CS-GY 6923: Lecture 4
Naive Bayes Classifier, Bayesian Linear
Regression + Regularization

NYU Tandon School of Engineering, Prof. Christopher Musco

$$(x, y) \rightarrow Pr(x, y)$$

Specific approach to machine learning:

1. Design parametric probabilistic model (usual pretty simple) that plausibly could have generated our data $(x_1, y_1), \ldots, (x_n, y_n)$.

2. Learn unknown parameters of the model based on observed training data.

3. Given new input x, predict the label y which is most likely under the now fixed probabilistic model.

Two strategies covered: Maximum a posteriori (MAP) estimation and maximum likelihood estimation (MLE).

$$Pr\ (x, \bigcirc)$$

Goal:

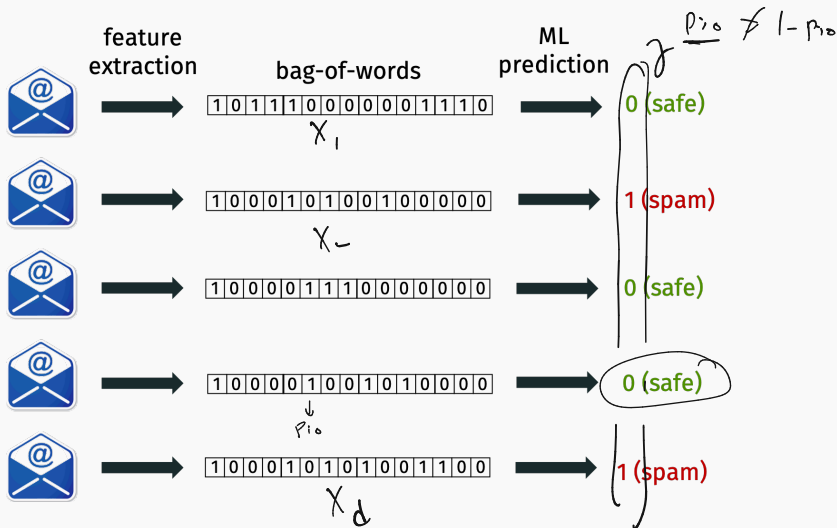$$\begin{bmatrix} 0101100 \\ \\ \chi \\ \\ \\ \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} y$$

- Build a probabilistic model for a binary classification problem.
- Estimate parameters of the model.
- From the model derive a Maximum a posteriori (MAP) estimation classification rule for future predictions. Specifically we will see the Naive Bayes Classifier.

x_1	x_2	x_3	

3

feature extraction — bag-of-words — ML prediction

$$\gamma \frac{p_{i0}}{} \neq 1 - p_{i0}$$

1 0 1 1 1 0 0 0 0 0 0 1 1 1 0 x_1 0 (safe)

1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 x_2 1 (spam)

1 0 0 0 0 1 1 1 0 0 0 0 0 0 0   0 (safe)

1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 p_{i0} 0 (safe)

1 0 0 0 1 0 1 0 1 0 0 1 1 0 0 x_d 1 (spam)

Both target labels and data vectors are binary.

Probabilistic model for (bag-of-words, label) pair (x, y):

- Set $y = 0$ with probability p_0, $y = 1$ with probability $p_1 = 1 - p_0$.
  - p_0 is probability an email is not spam (e.g. 99%).
  - p_1 is probability an email is spam (e.g. 1%).
- If $y = 0$, for each i, set $x_i = 1$ with prob. p_{i0}.
- If $y = 1$, for each i, set $x_i = 1$ with prob. p_{i1}.

Unknown model parameters: x $[- - O - - - - - - - -]$

$i \to$ "hello"

- p_0, p_1,
- $p_{10}, p_{20}, \ldots p_{n0}$, one for each of the n vocabulary words.
- $p_{11}, p_{21}, \ldots p_{n1}$, one for each of the n vocabulary words.

not spam $\leftarrow (p_{i0})$ $(p_{i1}) \longrightarrow$ is spam
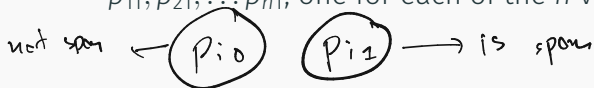
How to set parameters:

- Set p_0 and p_1 to the empirical fraction of not spam/spam emails.
- For each word i, set p_{i0} to the empirical probability word i appears in a non-spam email.
- For each word i, set p_{i1} to the empirical probability word i appears in a spam email.

Estimating these parameters from previous data examples is the only "training" we will do.

DONE WITH MODELING

ON TO PREDICTION

random event $\quad p(x), \, p(y)$

$p(x,y)$

- **Probability:** $p(x)$ – the probability event *x* happens.
- **Joint probability:** $p(x,y)$ – the probability that event *x* and event *y* happen.
- **Conditional Probability** $p(x \mid y)$ – the probability *x* happens given that *y* happens.

