Introduction to Machine Learning (IML)

LECTURE #9: REGRESSION

236756 – 2024 SPRING – TECHNION

LECTURER: NIR ROSENFELD

Today

- **Part III**: *more supervised learning*
    1. Regression (today)
    2. Bagging and boosting
    3. Deep learning

Learning setup

<u>**Classification:**</u>

- **Features:** $x \in \mathcal{X} = \mathbb{R}^d$
- **Labels**: $y \in \mathcal{Y} = \{\pm 1\}$
- **Sample set**: $S = \{(x_i, y_i)\}_{i=1}^{m} \overset{iid}{\sim} D^m$
- **Model class:** $H = \{h \mid h\colon \mathcal{X} \to \mathcal{Y}\}$

- **Expected error**:

  $L_D(h) = \mathbb{P}_D[y \neq h(x)]$

- **Empirical error**:

  $L_S(h) = \mathbb{P}_S[y \neq h(x)]$

Learning setup

Classification:

- **Features:** $x \in \mathcal{X} = \mathbb{R}^d$
- **Labels**: $y \in \mathcal{Y} = \{\pm 1\}$
- **Sample set**: $S = \{(x_i, y_i)\}_{i=1}^m \overset{iid}{\sim} D^m$
- **Model class:** $H = \{h \mid h: \mathcal{X} \to \mathcal{Y}\}$

- **Loss function:** $\ell^{0/1}(y, \hat{y}) = \mathbb{1}\{y \neq \hat{y}\}$
- **Expected error**:

  $L_D(h) = \mathbb{E}_D\left[\ell^{0/1}(y, h(x))\right]$
- **Empirical error**:

  $L_S(h) = \frac{1}{m}\sum_{i=1}^m \ell^{0/1}(y_i, h(x_i))$

Learning setup

~~**Classification:**~~ **Regression:**

- **Features:** $x \in \mathcal{X} = \mathbb{R}^d$
- **Labels:** $y \in \mathcal{Y} = \mathbb{R}$
- **Sample set:** $S = \{(x_i, y_i)\}_{i=1}^m \overset{iid}{\sim} D^m$
- **Model class:** $H = \{h \mid h: \mathcal{X} \to \mathcal{Y}\}$

- **Loss function:** $\ell^{0/1}(y, \hat{y}) = \mathbb{1}\{y \neq \hat{y}\}$
- **Expected error:**

  $L_D(h) = \mathbb{E}_D\left[\ell^{0/1}(y, h(x))\right]$
- **Empirical error:**

  $L_S(h) = \frac{1}{m}\sum_{i=1}^m \ell^{0/1}(y_i, h(x_i))$

Learning setup

~~Classification:~~ **Regression:**

- **Features:** $x \in \mathcal{X} = \mathbb{R}^d$
- **Labels**: $y \in \mathcal{Y} = \mathbb{R}$
- **Sample set**: $S = \{(x_i, y_i)\}_{i=1}^m \overset{iid}{\sim} D^m$
- **Model class:** $H = \{h \mid h: \mathcal{X} \to \mathcal{Y}\}$

- **Loss function:** $\ell^{0/1}(y, \hat{y}) = \mathbb{1}\{y \neq \hat{y}\}$
- **Expected error**:

  $L_D(h) = \mathbb{E}_D\left[\ell^{0/1}(y, h(x))\right]$
- **Empirical error**:

  $L_S(h) = \frac{1}{m}\sum_{i=1}^m \ell^{0/1}(y_i, h(x_i))$
- Possible, but somewhat silly:
  - penalizes harshly even tiny errors
  - same penalty regardless of error size
- Solution is fairly straightforward,

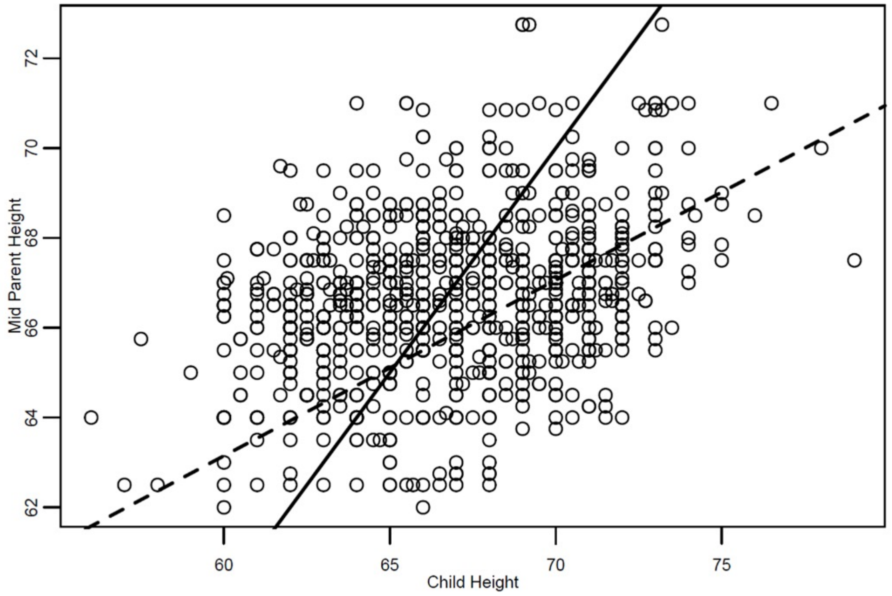
  but we'll take the long road there!

Regression via statistical modeling

History

- As old as time (Legendre 1805, Gauss 1809)
- **Why is it called "regression"?**
  - Galton (1889) and later Pearson (his student) observed that:
    - ➢ children of *tall* parents are *shorter* than their parents (on average)
    - ➢ children of *short* parents are *taller* than their parents (on average)
  - They concluded that observations have a statistical tendency to "regress" (i.e., "return") to the mean (following extreme observations)
- Regression is deeply rooted in statistics; basis for many statistical tasks (beyond prediction)
- Way before ML came to be (and was popularized through classification)
- Today we'll think about regression like *statisticians* would

$y = x$



Statistical modeling

- **Recall**: in prediction, we care about $p(y|x)$
- In discriminative learning, we've assumed $y \overset{iid}{\sim} D_{Y|X=x}$
- Alternatively, **statistical modeling** assumes:

random noise: unobserved, but sampled from assumed "error" distribution

$$y = f^*(x) + \epsilon, \qquad f^* \in F, \qquad \epsilon \sim D_{\text{ERR}}$$

true model: unknown, but from assumed model class

- This is a direct assumption on the data generating process
  (more concretely, on $p(y|x)$, with arbitrary $p(x)$; still assume x sampled iid)
- Seems strange from a purely discriminative perspective
  (we've worked hard to assume as little as possible!)
- **But remember**: regression emerged in statistics
  (where explicit assumptions are routine)
- Like everything, such assumptions have pros and cons
  (**example**: easier to reason about non-iid data by modeling correlations in noise)

Linear regression

- In **linear regression**, we assume:
  1. True model is linear: $f^* \in \{f(x) = w^\top x : w \in \mathbb{R}^d\} = F$
  2. Error distribution is normal: $\epsilon \overset{iid}{\sim} N(0, \sigma^2)$
     - Variance σ^2 is **unknown** but **uniform** across x ("homoscedastic")
  3. Features sampled from some (unknown) $p(x)$

- Conditional distributions become:

$$y = w^\top x + \epsilon, \quad \epsilon \overset{iid}{\sim} N(0, \sigma^2) \quad \Rightarrow \quad P(y|x; w) = N(w^\top x, \sigma^2) \quad \textit{(from linearity)}$$

- **Interpretation**:
  - Relationship between x and y is, in principle, linear
  - Observations corrupted by many additive noise (e.g., a sum of many small errors + CLT = Normal!)
- **Take away**: structural assumptions are ok when we know something about the problem (conversely, with knowledge, not assuming anything (except iid) can be suboptimal)

Learning

density

- Just made a *parametric* distributional assumption: $P(y|x;w) = N(w^\top x, \sigma^2)$ for some w
- **Q**: How can we use this to derive an appropriate learning objective?
- **A**: Maximize data *likelihood*

- **Observation**: two ways to interpret $P(y|x;w)$
  - as function of data: given w, what is the probability of observing y given x?
  - as function of parameters: given (x, y), what it is the likelihood it was generated by w?
- More generally, given sample set $S = \{(x_i, y_i)\}_{i=1}^m$, can define (as a function of w):

Likelihood: $L(w;S) := P(S;w)$ $= P\big((x_1, y_1), \dots, (x_m, y_m); w\big)$

- **Idea**: learn w that "best explains" data:

Maximum-likelihood estimation (MLE): $\hat{w} = w_{\text{MLE}} = \underset{w \in \mathbb{R}^d}{\operatorname{argmax}} L(w;S)$

Maximum likelihood estimation

$w_{\mathrm{MLE}} = \mathrm{argmax}_w\, L(w; S)$

$= \mathrm{argmax}_w P(S; w)$ *definition*

$= \mathrm{argmax}_w \prod_i P(y_i, x_i; w)$ *iid*

$= \mathrm{argmax}_w \prod_i P(y_i | x_i; w) P(x_i | w)$ *chain rule*

$= \mathrm{argmax}_w \prod_i P(y_i | x_i; w) P(x_i)$ *x indep. of w*

$= \mathrm{argmax}_w \prod_i P(y_i | x_i; w)$ *$\prod_i P(x_i)$ is scalar*

$= \mathrm{argmax}_w \log \prod_i P(y_i | x_i; w)$ *log preserves argmax*

$= \mathrm{argmax}_w \sum_i \log P(y_i | x_i; w)$ $\log \Pi = \Sigma \log$

$= \mathrm{argmax}_w \sum_i \log \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(w^\top x_i - y_i)^2}{2\sigma^2}}$ $y|x \sim N(w^\top x, \sigma^2)$

$= \mathrm{argmax}_w \sum_i \left(\log \frac{1}{\sqrt{2\pi\sigma^2}} + \log e^{-\frac{(w^\top x_i - y_i)^2}{2\sigma^2}} \right)$

$= \mathrm{argmax}_w - \frac{1}{2\sigma^2} \sum_i (w^\top x_i - y_i)^2$ *indep. of w; $\log e^z = z$*

