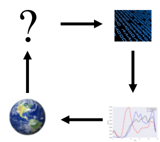
Classification & Logistic Regression
& maybe deep learning

Slides by:

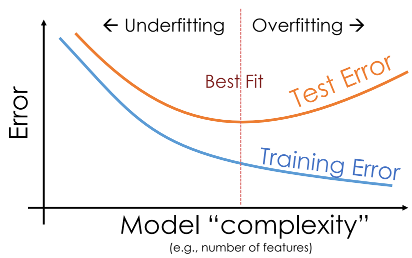
Joseph E. Gonzalez jegonzal@cs.berkeley.edu



Previously…

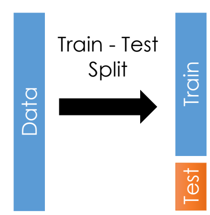
Training vs **Test** Error

Training error typically *under estimates* **test error.**



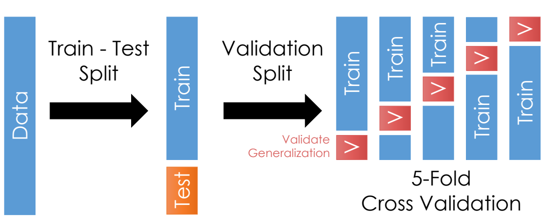
Generalization: *The Train-Test Split*

- **Training Data:** used to fit model
- **Test Data:** check generalization error
- How to split?
  - Randomly, Temporally, Geo…
  - Depends on application (usually randomly)
- What size? (90%-10%)
  - Larger training set → more complex models
  - Larger test set → better estimate of generalization error
  - Typically between 75%-25% and 90%-10%



You can only use the test dataset once after deciding on the model.

Generalization: *Validation Split*



Cross validation ***simulates multiple train test-splits*** *on the training data.*

Regularized Loss Minimization

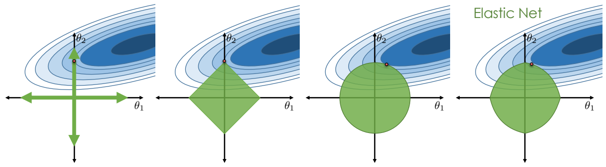
$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n}\sum_{i=1}^{n} \mathbf{Loss}\left(y_i, f_\theta(x_i)\right) \; + \lambda R(\theta)$$

Regularization Parameter

- Larger values of λ → more regularization
- **Confusing!:** Scikit-learn uses ***α* = 1/λ**
  - **Larger values of *α*** → less regularization

Different choices of $R(\theta)$

L[0] Norm Ball | L[1] Norm Ball | L[2] Norm Ball | L[1] + L[2] Norm
Elastic Net



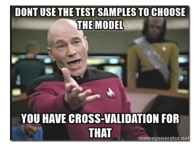
Ideal for Feature Selection but combinatorically difficult to optimize

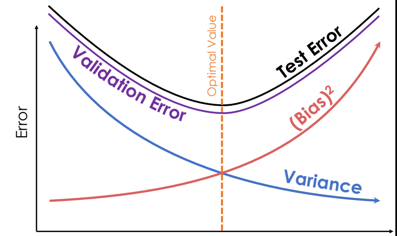
Encourages Sparse Solutions **Convex!**

Spreads weight over features **(robust)** does not encourage sparsity

Compromise Need to tune two regularization parameters

Determining the Optimal λ





- Value of λ determines bias-variance tradeoff
  - Larger values → more regularization → more bias → less variance
- Determined through cross validation

Using Scikit-Learn for Regularized Regression

`import sklearn.linear_model`

- **<u>Confusion Warning:</u>** Regularization parameter α = 1/λ
  - larger α ➔ less regularization → greater complexity → overfitting
- Lasso Regression (L1)
  - `linear_model.Lasso(alpha=3.0)`
  - `linear_model.LassoCV()` automatically picks α by cross-validation
- Ridge Regression (L2)
  - `linear_model.Ridge(alpha=3.0)`
  - `linear_model.RidgeCV()` automatically selects α by cross-validation
- Elastic Net (L1 + L2)
  - `linear_model.ElasticNet(alpha=3.0, l1_ratio = 2.0)`
  - `linear_model.ElasticNetCV()` automatically picks α by cross-validation

Standardization and the **Intercept** Term

`Height` = θ_1 `age_in_seconds` + θ_2 `weight_in_tons`

Small | Large

- Regularization penalized dimensions equally
- **Standardization**
  - Ensure that each dimensions has the same scale
  - centered around zero
- **Intercept Terms**
  - Typically don't regularize intercept term
  - Center *y* values (e.g., subtract mean)

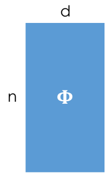
Standardization

For each dimension *k*:

$$z_k = \frac{x_k - \mu_k}{\sigma_k}$$

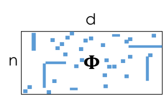
Regularization and **High-Dimensional Data**

Regularization is often used with high-dimensional data



Tall Skinny Matrix

- n >> d
- typically dense
- Regularization can help with complex feature transformations

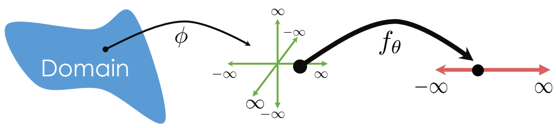


High-dimensional sparse matrix

- d > n
- **requires** regularization
- **Goal:** to determine **informative dimensions**
  - Consider L1 (Lasso) Regularization.
- **Goal:** is to make **robust predictions**
  - Consider L2 (+L1) Regularization

Today
Classification

So far ….

