Day 3: Classification, logistic regression

Introduction to Machine Learning Summer School
June 18, 2018 - June 29, 2018, Chicago

Instructor: Suriya Gunasekar, TTI Chicago

20 June 2018

TOYOTA TECHNOLOGICAL INSTITUTE AT CHICAGO

THE UNIVERSITY OF CHICAGO

NSF

Topics so far

- Supervised learning, linear regression
- Yesterday
  - Overfitting,
  - Ridge and lasso Regression
  - Gradient descent
- Today
  - Bias variance trade-off
  - Classification
  - Logistic regression
  - Regularization for logistic regression
  - Classification metrics

Bias-variance tradeoff

Empirical vs population loss

- **Population distribution** Let $(x, y) \sim \mathcal{D}$
- We have
  - Loss function $\ell(\hat{y}, y)$
  - Hypothesis class $\mathcal{H}$
  - Training data $S = \{(x^{(i)}, y^{(i)}) : i = 1,2, \ldots, N\} \sim_{\text{iid}} \mathcal{D}^N$
    - Think of *S* as random variable

e.g, $\Pr(x) = uniform(0,1)$
$y = w^*.x + \epsilon$ where $\epsilon = \mathcal{N}(0,0.1)$
$\Rightarrow \Pr(y|x) = \mathcal{N}(w^*.x, 0.1)$
$\Pr(x, y) = \Pr(x) \Pr(y|x)$

- What we really want $f \in \mathcal{H}$ to minimize **population loss**

$$L_{\mathcal{D}}(f) \triangleq \mathbf{E}_{\mathcal{D}}[\ell(f(x), y)] = \int_{(x,y)} \ell(f(x), y) \Pr(x, y)$$

- ERM minimizes **empirical loss**

$$L_S(f) \triangleq \hat{\mathbf{E}}_S[\ell(f(x), y)] = \frac{1}{N} \sum_{i=1}^{N} \ell\left(f\left(x^{(i)}\right), y^{(i)}\right)$$

Empirical vs population loss

$$L(f) \triangleq \mathbf{E}_{\mathcal{D}}[\ell(f(x), y)] = \int_{(x,y)} \ell(f(x), y)\ \Pr(x, y)$$

$$L_S(f) \triangleq \hat{\mathbf{E}}_S[\ell(f(x), y)] = \frac{1}{N}\sum_{i=1}^{N} \ell\left(f\left(x^{(i)}\right), y^{(i)}\right)$$

- $\hat{f}_S$ from some model **overfits** to S if there is $f^* \in \mathcal{H}$ with
  $\hat{\mathbf{E}}_S[\ell(\hat{f}_S(x), y)] \leq \hat{\mathbf{E}}_S[\ell(f^*(x), y)]$ but $\mathbf{E}_{\mathcal{D}}[\ell(\hat{f}_S(x), y)] \gg \mathbf{E}_{\mathcal{D}}[\ell(f^*(x), y)]$
- If f is independent of S_{train} then both $L_{S_{train}}(f)$ and $L_{S_{test}}(f)$ are good approximations of $L_{\mathcal{D}}(f)$
- But generally, $\hat{f}$ depends on S_{train}. Why?
  - $L_{S_{train}}(\hat{f}_{S_{train}})$ is no more a good approximation of $L_{\mathcal{D}}(f)$
  - $L_{S_{test}}(\hat{f}_{S_{train}})$ is still a good approximation of $L_{\mathcal{D}}(f)$ since $\hat{f}_{S_{train}}$ is independent of S_{test}

Optimum Unrestricted Predictor

- Consider population squared loss

$$\underset{f\in\mathcal{H}}{\operatorname{argmin}}\, L(f) \triangleq \mathbf{E}_{\mathcal{D}}[\ell(f(x),y)] = \mathbf{E}_{(x,y)}[(f(x)-y)^2]$$

- Say $\mathcal{H}$ is unrestricted – any function $f: x \to y$ is allowed

$$L(f) = \mathbf{E}_{(x,y)}[(f(x)-y)^2] = \mathbf{E}_x\left[\mathbf{E}_y[(f(x)-y)^2\,|x]\right]$$

$$= \mathbf{E}_x\left[\mathbf{E}_y\left[\left(f(x)-\mathbf{E}_y[y|x]+\mathbf{E}_y[y|x]-y\right)^2|x\right]\right]$$

$$= \mathbf{E}_x\left[\mathbf{E}_y\left[\left(f(x)-\mathbf{E}_y[y|x]\right)^2|x\right]\right] + \mathbf{E}_x\left[\mathbf{E}_y\left[\left(\mathbf{E}_y[y|x]-y\right)^2|x\right]\right]$$

$$+ 2\,\mathbf{E}_x\left[\mathbf{E}_y\left[\underbrace{(f(x)-\mathbf{E}_y[y|x])}_{\text{not a function of } y}\underbrace{(\mathbf{E}_y[y|x]-y)}_{=0}|x\right]\right]$$

$$= \underbrace{\mathbf{E}_x[\left(f(x)-\mathbf{E}_y[y|x]\right)^2]}_{\text{minimized for } f=\mathbf{E}_y[y|x]} + \underbrace{\mathbf{E}_{x,y}\left[\left(\mathbf{E}_y[y|x]-y\right)^2\right]}_{\text{Noise}}$$

Bias variance decomposition

- Best unrestricted predictor $f^{**}(x) = \mathbf{E}_y[y|x]$
- $L(f_S) = \mathbf{E}_x[(f_S(x) - f^{**}(x))^2] + \mathbf{E}_{x,y}[(f^{**}(x) - y)^2]$
- $\mathbf{E}_S L(f_S) = \mathbf{E}_S \mathbf{E}_x \left[(f_S(x) - f^{**}(x))^2\right] + noise$

$$\mathbf{E}_S \mathbf{E}_x \left[(f_S(x) - f^{**}(x))^2\right] = \mathbf{E}_x \left[\mathbf{E}_S \left[(f_S(x) - f^{**}(x))^2 | x\right]\right]$$

$$= \mathbf{E}_x \mathbf{E}_S \left[(f_S(x) - \mathbf{E}_S[f_S(x)] + \mathbf{E}_S[f_S(x)] - f^{**}(x))^2 | x\right]$$

$$= \mathbf{E}_x \mathbf{E}_S [(f_S(x) - \mathbf{E}_S[f_S(x)])^2 | x] + \mathbf{E}_x \left[(\mathbf{E}_S[f_S(x)] - f^{**}(x))^2\right]$$

$$+ 2\,\mathbf{E}_x \left[\mathbf{E}_S [(\mathbf{E}_S[f_S(x)] - f^{**}(x))(f_S(x) - \mathbf{E}_S[f_S(x)]) | x]\right]$$

$$= \mathbf{E}_{S,x}[(f_S(x) - \mathbf{E}_S[f_S(x)])^2] + \mathbf{E}_x \left[(\mathbf{E}_S[f_S(x)] - f^{**}(x))^2\right]$$

$$\mathbf{E}_S L(f_S) = \mathbf{E}_{S,x}[(f_S(x) - \mathbf{E}_S[f_S(x)])^2]$$
$$+ \mathbf{E}_x \left[(\mathbf{E}_S[f_S(x)] - f^{**}(x))^2\right]$$
$$+ \mathbf{E}_{x,y}[(f^{**}(x) - y)^2]$$

= variance
+ bias^2
+ noise

Bias-variance tradeoff

$$\mathbf{E}_S L(f_S) = \mathbf{E}_{S,x}\left[(f_S(x) - \mathbf{E}_S[f_S(x)])^2\right]$$
$$+\mathbf{E}_x\left[\left(\mathbf{E}_S\left[f_S(x)\right] - f^{**}(x)\right)^2\right]$$
$$+\mathbf{E}_{x,y}[(f^{**}(x) - y)^2\,]$$

= variance
+ bias^2
+ noise

- $f_S \in \mathcal{H}$
- noise is irreducible
- variance can reduced by
  - get more data
  - make f_S less sensitive to S
    - less number of candidates in $\mathcal{H}$ to choose from→less variance
    - reducing the "complexity" of model class $\mathcal{H}$ decreases variance
- bias^2 $\geq \min_{f \in \mathrm{conv}(\mathcal{H})} \mathbf{E}_x\left[\left(f(x) - f^{**}(x)\right)^2\right]$
  - expanding model class $\mathcal{H}$ decreases bias