$\boxed{= \mathrm{argmin}_w \frac{1}{m} \sum_i (w^\top x_i - y_i)^2}$ *negate; scalar prod.*

$= \mathrm{argmin}_w \frac{1}{m} \sum_i \ell^{\mathrm{sqr}} (y_i, w^\top x_i)$

$$w_{\mathrm{MLE}} = \underset{w \in \mathbb{R}^d}{\mathrm{argmax}}\, L(w; S) \;=\; \underset{w \in \mathbb{R}^d}{\mathrm{argmax}} \log L(w; S) \;=\; \underset{w \in \mathbb{R}^d}{\mathrm{argmin}} - \log L(w; S)$$

likelihood *log-likelihood* *negative log-likelihood (NLL)*

Optimization

- MLE objective for linear regression is **Ordinary Least Squares**:

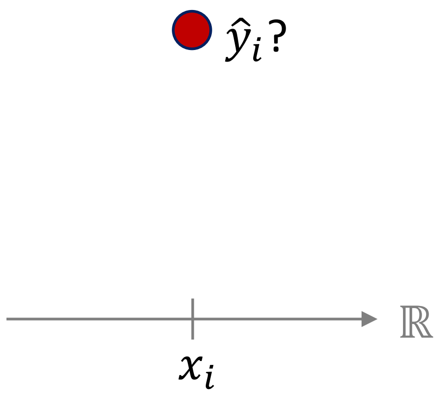
$$\underset{w \in \mathbb{R}}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} (y_i - w^\top x_i)^2$$

- **Possible ways to optimize:** (will not go in depth here)
  1. Gradient descent!
     (objective is convex – think why)
  2. Closed form
     (when $X^\top X$ is invertible; as seen in numerical algorithms)
  3. Using SVD
     (when $X^\top X$ is non-invertible ; as seen in numerical algorithms)

The story behind least squares

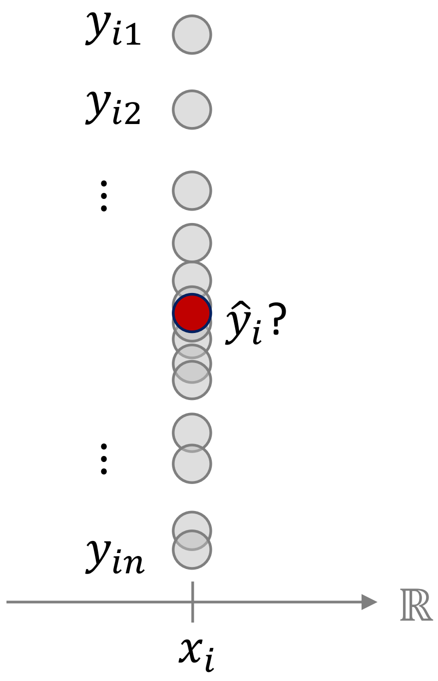
Interpretation

- Consider some x_i
- What would be a good prediction $\hat{y}_i$?



Interpretation

- Consider some x_i
- What would be a good prediction $\hat{y}_i$?
- To simplify, first assume we've observed multiple $y_{ij} \sim p(y|x_i)$ for this particular x_i
- What would be a good prediction now?



Interpretation

- **Puzzle**:
  Let $y_1, \dots, y_n \in \mathbb{R}$.
  What is $a \in \mathbb{R}$ which minimizes mean squared distances, $\frac{1}{n}\sum_i (y_i - a)^2$?

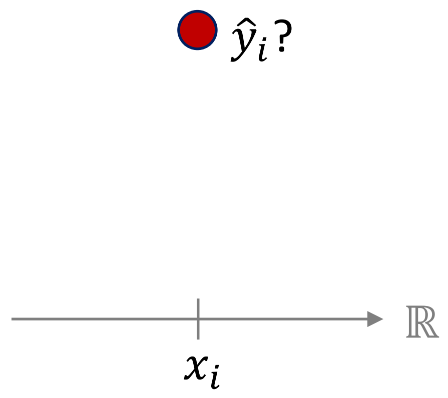
$$\frac{1}{n}\sum_i (y_i - a)^2 = \frac{1}{n}\sum_i y_i^2 - 2a\frac{1}{n}\sum_i y_i + a^2$$
$$= \bar{\bar{y}} - 2a\bar{y} + a^2 = \cdots$$

$$\dots = a^2 - 2a\bar{y} + \bar{y}^2 - \bar{y}^2 + \bar{\bar{y}}$$
$$= (a - \bar{y})^2 - \bar{y}^2 + \bar{\bar{y}}$$

- **Solution**: mean squared distances are minimized by the average, $\bar{y}$
- The average $\bar{y}$ is a **statistic**: a useful way to summarize an entire sample as a single scalar

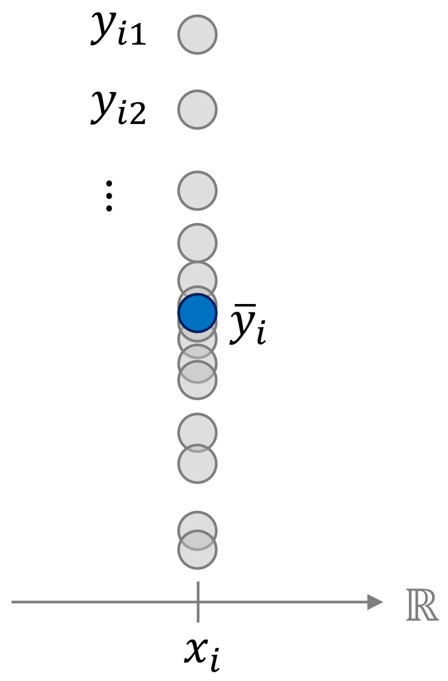
Interpretation

- Back to our x_i
- **Q**: What would be a good prediction $\hat{y}_i$?



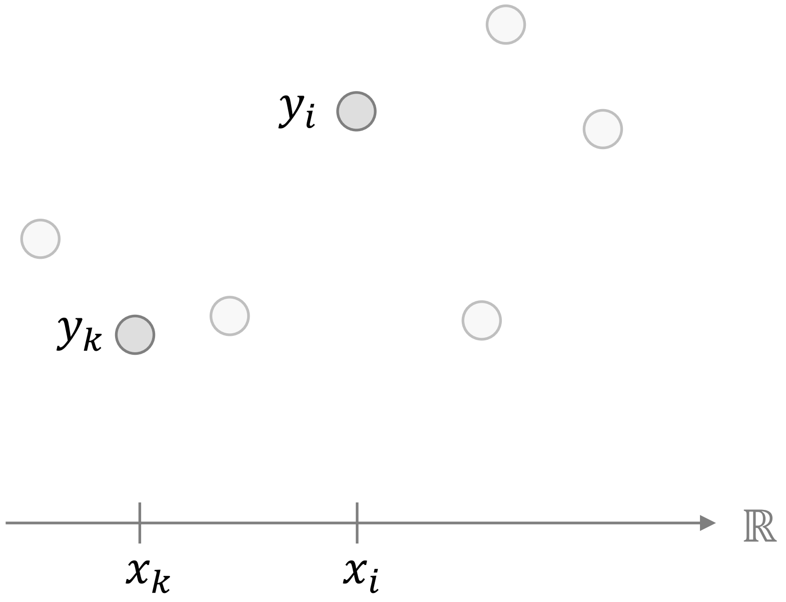
Interpretation

- Back to our x_i
- **Q**: What would be a good prediction $\hat{y}_i$?
- **A**: If we observed multiple $y_{ij} \sim p(y|x_i)$, then $\bar{y}_i$ would be a good prediction
- Unfortunately:
  - At *train time*, we only see <u>one</u> y_i per x_i
  - At *test time*, we don't see <u>any</u> y_i-s!



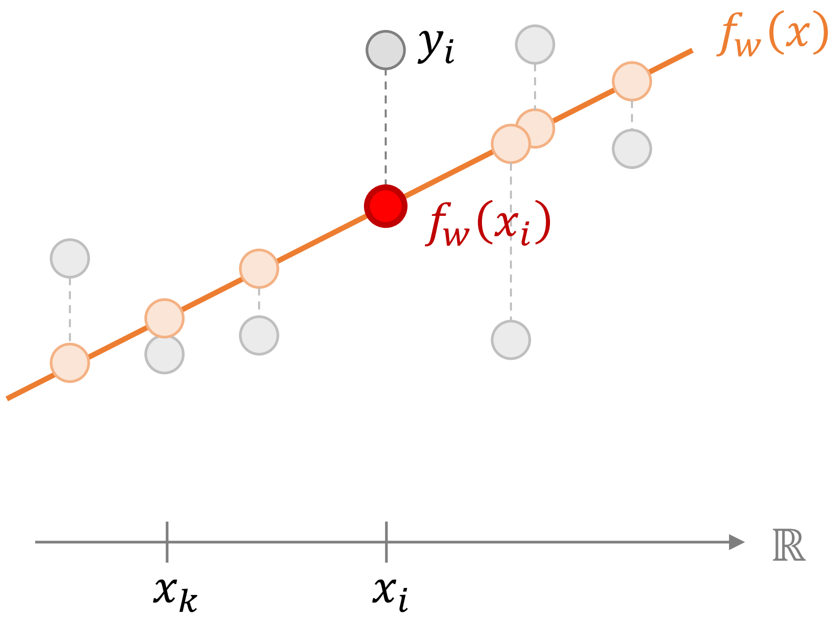
Interpretation

- Fortunately, training data includes *multiple* x_i, each with it's own (single) $y_i \sim P(y|x_i)$