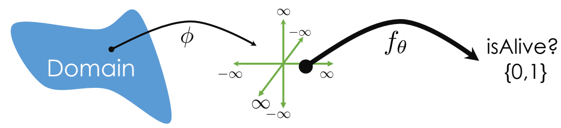


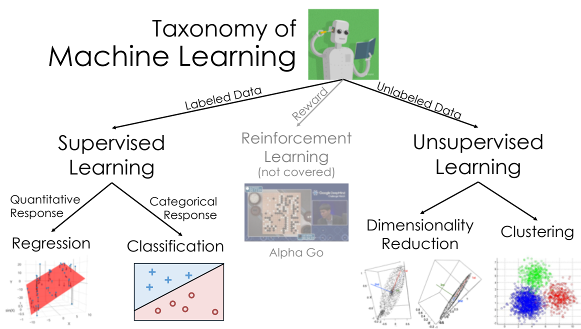
Squared Loss

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} (y_i - f_\theta(x_i))^2 + \lambda \mathbf{R}(\theta)$$

Regularization

Classification



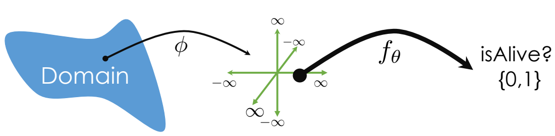


Kinds of Classification

Predicting a categorical variable

- **Binary** classification: Two classes
  - Examples: *Spam/Not Spam, churn/stay*
- **Multiclass** classification: Many classes (>2)
  - Examples: *Image labeling (Cat, Dog, Car), Next word in a sentence …*
- **Structured prediction** tasks (Classification)
  - Multiple related predictions
  - Examples: *Translation, Voice recognition*

Classification



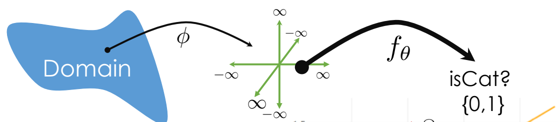
Can we just use least squares?

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} (y_i - f_\theta(x_i))^2 + \lambda \mathbf{R}(\theta)$$

Squared Loss

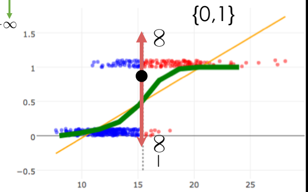
Python Demo

Classification

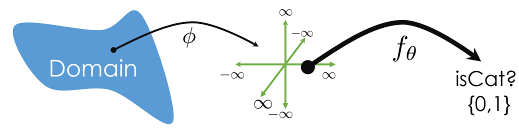


Can we just use least squares?

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n}\sum_{i=1}^{n}(y_i - f_\theta(x_i))^2 + \lambda \mathbf{R}(\theta)$$



Classification



Can we just use least squares?

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n}\sum_{i=1}^{n}(y_i - f_\theta(x_i))^2 + \lambda \mathbf{R}(\theta)$$

- Yes …
- Need …
- … interpret model …
- Sensitive to outliers

Don't use Least Squares for **Classification**

Defining a New Model for Classification

Logistic Regression

- Widely used models for **binary classification:**

$x =$ "Get a FREE sample …"

$\phi(x) = [2.0, 0.0, \ldots, 1.0, 0.5]$ ➡ $y = 1$

1 = "Spam"
0 = "Ham"

Why is ham good and spam bad? … (https://www.youtube.com/watch?v=anwy2MPT5RE)

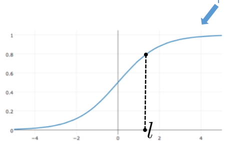
- Models the probability of $y=1$ given x

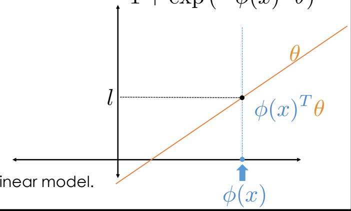
$$\hat{\mathbf{P}}_\theta(y = 1 \mid x) = \sigma(\phi(x)^T\theta) = \frac{1}{1 + \exp(-\phi(x)^T\theta)}$$

Logistic Regression

Model: $\hat{\mathbf{P}}_\theta(y = 1 \mid x) = \sigma(\phi(x)^T\theta) = \frac{1}{1 + \exp(-\phi(x)^T\theta)}$

Linear Model





Generalized Linear Model:
Non-linear transformation of a linear model.

- Widely used models for **binary classification:**

$x =$ "Get a FREE sample …"

$\phi(x) = [2.0, 0.0, \ldots, 1.0, 0.5]$ ➡ $y = 1$

1 = "Spam"
0 = "Ham"

Why is ham good and spam bad? … (https://www.youtube.com/watch?v=anwy2MPT5RE)

- Models the probability of $y=1$ given x

$$\hat{\mathbf{P}}_\theta(y = 1 \mid x) = \sigma(\phi(x)^T\theta) = \frac{1}{1 + \exp(-\phi(x)^T\theta)}$$

$$\hat{\mathbf{P}}_\theta(y = 0 \mid x) = 1 - \hat{\mathbf{P}}_\theta(y = 1 \mid x)$$

Python Demo

The Logistic Regression Model

Model: $\hat{\mathbf{P}}_\theta\left(y = 1 \,|\, x\right) \;= \sigma(\phi(x)^T\theta) \;= \dfrac{1}{1 + \exp\left(-\phi(x)^T\theta\right)}$

How do we fit the model to the data?