Model complexity

- reducing the complexity of model class $\mathcal{H}$ decreases variance
- expanding model class $\mathcal{H}$ decreases bias
- Complexity $\approx$ number of choices in $\mathcal{H}$
  - For any loss L, for all $f \in \mathcal{H}$ with probability greater than $1-\delta$

$$L(f) \leq L_S(f) + \sqrt{\frac{\log|\mathcal{H}| + \log\frac{1}{\delta}}{N}}$$

  - many other variants for infinite cardinality classes
  - often bounds are loose
- Complexity $\approx$ number of degrees of freedom
  - e.g., number of parameters to estimate
  - more data ➔can fit more complex models
- Is $\mathcal{H}_1 = \{\boldsymbol{x} \rightarrow w_0 + \boldsymbol{w_1}.\boldsymbol{x} - \boldsymbol{w_2}.\boldsymbol{x}\}$ more complex than $\mathcal{H}_2 = \{\boldsymbol{x} \rightarrow w_0 + \boldsymbol{w_1}.\boldsymbol{x}\}$?
  - What we need is how many different "behaviors" we can get on same S

Summary

- Overfitting
  - What is overfitting?
  - How to detect overfitting?
  - Avoiding overfitting using model selection
- Bias – variance tradeoff

Classification

- Supervised learning: estimate a mapping f from input $x \in \mathcal{X}$ to output $y \in \mathcal{Y}$
  - **Regression** $\mathcal{Y} = \mathbb{R}$ or other continuous variables
  - **Classification** $\mathcal{Y}$ takes discrete set of values
    - Examples:
      - $\mathcal{Y} = \{\text{spam}, \text{nospam}\}$,
      - digits (not values) $\mathcal{Y} = \{0,1,2, \dots, 9\}$
- Many successful applications of ML in vision, speech, NLP, healthcare

Classification vs Regression

- Label-values do not have meaning
  - $\mathcal{Y} = \{\text{spam}, \text{nospam}\}$ or $\mathcal{Y} = \{0,1\}$ or $\mathcal{Y} = \{-1,1\}$
- Ordering of labels does not matter (for most parts)
  - $f(x) =$ "0" when $y =$ "1" is as bad as $f(x) =$ "9" when $y =$ "1"
- Often $f(x)$ does not return labels y
  - e.g. in binary classification with $\mathcal{Y} = \{-1,1\}$ we often estimate $f: \mathcal{X} \to \mathbb{R}$ and then post process to get $\hat{y}\big(f(x)\big) = \mathbf{1}[f(x) \geq 0]$
  - mainly for computational reasons
    - remember, we need to solve $\min_{f \in \mathcal{H}} \sum_i \ell(f\big(x^{(i)}\big), y^{(i)})$
    - discrete values → combinatorial problems → hard to solve
  - more generally $\mathcal{H} \subset \{f: \mathcal{X} \to \mathbb{R}\}$ and loss $\ell: \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$
    - compare to regression, where typically $\mathcal{H} \subset \{f: \mathcal{X} \to \mathcal{Y}\}$ and loss $\ell: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$

Non-parametric classifiers

Nearest Neighbor (NN) Classifier

- Training data $S = \{(x^{(i)}, y^{(i)}): i = 1,2, \dots, N\}$
- Want to predict label of new point x
- Nearest Neighbor Rule
  - Find the closest training point: $i^* = \arg\min_i \rho(x, x^{(i)})$
  - Predict label of x as $\hat{y}(x) = y^{(i^*)}$
- Computation
  - Training time: Do nothing
  - Test time: search the training set for a NN

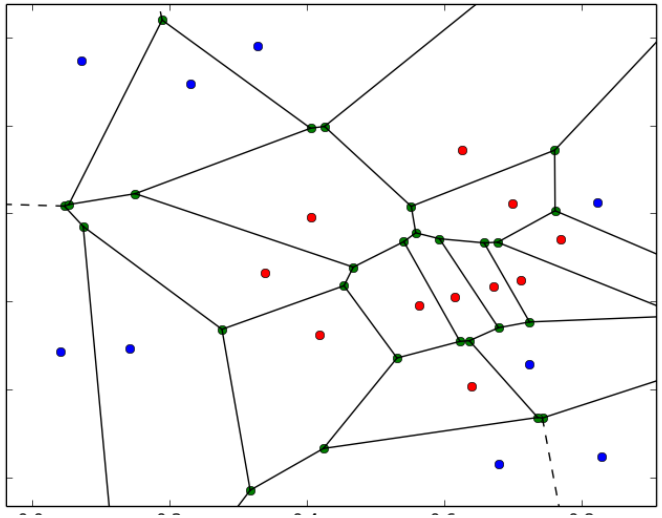
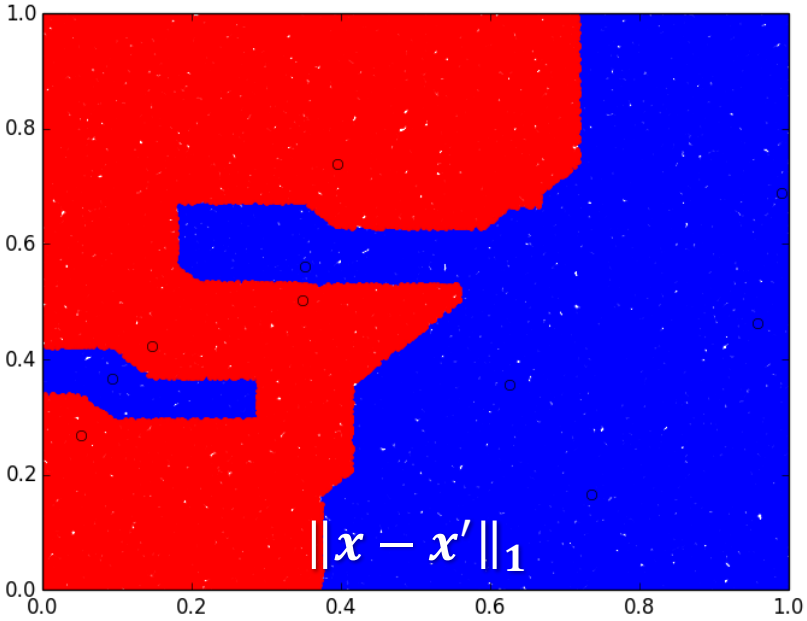
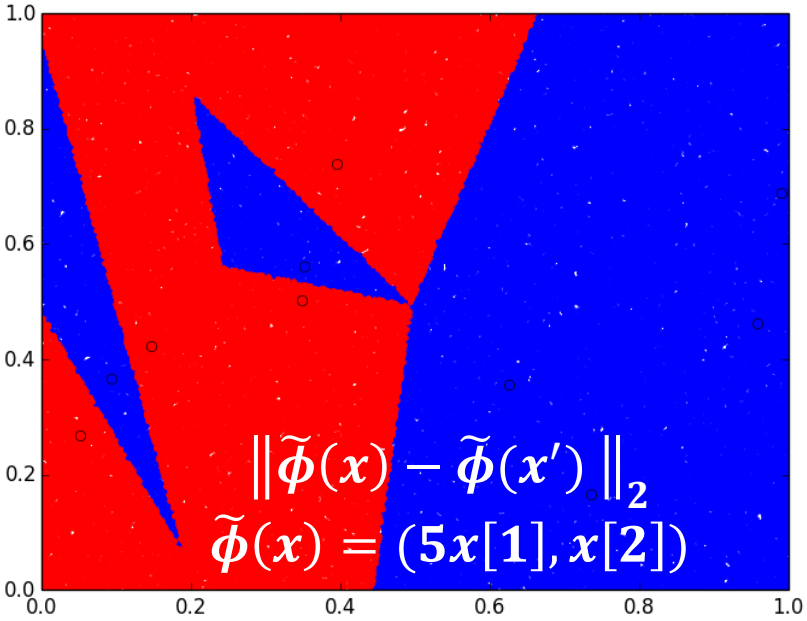


Figure credit: Nati Srebro

Nearest Neighbor (NN) Classifier

- Where is the main model?
  - $i^* = \arg\min_i \rho(x, x^{(i)})$
  - What is the right "distance" between images? Between sound waves? Between sentences?
  - Often $\rho(x, x') = \|\phi(x) - \phi(x')\|_2$ or other norms $\|x - x'\|_1$