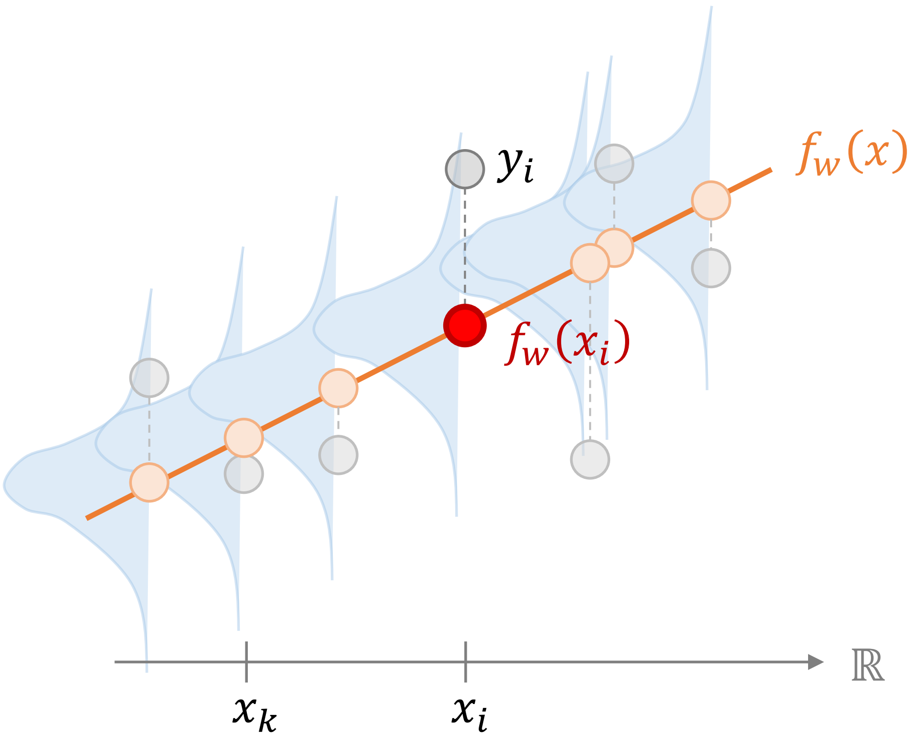
Interpretation

- Fortunately, training data includes *multiple* x_i, each with it's own (single) $y_i \sim P(y|x_i)$
- Linear regression allows us to share label information across examples
- By assuming a parametric model $f_w(x) = w^\top x$, we can estimate $\bar{y}_i$ for x_i using all <u>other</u> (x_j, y_j)
- Minimizing mean squared errors $(w^\top x - y)^2$ means we aim for the "line" $f_w(x) = w^\top x$ to pass through all true averages $\bar{y}_i$



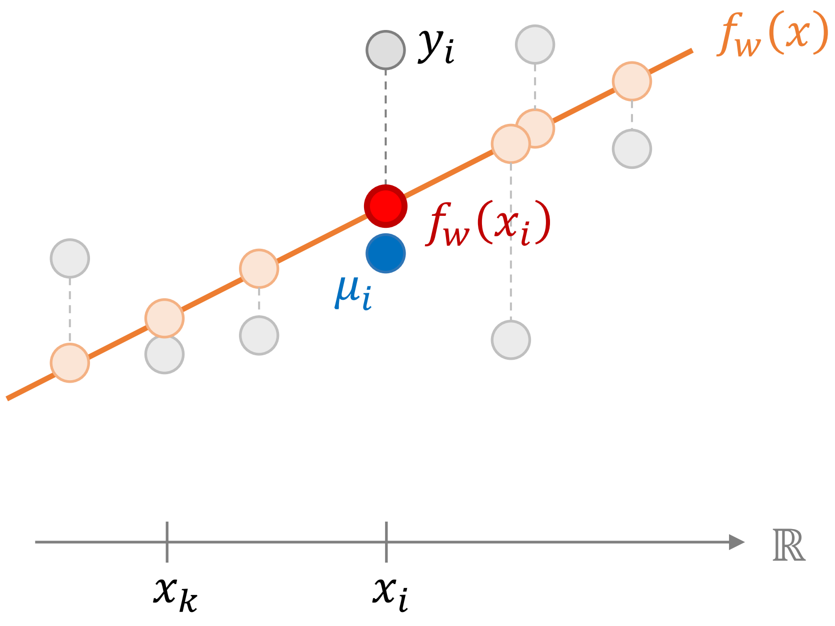
Interpretation

- Fortunately, training data includes *multiple* x_i each with it's own (single) $y_i \sim P(y|x_i)$
- Linear regression allows us to share label information across examples
- By assuming a parametric model $f_w(x) = w^\top x$, we can estimate $\bar{y}_i$ for x_i using all <u>other</u> (x_j, y_j)
- Minimizing mean squared errors $(w^\top x - y)^2$ means we aim for the "line" $f_w(x) = w^\top x$ to pass through all true averages $\bar{y}_i$
- This comes from our assumption: $y \sim N(\mu = w^\top x, \sigma^2)$



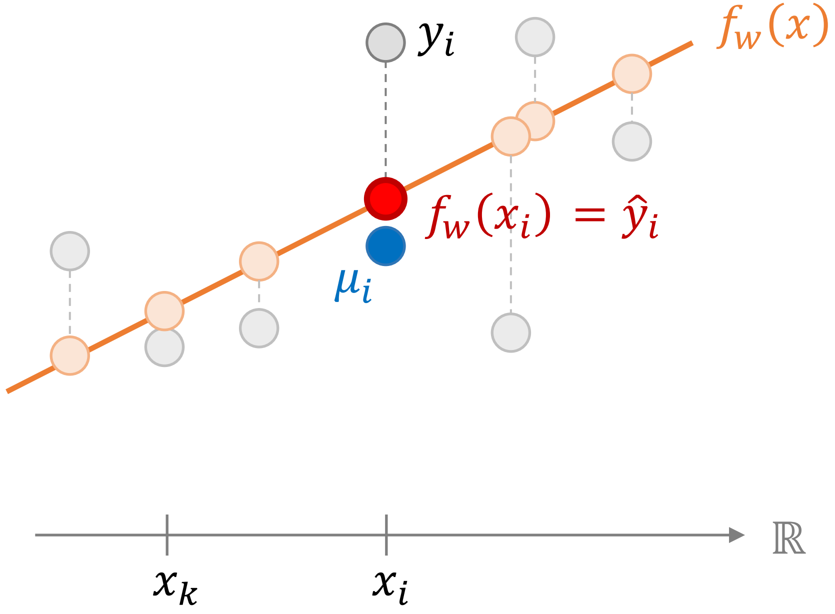
Interpretation

- Fortunately, training data includes *multiple* x_i each with it's own (single) $y_i \sim P(y|x_i)$
- Linear regression allows us to share label information across examples
- By assuming a parametric model $f_w(x) = w^\top x$, we can estimate $\bar{y}_i$ for x_i using other (x_j, y_j)
- Minimizing mean squared errors $(w^\top x - y)^2$ means we aim for the "line" $f_w(x) = w^\top x$ to pass through all true averages $\bar{y}_i$
- This comes from our assumption: $y \sim N(\mu = w^\top x, \sigma^2)$
- $f_w(x) \approx \bar{y}$ only *estimates* μ since we learn w from finite data



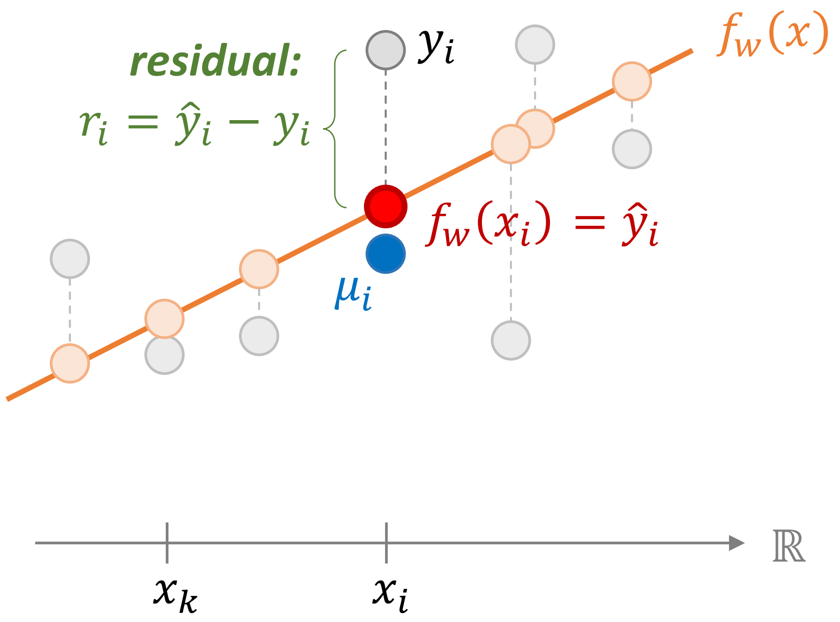
Interpretation

- Estimates become **predictions**: $\hat{y}_i = f_w(x_i) \approx \bar{y}_i$
- Ideally, we'd like $\mathbb{E}[\hat{y}] = \mathbb{E}[\bar{y}] = \mu$ for new (x, y)
- Hope is to approach this as m increases
  (i.e., more training examples (x_i, y_i))
- Won't work if we made the wrong assumptions!



Interpretation

- Estimates become **predictions**: $\hat{y}_i = f_w(x_i) \approx \bar{y}_i$
- Ideally, we'd like $\mathbb{E}[\hat{y}] = \mathbb{E}[\bar{y}] = \mu$ for new (x, y)
- Hope is to approach this as m increases
  (i.e., more training examples (x_i, y_i))
- Won't work if we made the wrong assumptions!

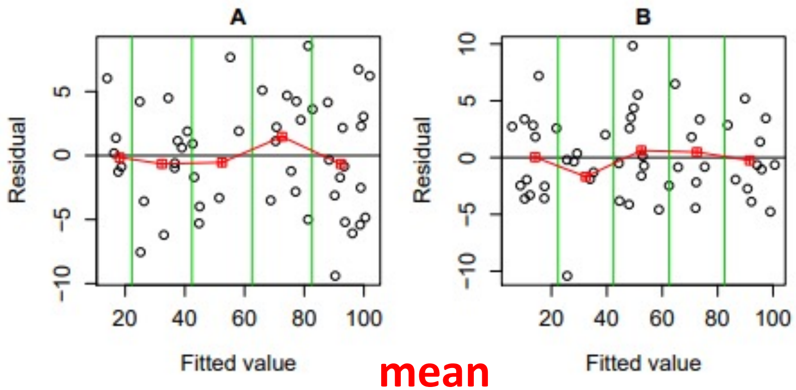


- ***Residual analysis*** can help us understand (in hindsight) if our assumptions were sensible
- Prediction "errors" are called **residuals:** $r_i = \hat{y}_i - y_i$
- Note "$\mathbb{E}[\hat{y}] = \mathbb{E}[\bar{y}]$?" is like asking "$\mathbb{E}[r] = 0$?"

