.
    - E.g. number of users.
    - Bike trips taken.
- Want to predict an $x(t_i)$ in the future.
- Assume that t_i are periodic and equidistant.

AUTO-REGRESSIVE MODELS

- Fit

$$x(t) \approx \hat{f}\,(x(t)) = \hat{f}\,(x(t-\Delta_1), \ldots, x(t-\Delta_p))$$

  for chosen shifts Δ_k.
- Need to take into account periodicity in data.
  - E.g. $\Delta_1 = 1 \text{ day}$, $\Delta_2 = 1 \text{ week}$.

DETECTING ANOMALIES

- Predict values using $\hat{f}$, compare to actuals.
- Let $\delta_i = \hat{f}\,(x(t_i)) - x(t_i)$.
  - For an unbiased model, should have $\mathrm{E}[\delta] = 0$.
- Let $\sigma = \sqrt{\frac{1}{N_t} \sum_i (\delta_i - \hat{\delta}\,)^2}$.
- Let $z_i = \delta_i / \sigma$.
- Flag data points as anomalous if $z_i > z_{\mathrm{max}}$.

ENSEMBLE METHODS

GOALS

- Reduce over-fitting and increase accuracy by using *multiple* models.
  - Reduce variance.
  - Possibly increase bias.
- Most commonly used:
  - Boosting.
  - Bagging.

BOOSTING

ADDITIVE MODELS

Instead of using one complex predictor, use many instances of a very basic one b (e.g. a tree with one split).

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m)$$

The parameters are given by

$$\min_{\beta,\gamma} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta_m b(x; \gamma_m)\right).$$

FORWARD STEPWISE ADDITIVE MODELING

1. Set $f_0 \equiv 0$.
2. For $m = 1, \ldots, M$
   - Set

$$(\beta_m, \gamma_m) = \underset{\beta,\gamma}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)).$$

   - Set

$$f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m).$$

3. Return f_m.

FSAM AND SQUARE ERROR LOSS

For square error loss, we fit in each step to the residuals of the previous step.

$$\begin{aligned} L\left(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)\right) &= \left[y_i - f_{m-1}(x_i) - \beta b(x_i, \gamma)\right]^2 \\ &= \left[r_{im} - \beta b(x_i, \gamma)\right]^2 \end{aligned}$$

ADABOOST

Using

$$L(y, f(x)) = \exp(-yf(x))$$

for targets $y \in \{-1, 1\}$, gives raise to the AdaBoost algorithm.

ADABOOST

1. Initialize $w_i = 1/N,\ i = 1, \ldots, N$.
2. For $m = 1, \ldots, M$
   - Fit classifier $b_m(x)$ to data with weights w_i.
   - Set