Slide credit: Nati Srebro

k-Nearest Neighbor (kNN) classifier

- Training data $S = \{(x^{(i)}, y^{(i)}) : i = 1,2, \ldots, N\}$
- Want to predict label of new point x
- k-Nearest Neighbor Rule
  - Find the $\boldsymbol{k}$ closest training point: $i_1^*, i_2^*, \ldots, i_k^*$
  - Predict label of x as

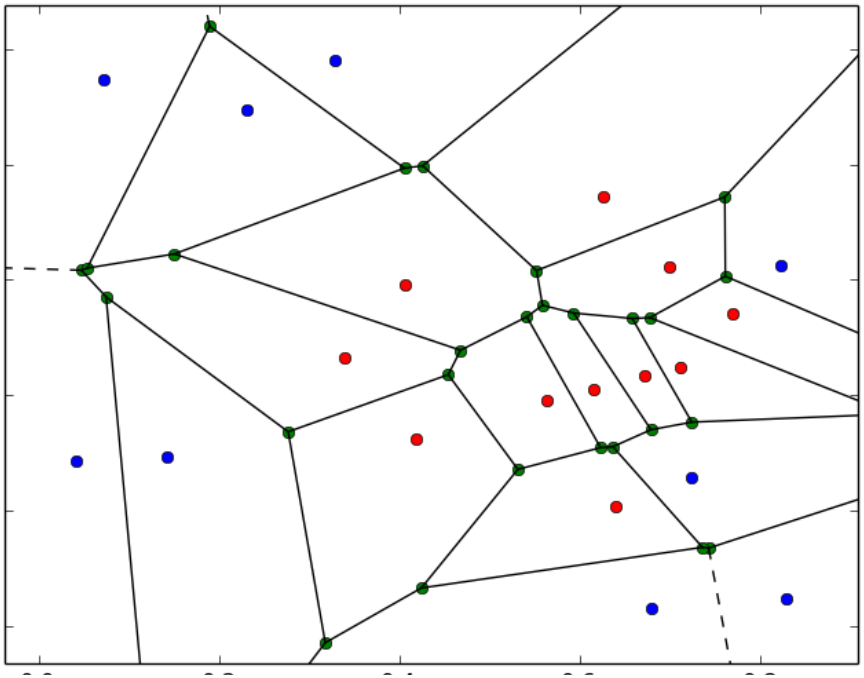
$$\hat{y}(x) = \text{majority}\left(y^{(i_1^*)}, y^{(i_2^*)}, \ldots, y^{(i_k^*)}\right)$$

- Computation
  - Training time: Do nothing
  - Test time: search the training set for k NNs

k-Nearest Neighbor

- Advantages
  - no training
  - universal approximator – non-parametric
- Disadvantages
  - not scalable
    - test time memory requirement
    - test time computation
  - easily overfits with small data

Training vs test error



1-NN

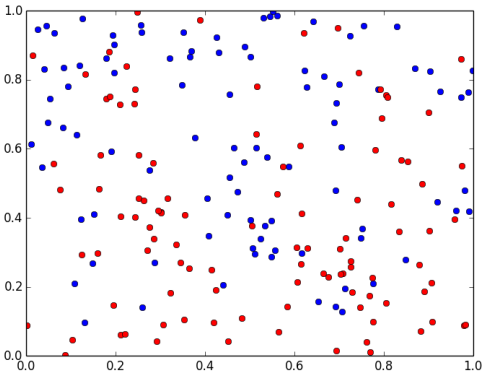
- Training error?
  - 0
- Test error?
  - Depends on $\Pr(x, y)$

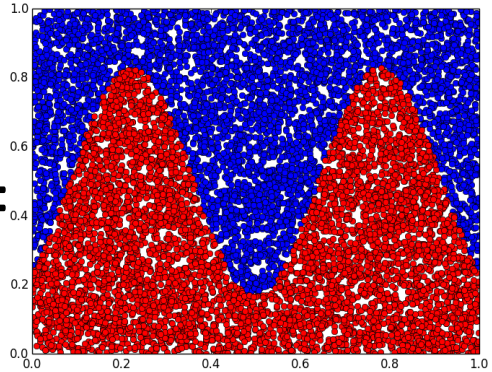
k-NN

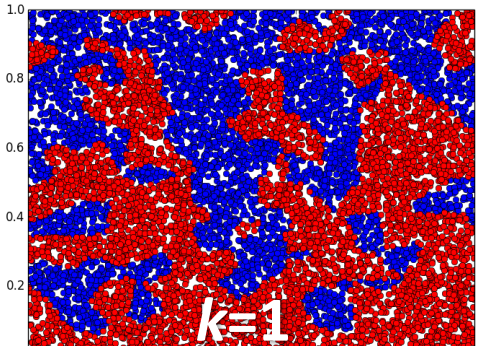
- Training error: can be greater than 0
- Test error: again depends on $\Pr(x, y)$

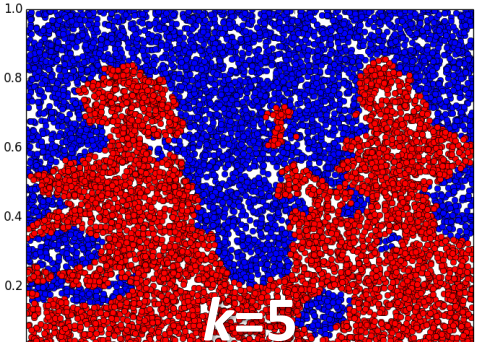
Figure credit: Nati Srebro

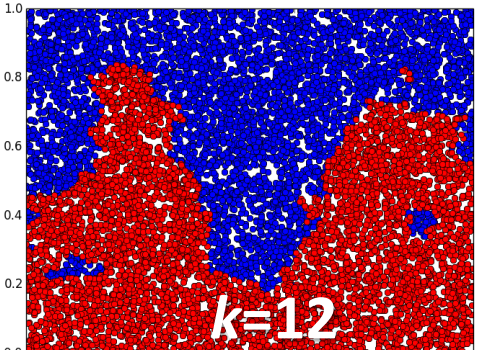
k-Nearest Neighbor: Data Fit / Complexity Tradeoff

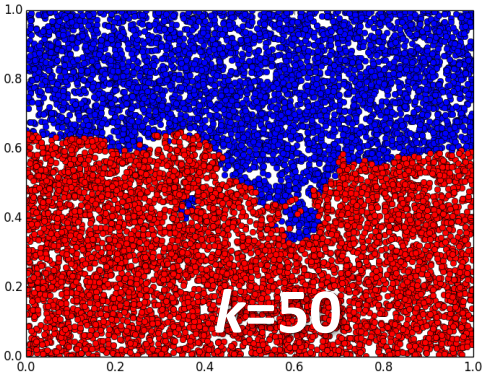
S= 

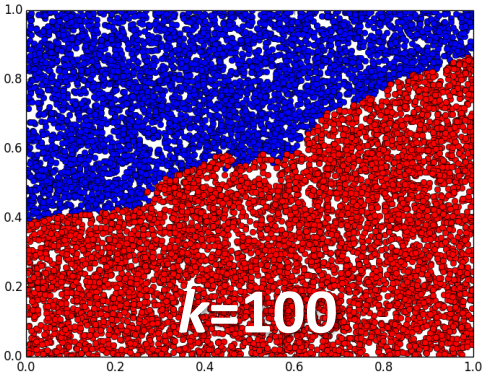
*h**= 

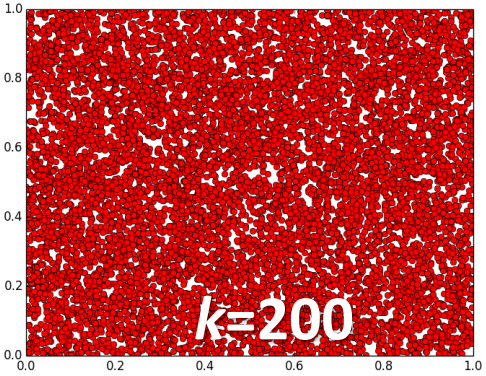












Slide credit: Nati Srebro

Space partition

- kNN partitioning of $\mathcal{X}$ (or $\mathbb{R}^d$) into regions of +1 and -1
- What about discrete valued features x?
- Even for continuous x, can we get more structured partitions?
  - easy to describe
    - e.g., $R_2 = \{x: x_1 < t_1 \text{ and } x_2 > t_2\}$
  - reduces degrees of freedom
- Any non-overlapping partition using only (hyper) rectangles
  → representable by a tree

