CS-UY 4563: Lecture 8
Finishing the Bayesian Perspective, Linear Classifiers

NYU Tandon School of Engineering, Prof. Christopher Musco

PROBABILISTIC MODELING

Bayesian or Probabilistic approach to machine learning:

- Decide on simple probabilistic model with parameters $\vec{\theta}$ which could explain our data $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$.
- Learn $\vec{\theta}$ from past data.
- Given a new input $\vec{x}$, predict y (either a class label or regression value) using the probabilistic model.

Typically prediction y is chosen to be the **maximum a posterior (MAP) estimate** under the assumption that data comes from our chosen probabilistic model.

NAIVE BAYES CLASSIFIER

Example from last class:

- Given binary inputs $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$ (e.g. email bag-of-words vectors and binary labels)
- Came up with model for how $\vec{x}_i, y_i$ might be generated.
- Computed MAP estimate using Bayes rule.

This gave us the Naive Bayes Classifier.

OTHER APPLICATIONS OF THE BAYESIAN PERSPECTIVE

BAYESIAN REGRESSION

The Bayesian view offers an interesting alternative perspective on many machine learning techniques.

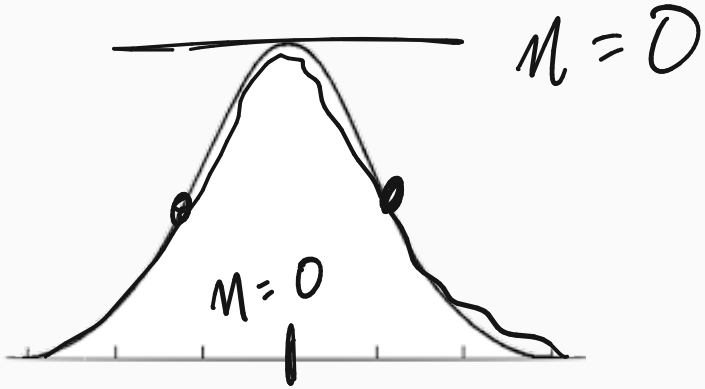
Example: Linear Regression.

Probabilistic model:

$$y_i = \langle \vec{x}_i, \vec{\beta} \rangle + \eta$$

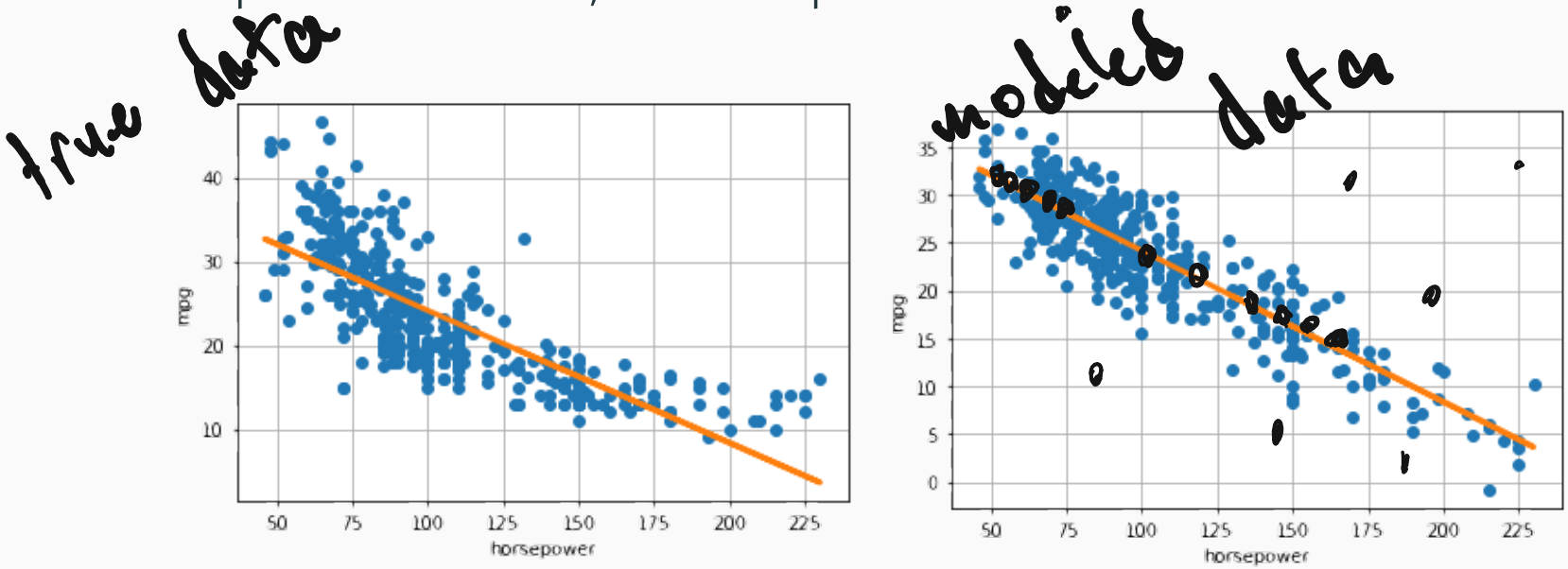
where η is a **Gaussian random variable** with variance σ^2.

(Here we assume $\vec{x}_1, \ldots, \vec{x}_n$ are **fixed**, not random. This is called a "fixed design" setting.)



$$Pr(\eta = z) \sim \frac{1}{\sqrt{2\pi\sigma^2}} e^{-z^2/\sigma^2}$$

Not a perfect model, but simple and reasonable:



To make the plot on right I used numpy's `random` library and the `randn` function for generating Gaussian (normal) random numbers:

```
ypred = beta1*x + beta0
var = 3
ypred_with_noise = ypred + var*np.random.randn(ypred.shape[0])
```

Example: Linear Regression.