$p(x,y) = p(x|y) \cdot p(y)$

$$p(x|y) = \frac{p(x,y)}{p(y)} = \frac{2/6}{1/2} = \frac{4}{6} = \frac{2}{3}$$

X = event that dice comes up odd

y = event that dice value ≤ 3

$p(x) = 1/2$

$p(x|y) = \boxed{\frac{2}{3}}$

7

$$p(x|y) = \frac{p(x,y)}{p(y)} \qquad p(y|x) = \frac{p(x,y)}{p(x)}$$

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

Proof:

$$\frac{p(x,y)}{p(y)} = \frac{\frac{p(x,y)}{p(x)} \cdot p(x)}{p(y)}$$

$$p(x,y) = p(x,y)$$

$$p(x)\,p(y|x) = p(y) \cdot p(x|y)$$

Given unlabeled input (w, ___) choose the label $y \in \{0, 1\}$ which is <u>most likely</u> given the data. Recall $\mathbf{w} = [0, 0, 1, \ldots, 1, 0]$.

Classification rule: maximum a posterior (MAP) estimate.

Step 1. Compute:

· $p(y = 0 \mid \mathbf{w})$: prob. $y = 0$ given observed data vector **w**.

· $p(y = 1 \mid \mathbf{w})$: prob. $y = 1$ given observed data vector **w**.

Step 2. Output: 0 or 1 depending on which probability is larger.

$p(y = 0 \mid \mathbf{w})$ and $p(y = 1 \mid \mathbf{w})$ are called **posterior** probabilities.

prior probability $\longrightarrow$ $p_0 \rightarrow p(y = 0)$

$\longrightarrow p_1 \rightarrow p(y = 1)$

9

How to compute the posterior? **Bayes rule!**

$$p(y = 0 \mid \mathbf{w}) = \frac{p(\mathbf{w} \mid y = 0)\, p(y = 0)}{p(\mathbf{w})} \tag{1}$$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}} \tag{2}$$

- **Prior:** Probability in class 0 <u>prior</u> to seeing any data.
- **Posterior:** Probability in class 0 <u>after</u> seeing the data.

10

Goal is to determine which is larger:

likelihood

$$p(y = 0 \mid \mathbf{w}) = \frac{p(\mathbf{w} \mid y = 0)p(y = 0)}{p(\mathbf{w})} \qquad \text{vs.}$$

$$p(y = 1 \mid \mathbf{w}) = \frac{p(\mathbf{w} \mid y = 1)p(y = 1)}{p(\mathbf{w})}$$

- We can ignore the evidence $p(\mathbf{w})$ since it is the same for both sides!

- $p(y = 0)$ and $p(y = 1)$ already known (computed from training data). These are our computed parameters p_0, p_1.

- $p(\mathbf{w} \mid y = 0) = ?$ $p(\mathbf{w} \mid y = 1) = ?$

11

$$pr(v_1 = 0 \mid y = 0) = 1 - pr(v_1 = 1 \mid y = 0) = \underline{1 - p_{10}}$$

Consider the example $\mathbf{w} = [0, 1, 1, 0, 0, 0, 1, 0].$

Recall that, under our model, index i is 1 with probability p_{i0} if we are not spam, and 1 with probability p_{i1} if we are spam .

$$p(\mathbf{w} \mid y = 0) = pr(w_1 = 0 \mid y = 0) \, pr(w_2 = 1 \mid y = 0) \ldots$$
$$pr(w_n = 0 \mid y = 0)$$
$$= (1 - p_{10})(p_{20})(p_{30}) \cdots (1 - p_{m0})(p_{m+1,0})$$

$$\underline{p(\mathbf{w} \mid y = 1) = (1 - p_{11})(p_{21})(p_{31}) \cdots (1 - p_{m1})}$$

$$(1 - p_{10}) \, p_{20} \, p_{30} \, (1 - p_{40})(1 - p_{50}) \cdots$$

Final Naive Bayes Classifier

Training/Modeling: Use existing data to compute:

- $p_0 = p(y = 0), p_1 = p(y = 1)$ $\arg\max\ (a, b)$

  $\arg\max\ (\log(a), \log(b))$
- For all i compute:
  - $p_{i0} = p(x_i = 1 \mid y = 0)$ and $(1 - p_{i0}) = p(x_i = 0 \mid y = 0)$
  - $\bar{p}_{i1} = p(x_i = 1 \mid y = 1)$ and $(1 - p_{i1}) = p(x_i = 0 \mid y = 1)$

Prediction:

- For new input $\mathbf{w}$:
  - Compute $p(\mathbf{w} \mid y = 0) = \prod_i p(w_i \mid y = 0)$ $\rightarrow$ likelihood
  - Compute $p(\mathbf{w} \mid y = 1) = \prod_i p(w_i \mid y = 1)$
- Return

$$\arg\max \left[p(\mathbf{w} \mid y = 0) \cdot p(y = 0), \; p(\mathbf{w} \mid y = 1) \cdot p(y = 1) \right].$$

p_0 p_1

13

OTHER APPLICATIONS OF
THE BAYESIAN PERSPECTIVE

The Bayesian view offers an interesting alternative perspective on <u>many</u> machine learning techniques.
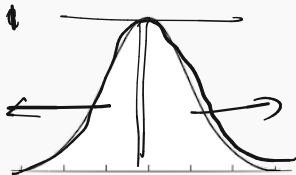
$$X \beta = y + \tilde{m}$$
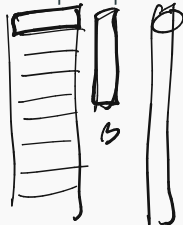
Example: Linear Regression.

