Natural Language Processing
Text classification

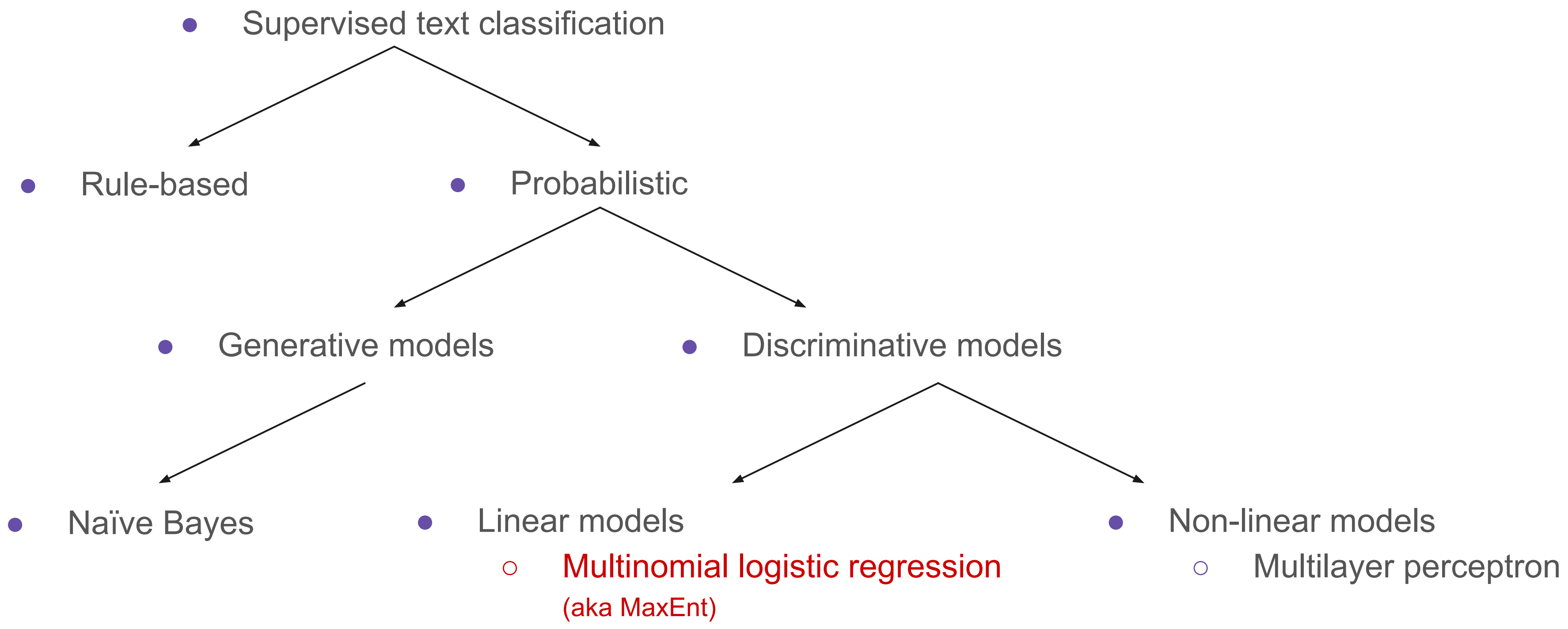
Sofia Serrano
sofias6@cs.washington.edu

Credit to Yulia Tsvetkov and Noah Smith for slides

Announcements

- A1 is due at 11:59pm this Friday
  - For details on how to tag and submit your assignment, see Leo's tutorial video linked in last night's Ed announcement
- We'll be holding extra office hours this week
  - See the office hour schedule google doc from the Ed announcement, also linked on the course website under "Announcements"
- Quiz 2 will be released on Canvas on Wednesday (1/25)
  - Available Wednesday 2:20pm through Thursday 2:20pm
  - 5 questions, 10 minutes

Logistic regression



Remaining loose ends for (multinomial) logistic regression

Regularization

A solution for overfitting

Add a **regularization** term $R(\theta)$ to the loss function (for now written as maximizing logprob rather than minimizing loss)

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) - \alpha R(\theta)$$

Idea: choose an $R(\theta)$ that penalizes large weights

- fitting the data well with lots of big weights not as good as fitting the data a little less well, with small weights