$$e_m = \frac{\sum_i w_i I(y_i \neq b_m(x_i))}{\sum_i w_i}.$$

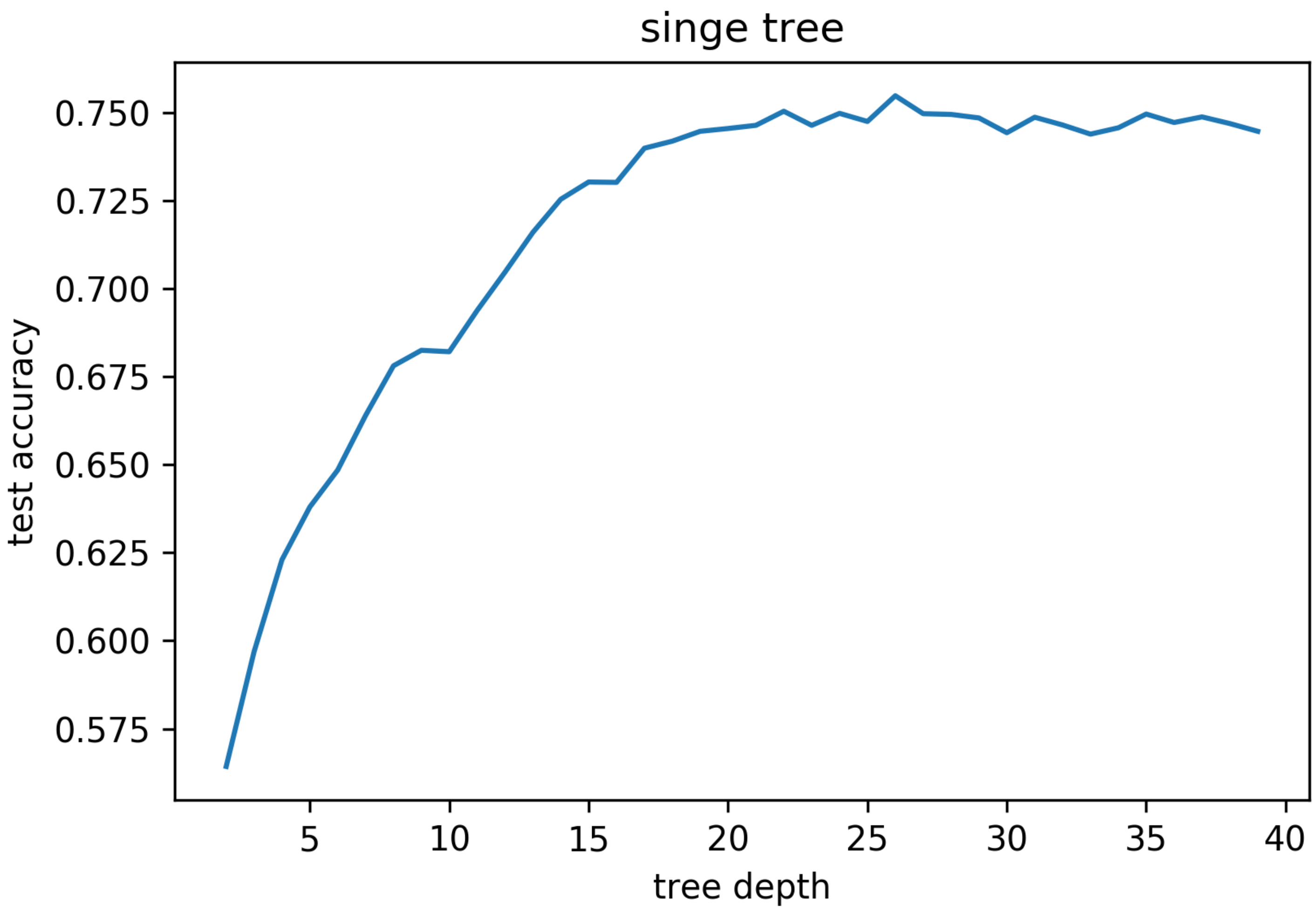
   - Set $\alpha_m = \log((1 - e_m)/e_m)$.
   - Set $w_i \leftarrow w_i \exp[\alpha_m I(y_i \neq b_m(x_i))]$.
3. Return $f(x) = \operatorname{sign}\left[\sum_{m=1}^{M} \alpha_m b_m(x)\right]$.

This can be adapted to *regression* as well.