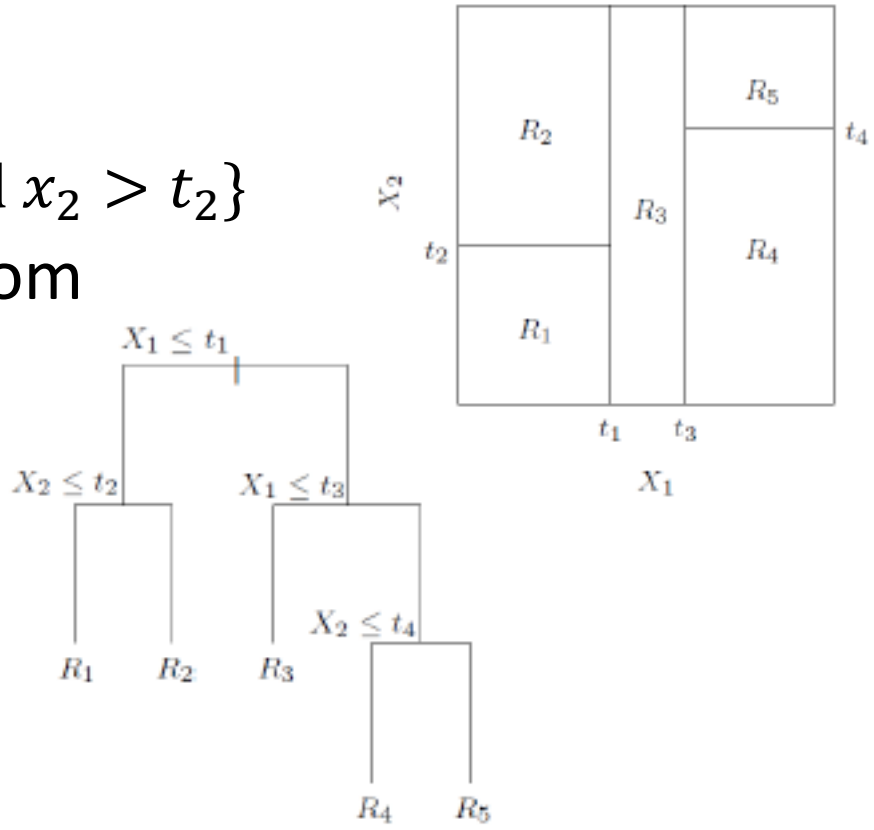


Figure credit: Greg Shaknarovich

Decision trees

- Focus on binary trees (trees with at most two children at each node)
- How to create trees?
- What is a "good" tree?
  - Measure of "purity" at each leaf node where each leaf node corresponding to a region R_i

$$\text{purity}(tree) = \sum_{R_i} |\#\text{ blue at } R_i - \#\text{ red at } R_i|$$

There are various metrics of (im)purity that are used in practice, but the rough idea is the same

Decision trees

- How to create trees?
- Training data $S = \{(x^{(i)}, y^{(i)}): i = 1,2, \dots, N\}$, where $y^{(i)} \in \{\text{blue}, \text{red}\}$
- At each point,

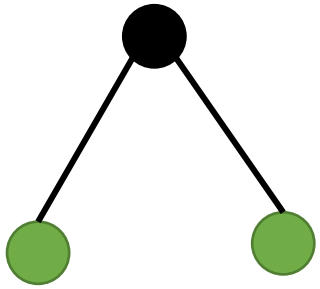
$$\text{purity}(tree) = \sum_{\text{leaf}} |\#\text{ blue at leaf} - \#\text{ red at leaf}|$$

- Start with all data at root
  - only one $leaf = root$. What is $\text{purity}(tree)$?

Decision trees

- How to create trees?
- Training data $S = \{(x^{(i)}, y^{(i)}) : i = 1,2, \dots, N\}$, where $y^{(i)} \in \{\text{blue}, \text{red}\}$
- At each point,

$$\text{purity}(tree) = \sum_{\text{leaf}} |\#\text{ blue at leaf} - \#\text{ red at leaf}|$$



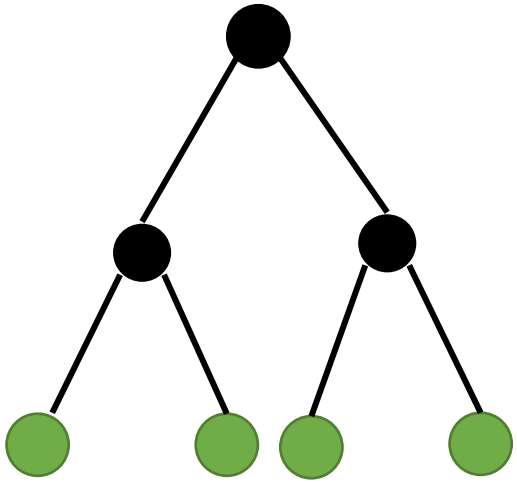
- Start with all data at root
  - only one $leaf = root$. What is $\text{purity}(tree)$?
- Create a split based on a rule that **increases** the amount of "**purity**" of tree.
  - How complex can the rules be?

Decision trees

- How to create trees?
- Training data $S = \{(x^{(i)}, y^{(i)}): i = 1,2, \dots, N\}$, where $y^{(i)} \in \{\text{blue}, \text{red}\}$
- At each point,

$$\text{purity}(tree) = \sum_{\text{leaf}} |\#\text{ blue at leaf} - \#\text{ red at leaf}|$$

- Start with all data at root
  - only one $leaf = root$. What is $\text{purity}(tree)$?
- Create a split based on a rule that **increases** the amount of "**purity**" of tree.
  - How complex can the rules be?
- Repeat



When to stop?

what is the complexity of a DT?

- Limit the number of leaf nodes

Decision trees

- Advantages
  - interpretable
  - easy to deal with non-numeric features
  - natural extensions to multi-class, multi-label
- Disadvantages
  - not scalable
  - hard decisions – non-smooth decisions
  - often overfits in spite of regularization
- Check CART package in scikit-learn

Parametric classifiers

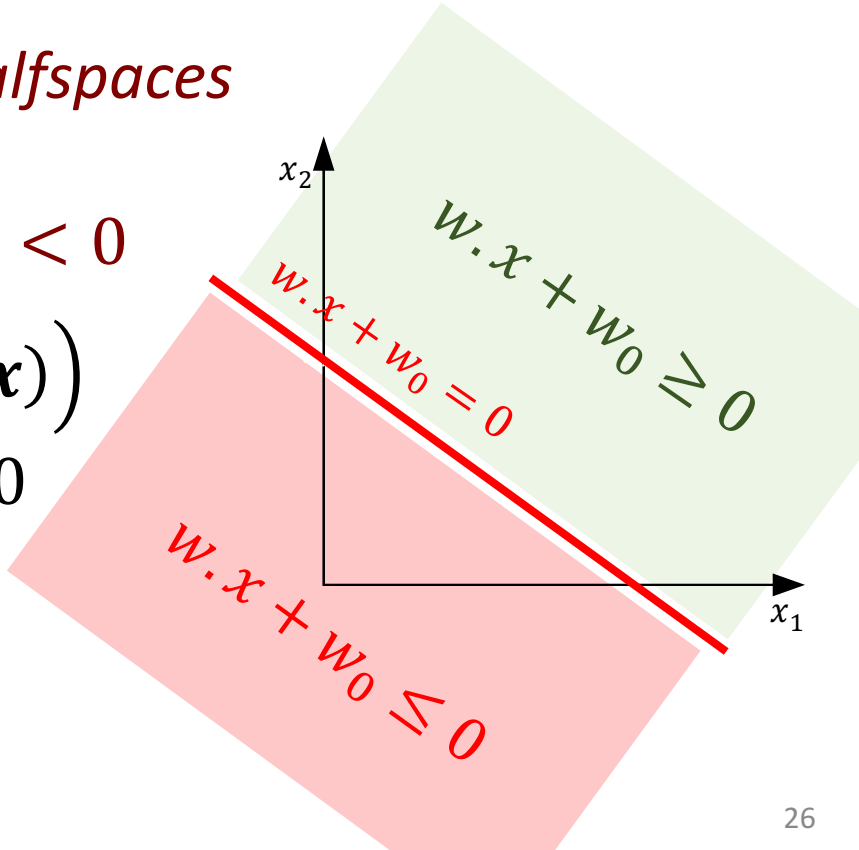
- What is the equivalent of linear regression?
  - something easy to train
  - something easy to use at test time
- $f(\boldsymbol{x}) = f_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}.\boldsymbol{x} + w_0$
- $\mathcal{H} = \{f_{\boldsymbol{w}} = \boldsymbol{x} \to \boldsymbol{w}.\boldsymbol{x} + w_0 : \boldsymbol{w} \in \mathbb{R}^d, w_0 \in \mathbb{R}\}$
- but $f(\boldsymbol{x}) \notin \{-1,1\}$! how do we get labels?
  - reasonable choice