Defining the Loss

Could we use the Squared Loss

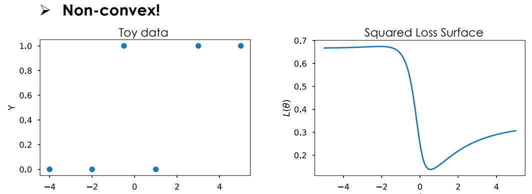
- What about squared loss and the new model:

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sigma\left(\phi(x_i)^T\theta\right)\right)^2$$

  - Tries to match probability with 0/1 labels.
  - Occasionally used in some neural network applications
  - **Non-convex!**

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sigma\left(\phi(x_i)^T\theta\right)\right)^2$$

- Tries to match probability with 0/1 labels.
- Occasionally used in some neural network applications
- **Non-convex!**



Defining the Cross Entropy Loss

Loss Function

- We want our model to be close to the data:

$$\hat{\mathbf{P}}_\theta\,(y = 1 \,|\, x) \;\approx\; \mathbf{P}\,(y = 1 \,|\, x)$$

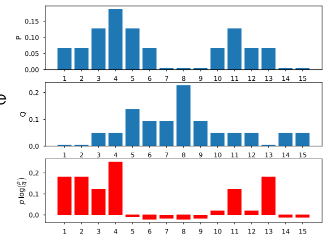
- Kullback–Leibler (KL) Divergence provides a measure of similarity between two distributions:
  - Between two discrete distributions P and Q

$$\mathbf{D}(P||Q) = \sum_{k=1}^{K} P(k) \log\left(\frac{P(k)}{Q(k)}\right)$$

Kullback–Leibler (KL) Divergence

$$\mathbf{D}(P||Q) = \sum_{k=1}^{K} P(k) \log\left(\frac{P(k)}{Q(k)}\right)$$

- The average log difference between P and Q weighted by P
- Does not penalize mismatch for rare events with respect to P



Loss Function

- We want our model to be close to the data:

$$\hat{\mathbf{P}}_\theta\,(y = 1 \,|\, x) \;\approx\; \mathbf{P}\,(y = 1 \,|\, x)$$

- Kullback–Leibler (KL) divergence for classification
  - For a **single** (x,y) data point

Binary Classification ($K=2$)

$$\mathbf{D}_{KL}(\mathbf{P}\,||\,\hat{\mathbf{P}}_\theta) = \sum_{k=1}^{K} \mathbf{P}\,(y = k \,|\, x) \log\left(\frac{\mathbf{P}\,(y = k \,|\, x)}{\hat{\mathbf{P}}_\theta\,(y = k \,|\, x)}\right)$$

  - Average KL Divergence for all the data:

- Kullback–Leibler (KL) divergence for classification
  - For a **single** (x,y) data point

Binary Classification ($K=2$)

$$\mathbf{D}_{KL}(\mathbf{P}\,||\,\hat{\mathbf{P}}_\theta) = \sum_{k=1}^{K} \mathbf{P}\,(y = k \,|\, x) \log\left(\frac{\mathbf{P}\,(y = k \,|\, x)}{\hat{\mathbf{P}}_\theta\,(y = k \,|\, x)}\right)$$

  - Average KL Divergence for all the data:

$$\arg\min_\theta \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} \mathbf{P}\,(y_i = k \,|\, x_i) \log\left(\frac{\mathbf{P}\,(y_i = k \,|\, x_i)}{\hat{\mathbf{P}}_\theta\,(y_i = k \,|\, x_i)}\right)$$

Log(a/b) = Log(a) – Log(b)

Doesn't depend on θ

$$\mathbf{P}\,(y_i = k \,|\, x_i) \log\left(\mathbf{P}\,(y_i = k \,|\, x_i)\right)$$

$$-\;\mathbf{P}\,(y_i = k \,|\, x_i) \log\left(\hat{\mathbf{P}}_\theta\,(y_i = k \,|\, x_i)\right)$$

- Average **cross entropy loss**

$$\arg\min_\theta \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} -\,\mathbf{P}\,(y_i = k \,|\, x_i) \log\left(\hat{\mathbf{P}}_\theta\,(y_i = k \,|\, x_i)\right)$$

Summing from k = 0 to 1 and not k = 1 to 2 (to be consistent with 0/1 labels):

$$\arg\min_\theta -\frac{1}{n}\sum_{i=1}^{n}\left(\begin{array}{l}\mathbf{P}\,(y_i = 0 \,|\, x_i) \log\left(\hat{\mathbf{P}}_\theta\,(y_i = 0 \,|\, x_i)\right) + \\ \quad \mathbf{P}\,(y_i = 1 \,|\, x_i) \log\left(\hat{\mathbf{P}}_\theta\,(y_i = 1 \,|\, x_i)\right)\end{array}\right)$$

$\mathbf{P}\,(y_i = 1 \,|\, x_i) = y_i$	$\hat{\mathbf{P}}_\theta\,(y_i = 1 \,|\, x_i) = \sigma(\phi(x_i)^T\theta)$
$\mathbf{P}\,(y_i = 0 \,|\, x_i) = (1 - y_i)$	$\hat{\mathbf{P}}_\theta\,(y_i = 0 \,|\, x_i) = 1 - \sigma(\phi(x_i)^T\theta)$

- Average **cross entropy loss**

Binary Classification ($K=2$)

$$\arg\min_\theta \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} -\,\mathbf{P}\,(y_i = k \,|\, x_i) \log\left(\hat{\mathbf{P}}_\theta\,(y_i = k \,|\, x_i)\right)$$

$$\arg\min_\theta -\frac{1}{n}\sum_{i=1}^{n}\left(\begin{array}{l}(1 - y_i) \log\left(1 - \sigma(\phi(x_i)^T\theta)\right) + \\ \quad y_i \quad \log\left(\sigma(\phi(x_i)^T\theta)\right)\end{array}\right)$$

Rewriting on one line:

$$\arg\min_\theta -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log\left(\sigma\left(\phi(x_i)^T\theta\right)\right) + (1 - y_i)\log\left(1 - \sigma\left(\phi(x_i)^T\theta\right)\right)\right)$$