Probabilistic model:

$$y = \langle \mathbf{x}, \boldsymbol{\beta} \rangle + \eta$$

where the η drawn from $N(0, \sigma^2)$ is **random Gaussian noise**.
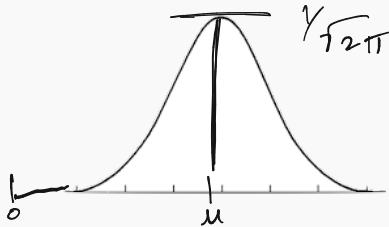
$$Pr(\eta = z) \sim e^{-z^2}$$

The symbol $\sim$ means "is proportional to".

Names for same thing: Normal distribution, Gaussian distribution, bell curve.

Parameterized by mean μ and variance σ^2.

$$\frac{y}{\sqrt{2\pi}}$$

η is a continuous random variable, so it has a probability density function $p(\eta)$ with $\int_{-\infty}^{\infty} p(\eta)d\eta = 1$

$$p(\eta) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{\eta-\mu}{\sigma})^2}$$

normalizing

The important thing to remember is that the the PDF falls off exponentially as we move further from the mean.
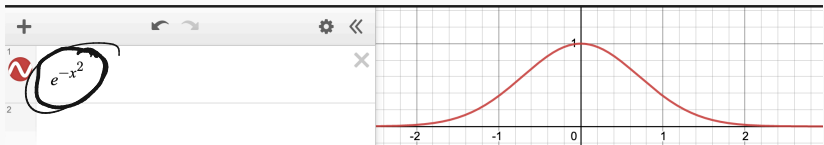
The normalizing constant in front 1/2, etc. don't matter so much.

Example: Linear Regression.

Probabilistic model:

(x, Θ) β

$\downarrow$ $\downarrow$

known   known

$$y = \langle \mathbf{x}, \boldsymbol{\beta} \rangle + \eta$$
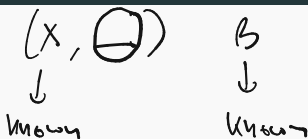
where the η drawn from $N(0, \sigma^2)$ is **random Gaussian noise**. The noise is <u>independent</u> for different inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$.

If we knew β what is the <u>maximum a posterior (MAP)</u> estimate for y given new data observation x?

I.e. what value of y maximizes $\Pr(y \mid \mathbf{x})$?

$y = \langle x, \beta \rangle + \mu$ mean of η

$\langle x, \beta \rangle$ $\downarrow$ $= 0$

17

How should be select β for our model?

Also use a Bayesian approach!

Choose β to maximize:

$$\text{posterior} = \Pr(\beta \mid X, y) = \frac{\Pr(X, y \mid \beta)\Pr(\beta)}{\Pr(X, y)} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}.$$

training set

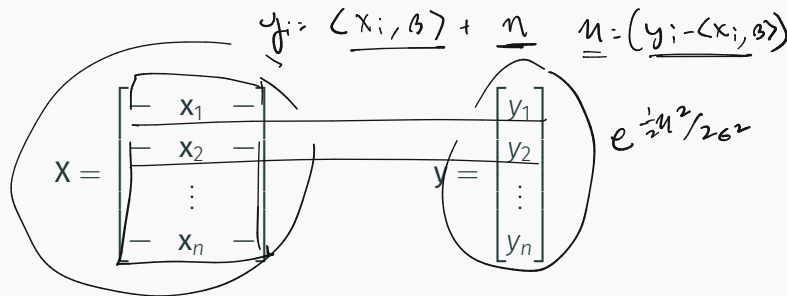In this case, we don't have a prior – no values of β are inherently more likely than others.

Choose β to maximize just the likelihood:

$$\frac{\Pr(X, y \mid \beta)\Pr(\beta)}{\Pr(X, y)} = \frac{\text{likelihood} \times \cancel{\text{prior}}}{\cancel{\text{evidence}}}.$$

This is called the maximum likelihood estimate.

Data:

$$y_i = \langle x_i, \beta \rangle + \eta \qquad \eta = (y_i - \langle x_i, \beta \rangle)$$

$$X = \begin{bmatrix} - & x_1 & - \\ - & x_2 & - \\ & \vdots & \\ - & x_n & - \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \qquad e^{\frac{1}{2}\eta^2/2\sigma^2}$$

Model: $y_i = \langle x_i, \beta \rangle + \eta_i$ where $p(\eta_i = z) \sim e^{-z^2/2\sigma^2}$ and $\eta_1, \ldots, \eta_n$ are independent.

$$Pr(y \mid \beta) \sim P_r(x_1, y_1 \mid \beta) \cdot P_r(x_2, y_2 \mid \beta) \ldots P_r(x_n, y_n \mid \beta)$$

$$\downarrow \qquad\qquad\qquad \downarrow$$

$$e^{-\frac{1}{2}(y_1 - \langle x_1, \beta \rangle)^2/2\sigma^2} \bullet \ldots \bullet e^{-\frac{1}{2}(y_i - \langle x_i, \beta \rangle)^2/2\sigma^2}$$