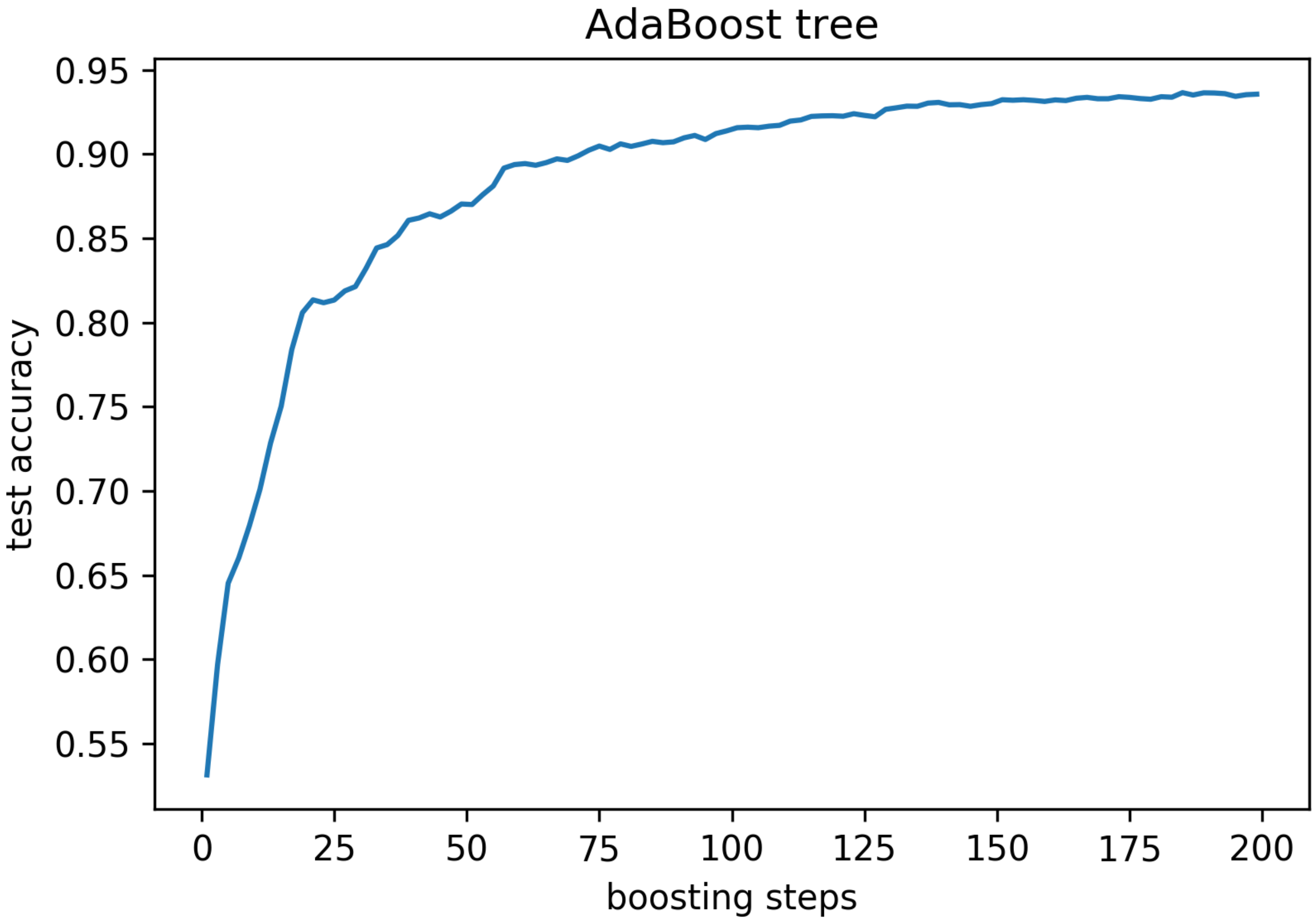
GENERATED DATA

- Taken from Elements of Statistical Learning.
- $X_1, \ldots, X_{10}$ standard Gaussians.
- $Y = \begin{cases} 1 & \text{if } \sum_j X_j^2 > 9.34 = \chi^2_{10}(0.5) \\ -1 & \text{else} \end{cases}$
- 2k training cases, 10k test cases.