➢ Average **cross entropy loss**

$$\arg\min_\theta \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} - \mathbf{P}\left(y_i = k \,|\, x_i\right)\log\left(\hat{\mathbf{P}}_\theta\left(y_i = k \,|\, x_i\right)\right)$$

$$\arg\min_\theta -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log\left(\sigma\left(\phi(x_i)^T\theta\right)\right) + (1-y_i)\log\left(1-\sigma\left(\phi(x_i)^T\theta\right)\right)\right)$$

Expanding

$$\log\left(1-\sigma\left(\phi(x_i)^T\theta\right)\right) - y_i\log\left(1-\sigma\left(\phi(x_i)^T\theta\right)\right)$$

Grouping Terms

Copy & Paste

$$\arg\min -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log\left(\frac{\sigma\left(\phi(x_i)^T\theta\right)}{1-\sigma\left(\phi(x_i)^T\theta\right)}\right) + \log\left(1-\sigma\left(\phi(x_i)^T\theta\right)\right)\right)$$

$$\arg\min -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log\left(\frac{\sigma\left(\phi(x_i)^T\theta\right)}{1-\sigma\left(\phi(x_i)^T\theta\right)}\right) + \log\left(1-\sigma\left(\phi(x_i)^T\theta\right)\right)\right)$$

Definition

$$\sigma(\phi(x)^T\theta) = \frac{1}{1+\exp\left(-\phi(x)^T\theta\right)}$$

$$\log \frac{\frac{1}{1+\exp(-\phi(x_i)^T\theta)} \times \left(1+\exp\left(-\phi(x_i)^T\theta\right)\right)}{1-\frac{1}{1+\exp(-\phi(x_i)^T\theta)} \times \left(1+\exp\left(-\phi(x_i)^T\theta\right)\right)}$$

$$\overset{\text{Alg.}}{=} \log\frac{1}{1+\exp\left(-\phi(x_i)^T\theta\right)-1} \overset{\text{Alg.}}{=} \log\exp\left(\phi(x_i)^T\theta\right) \overset{\text{Alg.}}{=} \phi(x_i)^T\theta$$

$$\arg\min -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log\left(\frac{\sigma\left(\phi(x_i)^T\theta\right)}{1-\sigma\left(\phi(x_i)^T\theta\right)}\right) + \log\left(1-\sigma\left(\phi(x_i)^T\theta\right)\right)\right)$$

$$= \phi(x_i)^T\theta$$

Definition

$$\sigma(\phi(x)^T\theta) = \frac{1}{1+\exp\left(-\phi(x)^T\theta\right)}$$

A Linear mode of the "Log odds"

$$\log\left(\frac{\sigma\left(\phi(x_i)^T\theta\right)}{1-\sigma\left(\phi(x_i)^T\theta\right)}\right) = \underbrace{\phi(x_i)^T\theta}_{\text{Linear Model}} = \underbrace{\log\left(\frac{\hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right)}{\hat{\mathbf{P}}_\theta\left(y_i = 0 \,|\, x_i\right)}\right)}_{\text{Log odds}}$$

A Linear mode of the "Log odds"

$$\log\left(\frac{\sigma\left(\phi(x_i)^T\theta\right)}{1-\sigma\left(\phi(x_i)^T\theta\right)}\right) = \underbrace{\phi(x_i)^T\theta}_{\text{Linear Model}} = \underbrace{\log\left(\frac{\hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right)}{\hat{\mathbf{P}}_\theta\left(y_i = 0 \,|\, x_i\right)}\right)}_{\text{Log odds}}$$

Implications?

$$\phi(x_i)^T\theta = 0 \overset{\exp(0)=1}{\Rightarrow} \hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right) = \hat{\mathbf{P}}_\theta\left(y_i = 0 \,|\, x_i\right)$$

$$\phi(x_i)^T\theta > 0 \overset{\exp(\epsilon)>1}{\Rightarrow} \hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right) > \hat{\mathbf{P}}_\theta\left(y_i = 0 \,|\, x_i\right)$$

$$\phi(x_i)^T\theta < 0 \overset{\exp(-\epsilon)<1}{\Rightarrow} \hat{\mathbf{P}}_\theta\left(y_i = 1 \,|\, x_i\right) < \hat{\mathbf{P}}_\theta\left(y_i = 0 \,|\, x_i\right)$$

for any positive ϵ

$$\arg\min -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log\left(\frac{\sigma\left(\phi(x_i)^T\theta\right)}{1-\sigma\left(\phi(x_i)^T\theta\right)}\right) + \log\left(1-\sigma\left(\phi(x_i)^T\theta\right)\right)\right)$$

$$= \phi(x_i)^T\theta$$

Definition

$$\sigma(\phi(x)^T\theta) = \frac{1}{1+\exp\left(-\phi(x)^T\theta\right)}$$

Substituting the above result:

$$\arg\min_\theta -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(1-\sigma\left(\phi(x_i)^T\theta\right)\right)\right)$$

Defn. of σ

$$1-\frac{1}{1+\exp\left(-\phi(x_i)^T\theta\right)} \overset{\text{Alg.}}{=} \frac{\exp\left(-\phi(x_i)^T\theta\right)}{1+\exp\left(-\phi(x_i)^T\theta\right)} \overset{\text{Alg.}}{=} \frac{1}{1+\exp\left(\phi(x_i)^T\theta\right)}$$

$\times \exp\left(\phi(x_i)^T\theta\right)$ (numerator and denominator)

$$\arg\min_\theta -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(1-\sigma\left(\phi(x_i)^T\theta\right)\right)\right)$$