19

$\mu = 0$

Easier to work with the log likelihood:

$$\arg\max_{\boldsymbol{\beta}} \Pr(\mathbf{X}, \mathbf{y} \mid \boldsymbol{\beta}) = \arg\max_{\boldsymbol{\beta}} \prod_{i=1}^{n} e^{-(y_i - \langle \mathbf{x}_i, \boldsymbol{\beta}\rangle)^2/2\sigma^2}$$

$$= \arg\max_{\boldsymbol{\beta}} \ \log\left(\prod_{i=1}^{n} e^{-(y_i - \langle \mathbf{x}_i, \boldsymbol{\beta}\rangle)^2/2\sigma^2}\right)$$

$$= \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{n} -(y_i - \langle \mathbf{x}_i, \boldsymbol{\beta}\rangle)^2 /2\sigma^2$$

$$\longrightarrow \log\left(e^{-(y_i - \langle x_i, \beta\rangle)^2/2\sigma^2}\right)$$

$$= \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \langle \mathbf{x}_i, \boldsymbol{\beta}\rangle)^2.$$

$$\Downarrow$$

$$\| X\beta - y \|_2^2$$

20
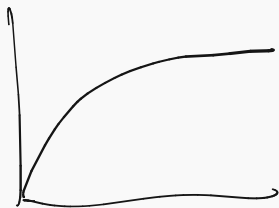
$$\text{Min } \|y - X\beta\|_2^2$$

Conclusion: Choose β to minimize:

$$\beta = (X^T X)^{-1} X^T y$$

$$\sum_{i=1}^{n} (y_i - \langle x_i, \beta \rangle)^2 = \|y - X\beta\|_2^2.$$

This is a completely different justification for squared loss!

Minimizing the ℓ_2 loss is optimal in a certain sense when you assume your data follows a linear model with i.i.d. Gaussian noise.

If we had modeled our noise η as Laplace noise, we would have found that minimizing $\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_1$ was optimal.
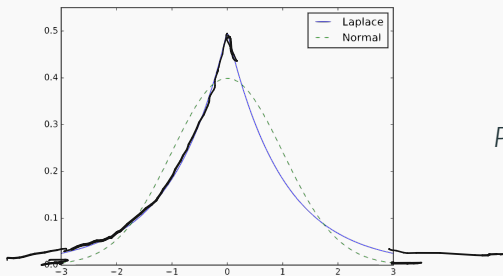
$$e^{-z^2} \to \text{gaussian}$$

$$Pr(\eta = z) \sim e^{-|z|}$$

Laplace noise has "heavier tails", meaning that it results in more outliers.

This is a completely different justification for ℓ_1 loss.

We can add another layer of probabilistic modeling by also assuming $\boldsymbol{\beta}$ is random and comes from some distribution, which encodes our <u>prior</u> belief on what the parameters are.

Return to Maximum a posteriori (MAP estimation):

$$\Pr(\boldsymbol{\beta} \mid X, y) = \frac{\Pr(X, y \mid \boldsymbol{\beta}) \Pr(\boldsymbol{\beta})}{\Pr(X, y)}.$$

Assume values in $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_d]$ come from some distribution.

- **Common model:** Each β_i drawn from $N(0, \gamma^2)$, i.e. normally distributed, independent.
- Encodes a belief that we are unlikely to see models with very large coefficients.

Goal: choose $\boldsymbol{\beta}$ to maximize:

$$\beta^{*} = \arg\max \Pr(\boldsymbol{\beta} \mid X, y) = \frac{\boxed{\Pr(X, y \mid \boldsymbol{\beta}) \Pr(\boldsymbol{\beta})}}{\cancel{\Pr(X, y)}}.$$

- We can still ignore the "evidence" term $\Pr(X, y)$ since it is a constant that does not depend on $\boldsymbol{\beta}$.

- $\underline{\Pr(\boldsymbol{\beta})} = \underline{\Pr(\beta_1)} \cdot \underline{\Pr(\beta_2)} \cdot \ldots \cdot \underline{\Pr(\beta_d)}$

- $\Pr(\boldsymbol{\beta}) \sim \displaystyle\prod_{i=1}^{d} e^{-\frac{1}{2} \beta_i^2 / 6^2}$

Easier to work with the **log likelihood**:

$$\arg\max_{\boldsymbol{\beta}} \Pr(X, y \mid \boldsymbol{\beta}) \cdot \Pr(\boldsymbol{\beta})$$

$$= \arg\max_{\boldsymbol{\beta}} \prod_{i=1}^{n} e^{-(y_i - \langle x_i, \boldsymbol{\beta} \rangle)^2 / 2\sigma^2} \cdot \left(\prod_{i=1}^{n} e^{-(\beta_i)^2 / 2\gamma^2} \right)$$

$$= \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{n} -(y_i - \langle x_i, \boldsymbol{\beta} \rangle)^2 / 2\sigma^2 + \left(\sum_{i=1}^{d} -(\beta_i)^2 / 2\gamma^2 \right)$$

$$= \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \langle x_i, \boldsymbol{\beta} \rangle)^2 + \frac{\sigma^2}{\gamma^2} \sum_{i=1}^{d} (\beta_i)^2 .$$

Choose $\boldsymbol{\beta}$ to minimize $\|y - X\boldsymbol{\beta}\|_2^2 + \frac{\sigma^2}{\gamma^2} \|\boldsymbol{\beta}\|_2^2$.

Completely different justification for ridge regularization!