    $\hat{y}(\boldsymbol{x}) = 1$ if $f_{\widehat{\boldsymbol{w}}}(\boldsymbol{x}) \geq 0$ and $\hat{y}(\boldsymbol{x}) = -1$ otherwise
  - linear classifier: $\hat{y}(\boldsymbol{x}) = \text{sign}(\widehat{\boldsymbol{w}}.\boldsymbol{x} + \widehat{w}_0)$

Parametric classifiers

- $\mathcal{H} = \{f_{\boldsymbol{w}} = \boldsymbol{x} \to \boldsymbol{w}.\boldsymbol{x} + w_0 : \boldsymbol{w} \in \mathbb{R}^d, w_0 \in \mathbb{R}\}$
- $\hat{y}(\boldsymbol{x}) = \text{sign}(\hat{\boldsymbol{w}}.\boldsymbol{x} + \hat{w}_0)$
- $\hat{\boldsymbol{w}}.\boldsymbol{x} + \hat{w}_0 = 0$ (linear) decision boundary or separating *hyperplane*
  - that separates $\mathbb{R}^d$ into two *halfspaces (regions)*
    $\hat{\boldsymbol{w}}.\boldsymbol{x} + \hat{w}_0 > 0$ and $\hat{\boldsymbol{w}}.\boldsymbol{x} + \hat{w}_0 < 0$
- more generally, $\hat{y}(\boldsymbol{x}) = \text{sign}\left(\hat{f}(\boldsymbol{x})\right)$

  → decision boundary is $\hat{f}(\boldsymbol{x}) = 0$



Linear classifier

Classification vs Regression

- Label-values do not have meaning
  - $\mathcal{Y} = \{\text{spam}, \text{nospam}\}$ or $\mathcal{Y} = \{0,1\}$ or $\mathcal{Y} = \{-1,1\}$
- Ordering of labels does not matter (for most parts)
  - $f(x) =$ "0" when $y =$ "1" is as bad as $f(x) =$ "9" when $y =$ "1"
- Often $f(x)$ does not return labels y
  - e.g. in binary classification with $\mathcal{Y} = \{-1,1\}$ we often estimate $f: \mathcal{X} \to \mathbb{R}$ and then post process to get $\hat{y}\big(f(x)\big) = \mathbf{1}[f(x) \geq 0]$
  - mainly for computational reasons
    - remember, we need to solve $\min_{f \in \mathcal{H}} \sum_i \ell(f\big(x^{(i)}\big), y^{(i)})$
    - discrete values → combinatorial problems → hard to solve
  - more generally $\mathcal{H} \subset \{f: \mathcal{X} \to \mathbb{R}\}$ and loss $\ell: \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$
    - compare to regression, where typically $\mathcal{H} \subset \{f: \mathcal{X} \to \mathcal{Y}\}$ and loss $\ell: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$

Classification vs Regression

- Label-values do not have meaning
  - $\mathcal{Y} = \{s$
- Ordering o
  - $f(x) =$ $y = \text{“1”}$
- Often $f(x)$
  - e.g. in l stimate
    $f: \mathcal{X} \rightarrow$ $(x) \geq 0]$
  - mainly
    - rer)
    - discrete values → combinatorial problems → hard to solve
  - more generally $\mathcal{H} \subset \{f: \mathcal{X} \rightarrow \mathbb{R}\}$ and loss $\ell: \mathbb{R} \times \mathcal{Y} \rightarrow \mathbb{R}$
    - compare to regression, where typically $\mathcal{H} \subset \{f: \mathcal{X} \rightarrow \mathcal{Y}\}$ and loss $\ell: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$

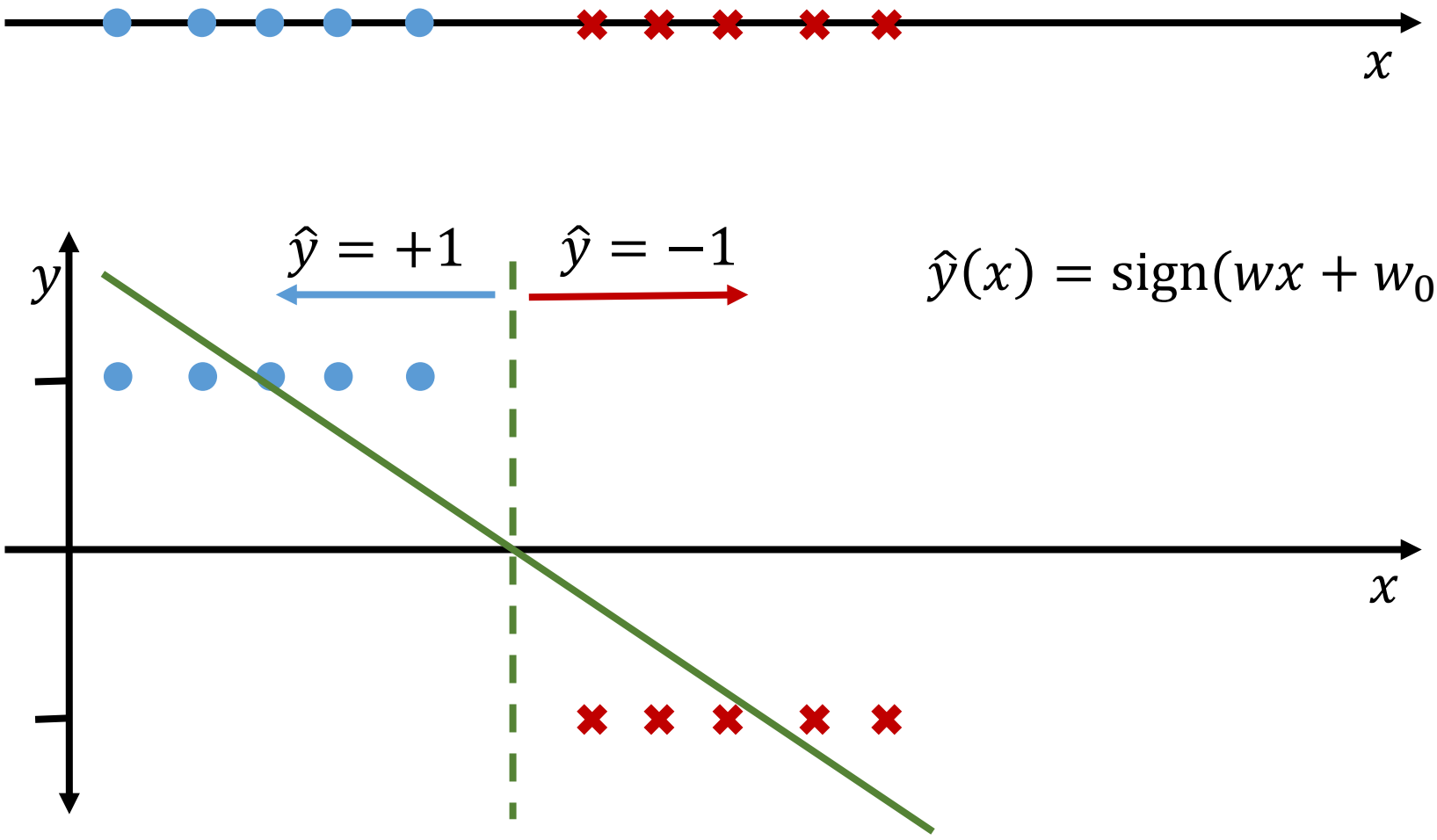
What if we ignore above and solve classification using regression?