L2 regularization (ridge regression)

The sum of the squares of the weights

$$R(\theta) = ||\theta||_2^2 = \sum_{j=1}^{n} \theta_j^2$$

L2 regularized objective function:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \left[\sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) \right] - \alpha \sum_{j=1}^{n} \theta_j^2$$

L1 regularization (aka “lasso regression”)

The sum of the (absolute value of the) weights

$$R(\theta) \;=\; ||\theta||_1 = \sum_{i=1}^{n} |\theta_i|$$

L1 regularized objective function:

$$\hat{\theta} \;=\; \underset{\theta}{\operatorname{argmax}} \left[\sum_{1=i}^{m} \log P(y^{(i)}|x^{(i)}) \right] - \alpha \sum_{j=1}^{n} |\theta_j|$$

L1 regularization prefers sparse solutions. Why?

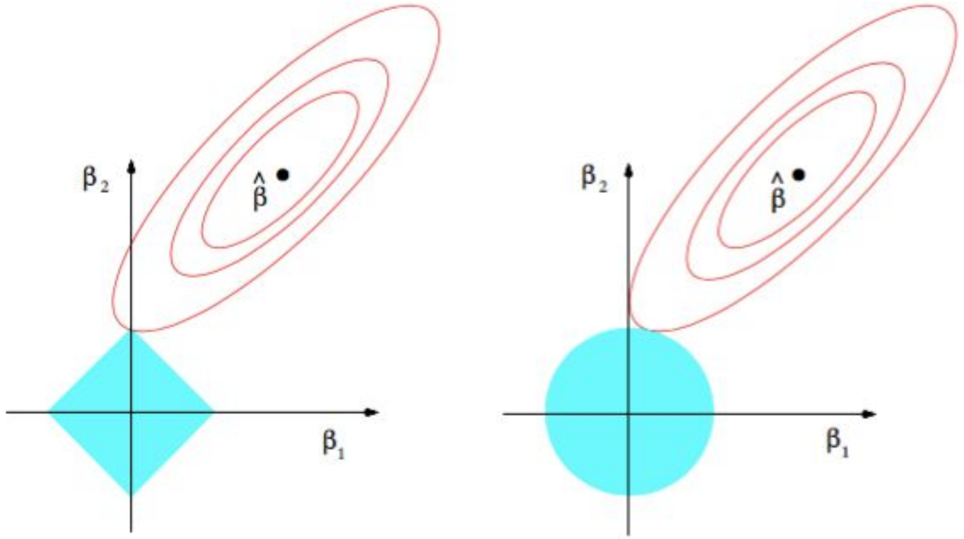
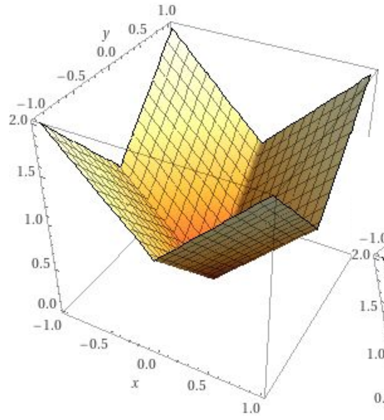
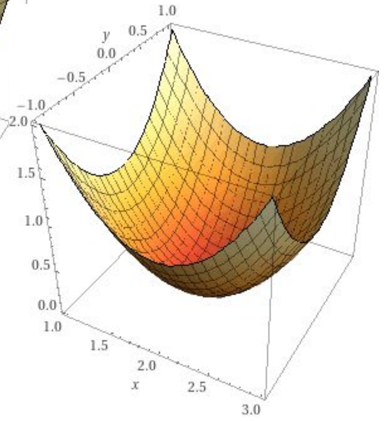