Residuals

- Recall our assumptions: $y = w^\top x + \epsilon, \quad \epsilon \overset{iid}{\sim} N(0, \sigma^2)$

linear



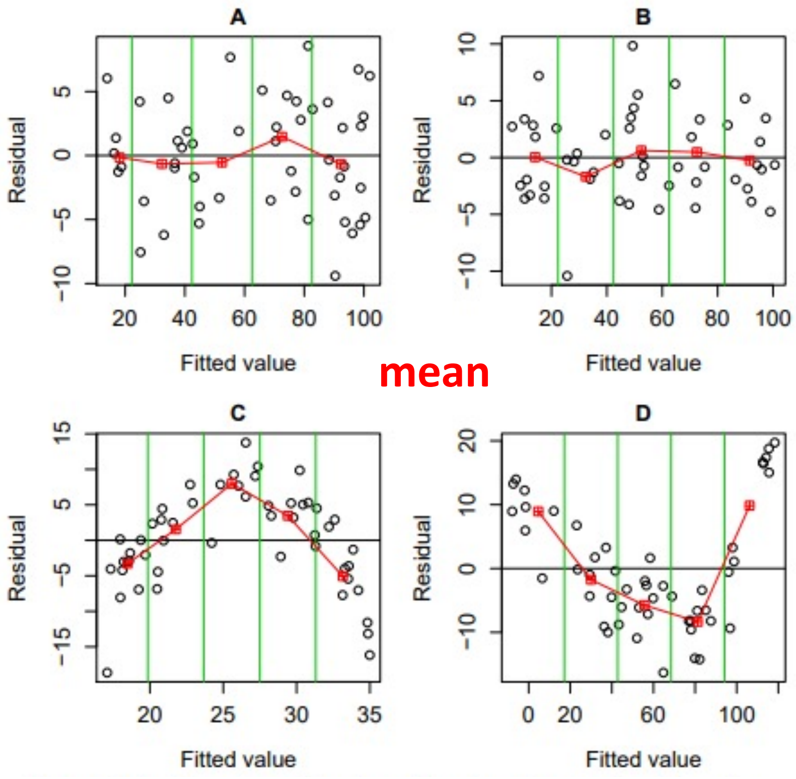
mean

Residuals

- Recall our assumptions: $y = w^\top x + \epsilon, \quad \epsilon \overset{iid}{\sim} N(0, \sigma^2)$

linear

non-linear



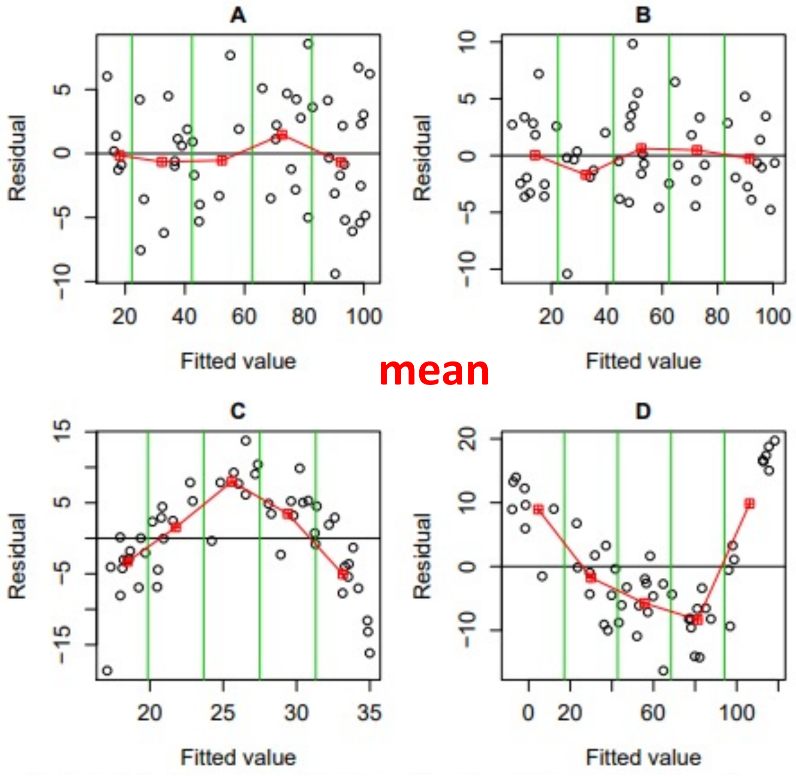
https://www.stat.cmu.edu/~hseltman/309/Book/chapter9.pdf

Residuals

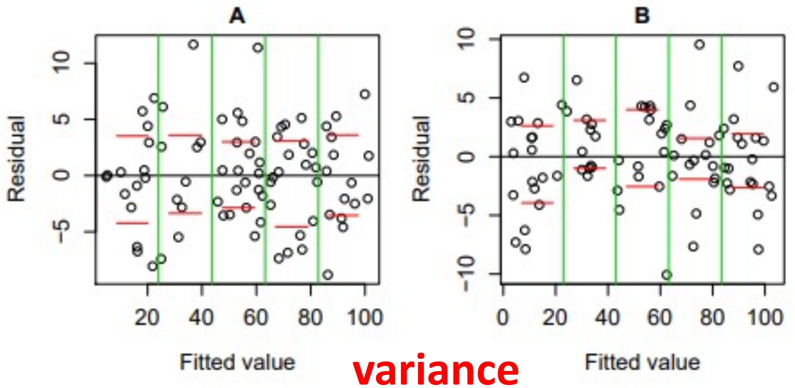
- Recall our assumptions: $y = w^\top x + \epsilon, \quad \epsilon \overset{iid}{\sim} N(0, \sigma^2)$

linear

non-linear



mean



variance

homoscedastic

https://www.stat.cmu.edu/~hseltman/309/Book/chapter9.pdf

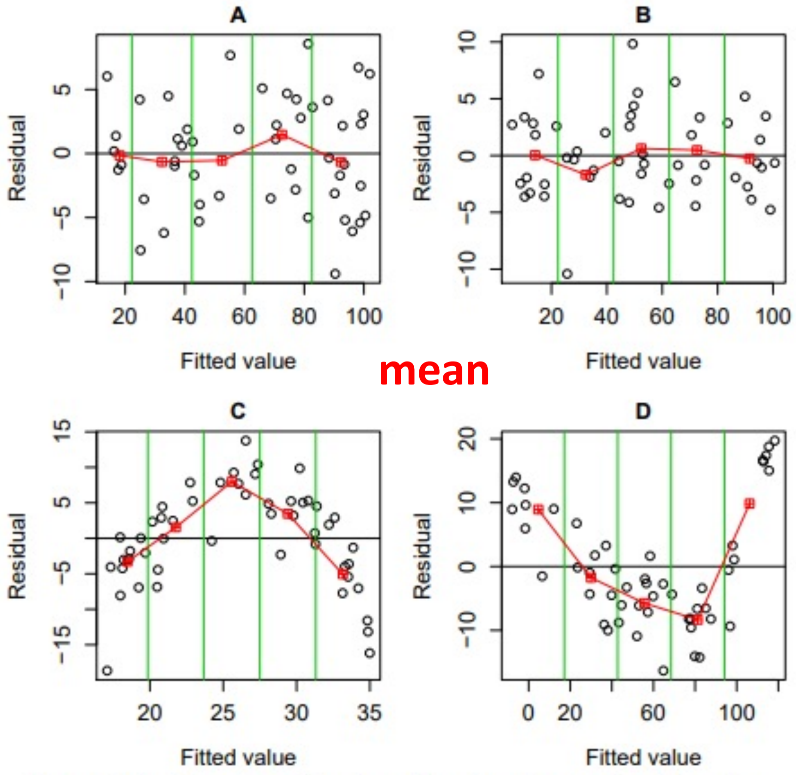
Residuals

- Recall our assumptions: $y = w^\top x + \epsilon,\ \ \epsilon \overset{iid}{\sim} N(0, \sigma^2)$

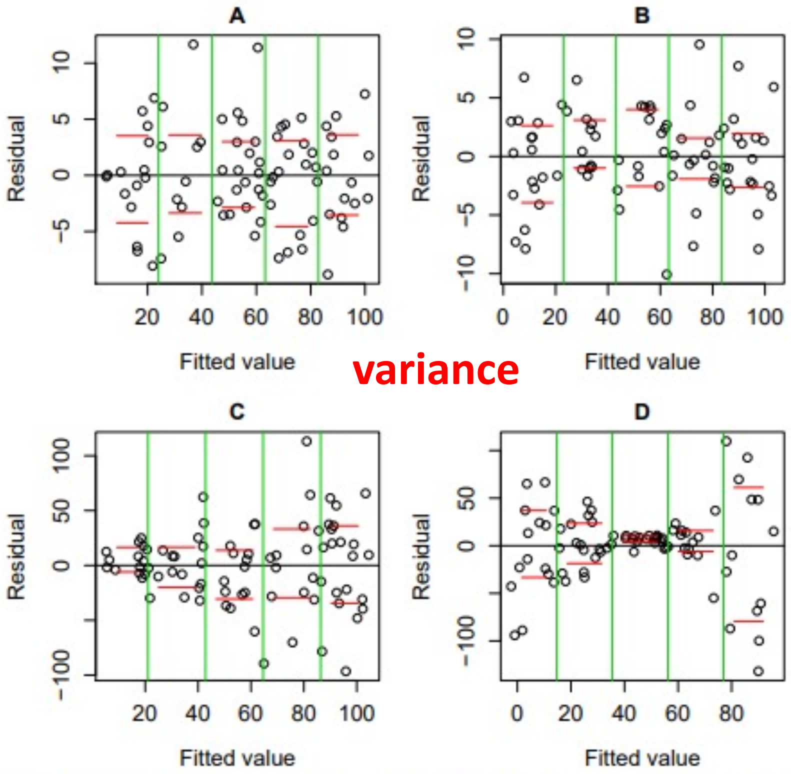
linear

non-linear

mean



variance



homoscedastic

heteroscedastic

- Residuals will play big role next week!

https://www.stat.cmu.edu/~hseltman/309/Book/chapter9.pdf

Regression as loss minimization

Regression $(y \in \mathbb{R})$

MLE objective:

$$\underset{f \in F}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\left(y_i - f(x_i)\right)^2}_{= \ell^{\mathrm{sqr}}(y, f(x))}$$

Classification $(y \in \{\pm 1\})$

ERM objective:

$$\underset{h \in H}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\mathbb{1}\{y_i \neq h(x_i)\}}_{= \ell^{0/1}(y, h(x))}$$

Regression ($y \in \mathbb{R}$)

~~MLE~~ ERM objective!

$$\underset{f \in F}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\left(y_i - f(x_i)\right)^2}_{= \ell^{\text{sqr}}\left(y, f(x)\right)}$$

Classification ($y \in \{\pm 1\}$)

ERM objective:

$$\underset{h \in H}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\mathbb{1}\{y_i \neq h(x_i)\}}_{= \ell^{0/1}\left(y, h(x)\right)}$$

- Squared error makes sense because:
  - correct prediction $\Rightarrow$ loss=0
  - loss gradually increases with distance $y - \hat{y}$
  - symmetric: $\ell(y, y + a) = \ell(y, y - a)$
- Regression vs. classification – **just different losses!**
- (Or is it?)