Defn. of σ

$$1-\frac{1}{1+\exp\left(-\phi(x_i)^T\theta\right)} \overset{\text{Alg.}}{=} \frac{\exp\left(-\phi(x_i)^T\theta\right)}{1+\exp\left(-\phi(x_i)^T\theta\right)} \overset{\text{Alg.}}{=} \frac{1}{1+\exp\left(\phi(x_i)^T\theta\right)}$$

$\times \exp\left(\phi(x_i)^T\theta\right)$ (numerator and denominator)

Definition

$$\sigma(\phi(x)^T\theta) = \frac{1}{1+\exp\left(-\phi(x)^T\theta\right)}$$

Defn. of σ

$$= \sigma\left(-\phi(x_i)^T\theta\right)$$

Simplified Loss Minimization Problem

$$\arg\min_\theta -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

The Loss for Logistic Regression

- Average **cross entropy** (simplified):

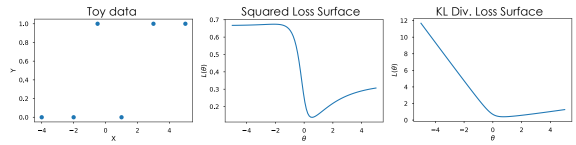
$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

- Equivalent to (derived from) **minimizing the KL divergence**
- Also equivalent to **maximizing the log-likelihood of the data** … (not covered in DS100 this semester)

Is this loss function reasonable?

Convexity Using Pictures

$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

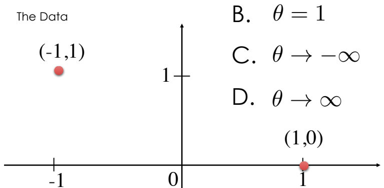


What is the value of θ?

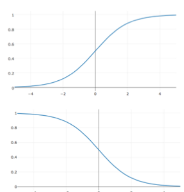
$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

http://bit.ly/ds100-sp18-cla

Assume: $\phi(x) = x$



A. $\theta = -1$
B. $\theta = 1$
C. $\theta \to -\infty$
D. $\theta \to \infty$



What is the value of θ?

$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

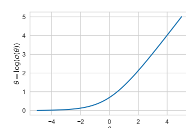
Assume: $\phi(x) = x$

For the point (-1,1): $y_i\phi(x_i)^T = -1$

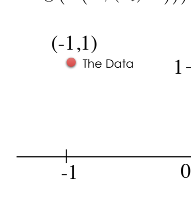
$-\phi(x_i)^T = 1$

Objective:

$\theta - \log\left(\sigma\left(\theta\right)\right)$



$\theta \to -\infty$



What is the value of θ?

$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

Assume: $\phi(x) = x$

For the point (-1,1): $\theta - \log\left(\sigma\left(\theta\right)\right)$

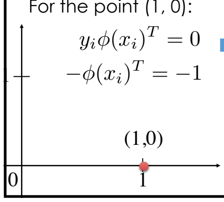
$\theta \to -\infty$

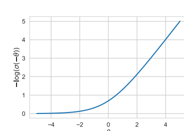
For the point (1, 0):

$y_i\phi(x_i)^T = 0$

$-\phi(x_i)^T = -1$

$0 - \log(\sigma(-\theta))$





$\theta \to -\infty$

What is the value of θ?

$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

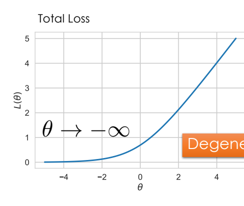
Assume: $\phi(x) = x$

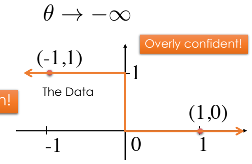
For the point (-1,1): $\theta - \log\left(\sigma\left(\theta\right)\right)$

$\theta \to -\infty$

For the point (1, 0): $0 - \log(\sigma(-\theta))$

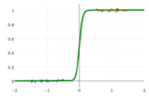
$\theta \to -\infty$





Linearly Separable Data

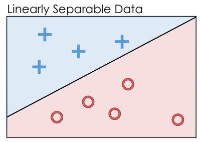
- A classification dataset is said to be linearly separable if there exists a hyperplane that separates the two classes.
- If data is linearly separable, logistic regression requires regularization



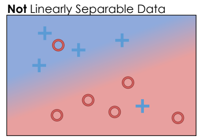
Weights go to infinity!

Solution?

Linearly Separable Data



Not Linearly Separable Data

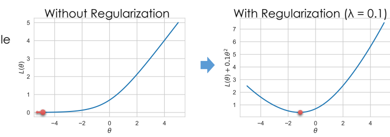


Adding Regularization to Logistic Regression

$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right) + \lambda \sum_{j=1}^{d}\theta_j^2$$

- Prevents weights from diverging on linearly separable data

Earlier Example



Minimize the Loss

Logistic Loss Function

- Average KL divergence (simplified)

$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

- Take Derivative:

$$\nabla_\theta \mathbf{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \nabla_\theta y_i \phi(x_i)^T\theta + \nabla_\theta \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)$$

$$= -\frac{1}{n}\sum_{i=1}^{n} y_i \phi(x_i) + \nabla_\theta \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)$$

- Average KL divergence (simplified)

$$\arg\min_{\theta} -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

- Take Derivative:

$$\nabla_\theta \mathbf{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \nabla_\theta y_i \phi(x_i)^T\theta + \nabla_\theta \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)$$

$$= -\frac{1}{n}\sum_{i=1}^{n} y_i \phi(x_i) + \nabla_\theta \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)$$

$$= -\frac{1}{n}\sum_{i=1}^{n} y_i \phi(x_i) + \frac{1}{\sigma\left(-\phi(x_i)^T\theta\right)} \nabla_\theta \sigma\left(-\phi(x_i)^T\theta\right)$$