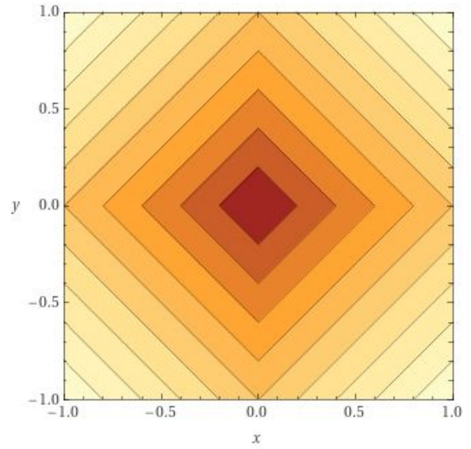
FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions $|\beta_1| + |\beta_2| \leq t$ and $\beta_1^2 + \beta_2^2 \leq t^2$, respectively, while the red ellipses are the contours of the least squares error function.*

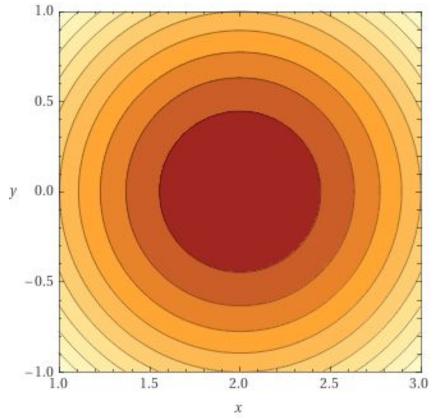
From *Elements of Statistical Learning* by Hastie, Tibshirani, and Friedman (Fig. 3.11)

Explaining those contour plots a bit more









Hyperparameters

Hyperparameters:

- Briefly, a special kind of parameter for an ML model
- Instead of being learned by algorithm from supervision (like regular parameters), they are chosen by algorithm designer.

The coefficient multiplied by a regularization term is an example of a **hyperparameter**.

The learning rate η is another hyperparameter.

- too high: the learner will take big steps and overshoot
- too low: the learner will take too long

Choosing hyperparameters

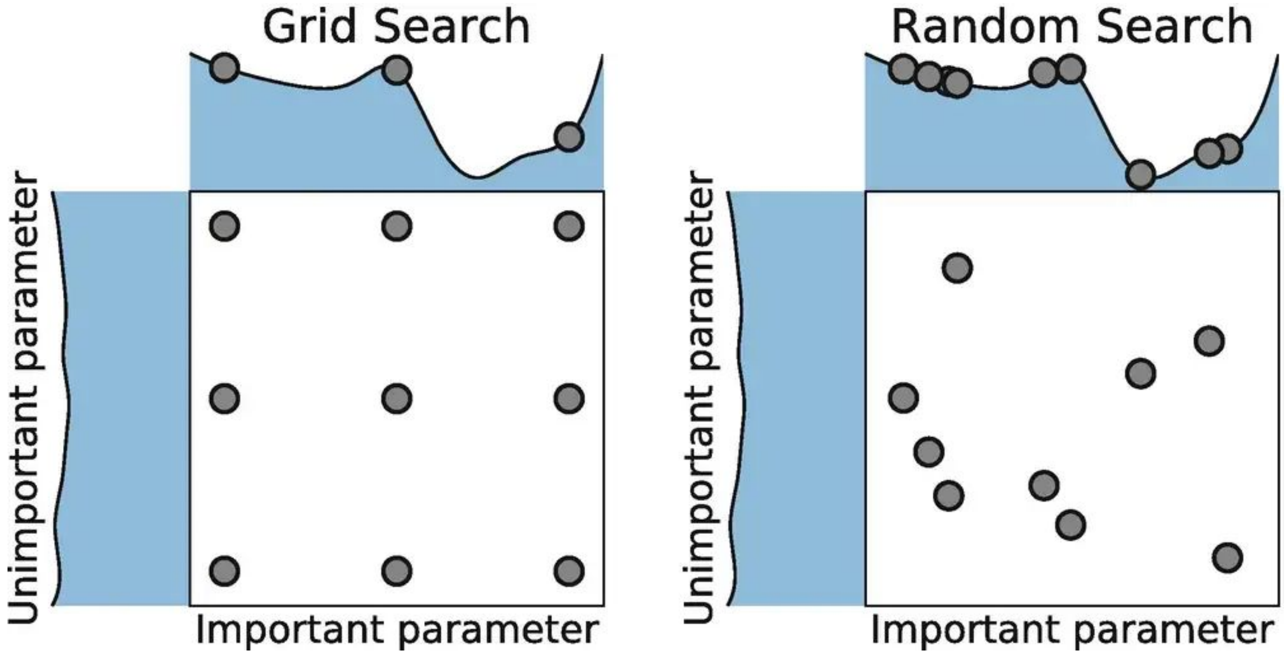


Figure by Elyse Lee

Components of a probabilistic machine learning classifier

Given m input/output pairs $(x^{(i)}, y^{(i)})$:

1. A **feature representation** for the input. For each input observation $x^{(i)}$, a vector of features $[x_1, x_2, \ldots, x_n]$. Feature j for input $x^{(i)}$ is x_j, more completely $x_1^{(i)}$, or sometimes $f_j(x)$.

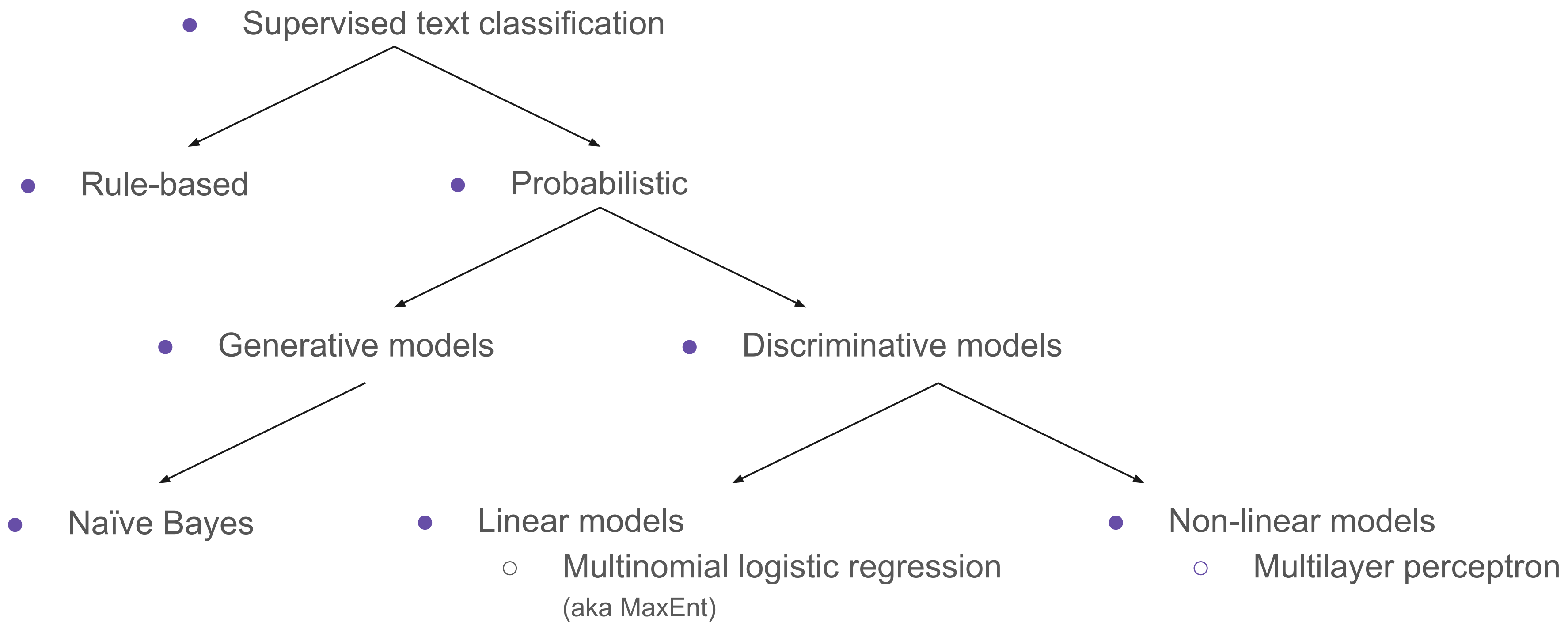
2. A **classification function** that computes $\hat{y}$ the estimated class, via $p(y|x)$, like the **sigmoid** or **softmax** functions