λ

25

$$\beta_i \sim L(0, \gamma)$$
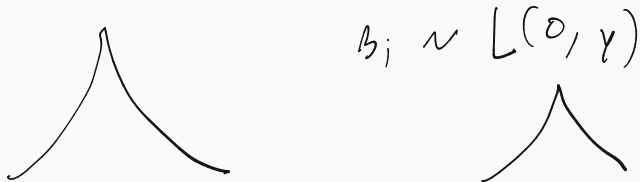
Test your intuition: What modeling assumption justifies LASSO regularization: $\min_{\beta} \|y - X\beta\|_2^2 + \lambda\|\beta\|_1$?

LINEAR CLASSIFICATION

Breast Cancer Biopsy: Determine if a breast lump in a patient is malignant (cancerous) or benign (safe).

- Collect cells from lump using fine needle biopsy.
- Stain and examine cells under microscope.
- Based on certain characteristics (shape, size, cohesion) determine if likely malignant or not).

cross section
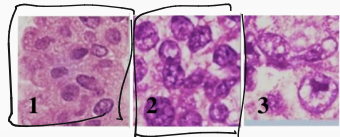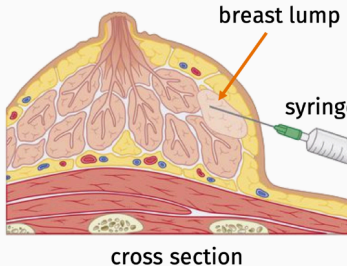
Demo: `demo_breast_cancer.ipynb`

Data: UCI machine learning repository

Breast Cancer Wisconsin (Original) Data Set
Download: Data Folder, Data Set Description

Abstract: Original Wisconsin Breast Cancer Database
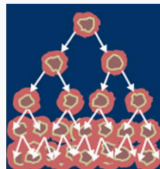
Data Set Characteristics:	Multivariate	Number of Instances:	699	Area:	Life
Attribute Characteristics:	Integer	Number of Attributes:	10	Date Donated	1992-07-15
Associated Tasks:	Classification	Missing Values?	Yes	Number of Web Hits:	564320

```
https://archive.ics.uci.edu/ml/datasets/breast+cancer+
                wisconsin+(original)
```

Features: 10 numerical scores about cell characteristics (Clump Thickness, Uniformity, Marginal Adhesion, etc.)

28

Data: $(x_1, y_1), \ldots, (x_n, y_n)$.

$x_i = [1, 5, 4 \ldots, 2]$ contains score values.

Label $y_i \in \{0, 1\}$ is 0 if benign cells, 1 if malignant cells.

Goal: Based on scores (which would be collected manually, or even learned on their own using an ML algorithm) predict if a sample of cells is malignant or benign.

Approach:

- Naive Bayes Classifier can be extended to x with numerical values (instead of binary values as seen before). Will see on homework.
- Today: Learn a different type of classifier.

We pick two variables, Margin Adhesion and Size Uniformity and plot a scatter plot. Points with label 1 (malignant) are plotted in blue, those with label 2 (benign) are plotted in green.
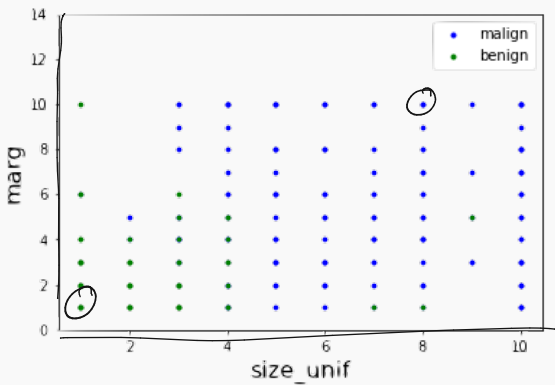
Lots of overlapping points! Hard to get a sense of the data. 30

Simple + Useful Trick: data jittering. Add tiny random noise (using e.g. `np.random.randn`) to data to prevent overlap.
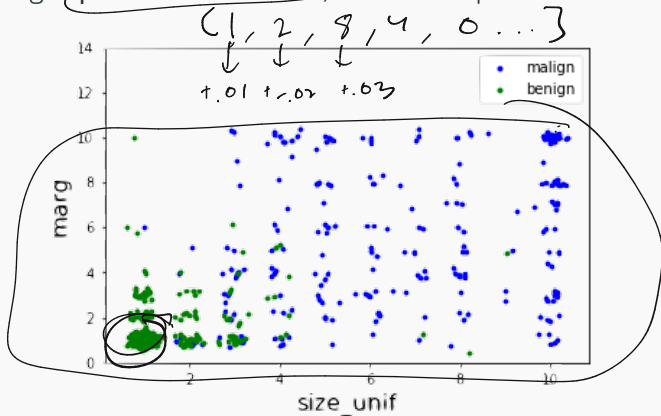
Noise is only for plotting. It is not added to the data for training, testing, etc.