- Take Derivative:

$$\nabla_\theta \mathbf{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y_i \phi(x_i) + \frac{1}{\sigma\left(-\phi(x_i)^T\theta\right)} \nabla_\theta \sigma\left(-\phi(x_i)^T\theta\right)$$

Useful Identity

$$\frac{\partial}{\partial t}\sigma(t) = \frac{\partial}{\partial t}\frac{1}{1+e^{-t}} \overset{\text{Chain Rule}}{=} \frac{-1}{(1+e^{-t})^2}\frac{\partial}{\partial t}\left(1+e^{-t}\right)$$

$$\overset{\text{Chain Rule}}{=} \frac{e^{-t}}{(1+e^{-t})^2} \overset{\text{Alg.}}{=} \left(\frac{1}{1+e^{-t}}\right)\left(\frac{e^{-t}}{1+e^{-t}}\right)$$

$$\overset{\text{Alg.}}{=} \left(\frac{1}{1+e^{-t}}\right)\left(\frac{1}{e^{t}+1}\right) \overset{\text{Defn. of } \sigma}{=} \sigma(t)\sigma(-t)$$

- Take Derivative:

$$\nabla_\theta \mathbf{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y_i \phi(x_i) + \frac{1}{\sigma\left(-\phi(x_i)^T\theta\right)} \nabla_\theta \sigma\left(-\phi(x_i)^T\theta\right)$$

Useful Identity $\frac{\partial}{\partial t}\sigma(t) = \sigma(t)\sigma(-t)$

$$= -\frac{1}{n}\sum_{i=1}^{n} y_i \phi(x_i) + \frac{\sigma\left(-\phi(x_i)^T\theta\right)}{\sigma\left(-\phi(x_i)^T\theta\right)} \sigma\left(\phi(x_i)^T\theta\right) \nabla_\theta(-\phi(x_i)^T\theta)$$

$$= -\frac{1}{n}\sum_{i=1}^{n} \left(y_i - \sigma\left(\phi(x_i)^T\theta\right)\right)\phi(x_i)$$

Logistic Loss Function

- Average KL divergence (simplified)

$$\arg\min_\theta -\frac{1}{n}\sum_{i=1}^{n}\left(y_i\phi(x_i)^T\theta + \log\left(\sigma\left(-\phi(x_i)^T\theta\right)\right)\right)$$

- Take Derivative:

$$\nabla_\theta \mathbf{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \left(y_i - \sigma\left(\phi(x_i)^T\theta\right)\right)\phi(x_i)$$

- Set derivative = 0 and solve for θ
  - No general analytic solution
  - Solved using numeric methods

The Gradient Descent Algorithm

$\theta^{(0)} \leftarrow$ initial vector (random, zeros …)

For τ from 0 to convergence:

$$\theta^{(\tau+1)} \leftarrow \theta^{(\tau)} - \rho(\tau)\left(\nabla_\theta \mathbf{L}(\theta)\Big|_{\theta=\theta^{(\tau)}}\right)$$

Evaluated at

- ***ρ(τ)*** is the step size (learning rate)
  - typically $1/\tau$
- Converges when gradient is ≈ 0 (or we run out of patience)

Gradient Descent for Logistic Regression

Logistic Regression

$\theta^{(0)} \leftarrow$ initial vector (random, zeros …)

For τ from 0 to convergence:

$$\theta^{(\tau+1)} \leftarrow \theta^{(\tau)} - \rho(\tau)\left(\frac{1}{n}\sum_{i=1}^{n}\left(\sigma\left(\phi(x_i)^T\theta^{(\tau)}\right) - y_i\right)\phi(x_i)\right)$$

- ***ρ(τ)*** is the step size (learning rate)
  - typically $1/\tau$
- Converges when gradient is ≈ 0 (or we run out of patience)

Stochastic Gradient Descent

- For many learning problems the gradient is a sum:

$$\nabla_\theta \mathbf{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(\sigma\left(\phi(x_i)^T\theta\right) - y_i\right)\phi(x_i)$$

  - For large n this can be costly
- What if we approximated the gradient by looking at a few random points:

$$\nabla_\theta \mathbf{L}(\theta) \approx \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\left(\sigma\left(\phi(x_i)^T\theta\right) - y_i\right)\phi(x_i)$$

- What if we approximated the gradient by looking at a few random points:

$$\nabla_\theta \mathbf{L}(\theta) \approx \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\left(\sigma\left(\phi(x_i)^T\theta\right) - y_i\right)\phi(x_i)$$

Batch Size

Random sample of records

- This is a reasonable estimator for the gradient
  - Unbiased …
- Often batch size is one! (why is this helpful)
  - Fast to compute!
- A key ingredient in the recent success of deep learning

Stochastic Gradient Descent

$\theta^{(0)} \leftarrow$ initial vector (random, zeros …)

For τ from 0 to convergence:

$\mathcal{B} \sim$ Random subset of indices

$$\theta^{(\tau+1)} \leftarrow \theta^{(\tau)} - \rho(\tau)\left(\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \nabla_\theta \mathbf{L}_i(\theta)\Big|_{\theta=\theta^{(\tau)}}\right)$$

Decomposable Loss $$\mathbf{L}(\theta) = \sum_{i=1}^{n} \mathbf{L}_i(\theta) = \sum_{i=1}^{n} \mathbf{L}(\theta, x_i, y_i)$$

Loss can be written as a sum of the loss on each record.