SINGLE TREE ON GENERATED DATA



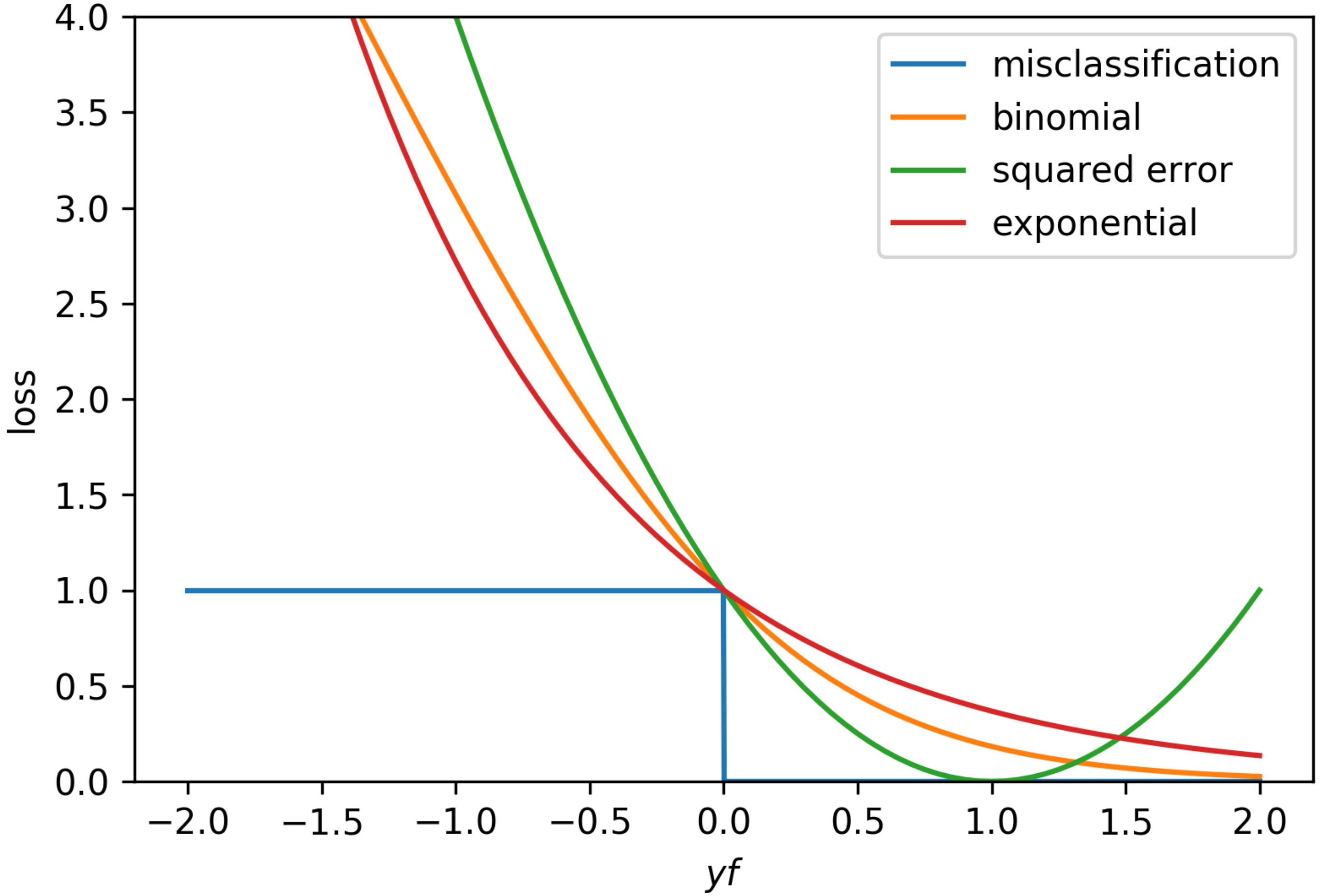
ADABOOST ON GENERATED DATA



MORE ON LOSS FUNCTIONS

- Let's compare some loss functions for **classification**.
- We'll classify to $\mathrm{sign}(f)$.
- Misclassification: $I(\mathrm{sign}(f) \neq y)$.
- Exponential: $\exp(-yf)$.
- Binomial: $\log(1 + \exp(-2fy))$.
- Squared error: $(y - f)^2$.