Any ideas for possible <u>classification rules</u> for this data?
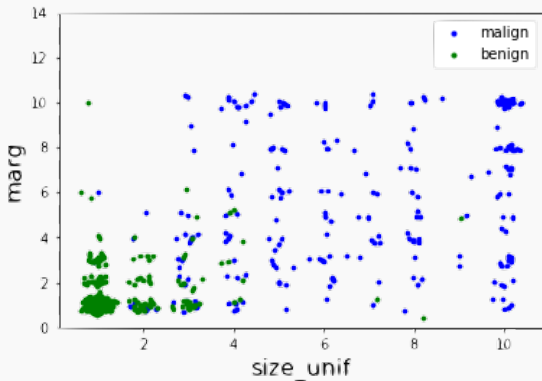
Given vector of predictors $\mathbf{x}_i \in \mathbb{R}^d$ (here $d = 2$) find a parameter vector $\boldsymbol{\beta} \in \mathbb{R}^d$ and threshold λ.

- Predict $y_i = 0$ if $\langle \mathbf{x}_i, \boldsymbol{\beta} \rangle \leq \lambda$.
- Predict $y_i = 1$ if $\langle \mathbf{x}_i, \boldsymbol{\beta} \rangle > \lambda$

Line has equation $\langle \mathbf{x}, \boldsymbol{\beta} \rangle = \lambda$.

33
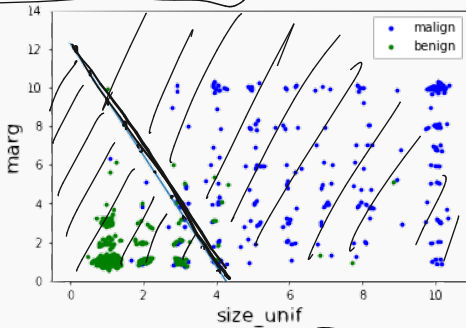
As long as we append a 1 onto each data vector $\mathbf{x}_i$ (i.e. a column of ones onto the data matrix $\mathbf{X}$) like we did for linear regression, an equivalent function is:

- Predict $y_i = 0$ if $\langle \mathbf{x}_i, \boldsymbol{\beta} \rangle \leq 0$.
- Predict $y_i = 1$ if $\langle \mathbf{x}_i, \boldsymbol{\beta} \rangle > 0$

$$[\text{marg, size, 1}] \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_\nu \end{bmatrix}$$
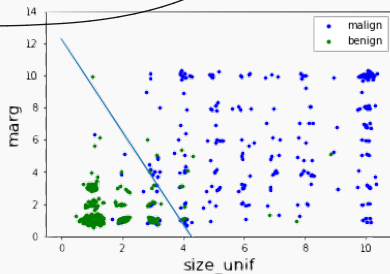
Line has equation $\langle \mathbf{x}, \boldsymbol{\beta} \rangle = 0$.

0 − 1 LOSS

Question: How do we find a good linear classifier
automatically?

Loss minimization approach (first attempt):

→ # of training examples

- Model[1]:

$$f_{\beta}(\mathbf{x}) = \mathbb{1}\left[\langle \mathbf{x}, \beta \rangle > 0\right]$$

- Loss function: "0 − 1 Loss"

$$L(\beta) = \sum_{i=1}^{n} |f_{\beta}(\mathbf{x}_i) - y_i|$$

0, 1

---

[1] $\mathbb{1}[\text{event}]$ is the indicator function: it evaluates to 1 if the argument inside is true, 0 if false.

Problem with $0 - 1$ loss:
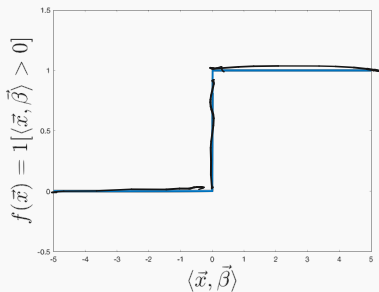
- The loss function $L(\boldsymbol{\beta})$ is not differentiable because $f_{\boldsymbol{\beta}}(\mathbf{x})$ is discontinuous.
- Impossible to take the gradient, very hard to minimize loss to find optimal $\boldsymbol{\beta}$.
- Non-convex function (will make more sense next lecture).

Loss minimization approach (second attempt):

- Model:

$$f_{\boldsymbol{\beta}}(\mathbf{x}) = \mathbb{1}[\langle \mathbf{x}, \boldsymbol{\beta} \rangle > 1/2]$$

- Loss function: "Square Loss"

$$\min \quad L(\boldsymbol{\beta}) = \sum_{i=1}^{n} (\langle \mathbf{x}, \boldsymbol{\beta} \rangle - y_i)^2$$
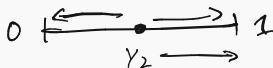
Intuitively tries to make $\langle \mathbf{x}, \boldsymbol{\beta} \rangle$ close to 0 for examples in class 0, close to 1 for examples in class 1.

37

We can solve for β by just solving a least squares multiple linear regression problem.
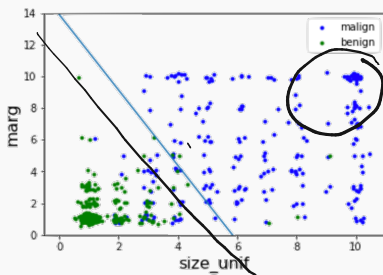
Do you see any issues here?