3. An **objective function** for learning, like **cross-entropy loss**

4. An algorithm for **optimizing** the objective function: **stochastic gradient descent**

The perceptron

Text classification models that we cover



- Supervised text classification
  - Rule-based
  - Probabilistic
    - Generative models
      - Naïve Bayes
    - Discriminative models
      - Linear models
        - Multinomial logistic regression (aka MaxEnt)
      - Non-linear models
        - Multilayer perceptron

Text classification models that we cover

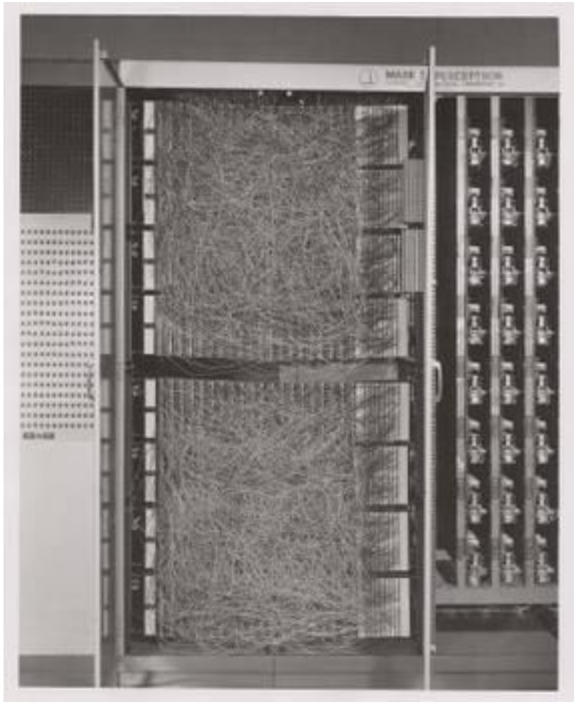
- Supervised text classification
  - Rule-based
  - Probabilistic
    - Generative models
      - Naïve Bayes
    - Discriminative models
      - Linear models
        - Multinomial logistic regression (aka MaxEnt)
      - Non-linear models
        - Multilayer perceptron

NOT the (vanilla) perceptron, though!

The perceptron

Developed with funding by the US Office of Naval Research in 1958

At the time, New York Times reported it to be "the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence." (from “A Sociological Study of the Official History of the Perceptrons Controversy”)



So what *is* this impressive perceptron?

$$y = \begin{cases} 0, & \text{if } \mathbf{w} \cdot \mathbf{x} + b \leq 0 \\ 1, & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \end{cases}$$

… isn't that exactly the same as binary logistic regression?

Well… yes and no!

If you're just doing classification? Yes.

If you're *learning* the model? No.

Here's our update rule for the perceptron:

$$\theta \leftarrow \theta + f(x, y) - f(x, \hat{y})$$

Comparing perceptron updates to binary LR updates

Binary LR update:

$$w_i \leftarrow w_i - \eta \frac{\partial L_{\mathrm{CE}}}{\partial w_i}$$

$$\frac{\partial L_{\mathrm{CE}}(\hat{y}, y)}{\partial w_j} = [\sigma(w \cdot x + b) - y]x_j$$

$$= (\hat{y} - y)\mathbf{x}_j$$

Perceptron update:

$$\theta \leftarrow \theta + f(x, y) - f(x, \hat{y})$$

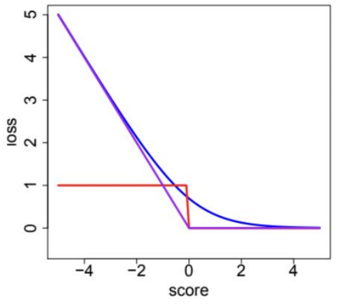
$$\theta \leftarrow \theta - (f(x, \hat{y} - f(x, y))$$

The perceptron uses the "hinge loss"

$$\text{log loss:}\quad \left(\log \sum_{\ell' \in \mathcal{L}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \ell')\right) - \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \ell)$$