Assuming Decomposable Loss Functions

Gradient Descent

$\theta^{(0)} \leftarrow$ initial vector (random, zeros …)

For τ from 0 to convergence:

$$\theta^{(\tau+1)} \leftarrow \theta^{(\tau)} - \rho(\tau)\left(\frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \mathbf{L}_i(\theta)\Big|_{\theta=\theta^{(\tau)}}\right)$$

Stochastic Gradient Descent

$\theta^{(0)} \leftarrow$ initial vector (random, zeros …)

Very Similar Algorithms

For τ from 0 to convergence:

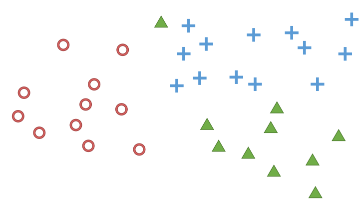
$\mathcal{B} \sim$ Random subset of indices

$$\theta^{(\tau+1)} \leftarrow \theta^{(\tau)} - \rho(\tau)\left(\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \nabla_\theta \mathbf{L}_i(\theta)\Big|_{\theta=\theta^{(\tau)}}\right)$$

Python Demo!

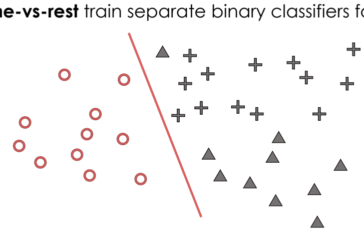
Multiclass (more than 2) Classification

- **One-vs-rest** train separate binary classifiers for each class



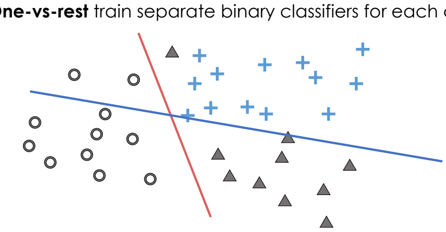
Multiclass (more than 2) Classification

- **One-vs-rest** train separate binary classifiers for each class



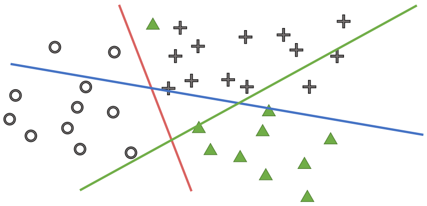
Multiclass (more than 2) Classification

- **One-vs-rest** train separate binary classifiers for each class



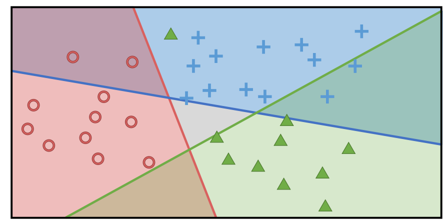
Multiclass (more than 2) Classification

- **One-vs-rest** train separate binary classifiers for each class



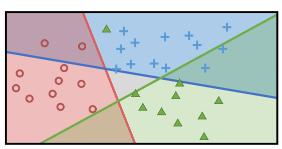
Multiclass (more than 2) Classification

- **One-vs-rest** train separate binary classifiers for each class



Multiclass (more than 2) Classification

- **One-vs-rest** train separate binary classifiers for each class
  - Class with highest confidence wins
  - Need to address class imbalance issue



- **Soft-Max** multiclass classification

- **Soft-Max** multiclass classification

$$\mathbf{P}(Y = j \mid x) = \frac{\exp\left(x^T \theta^{(j)}\right)}{\sum_{m=1}^{k} \exp\left(x^T \theta^{(m)}\right)}$$

- Separate $\theta^{(j)} \in \mathbb{R}^p$ for each class
- Trained using gradient descent methods
- Over parameterized. Why?
  - k sets of parameters one for each class
  - Only need k-1 parameters

$$\mathbf{P}(y = k \mid x) = 1 - \sum_{j=1}^{K} \mathbf{P}(y = j \mid x)$$

  - Often use k parameters + regularization to address "redundancy".

Python Demo!

Deep Learning Overview

Bonus Material

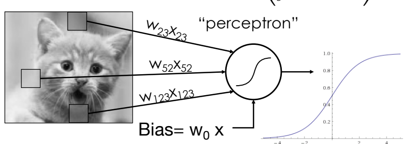
Borrowed heavily from excellent talks by:

- **Adam Coates:** http://ai.stanford.edu/~acoates/coates_dltutorial_2013.pptx
- **Fei-Fei Li and Andrej Karpathy:** http://cs231n.stanford.edu/syllabus.html

Logistic Regression as a "Neuron"

- Consider the simple function family: $\sigma(u) = \dfrac{1}{1+\exp(-u)}$

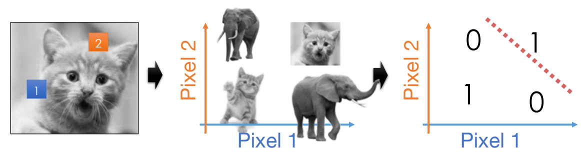
$$f_w(x) = \sigma\left(w^T x\right) = \sigma\left(\sum_{j=1}^{d} w_j x_j\right) = P(y = 1 \,|\, x)$$



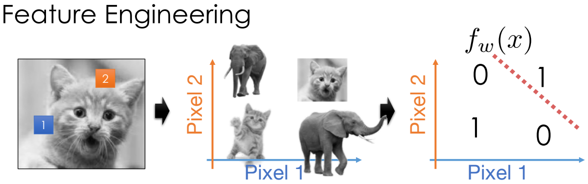
Neuron "fires" if weighted sum of input is greater than zero.