Regression ($y \in \mathbb{R}$)

~~MLE~~ ERM objective!

$$\underset{f \in F}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\left(y_i - f(x_i)\right)^2}_{= \ell^{\text{sqr}}(y, f(x))}$$

Classification ($y \in \{\pm 1\}$)

ERM objective:

$$\underset{h \in H}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\mathbb{1}\{y_i \neq h(x_i)\}}_{= \ell^{0/1}(y, h(x))}$$

- Fundamental difference: f is scalar, h is binary

Regression ($y \in \mathbb{R}$)

~~MLE~~ ERM objective!

$$\underset{f \in F}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\left(y_i - f(x_i)\right)^2}_{= \ell^{\text{sqr}}(y, f(x))}$$

Classification ($y \in \{\pm 1\}$)

ERM objective:

$$\underset{f \in F}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\mathbb{1}\{y_i \neq \operatorname{sign}(f(x_i))\}}_{= \ell^{0/1}(y, f(x))}$$

- Fundamental difference: f is scalar, h is binary
- But…
  - most classifiers we've considered are based on scalar functions

Regression ($y \in \mathbb{R}$)

~~MLE~~ ERM objective!

$$\underset{f \in F}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\left(y_i - f(x_i)\right)^2}_{= \ell^{\text{sqr}}(y, f(x))}$$

Classification ($y \in \{\pm 1\}$)

ERM objective:

$$\underset{f \in F}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \underbrace{\max\{0, 1 - y_i f(x_i)\}}_{= \ell^{\text{hinge}}(y, f(x))}$$

- Fundamental difference: f is scalar, h is binary
- But…
  - most classifiers we've considered are based on scalar functions
  - and what we actually optimize is a continuous proxy loss
- **Surprise: we've been doing regression all along!**

Classification vs. regression

- On its face, difference appears minor:
  - **Classification**: $y \in \{\pm 1\} \;\Rightarrow\; \ell^{0/1}$
  - **Regression**: $y \in \mathbb{R} \;\Rightarrow\; \ell^{\mathrm{sqr}}$

  (and most learning textbooks end the discussion here)
- But digging deeper, it turns out that:
  - some things remain exactly the same
  - some things differ but are easy to adapt
  - some things are fundamentally different
- Let's see *what* changes, *how*, and *why* (and what to do about it!)

1) Losses and proxies

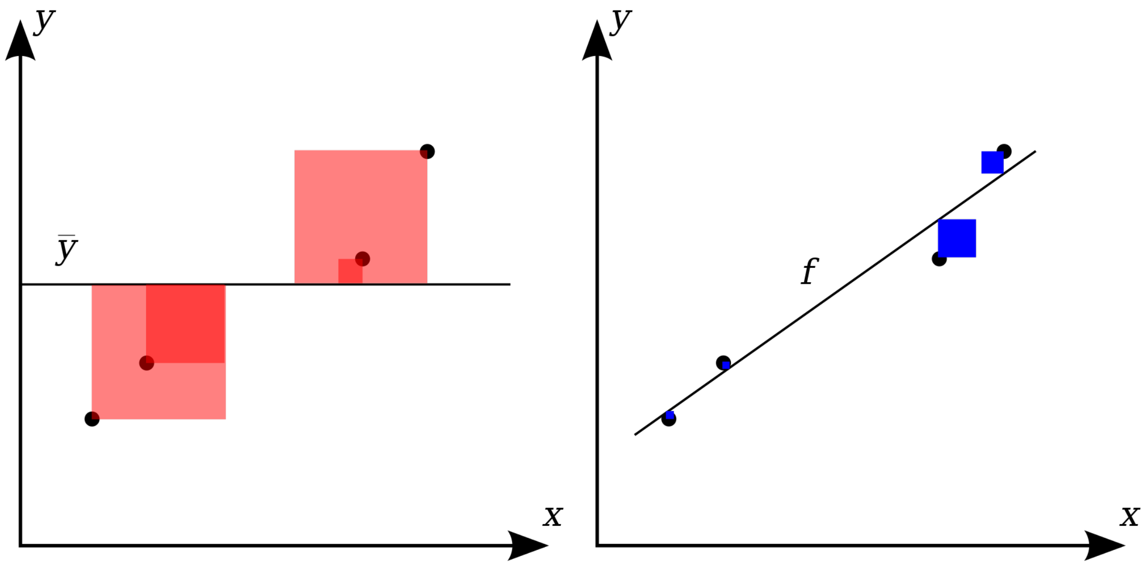
- **Classification**: optimize hinge loss, want 0/1 loss
- **Regression**: optimize squared loss, want… squared loss?

- In regression we typically don't need a proxy loss
- Problem is natively continuous! (this is good news)
- Crux is that there is no "natural" performance measure
  (i.e., why $(y - \hat{y})^2$, and not $(y - \hat{y})^4$? or $|y - \hat{y}|$? or $|y - \hat{y}|^{1.5}$? or $|y - \hat{y}|^{0.5}$? or …)
- Statistical modeling gives partial answer
  (that is not entirely satisfactory from a discriminative perspective)
- But still raises the issue – *how should we interpret performance results*?

2) The meaning of error

- **Classification**: $0/1\text{-error} = 0.08 \quad \Rightarrow \quad$ 92% future predictions are correct
- **Regression**: $\text{sqr-error} = 0.08 \quad \Rightarrow \quad$???
- **Q**: How can we
  - *interpret evaluated performance*?
  - *meaningfully compare performance?*
- **A**: $R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} = 1 - \frac{SS_{\text{RES}}}{SS_{\text{TOT}}}$

  $= \frac{1}{m} \sum_i y_i$



- Measures how much of the variance the model can explain by using the features x_i
  (think of $\bar{y}$ as "best guess" if you observed only y-s but no x-s)
- (Recall that even 0-1 error is not always meaningful (e.g., imbalanced data) and other measures are needed)

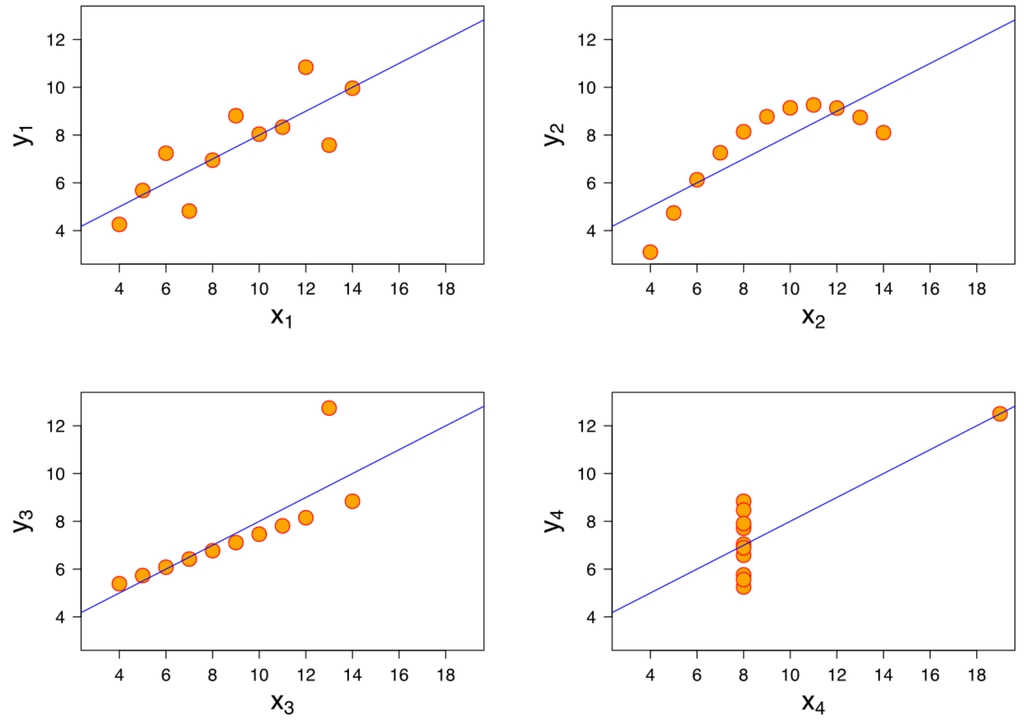
https://en.wikipedia.org/wiki/Coefficient_of_determination

2) The meaning of error

- **Classification**: $0/1\text{-error} = 0.08 \;\Rightarrow\;$ 92% future predictions are correct
- **Regression**: $\text{sqr-error} = 0.08 \;\Rightarrow\;$???
- **Q**: How can we
  - *interpret evaluated performance*?
  - *meaningfully compare performance?*
- **A**: $R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} = 1 - \frac{SS_{\mathrm{RES}}}{SS_{\mathrm{TOT}}}$