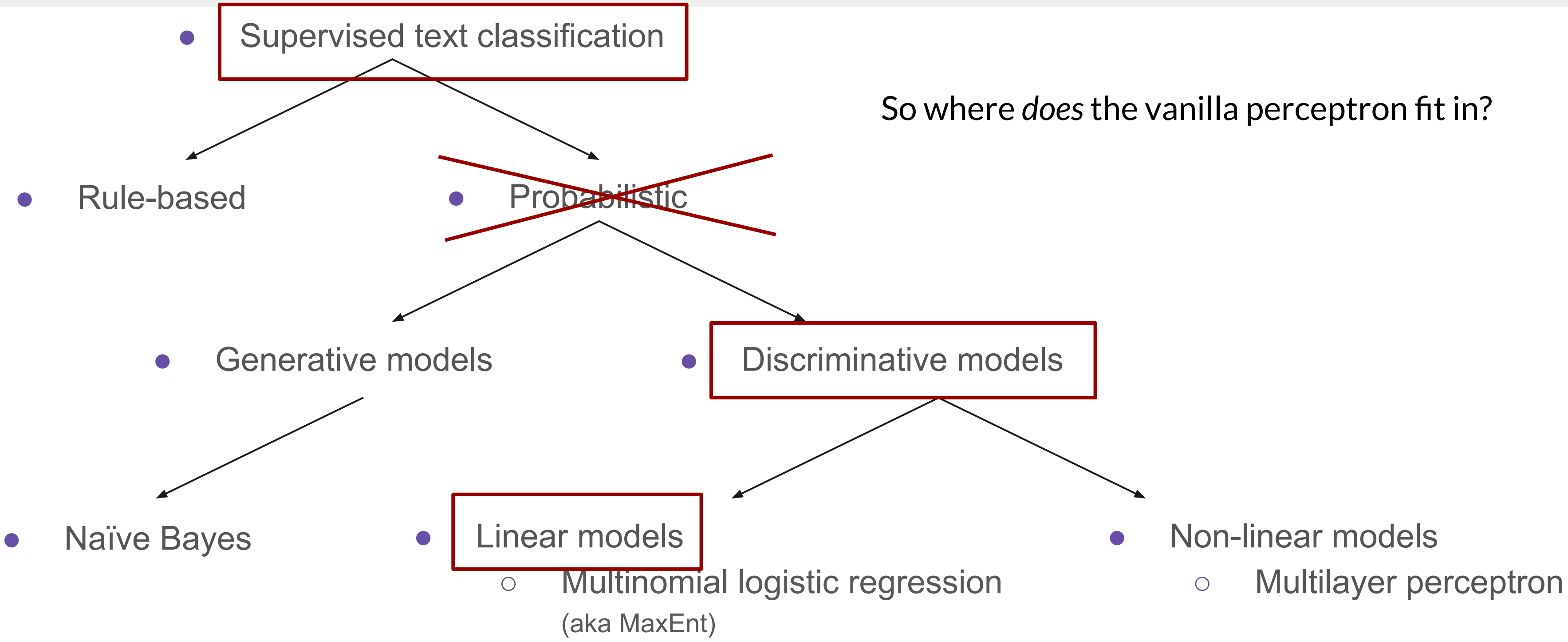
$$\text{hinge loss:}\quad \left(\max_{\ell' \in \mathcal{L}} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \ell')\right) - \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \ell)$$

In the binary case, where "score" is the linear score of the correct label:



In purple is the hinge loss, in blue is the log loss; in red is the "zero-one" loss (error).

Thinking back to our text classification tree



So where *does* the vanilla perceptron fit in?

So our tree's not really a tree?

Afraid not.