Probabilistic model:

$$y_i = \langle \vec{x}_i, \vec{\beta} \rangle + \eta$$

where η is a **Gaussian random variable** with variance σ^2.

Suppose we learn $\vec{\beta}$ using past data. What is the maximum a posterior (MAP) estimate y^* given observed data $\vec{x}$?

- Want to find y^* which maximizes $\max_y \Pr(y \mid \vec{x})$.
- Under our model, $y = \langle \vec{x}, \vec{\beta} \rangle + \eta$.
- So $\Pr(y \mid \vec{x})$ is equal to $\Pr(\eta = y - \langle \vec{x}, \vec{\beta} \rangle)$
- $\Pr(\eta = y - \langle \vec{x}, \vec{\beta} \rangle)$ is maximized at $y - \langle \vec{x}, \vec{\beta} \rangle = 0$.
- So $y^* = \langle \vec{x}, \vec{\beta} \rangle$ is the MAP estimate.

How should we learn $\vec{\beta}$ for our model from prior data?

Bayesian approach: Use MAP estimate again! But this time for the parameter vector itself, not just for prediction.

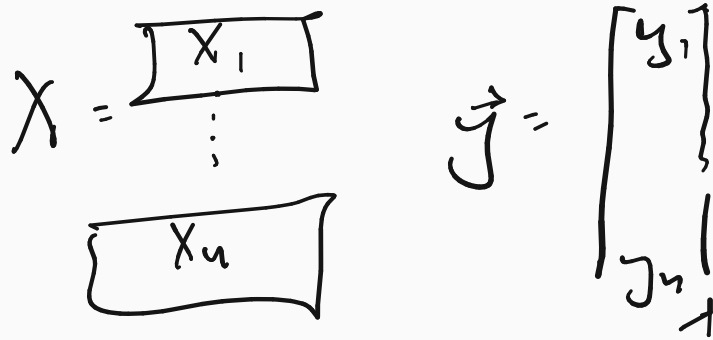
Give data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and target vector $\vec{y} \in \mathbb{R}^n$, choose $\vec{\beta}^*$ to maximize:

$$\max_{\vec{\beta}} \Pr(\vec{\beta} \mid \mathbf{X}, \vec{y}) = \max_{\vec{\beta}} \frac{\Pr(\mathbf{X}, \vec{y} \mid \vec{\beta}) \Pr(\vec{\beta})}{\Pr(\mathbf{X}, \vec{y})}.$$

- Assume all $\vec{\beta}$'s are equally likely. So both $\Pr(\vec{\beta})$ and $\Pr(\mathbf{X}, \vec{y})$ are fixed, independent of β.
- Need to find $\vec{\beta}^*$ to maximize the likelihood $\Pr(\mathbf{X}, \vec{y} \mid \vec{\beta})$.

LIKELIHOOD COMPUTATION

- $y_i = \langle \vec{x}_i, \vec{\beta} \rangle + \eta$
- where $p(\eta = z) \sim \frac{e^{-z^2/\sigma^2}}{\sqrt{2\pi\sigma^2}}$



$$\max_{\beta} \Pr(X, \vec{y} \mid \vec{\beta}) \sim$$

$$\prod_{i=1}^{n} p(x_i, y_i \mid \beta) = \prod_{i=1}^{n} p\left(\eta_i = y_i - \langle x_i, \beta \rangle\right)$$

$$= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(\langle x_i, \beta \rangle - y_i)^2/\sigma^2}$$

LOG LIKELIHOOD

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

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

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

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

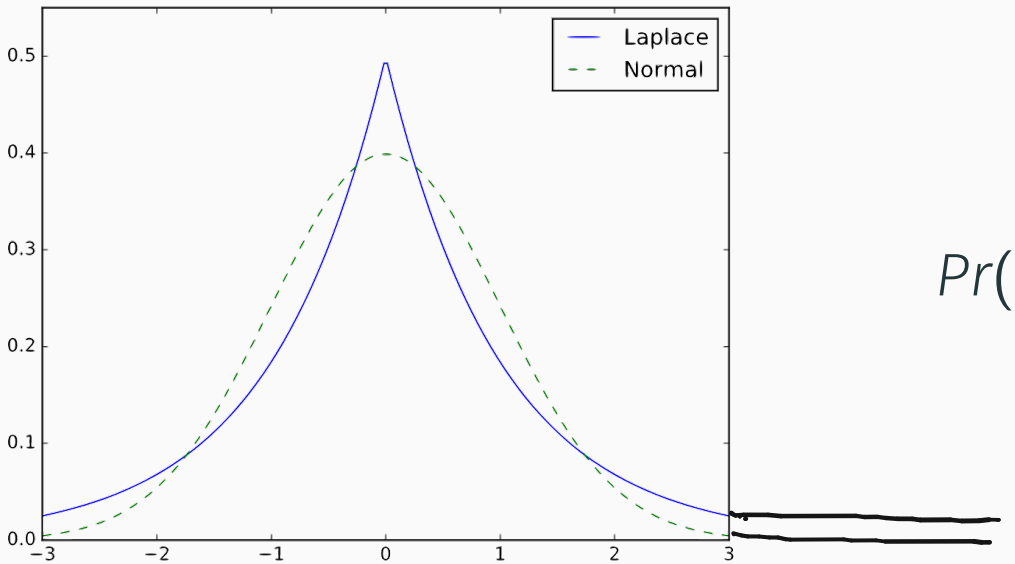
$$= \arg\min_{\vec{\beta}} \sum_{i=1}^{n} (y_i - \langle \vec{x}_i, \vec{\beta} \rangle)^2.$$

Choose $\vec{\beta}^*$ to minimize $\sum_{i=1}^{n} (y_i - \langle \vec{x}_i, \vec{\beta} \rangle)^2 = \|\vec{y} - \mathbf{X}\vec{\beta}\|_2^2$!