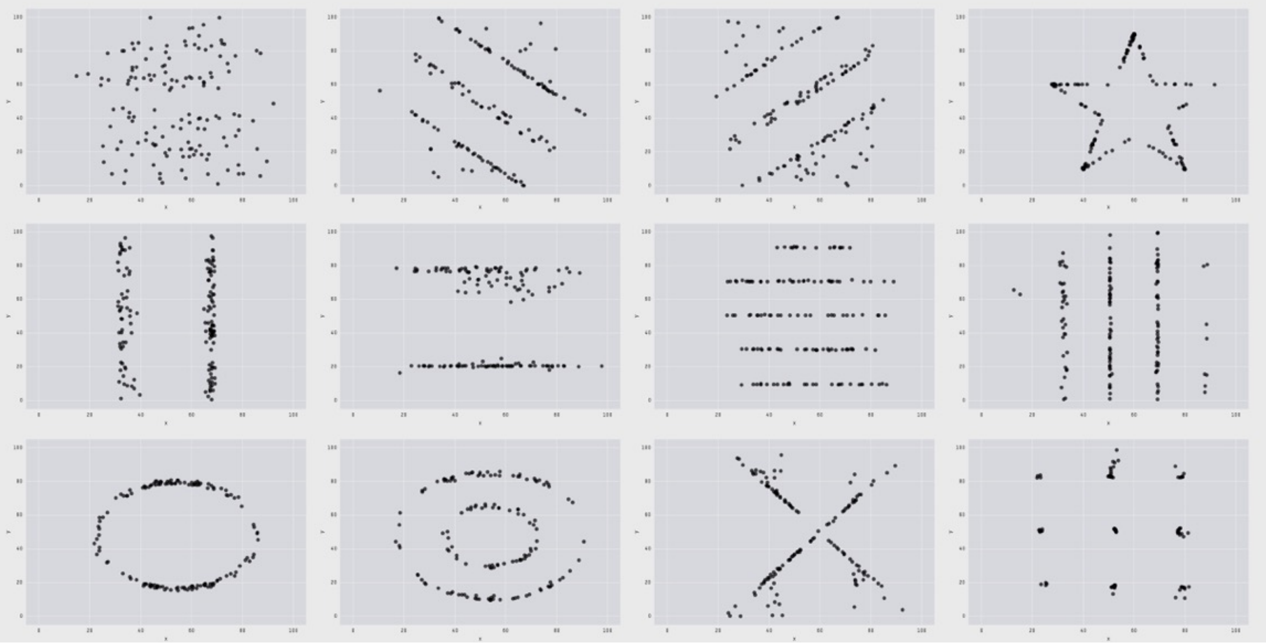
  $\bar{y} = \frac{1}{m}\sum_i y_i$
- Measures how much of the variance the model can explain by using the features x_i (think of $\bar{y}$ as "best guess" if you observed only y-s b
- (Recall that even 0-1 error is not always meaningful (e.g., imbalanced data) and other measures are needed)

But even R^2 has its limits...



https://en.wikipedia.org/wiki/Anscombe%27s_quartet

X Mean: 54.26
Y Mean: 47.83
X SD : 16.76
Y SD : 26.93
Corr. : -0.06



https://www.autodesk.com/research/publications/same-stats-different-graphs

3) Feature transformations

- **Classification**: in theory, unaffected by monotone feature transforms
  (in practice, this does matter, e.g. running GD on proxy loss)
- **Regression**: not true!

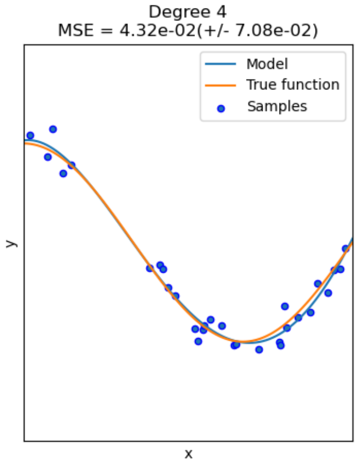
- **Example**: threshold classifiers – $\operatorname{sign}\big(f(x)\big) = \operatorname{sign}\big(\alpha f(x)\big)\ \forall \alpha \in \mathbb{R}$
- In regression, direct evaluation of $f(x)$ (i.e., no sign) means:
  - loss is sensitive to transforms
  - statistical modeling assumptions mean different things
- **Example:**
  - Consider $x \mapsto \log x$ (assume $x > 0$)
  - Compare: $y = a + bx \quad \Rightarrow$ one unit change in x results in b units change in y

    $y = a + b \log x \quad \Rightarrow$ one % change in x results in $\frac{b}{100}$ units change in y

4) Generalization

- **Classification**: overfitting: (i) quantified by VC, and (ii) reduced by norm regularizers
- **Regression**:
  - VC theory does not apply (specific to binary classification – think what shattering does!)
  - Min-norm as max-margin does not apply (there is no longer even a notion of "margin")

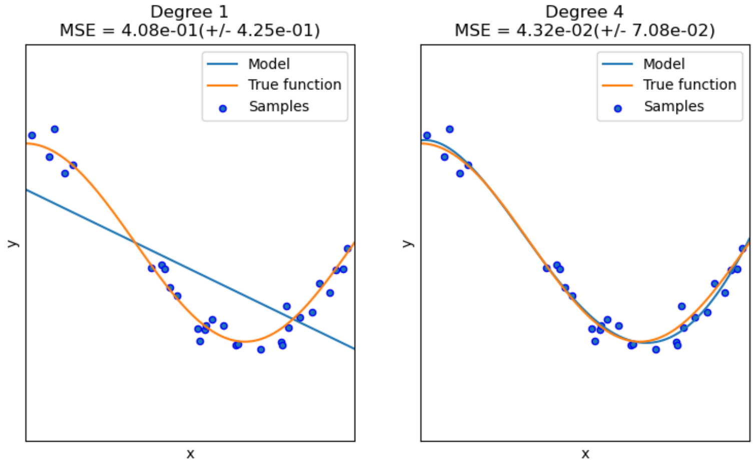
- **Q**: Does overfitting even happen?
- **A**: Most certainly yes! (and easier to draw)



4) Generalization

- **Classification**: overfitting: (i) quantified by VC, and (ii) reduced by norm regularizers
- **Regression**:
  - VC theory does not apply (specific to binary classification – think what shattering does!)
  - Min-norm as max-margin does not apply (there is no longer even a notion of “margin”)

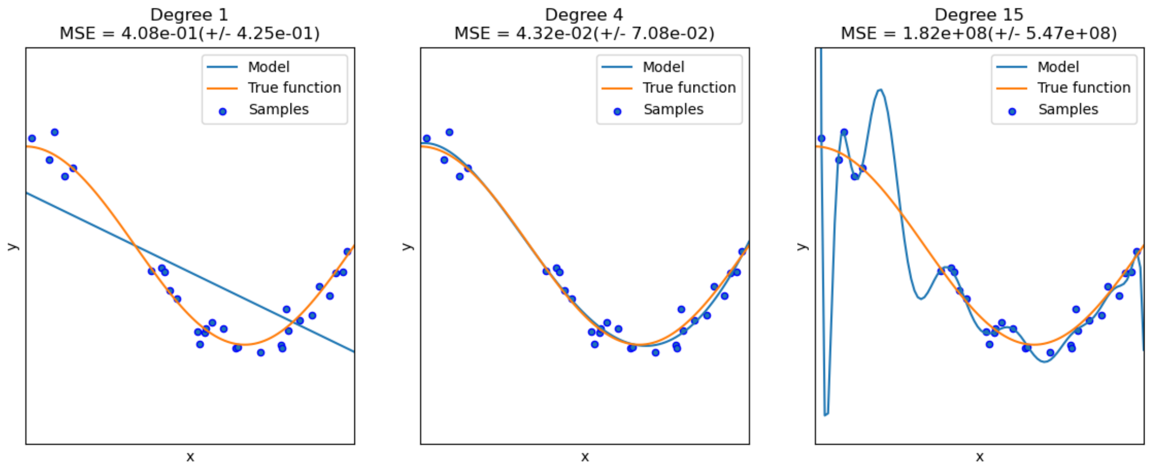
- **Q**: Does overfitting even happen?
- **A**: Most certainly yes! (and easier to draw)



4) Generalization

- **Classification**: overfitting: (i) quantified by VC, and (ii) reduced by norm regularizers
- **Regression**:
  - VC theory does not apply (specific to binary classification – think what shattering does!)
  - Min-norm as max-margin does not apply (there is no longer even a notion of "margin")

- **Q**: Does overfitting even happen?
- **A**: Most certainly yes! (and easier to draw)
- **Q**: Do norm regularizers help with overfitting?
- **A**: Yes! More on this in tirgul.



5) Least Squares vs Least Squares

Empirical Risk Minimization (ERM):

- Learning objective – **empirical loss**:

$$f_{\text{ERM}} = \underset{f \in F}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \left(y_i - f(x_i)\right)^2$$

Maximum Likelihood Estimation (MLE):

- Learning objective – **likelihood**:

$$f_{\text{MLE}} = \underset{f \in F}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \left(y_i - f(x_i)\right)^2$$

5) Least Squares vs Least Squares

Empirical Risk Minimization (ERM):

- Learning objective – **empirical loss**:
  $$f_{\text{ERM}} = \underset{f \in F}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \big(y_i - f(x_i)\big)^2$$
- Loss function: $\ell^{\text{sqr}}(y, \hat{y}) = (y - \hat{y})^2$
- Distribution-independent
- Arbitrary H
- Care about:
  - minimizing expected loss
  - generalization: $\mathbb{E}[(y - f_{\text{ERM}}(x))^2]$
  - finite sample performance

Maximum Likelihood Estimation (MLE):

- Learning objective – **likelihood**:
  $$f_{\text{MLE}} = \underset{f \in F}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \big(y_i - f(x_i)\big)^2$$
- Statistical model: $y = f^*(x) + \epsilon$
  - Assumed “true” model $f^* \in F$
  - Assumed error distribution $\epsilon \sim D_{\text{ERR}}$
- Care about: (for example; out of our scope)
  - consistency (asymptotic property)
  - identifiability (asymptotic property)
  - confidence intervals