Classification as regression

- Binary classification $\mathcal{Y} = \{-1,1\}$ and $\mathcal{X} \in \mathbb{R}^d$
- Treat it as regression with squared loss, say linear regression
  - Training data $S = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right): i = 1,2, \ldots, N\right\}$
  - ERM

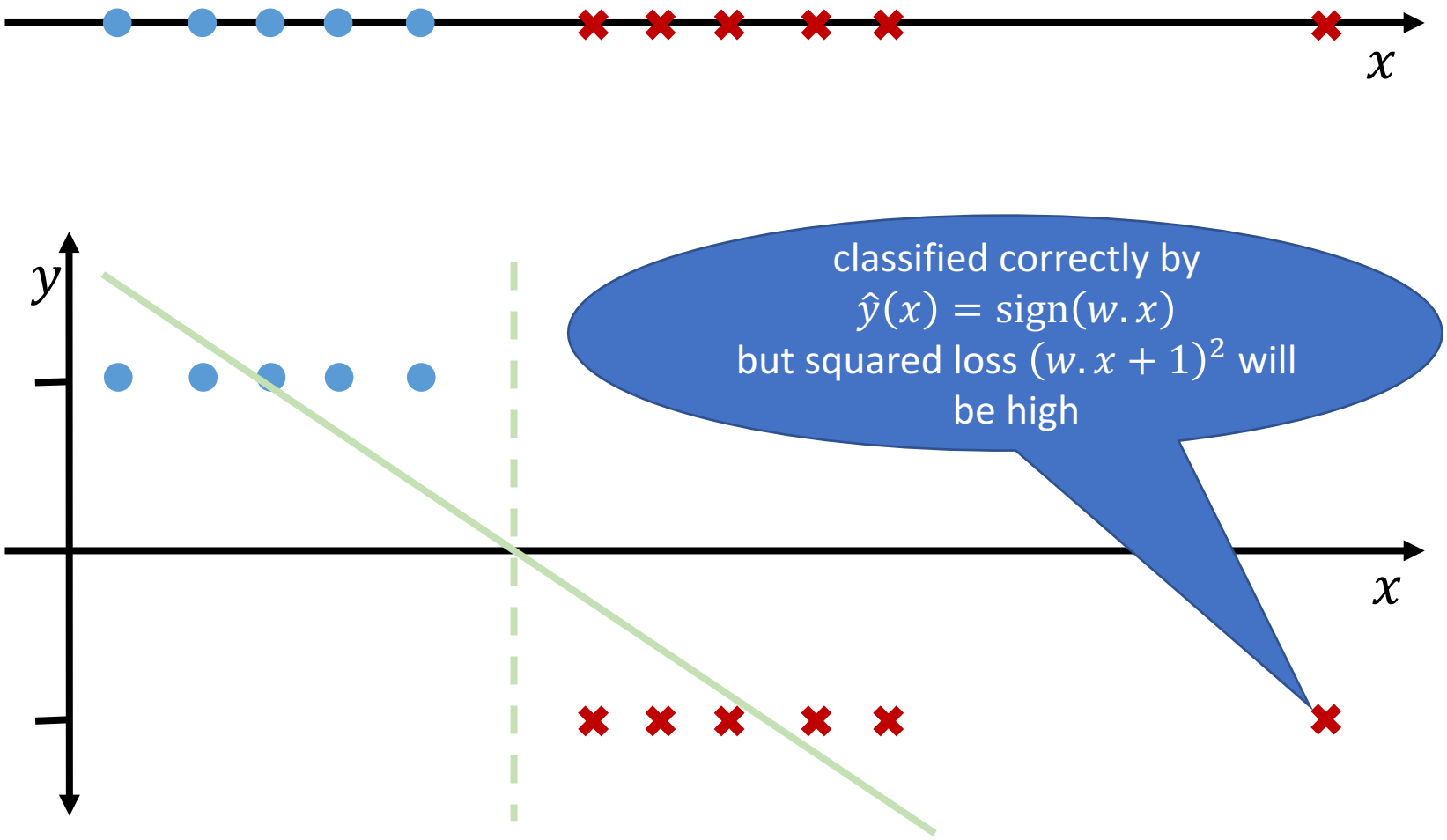
$$\widehat{\boldsymbol{w}}, \widehat{w}_0 = \underset{w, w_0}{\operatorname{argmin}} \sum_i \left(\boldsymbol{w}.\boldsymbol{x}^{(i)} + w_0 - y^{(i)}\right)^2$$