This is a completely different justification for squared loss.

BAYESIAN REGRESSION

If we had modeled our noise η as Laplace noise, we would have found that minimizing $\|\vec{y} - \mathbf{X}\vec{\beta}\|_1$ was optimal.



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

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

This is a completely different justification for ℓ_1 loss.

BAYESIAN REGULARIZATION

Recall goal is to maximize over $\vec{\beta}$:

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

~~assume all $\vec{\beta}$'s equally likely~~

Bayesian view: Assume values in $\vec{\beta} = [\beta_1, \ldots, \beta_d]$ are generated from some probabilistic model.

- **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 large coefficients.

BAYESIAN REGULARIZATION

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

$$\Pr(\vec{\beta} \mid \mathbf{X}, \vec{y}) = \frac{\Pr(\mathbf{X}, \vec{y} \mid \vec{\beta}) \Pr(\vec{\beta})}{\Pr(\mathbf{X}, \vec{y})}$$

- We can still ignore the "evidence" term $\Pr(\mathbf{X}, \vec{y})$ since it is a constant that does not depend on $\vec{\beta}$.
- $\Pr(\vec{\beta}) = \Pr(\beta_1) \cdot \Pr(\beta_2) \cdot \ldots \cdot \Pr(\beta_d)$
- $\Pr(\vec{\beta}) \sim \prod_{i=1}^{d} \frac{e^{-\beta_i^2/\gamma^2}}{\sqrt{2\pi\gamma^2}}$

BAYESIAN REGULARIZATION

$$\vec{\beta}* = \arg\max_{\vec{\beta}} \Pr(\mathbf{X}, \vec{y} \mid \vec{\beta}) \cdot \Pr(\vec{\beta})$$

$$= \arg\max_{\vec{\beta}} \prod_{i=1}^{n} e^{-(y_i - \langle \vec{x}_i, \vec{\beta} \rangle)^2/\sigma^2} \cdot \prod_{i=1}^{d} e^{-(\beta_i)^2/\gamma^2}$$

$$= \arg\max_{\vec{\beta}} \sum_{i=1}^{n} -(y_i - \langle \vec{x}_i, \vec{\beta} \rangle)^2/\sigma^2 + \sum_{i=1}^{d} -(\beta_i)^2/\gamma^2$$

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

Choose $\vec{\beta}*$ to minimize $\|\vec{y} - \mathbf{X}\vec{\beta}\|_2^2 + \frac{\sigma^2}{\gamma^2}\|\vec{\beta}\|_2^2$. $\lambda = \frac{\sigma^2}{\gamma^2}$

Completely different justification for ridge regularization!

BAYESIAN REGULARIZATION

Test your intuition: What modeling assumption justifies LASSO regularization: $\min \|\vec{y} - \mathbf{X}\vec{\beta}\|_2^2 + \lambda\|\vec{\beta}\|_1$?

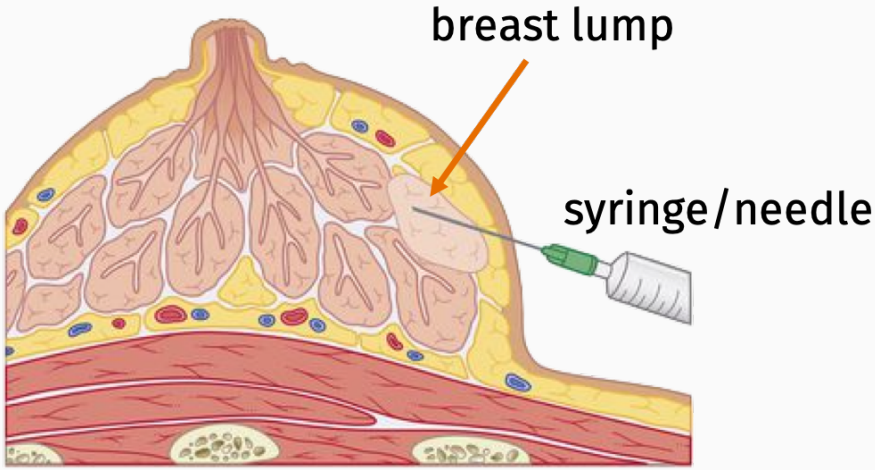
Each $\beta_i \sim e^{-|\beta_i|}$

LINEAR CLASSIFICATION

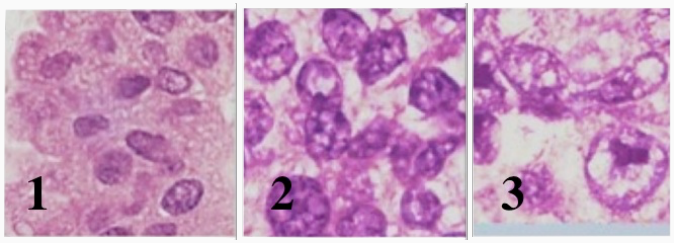
MOTIVATING PROBLEM

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



MOTIVATING PROBLEM

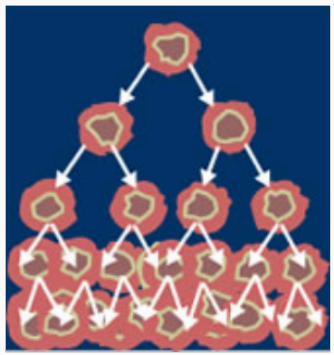
Demo: `demo_breast_cancer.ipynb`

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.)

MOTIVATING PROBLEM

Data: $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$.

$\vec{x}_i = [1, 5, 4 \ldots, 2]$ contains score values.