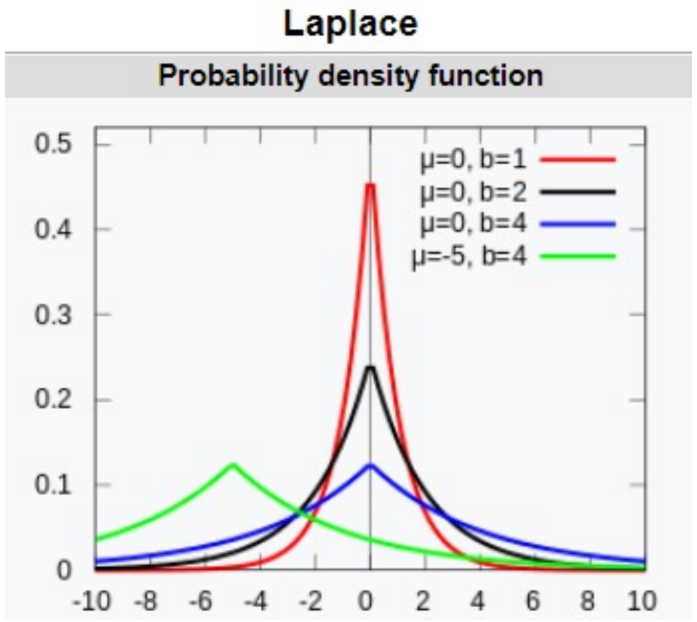
take away: same same (formula) but different (story; and language)

Statistical modeling, revisited

Other noise models

- We saw Normal noise ⇒ squared loss
- **Q**: What happens when assuming other forms of noise?
- **Example**:

$$\epsilon \sim \text{Laplace}(0, \eta)$$

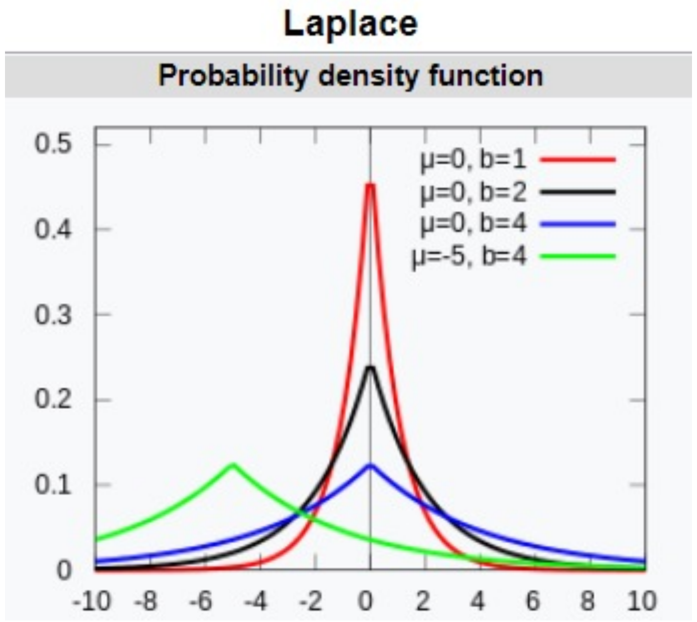


https://en.wikipedia.org/wiki/Laplace_distribution

Other noise models

- We saw Normal noise ⇒ squared loss
- **Q**: What happens when assuming other forms of noise?
- **Example**:

$$\epsilon \sim \text{Laplace}(0, \eta) = \frac{1}{2\eta} e^{-\frac{|x|}{\eta}}$$



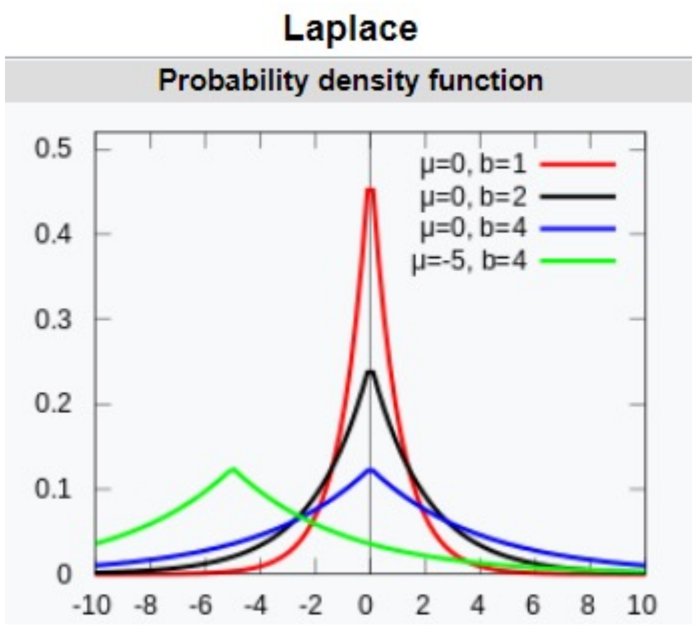
https://en.wikipedia.org/wiki/Laplace_distribution

Other noise models

- We saw Normal noise ⇒ squared loss
- **Q**: What happens when assuming other forms of noise?
- **Example**:

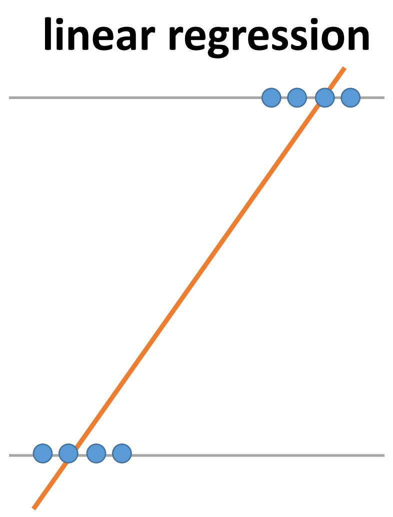
$$\epsilon \sim \text{Laplace}(0, \eta) = \frac{1}{2\eta} e^{-\frac{|x|}{\eta}}$$

- **A**: Gives absolute loss $\ell^{\text{abs}}(y, \hat{y}) = |y - \hat{y}|$
- (To get this, just follow MLE derivation for Normal)
- Called ***least absolute deviation***
- Solution estimate **median** of $p(y|x)$ (vs. average in squared loss)
  - **pros**: more robust to outliers
  - **cons**: not smooth



https://en.wikipedia.org/wiki/Laplace_distribution

What about classification?



- Can statistical modelling handle classification? ($y \in \{0,1\}$)
- In principle, can just use linear regression… (it’s well defined)
- …but it’s a little silly!
- E.g., we know y is bounded in $[0,1]$, but $f(x)$ is not
- **Fix:** change assumption on $p(y|x)$

Logistic regression

- In **linear regression**, we:
  1. Assumed $y|x \sim N(\mu_x, \sigma^2)$, where $\mu_x = \mathbb{E}[Y|X = x] \in \mathbb{R}$
  2. Modeled $\hat{y} = \mathbb{E}[Y|X = x] = w^\top x$
- **In logistic regression:**
  1. Assume $y|x \sim \text{Bernoulli}(p_x)$, for some $p_x \in [0,1]$
     - This gives:

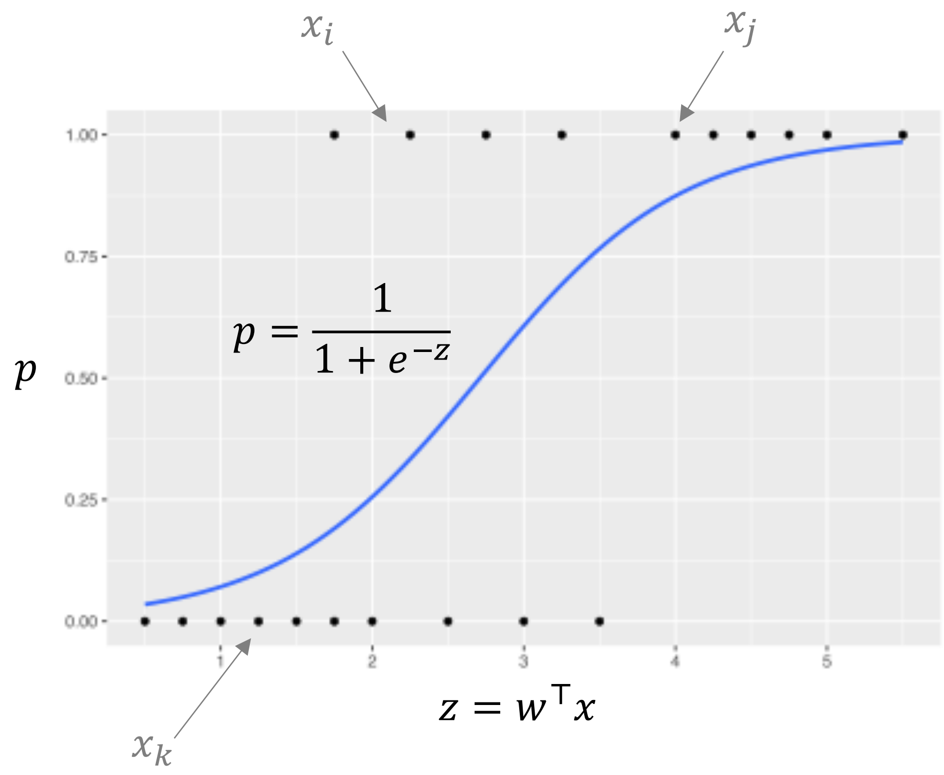
$$P(y|x) = \begin{cases} p_x & \text{if } y = 1 \\ 1 - p_x & \text{if } y = 0 \end{cases}$$

     - Recall Bernoulli $\Rightarrow$ $\mathbb{E}[Y|X = x] = p_x$
  2. Can now model $f(x) = \mathbb{E}[Y|X = x] = p_x$