LOSS FUNCTIONS FOR CLASSIFICATION

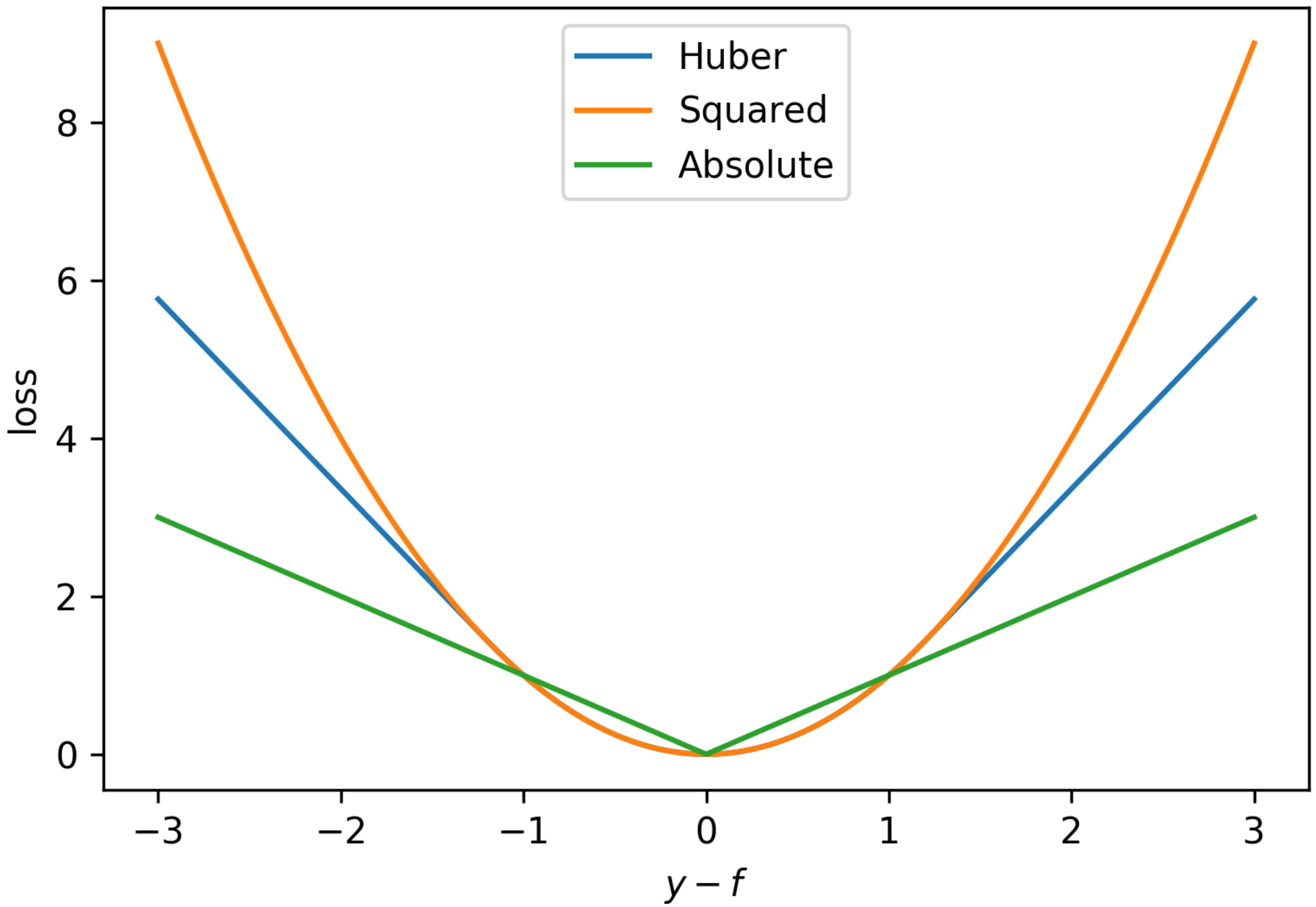


COMMON LOSS FUNCTIONS FOR REGRESSION

- Squared error, $(f(x) - y)^2$.
- Absolute error, $|f(x) - y|$.
- Huber loss,

$$L(f(x), y) = \begin{cases} (y - f(x))^2 & \text{if } |y - f(x)| \leq \delta \\ 2\delta|y - f(x)| - \delta^2 & \text{else.} \end{cases}$$

LOSS FUNCTIONS FOR REGRESSION



THE IDEAL LOSS FUNCTION

- Robust against outliers.
- Numerically easy.
- Not 'deteriorate'.

THE NEED FOR GRADIENTS.

AdaBoost works great, but we'd like to plug in arbitrary loss functions. This seems like a hard task looking at

$$\min_{\beta,\gamma} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta_m b(x;\gamma_m)\right).$$

GRADIENT BOOSTING.

Using a tree T as our basic model, we build up our model as

$$f_M(x) = \sum_{m=1}^{M} T(x; \gamma_m),$$

where $\gamma_m = \{R_j, c_j\}$ defines the *regions* R of the terminal nodes and c_j their predictions. We want to find

$$\hat{\gamma}_m = \operatorname*{agrmin}_{\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x; \gamma)).$$

MINIMIZATION

In each step, we want to minimize

$$L(f) = \sum_{i=1}^{N} L(y_i, f(x_i))$$

in order to find

$$\hat{\mathbf{f}} = \operatorname*{argmin}_{\mathbf{f}} L(f).$$

This can be thought of as point-wise optimization.

GRADIENT DESCENT

In order to optimize a multi-valued function, gradient descent constructs a stepwise solution