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

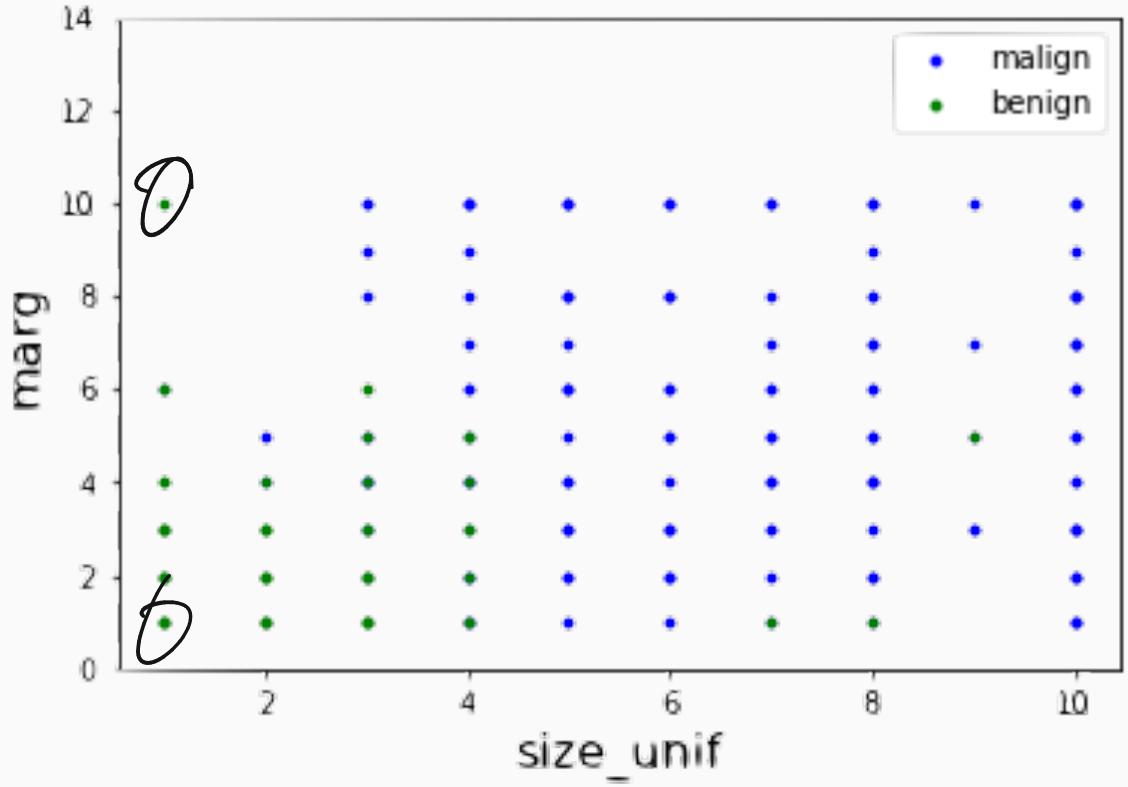
Goal: Based on scores (which would be collected manually, or even learned on their own using an ML algorithm) predict if a sample of cells is malignant or benign.

Approach:

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

BEGIN BY PLOTTING DATA

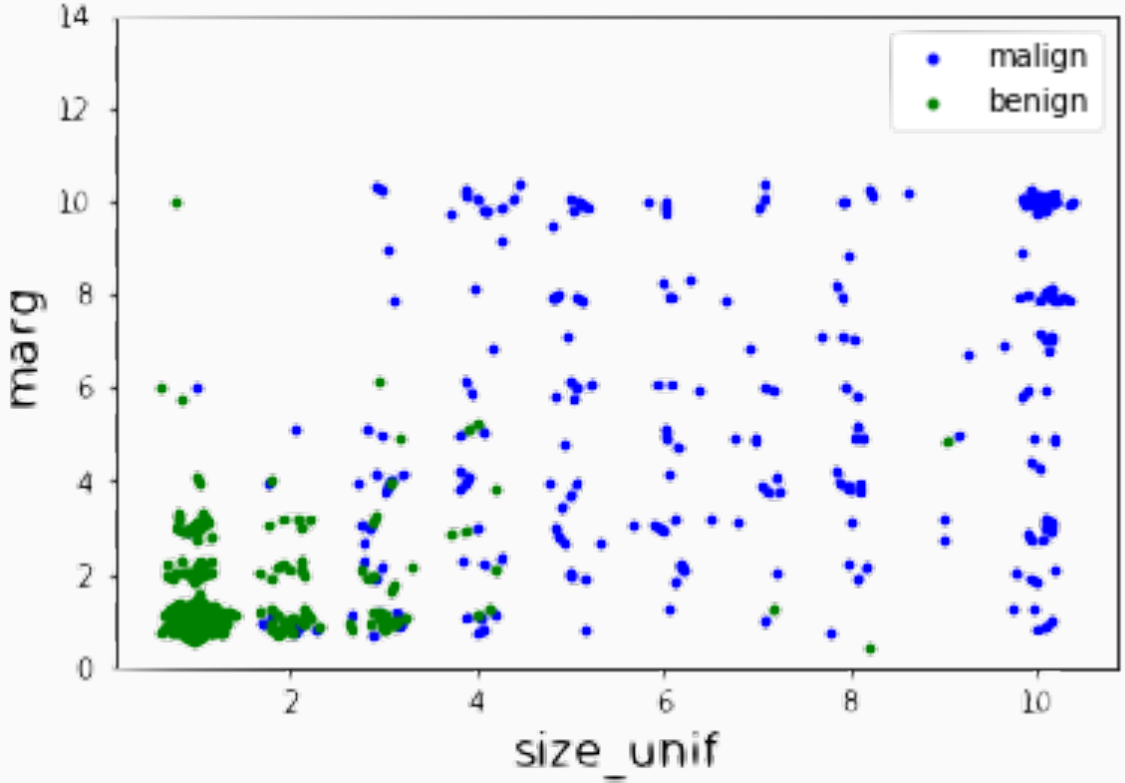
We pick two variables, Margin Adhesion and Size Uniformity and plot a scatter plot. Points with label 1 (malignant) are plotted in blue, those with label 2 (benign) are plotted in green.