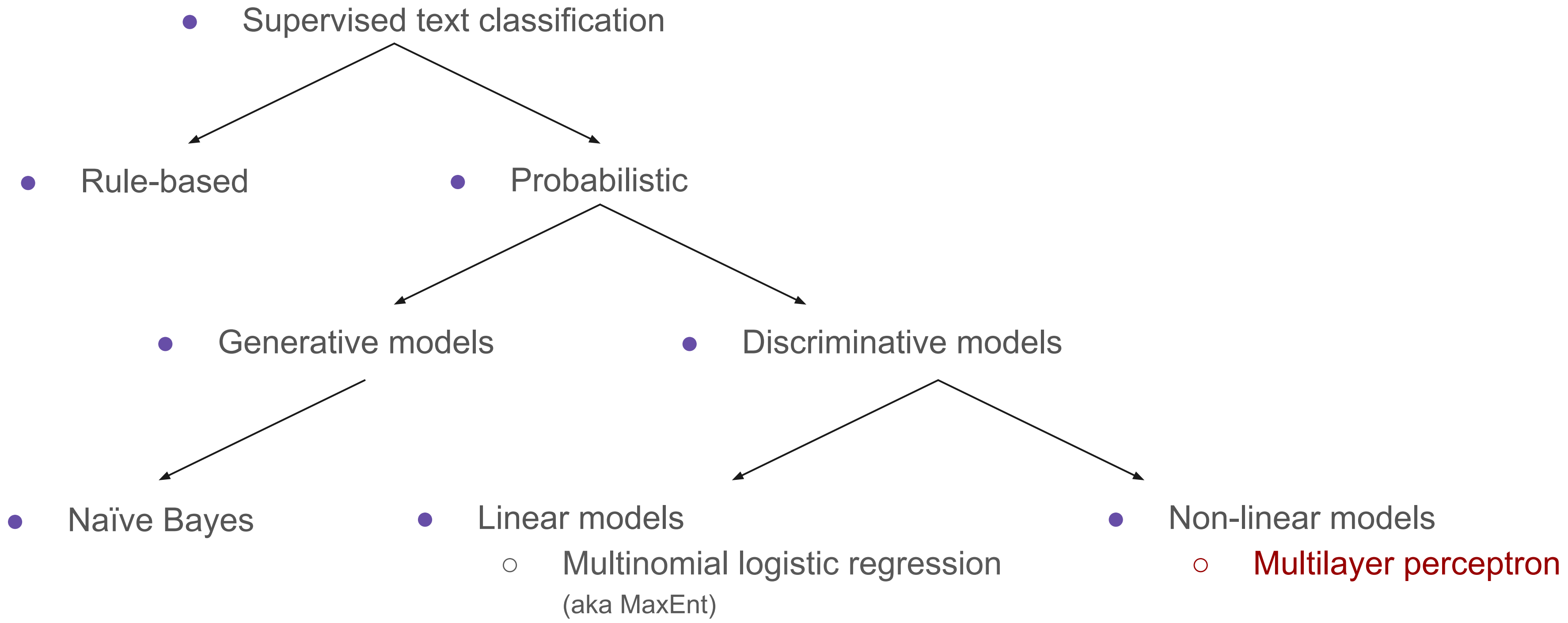
- Unsupervised, supervised, semi-supervised, weakly supervised all have associated research into probabilistic models
  - And rule-based doesn't actually care about these distinctions
- Some discriminative or generative models (usually discriminative, though) aren't actually probabilistic
  - Perceptron
  - Support Vector Machine (SVM)

  (unless you apply Platt scaling [fits a LR model to the outputs] or something)

- It *is* true that a model will either be linear or nonlinear, though (thank goodness)

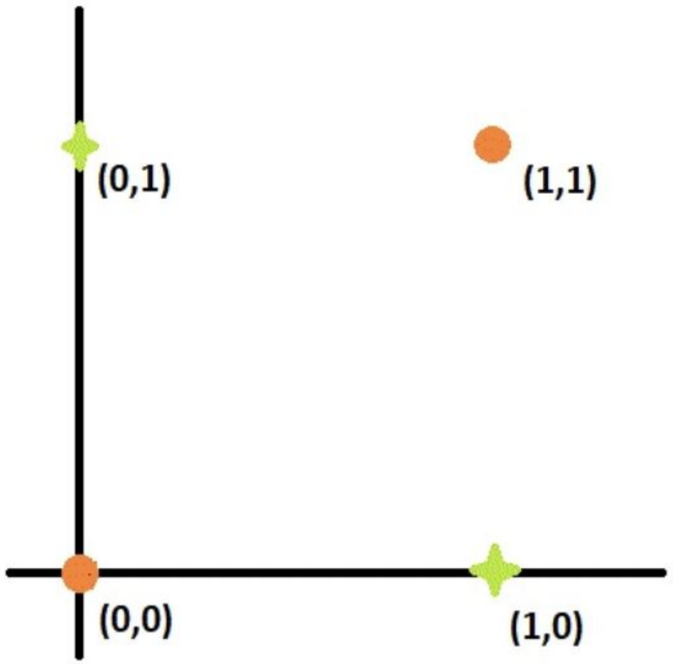
So what’s this multilayer perceptron, then?