Classification as regression



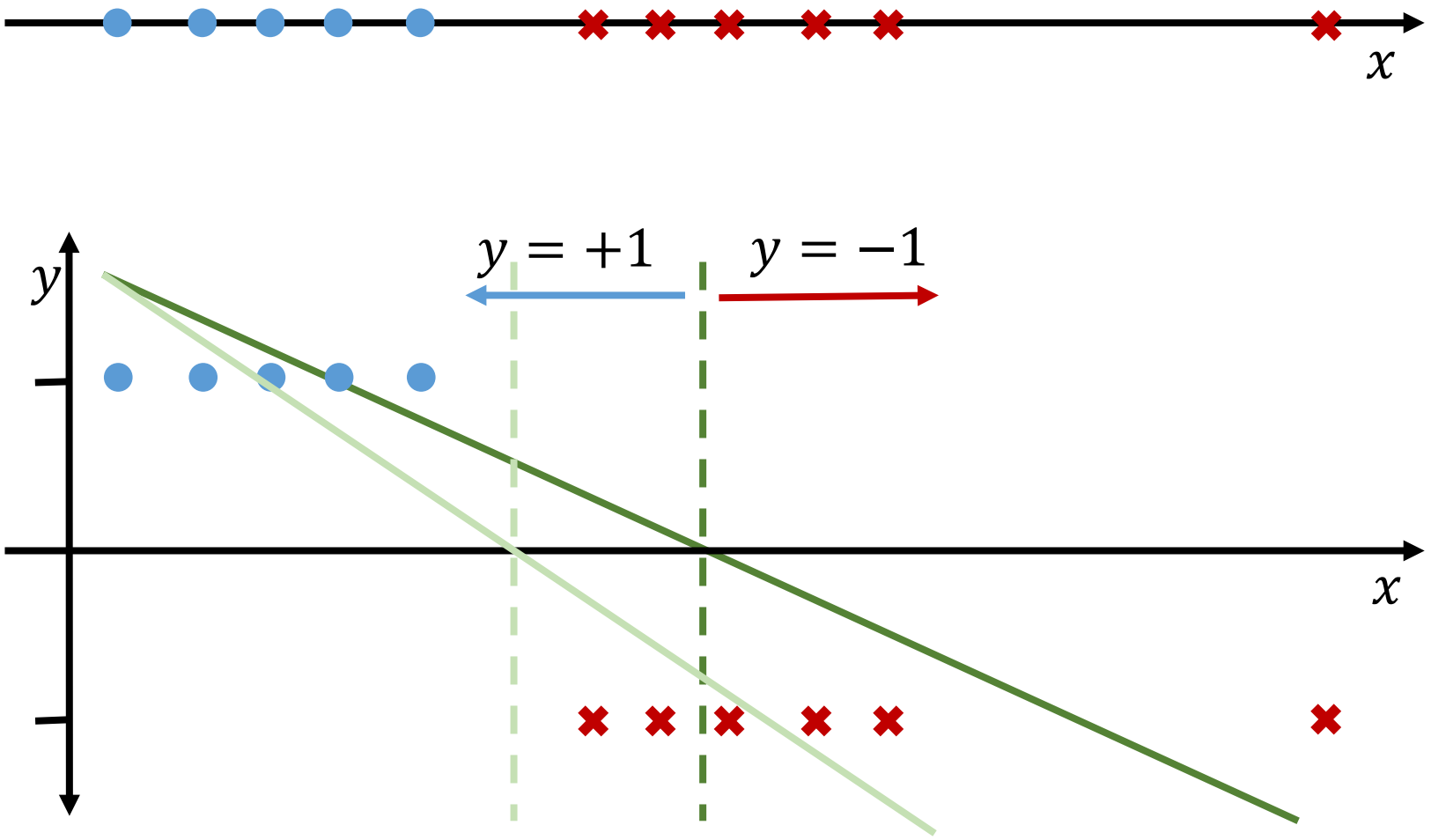
Example credit: Greg Shaknarovich

Classification as regression



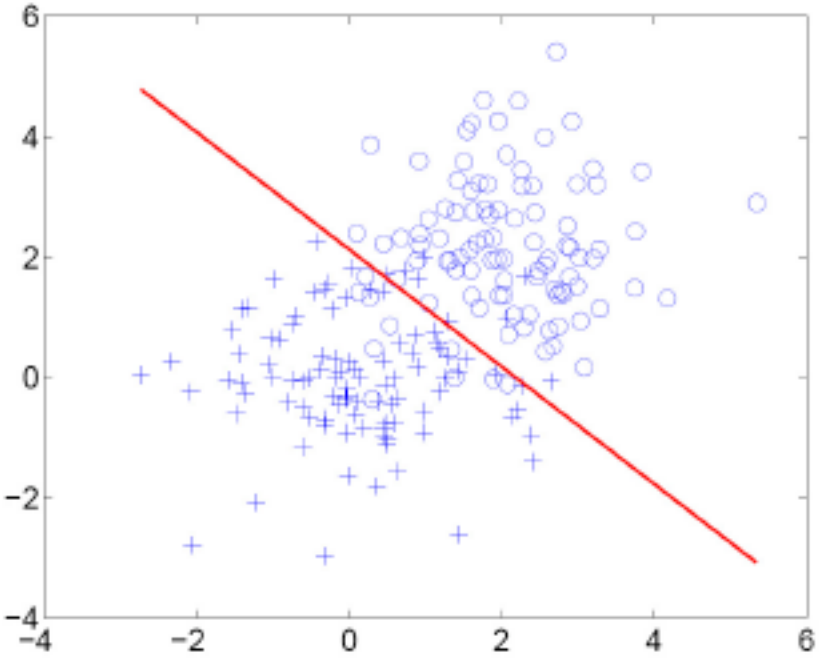
Example credit: Greg Shaknarovich

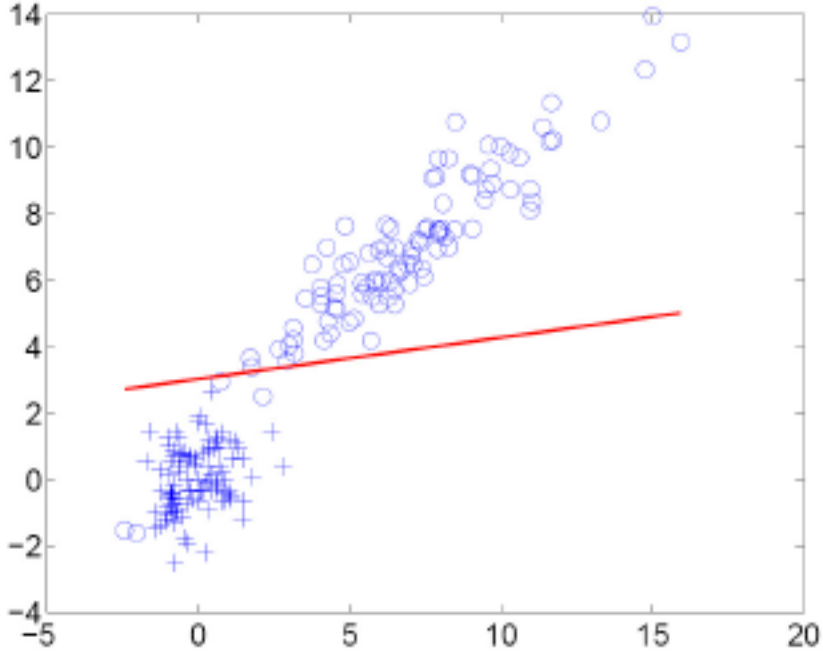
Classification as regression



Example credit: Greg Shaknarovich

Classification as regression



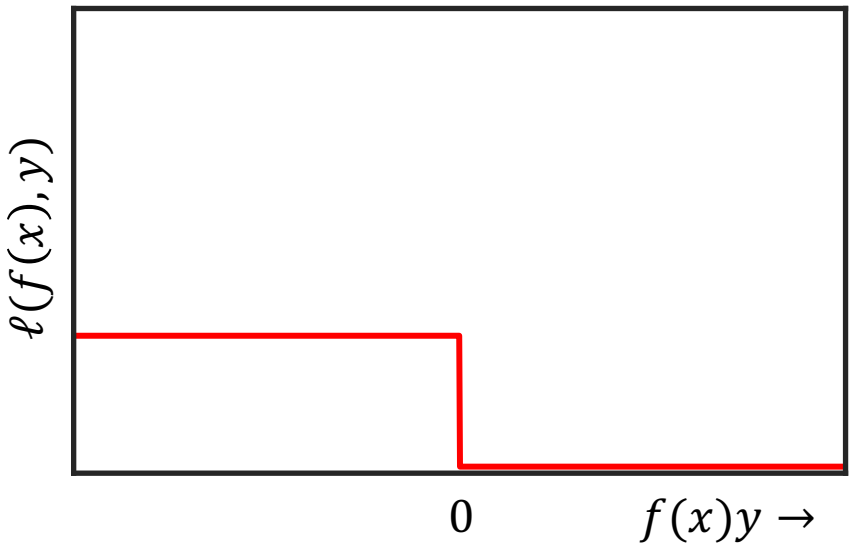


Slide credit: Greg Shaknarovich

Surrogate Losses

- The correct loss to use is 0-1 loss ***after*** thresholding

$$\ell^{01}(f(x), y) = \mathbf{1}\left[\text{sign}\left(f(x)\right) \neq y\right]$$
$$= \mathbf{1}[\text{sign}(f(x)y) < 0]$$

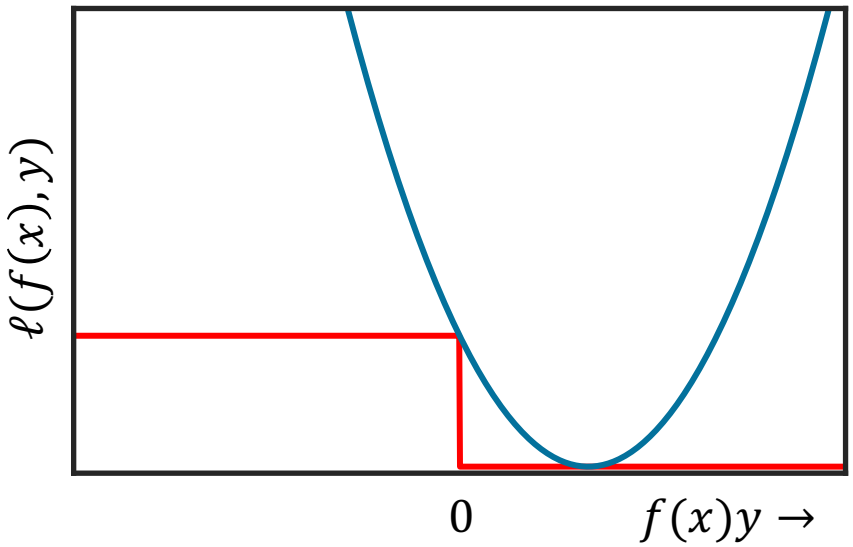


Surrogate Losses

- The correct loss to use is 0-1 loss ***after*** thresholding

$$\ell^{01}(f(x), y) = \mathbf{1}\left[\text{sign}\left(f(x)\right) \neq y\right]$$
$$= \mathbf{1}[\text{sign}(f(x)y) < 0]$$

- Linear regression uses $\ell^{LS}(f(x), y) = (f(x) - y)^2$



- Why not do ERM over $\ell^{01}(f(x), y)$ directly?
  - non-continuous, non-convex

Surrogate Losses

- Hard to optimize over ℓ^{01}, find another loss $\ell(\hat{y}, y)$
  - Convex (for any fixed y) → easier to minimize
  - An upper bound of ℓ^{01} → small $\ell \Rightarrow$ small ℓ^{01}
- Satisfied by squared loss