Lots of overlapping points! Hard to get a sense of the data.

PLOTTING WITH JITTER

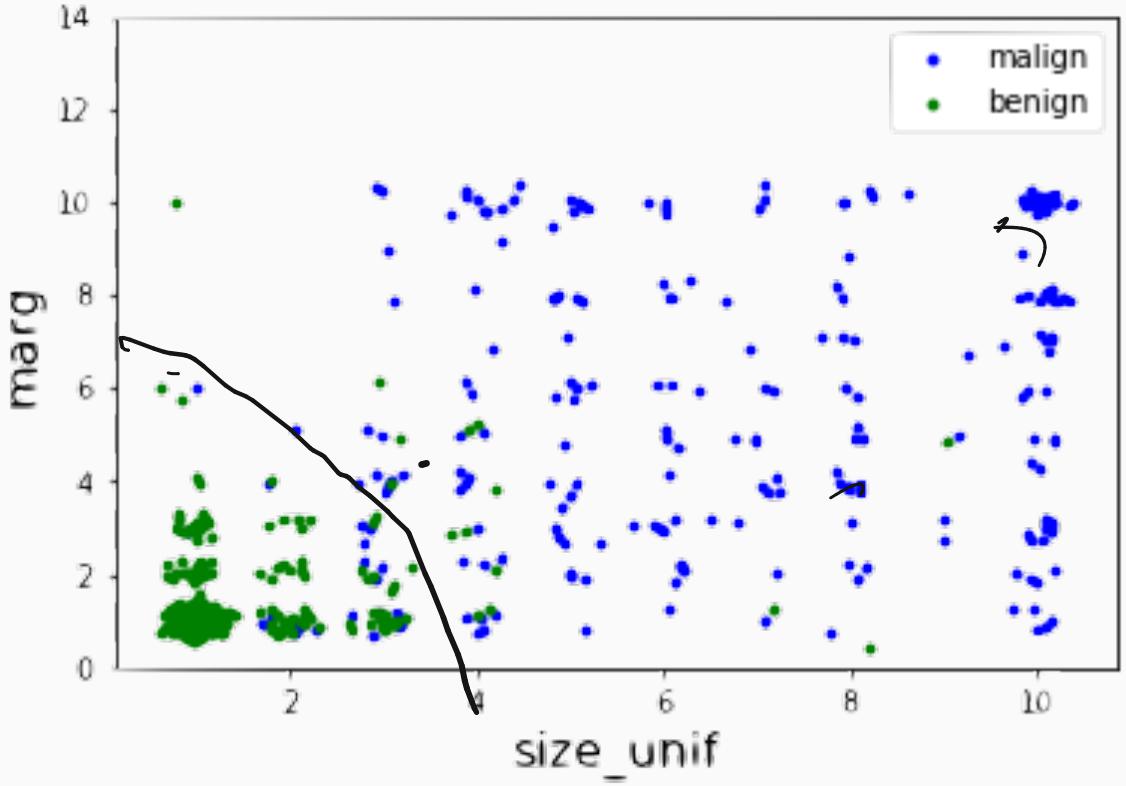
Simple + Useful Trick: data jittering. Add tiny random noise (using e.g. `np.random.randn`) to data to prevent overlap.



Noise is only for plotting. It is not added to the data for training, testing, etc.

1 = malignant 0 = benign

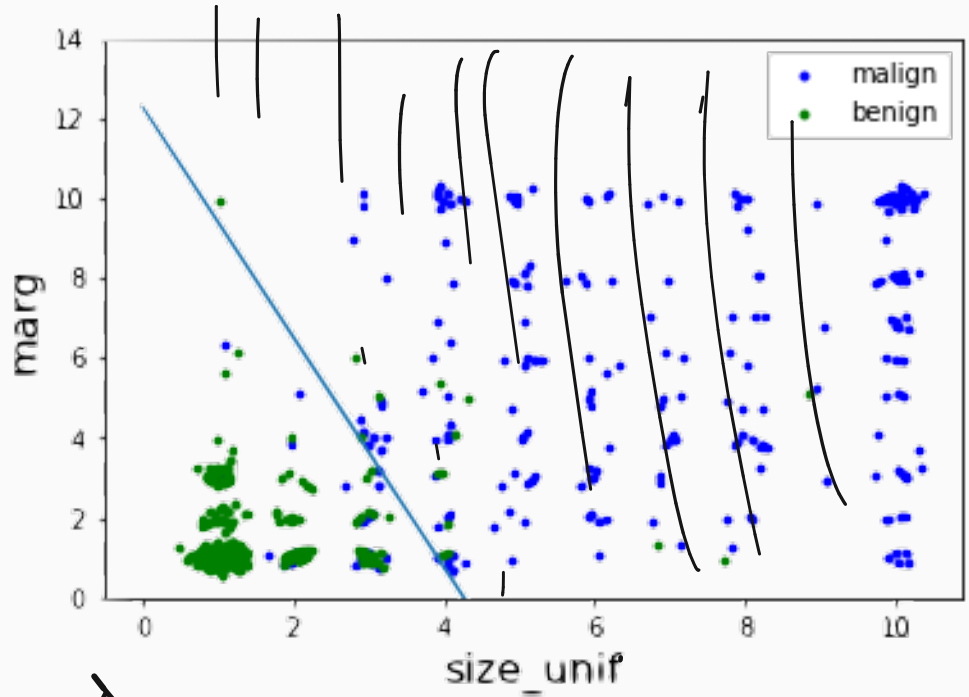
Any ideas for possible classification rules for this data?




```
If (marg > 4 or size_unif > 4)
    return 1;
```

Given vector of predictors $\vec{x}_i \in \mathbb{R}^d$ (here $d = 2$) find a parameter vector $\vec{\beta} \in \mathbb{R}^d$ and threshold λ.