Logistic regression



Why might we want a nonlinear classification model?

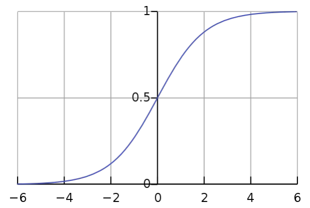
Simple motivating example: the XOR problem.

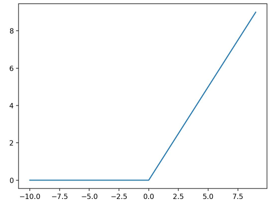


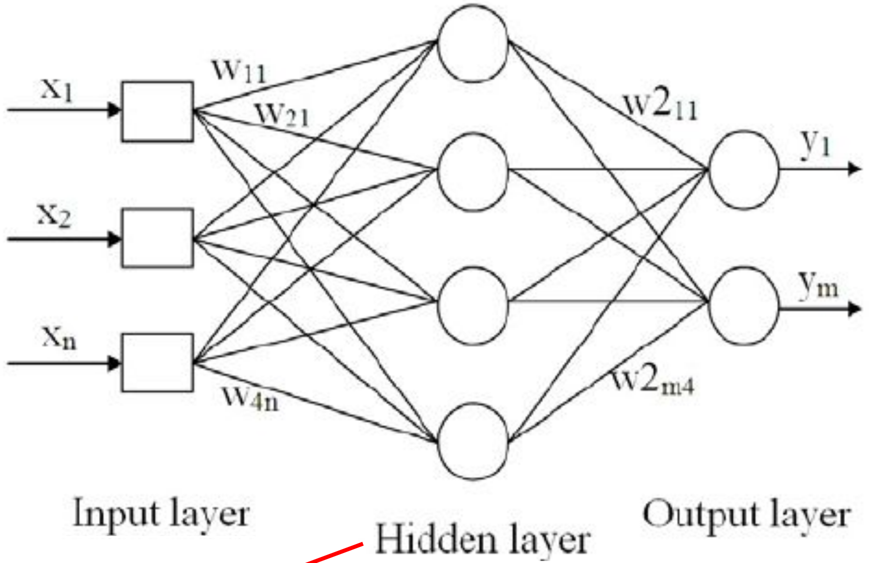
“Multilayer perceptron” is a bit of a misnomer

Perceptrons are linear, and this, well, is not.

A feedforward neural network where each layer includes a nonlinearity.







Contains an elementwise nonlinearity

Figure by Khursiah Zainal Mokhtar and Junita Mohamad-Saleh

Circling back to generative vs. discriminative models

Generative and discriminative models

imagenet

imagenet

Generative model

- Build a model of what's in a cat image
  - Knows about whiskers, ears, eyes
  - Assigns a probability to any image:
  - how cat-y is this image?
- Also build a model for dog images

imagenet

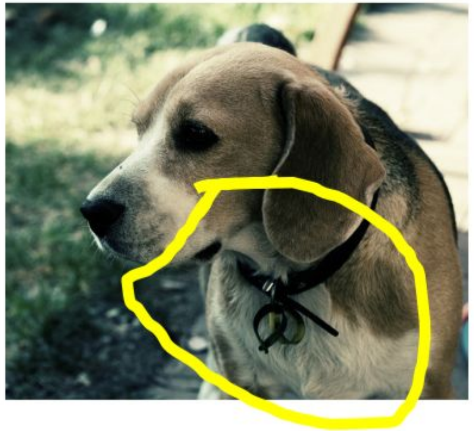
imagenet

Now given a new image:

Run both models and see which one fits better

Discriminative model

Just try to distinguish dogs from cats



Oh look, dogs have collars! Let's ignore everything else

Generative and discriminative models

- Generative model: a model that calculates the probability of the input data itself

$$P(X, Y)$$

joint

- Discriminative model: a model that calculates the probability of a latent trait given the data

$$P(Y \mid X)$$

conditional

Next class:

- Language modeling!