Logistic Regression: Strengths and Limitations

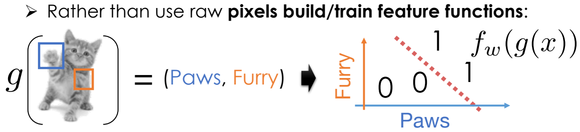
- Widely used machine learning technique
  - convex → efficient to learn
  - easy to interpret model weights
  - works well given good features
- Limitations:
  - Restricted to linear relationships → sensitive to choice of features



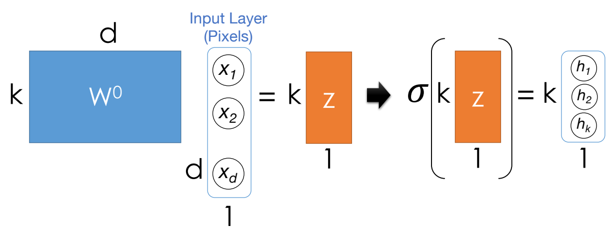
Feature Engineering



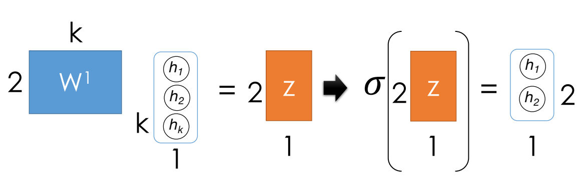
- Rather than use raw pixels **build/train feature functions**:



Composition Linear Models and Nonlinearities



Composition Linear Models and Nonlinearities

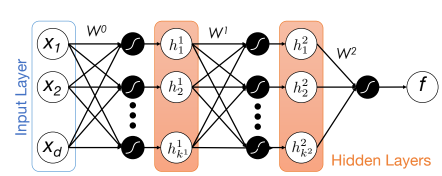


Neural Networks

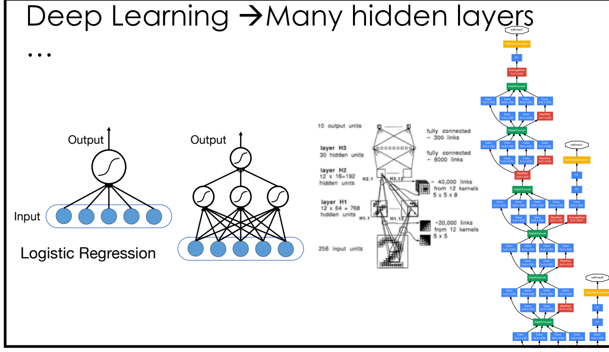
- Composing "perceptrons"

$$x \rightarrow \sigma(W^0 x) \rightarrow h^1 \rightarrow \sigma(W^1 h^1) \rightarrow h^2 \rightarrow \sigma(W^2 h^2) \rightarrow f$$

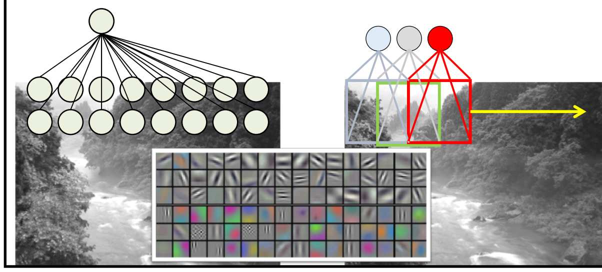
$$y = f_{W^0, W^1, W^2}(x) = \sigma\left(W^2\, \sigma\left(W^1\, \sigma\left(W^0 x\right)\right)\right)$$



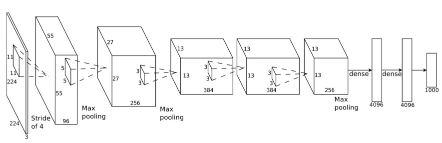
Deep Learning → Many hidden layers …



Convolutional Neural Networks: Exploiting Spatial Sparsity

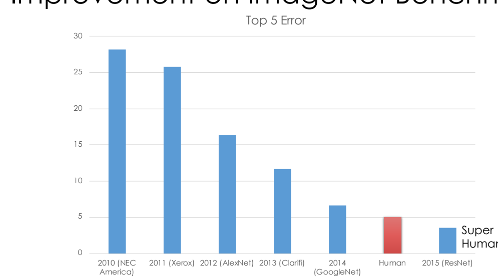


Example: AlexNet (Krizhevsky et al., NIPS 2012)

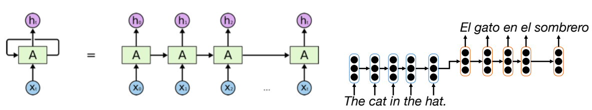


- Introduced in 2012, significantly outperformed state-of-the-art (top 5 error of 16% compared to runner-up with 26% error)

Improvement on ImageNet Benchmark

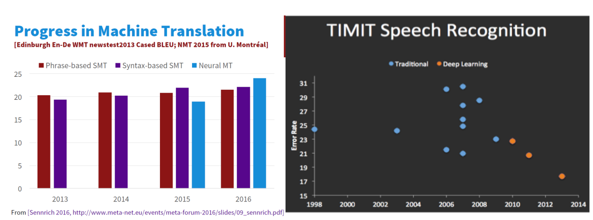


Recurrent Neural Networks: Modeling Sequence Structure

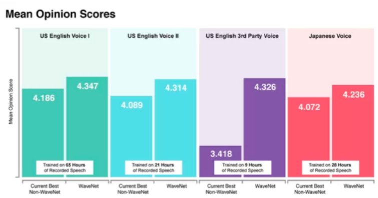


- input + previous output → new output
- State of the art in modeling sequential data
  - speech recognition and machine translation

Improvements in Machine Translation & Automatic Speech Recognition



State of the art in Text to Speech (TTS)



Interested in Deep Learning?

- RISE Lab Deep Learning Overview:
  - https://ucbrise.github.io/cs294-rise-fa16/deep_learning.html
- TensorFlow Python Tutorial
- Stanford CS231 Labs
  - http://cs231n.github.io/linear-classify/
  - http://cs231n.github.io/optimization-1/
  - http://cs231n.github.io/optimization-2/