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

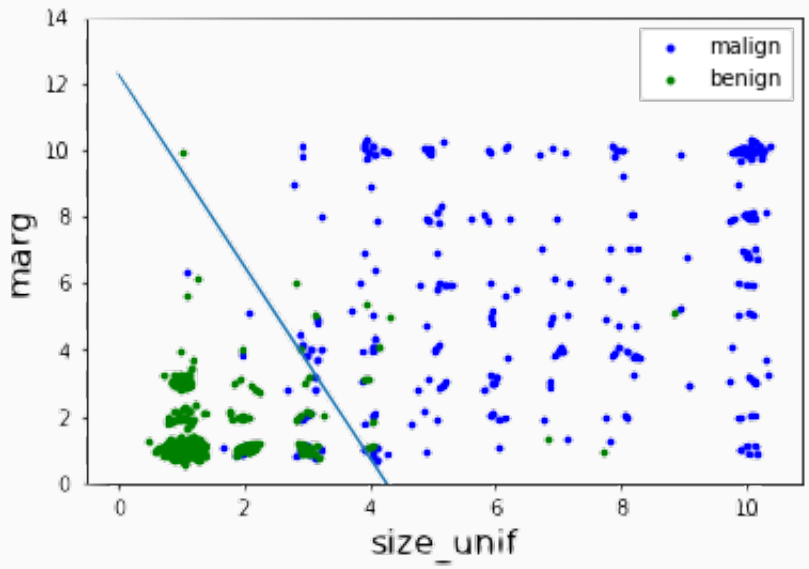


$\beta_1 x_1 + \beta_2 x_2 = \lambda$

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

As long as we append a 1 onto each data vector $\vec{x}_i$ (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 $\langle \vec{x}_i, \vec{\beta} \rangle \leq 0$.
- Predict $y_i = 1$ if $\langle \vec{x}_i, \vec{\beta} \rangle > 0$



Line has equation $\langle \vec{x}, \vec{\beta} \rangle = 0$.

$0-1$ loss

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

Loss minimization approach (first attempt):

- **Model**[1]:

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

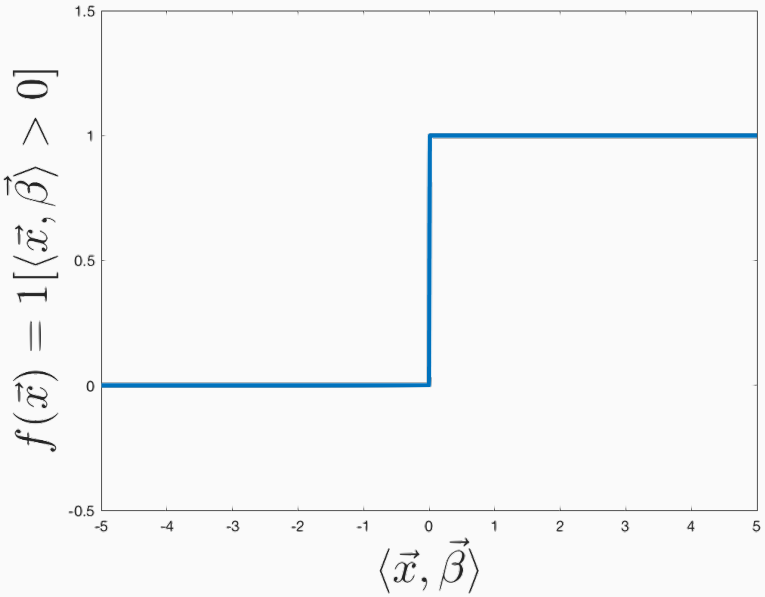
- **Loss function**: "$0-1$ Loss"

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

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

$0-1$ loss

Problem with $0-1$ loss:



- The loss function $L(\vec{\beta})$ is not differentiable because $f_{\vec{\beta}}(\vec{X})$ is discontinuous.
- Impossible to take the gradient, very hard to minimize loss to find optimal $\vec{\beta}$.
- Non-convex function (will make more sense next lecture).

LINEAR CLASSIFIER VIA SQUARE LOSS

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

Loss minimization approach (second attempt):

- **Model**:

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

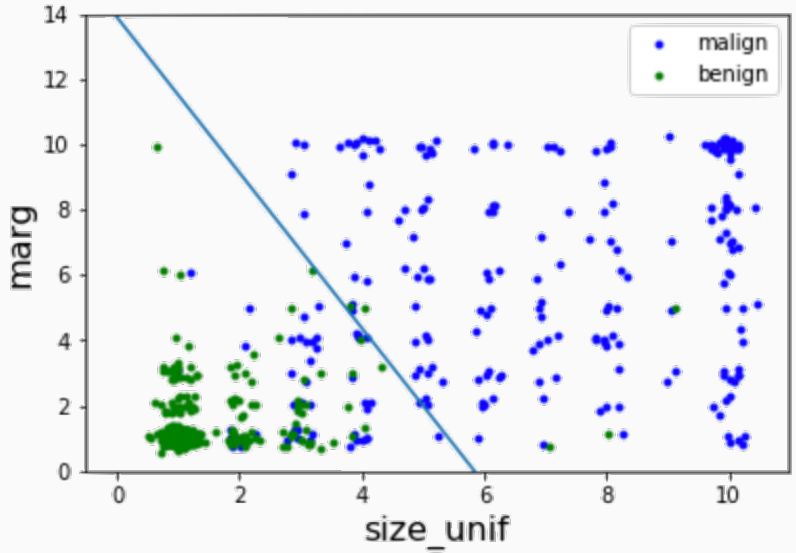
- **Loss function**: "Square Loss"

$$L(\vec{\beta}) = \sum_{i=1}^{n} (\langle \vec{x}, \vec{\beta} \rangle - y_i)^2$$

Intuitively tries to make $\langle \vec{x}, \vec{\beta} \rangle$ close to 0 for examples in class 0, close too 1 for examples in class 1.