→but has "large" loss even when $\ell^{01}(\hat{y}, y) = 0$

- Two more surrogate losses in in this course
  - Logistic loss

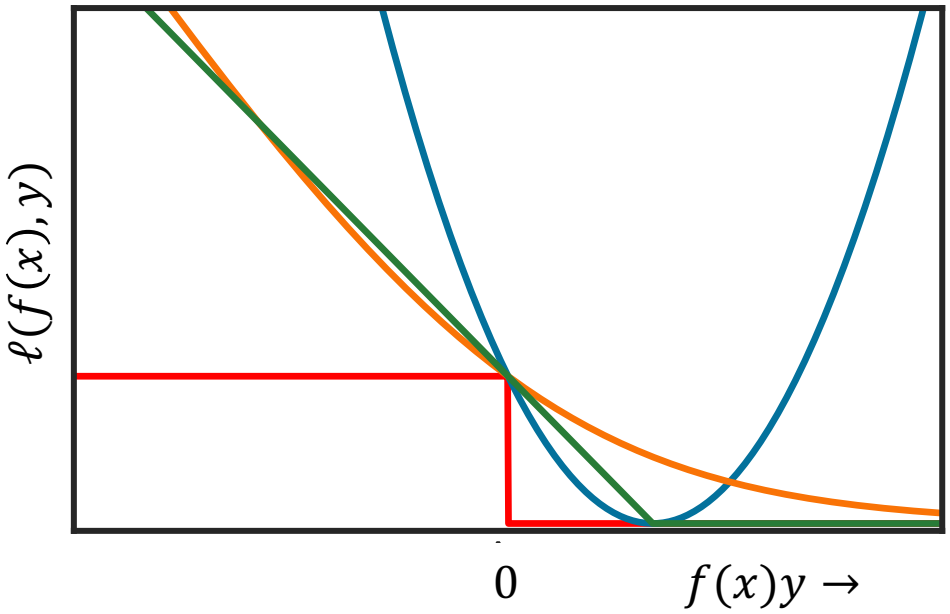
$\ell^{\log}(\hat{y}, y) = \log(1 + \exp(-\hat{y}y))$

(TODAY)

  - Hinge loss

$\ell^{hinge}(\hat{y}, y) = \max(0, 1 - \hat{y}y)$

(TOMORROW)

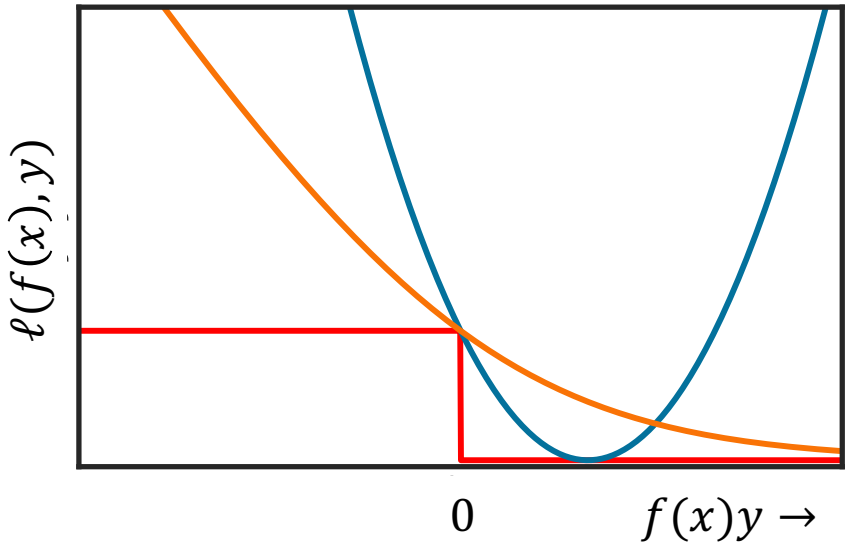


Logistic Regression

Logistic regression: ERM on surrogate loss

Logistic loss

$$\ell(f(x), y) = \log\left(1 + \exp(-f(x)y)\right)$$



- $S = \{(\boldsymbol{x}^{(i)}, y^{(i)}): i = 1,2, \ldots, N\}, \quad \mathcal{X} = \mathbb{R}^d, \quad \mathcal{Y} = \{-1,1\}$
- Linear model $f(\boldsymbol{x}) = f_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}.\boldsymbol{x} + w_0$
- Minimize training loss

$$\widehat{\boldsymbol{w}}, \widehat{w}_0 = \underset{\boldsymbol{w}, w_0}{\operatorname{argmin}} \sum_i \log\left(1 + \exp\left(-\left(\boldsymbol{w}.\boldsymbol{x}^{(i)} + w_0\right)y^{(i)}\right)\right)$$

- Output classifier $\hat{y}(\boldsymbol{x}) = \operatorname{sign}(\boldsymbol{w}.\boldsymbol{x} + w_0)$

Logistic regression

$$\hat{\boldsymbol{w}}, \hat{w}_0 = \underset{\boldsymbol{w}, w_0}{\operatorname{argmin}} \sum_i \log\left(1 + \exp\left(-\left(\boldsymbol{w}.\boldsymbol{x}^{(i)} + w_0\right) y^{(i)}\right)\right)$$

- Learns a linear decision boundary
  - $\{\boldsymbol{x}: \boldsymbol{w}.\boldsymbol{x} + w_0 = 0\}$ is a hyperplane in $\mathbb{R}^d$ - decision boundary
  - $\{\boldsymbol{x}: \boldsymbol{w}.\boldsymbol{x} + w_0 = 0\}$ divides $\mathbb{R}^d$ into two halfspace (regions)
  - $\{\boldsymbol{x}: \boldsymbol{w}.\boldsymbol{x} + w_0 \geq 0\}$ will get label +1 and $\{\boldsymbol{x}: \boldsymbol{w}.\boldsymbol{x} + w_0 < 0\}$ will get label -1

- Maps $\boldsymbol{x}$ to a 1D coordinate

$$x' = \frac{\boldsymbol{w}.\boldsymbol{x} + w_0}{\|\boldsymbol{w}\|}$$

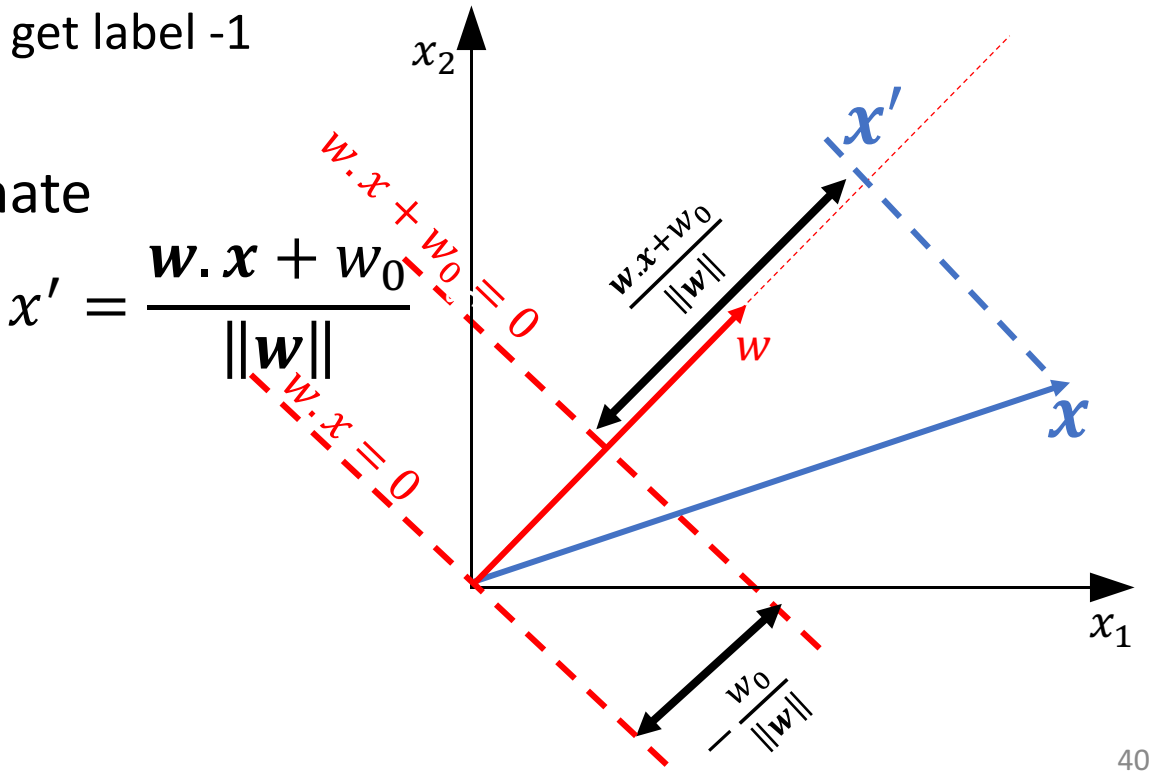


Figure credit: Greg Shaknarovich

Logistic Regression

$$\hat{\boldsymbol{w}}, \hat{w}_0 = \underset{\boldsymbol{w}, w_0}{\operatorname{argmin}} \sum_i \log\left(1 + \exp(-(\boldsymbol{w}.\boldsymbol{x} + w_0)y)\right)$$

- Convex optimization problem
- Can solve using gradient descent
- Can also add usual regularization: ℓ_2, ℓ_1
  - More details in the next session