Problem with square loss:
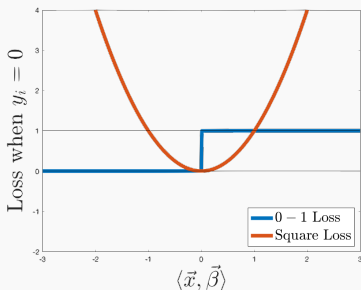
- Loss increases if $\langle \mathbf{x}, \boldsymbol{\beta} \rangle > 1$ even if correct label is 1. Or if $\langle \mathbf{x}, \boldsymbol{\beta} \rangle < 0$ even if correct label is 0.
- Intuitively we don't want to "punish" these cases.

Let $h_\beta(\mathbf{x})$ be the logistic function:

$$h_\beta(\mathbf{x}) = \frac{1}{1 + e^{-\langle \beta, \mathbf{x} \rangle}}$$

$$\frac{1}{1 + e^0} = \frac{1}{2}$$

40
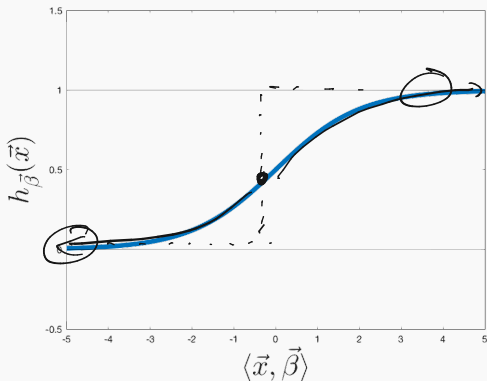
Loss minimization approach (this works!):

- **Model**: Let $h_\beta(x) = \frac{1}{1+e^{-\langle \beta, x \rangle}}$

  $\langle x, \beta \rangle > 0$

  $$f_\beta(x) = \mathbb{1}\left[h_\beta(x) > 1/2\right]$$

- **Loss function**: "Logistic loss" aka "binary cross-entropy loss"

$$L(\beta) = \left(-\right)\sum_{i=1}^{n} y_i \log(h_\beta(x)) + (1 - y_i) \log(1 - h_\beta(x))$$

$y_i = 0$

$y_i = 1$ $- \log(h_\beta(x))$
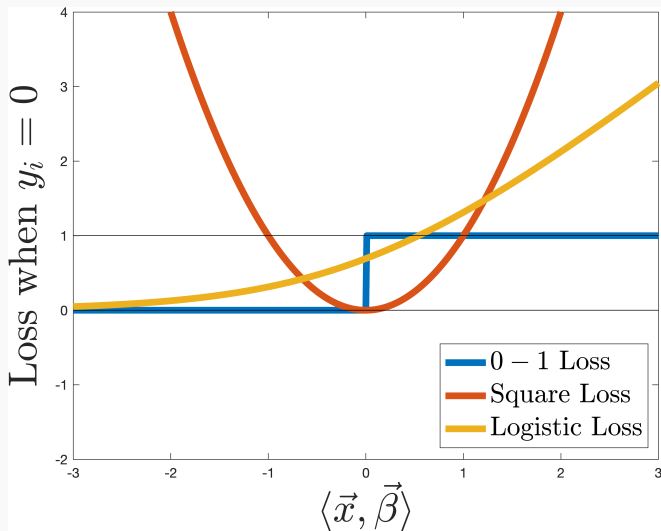
Logistic Loss:
$$L(\boldsymbol{\beta}) = -\sum_{i=1}^{n} y_i \log(h_{\boldsymbol{\beta}}(\mathsf{x})) + (1 - y_i) \log(1 - h_{\boldsymbol{\beta}}(\mathsf{x}))$$

Logistic Loss:
$$L(\boldsymbol{\beta}) = -\sum_{i=1}^{n} y_i \log(h_{\boldsymbol{\beta}}(\mathsf{x})) + (1 - y_i) \log(1 - h_{\boldsymbol{\beta}}(\mathsf{x}))$$

43
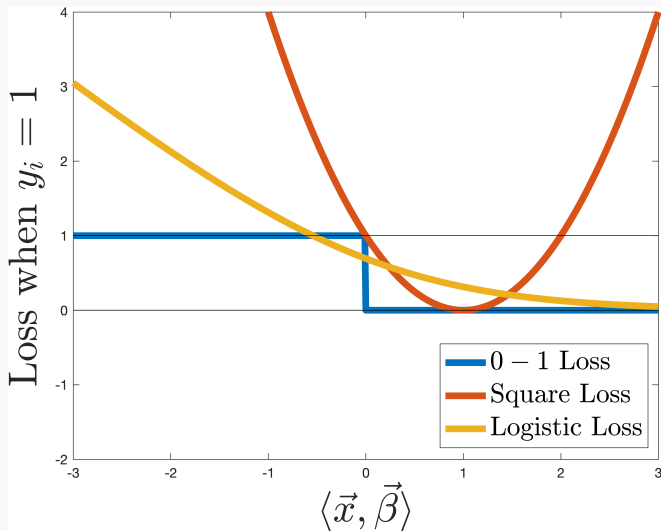
- Convex function in β, can be minimized using gradient descent (next lecture).
- Works well in practice.
- Good Bayesian motivation: see posted lecture notes if you are interested.
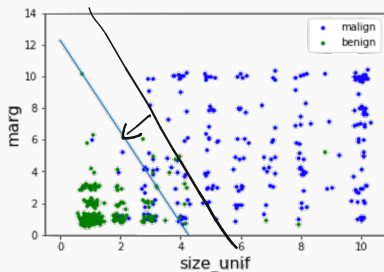
Fit using logistic regression/log loss.