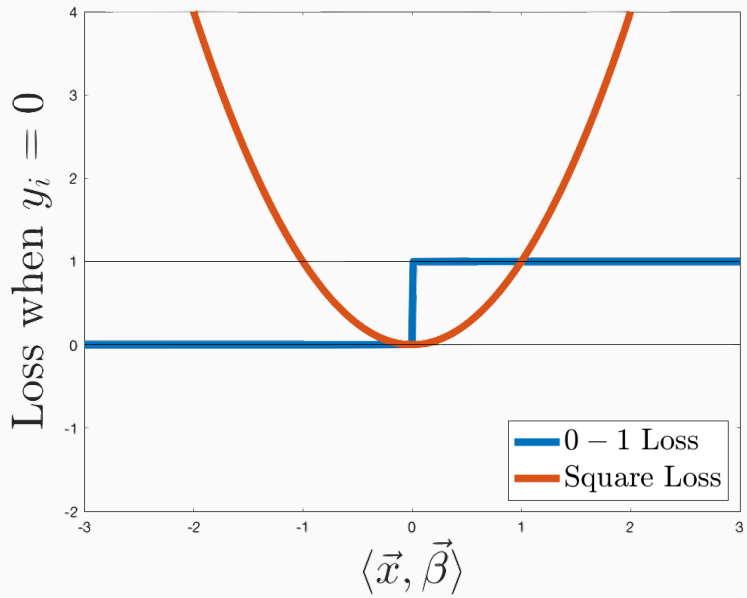
LINEAR CLASSIFIER VIA SQUARE LOSS

We can solve for $\vec{\beta}$ my just solving a least squares multiple linear regression problem.



Do you see any issues here?

Problem with square loss:



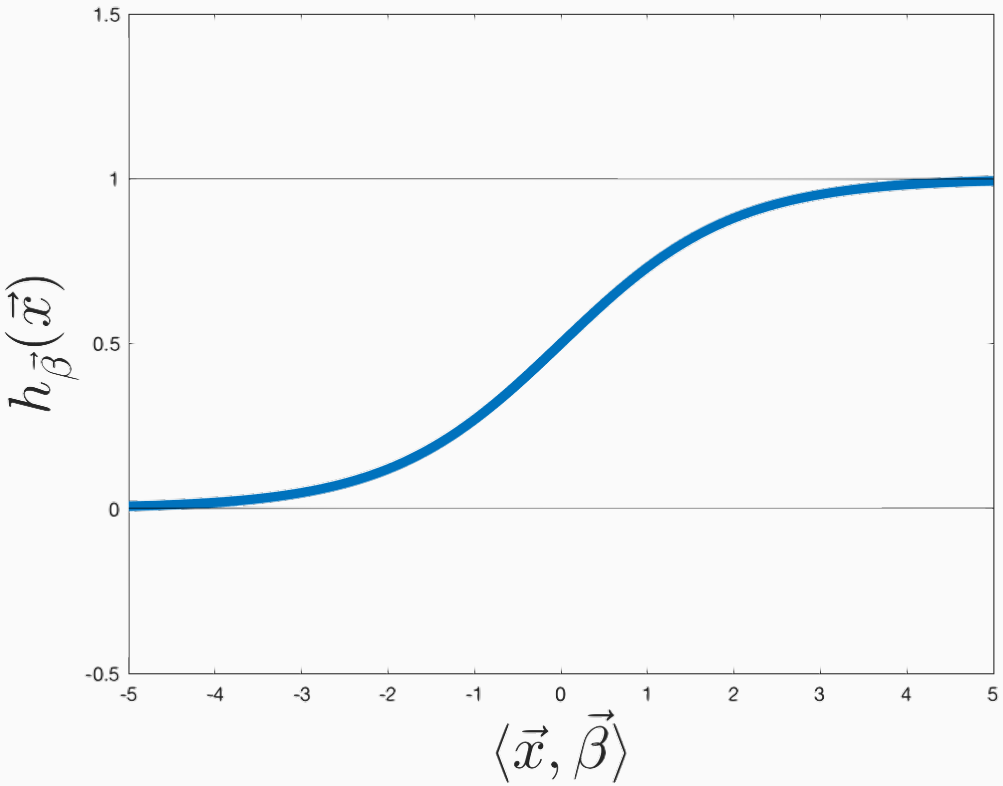
- Loss increases if $\langle \vec{x}, \vec{\beta} \rangle > 1$ even if correct label is 1. Or if $\langle \vec{x}, \vec{\beta} \rangle < 0$ even if correct label is 0.
- Intuitively we don't want to "punish" these cases.

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

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

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

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



Loss minimization approach (what works!):