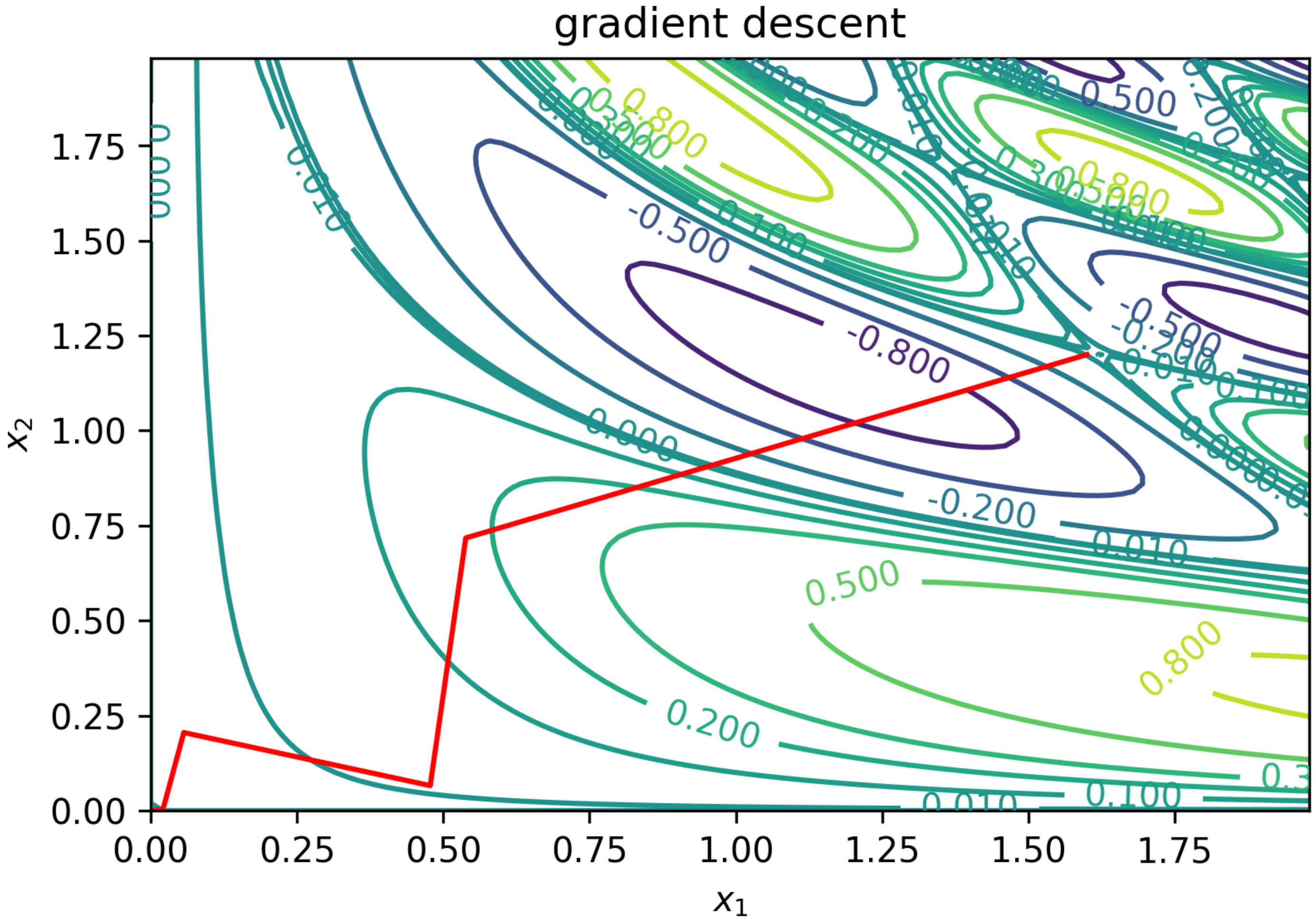
$$\mathbf{f}_M = \sum_{i=1}^{M} \mathbf{h}_m,$$

with

$$h_{im} = -\alpha_m g_{im} = -\alpha_m \left. \frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right|_{f(x_i)=f_{m-1}(x_i)},$$

$$\alpha_m = \underset{\alpha}{\operatorname{argmin}}\, L(\mathbf{f}_{m-1} - \alpha \mathbf{g}_m).$$

$$f(x_1, x_2) = \sin(x_1^2\, x_2) \cos(2\, x_1\, x_2^2)$$



GRADIENT BOOSTED TREES.

Now use trees to fit at each step

$$\hat{\gamma}_m = \operatorname*{argmin}_{\gamma} \sum_{i=1}^{N} (-g_{im} - T(x_i;\gamma))^2$$

and construct the final model

$$f_M(x) = \sum_{m=1}^{M} T(x;\hat{\gamma}_m).$$

REGULARIZATION

In addition to tuning M, one can introduce a parameter ν and set

$$f_m(x) = f_{m-1}(x) + \nu \sum_{j=1}^{J} \gamma_{jm} I(x \in R_{jm})$$

.

The number ν is often called the *learning rate*.

QUESTIONS?