Once we have a classification algorithm, how do we judge its performance?
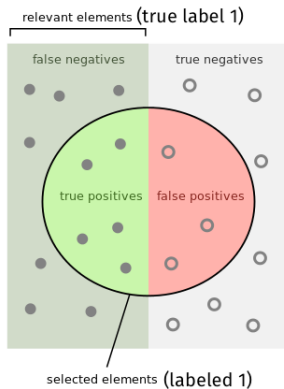
- **Simplest answer:** Error rate = fraction of data examples misclassified in test set.
- What are some issues with this approach?

- **Precision:** Fraction of positively labeled examples (label 1) which are correct.
- **Recall:** Fraction of true positives that we labeled correctly with label 1.

Question: Which should we optimize for medical diagnosis?

Logistic regression workflow:

- Select $\boldsymbol{\beta}$ via training and compute $h_{\boldsymbol{\beta}}(\mathbf{x}_i) = \frac{1}{1+e^{-\langle \mathbf{x}_i, \boldsymbol{\beta} \rangle}}$ for all $\mathbf{x}_i$.
- Predict $y_i = 0$ if $h_{\boldsymbol{\beta}}(\mathbf{x}_i) \leq \lambda$, $y_i = 1$ if $h_{\boldsymbol{\beta}}(\mathbf{x}_i) > \lambda$.
- Default value of λ is 1/2. Increasing λ improves <u>precision</u>. Decreasing λ improves <u>recall</u>.
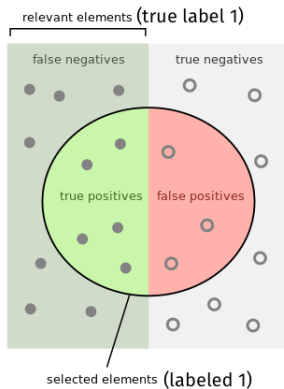
Once we have a classification algorithm, how do we judge its performance?

- **Simplest answer:** Error rate = fraction of data examples misclassified in test set.
- What are some issues with this approach?

- Precision: Fraction of positively labeled examples (label 1) which are correct.
- Recall: Fraction of true positives that we labeled correctly with label 1.

Question: Which should we optimize for medical diagnosis?

49

Possible logistic regression workflow:

- Learn $\vec{\beta}$ and compute $h_{\vec{\beta}}(\vec{x}_i) = \frac{1}{1+e^{-\langle \vec{x}_i, \vec{\beta} \rangle}}$ for all $\vec{x}_i$.
- Predict $y_i = 0$ if $h_{\vec{\beta}}(\vec{x}_i) \leq \lambda$, $y_i = 1$ if $h_{\vec{\beta}}(\vec{x}_i) > \lambda$.
- Default value of λ is 1/2. Increasing λ improves <u>precision</u>. Decreasing λ improves <u>recall</u>.

This is very heuristic. There are other methods for handling "class imbalance" which can often lead to good overall error, but poor precision or recall. Techniques include weighting the loss function to care more about false negatives, or subsampling the larger class.

What about when $y \in \{1, \ldots, q\}$ instead of $y \in \{0, 1\}$

Two options for multiclass data:

- One-vs.-all (most common, also called one-vs.-rest)
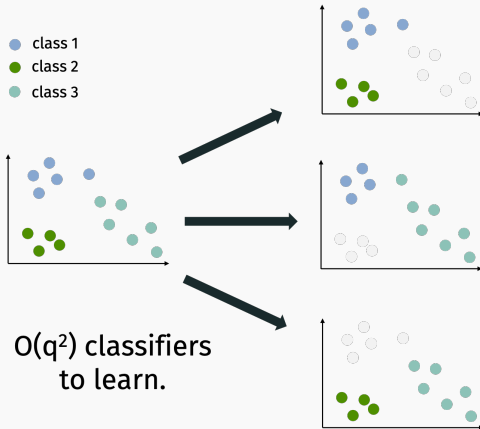- One-vs.-one (slower, but can be more effective)

In both cases, we convert to multiple <u>binary</u> classification problem.

q classifiers to learn.

- For q classes train q classifiers. Obtain parameters $\vec{\beta}_1, \ldots, \vec{\beta}_q$.
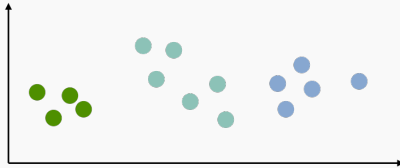- Assign y to class i with maximum $\langle \vec{\beta}_i, \vec{x} \rangle$.

class 1
class 2
class 3

$O(q^2)$ classifiers
to learn.

- For q classes train $\frac{q(q-1)}{2}$ classifiers.
- Assign y to class which i which wins in the most number of head-to-head comparisons.

Hard case for one-vs.-all.

- One-vs.-one would be a better choice here.
- Also tends to work better when there is class in balance.

We will see one more very common approach related to one-vs-rest either next lecture or when we do neural nets.

Confusion matrix for k classes:

Pred--> Real↓	1	2	...	K
1				
2				
...				
K				

- Entry i, j is the fraction of class i items classified as class j.
- Useful to see whole matrix to visualize where errors occur.