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

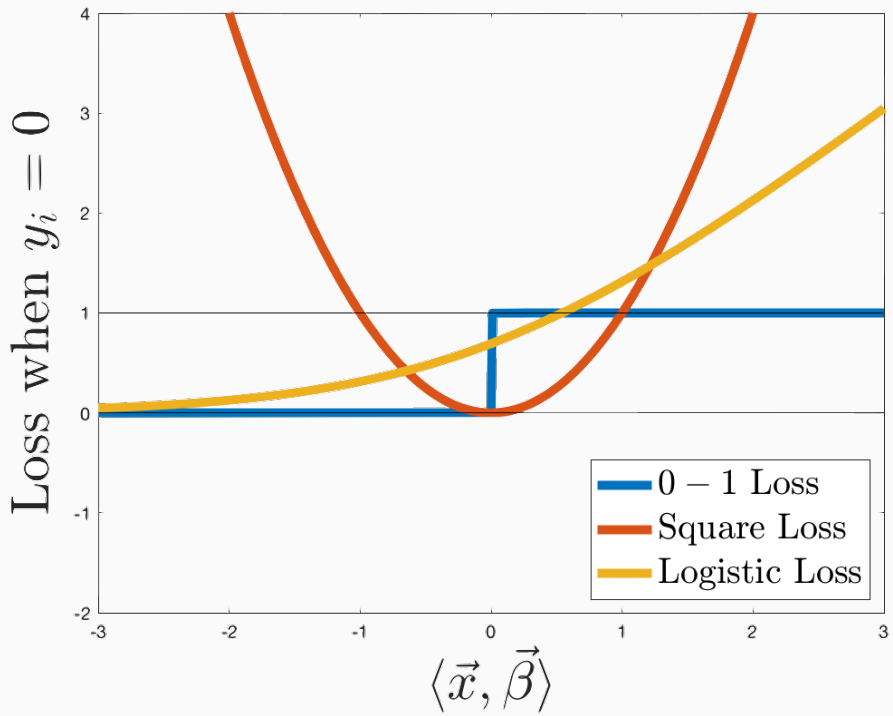
$$f_{\vec{\beta}}(\vec{x}) = \mathbb{1}\left[h_{\vec{\beta}}(\vec{x}) > 1/2\right]$$

- **Loss function**: "Logistic loss" aka "Cross-entropy loss"

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

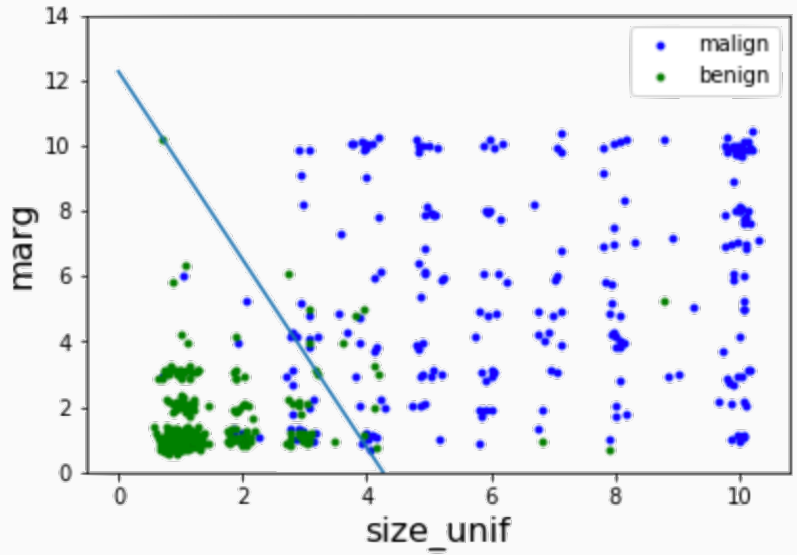
Logistic Loss:

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



LOGISTIC LOSS

- Convex function, 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.

ERROR IN CLASSIFICATION

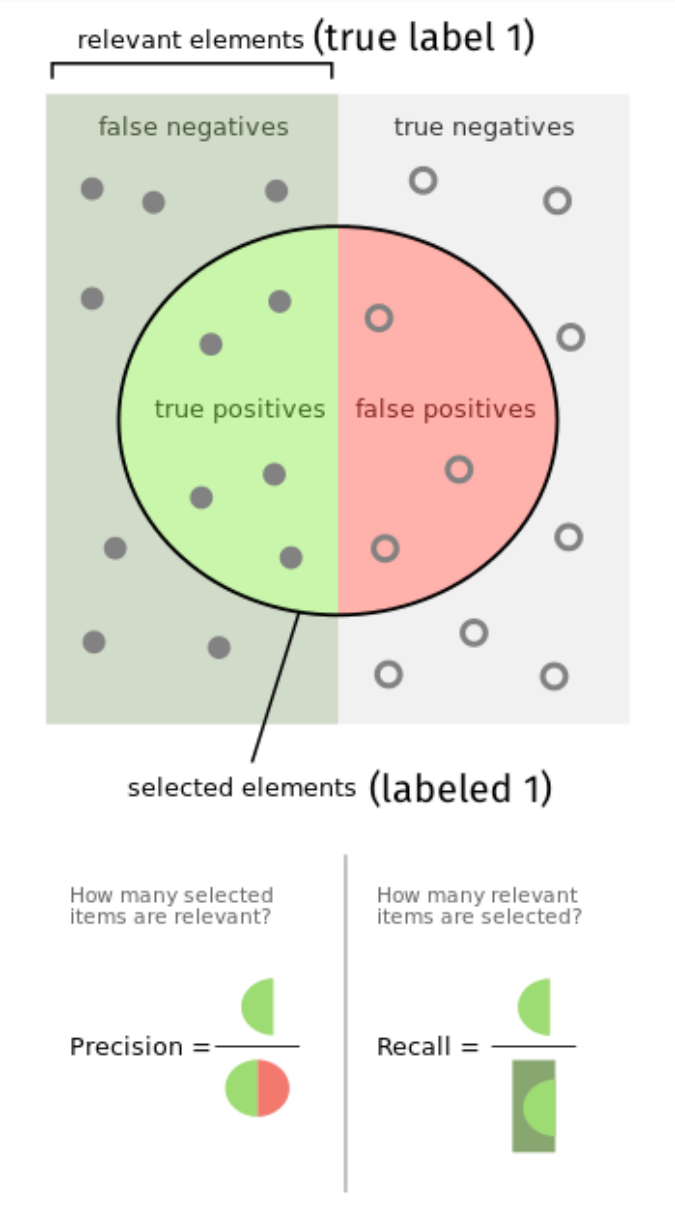
Once we have a classification algorithm, how do we judge its performance?

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

ERROR IN CLASSIFICATION

- **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 $\vec{\beta}$ via training 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 precision. Decreasing λ improves recall.