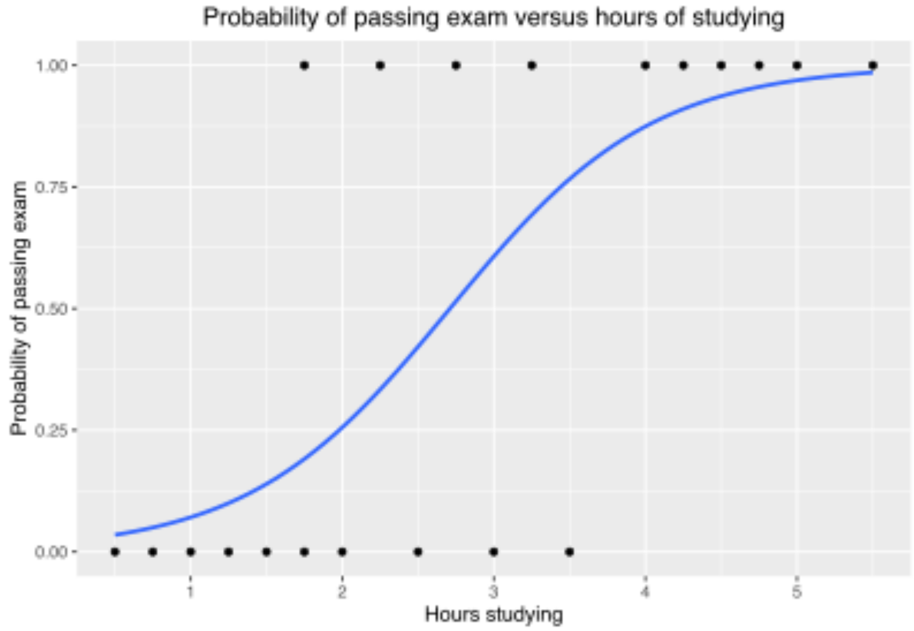
Logistic regression

- We would like to model $f(x) = \mathbb{E}[Y|X = x] = p_x$
- Need p_x to be <u>parametric</u>: $p_x = f(x; w)$
- This allows to *learn* w from data with MLE
- Keep it simple: use *linear* $w^\top x$!
- **Q**: Would $p_x = w^\top x$ work?
- **A**: No! $p_x \in [0,1]$, but $w^\top x$ isn't
- **Idea**: model probabilities by passing $w^\top x$ through "sigmoid"

$$p_x(w) = P(Y = 1|x; w) = \frac{1}{1 + e^{-w^\top x}} := \sigma(w^\top x) \in [0,1]$$



Logistic regression



- We would like to model $f(x) = \mathbb{E}[Y|X = x] = p_x$
- Need p_x to be parametric: $p_x = f(x; w)$
- This allows to *learn* w from data with MLE
- Keep it simple: use *linear* $w^\top x$!
- **Q**: Would $p_x = w^\top x$ work?
- **A**: No! $p_x \in [0,1]$, but $w^\top x$ isn't
- **Idea**: model probabilities by passing $w^\top x$ through "sigmoid"

$$p_x(w) = P(Y = 1|x; w) = \frac{1}{1 + e^{-w^\top x}} \coloneqq \sigma(w^\top x) \in [0,1]$$

- Relies on common empirical observation:
  as p reaches extremes, changes in x matter less

Logistic regression

- Learning w gives us a **predictor**:

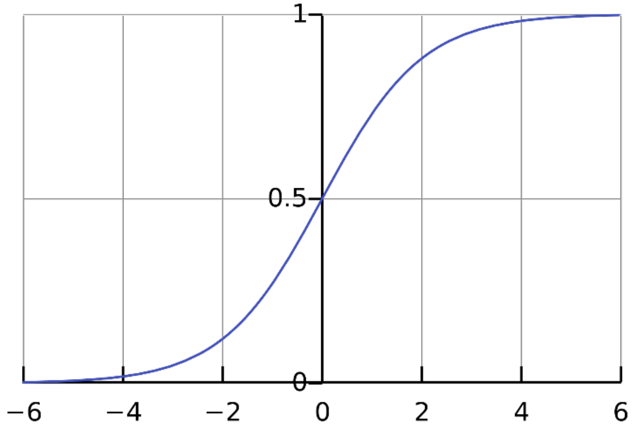
$$\hat{p} = \sigma(w^\top x + b) = \frac{1}{1 + e^{-w^\top x + b}} \in [0,1]$$

- σ is called a **logistic function**
- Note that labels $y \in \{\pm 1\}$ are *binary*, but predictions $\hat{p} \in [0,1]$ are *probabilities*
- To classify, can threshold: $\hat{y} = \mathbb{1}\{\hat{p} > 0.5\}$

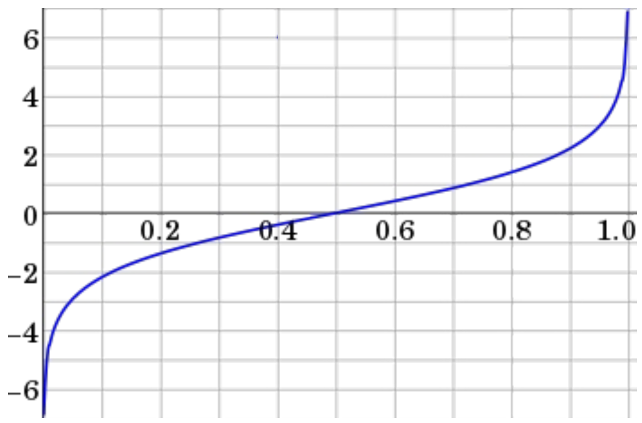
- But what does $w^\top x$ "mean"?
- Interpretation: invert to get *linear log-odds*:

$$w^\top x = \log \frac{p(x)}{1 - p(x)} \quad \text{(a.k.a. "logit")}$$

logistic: $\frac{1}{1+e^{-z}}$



logit/log-odds: $\log \frac{p}{1-p}$



Learning objective

$NLL(w;S) = -\log P(S;w)$

"soft" prediction
$\hat{p} \in [0,1]$

$= \ \ldots$

$= -\sum_i \log P(y_i|x_i;w) \ = \begin{cases} \hat{p}_i & \text{if } y_i = 1 \\ 1-\hat{p}_i & \text{if } y_i = 0 \end{cases}$

(remember $\hat{p}_i$ depends on w)

$= -\sum_i \log \hat{p}_i^{\,y_i}(1-\hat{p}_i)^{1-y_i}$

$= -\sum_i y_i \log \hat{p}_i + (1-y_i)\log(1-\hat{p}_i)$

- **Cross-entropy loss:**

  $\ell^{\mathrm{CE}}(y,\hat{p}) = -y\log\hat{p} - (1-y)\log(1-\hat{p})$

- **Interpretation**: logistic regression =

  cross_entropy(sigmoid(linear))

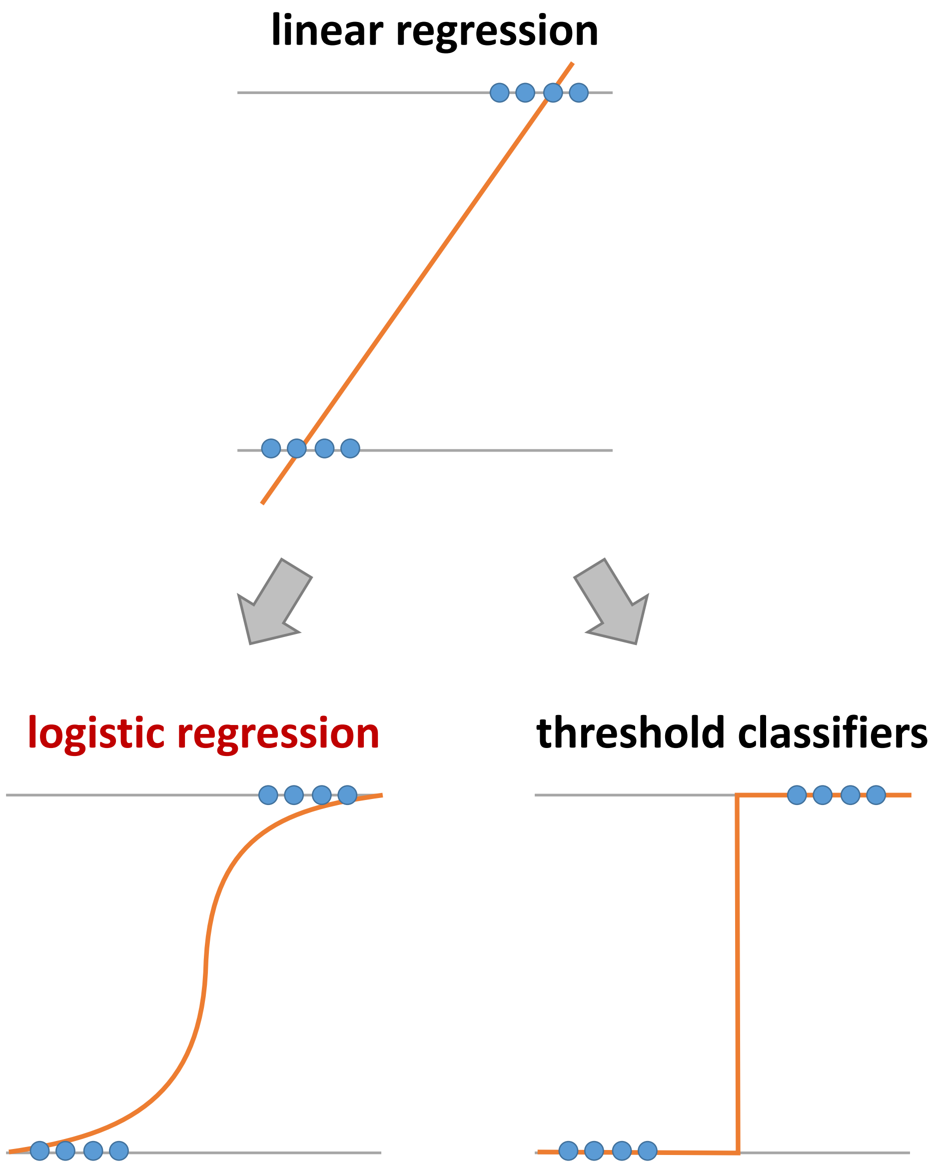
- Nonetheless – **convex** in w!
  (so can optimize with gradient descent)
- Naturally extends to multiclass
  (think multinomial distribution)
- Very popular in deep learning

Vs. discriminative

- **Statistical modeling** handles classification by wrapping linear model with *logistic* *"link function"*
- Compare this to **discriminative ML**, which wraps a linear model with a *step function* (e.g., sign, or $\mathbb{1}$)
- Not so different after all!
- In fact, sigmoid → step function in the limit:

$$\sigma_\alpha(z) = \frac{1}{1 + e^{-\alpha z}} \xrightarrow[\alpha \to \infty]{} \mathbb{1}\{z > 0\}$$

- Since ℓ^{CE} is continuous, can use as proxy for 0-1 loss (just like hinge is!)



Next week

- **Part III**: *more supervised learning*
    1. Regression (today)
    2. Bagging and boosting ← *residuals!*
    3. Deep learning