Lecture 4 Optimization for Machine Learning

Instructor: Lei Li, **Yu-Xiang Wang**

Recap: Supervised learning

- Data $(x_1, y_1), ..., (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$
- Hypothesis $h : \mathcal{X} \to \mathcal{Y}$ from $\mathcal{H}$
- Loss function $\ell(h, (x, y))$

- Example:
  - Linear regression
  - Linear classifiers
  - Decision tree classifiers

Recap: Unsupervised learning

- Data $x_1, ..., x_n \in \mathcal{X}$
- Hypothesis $h \in \mathcal{H}$
- Loss function $\ell : \mathcal{H} \times \mathcal{X} \to \mathbb{R}$

- Examples:
  - k-means clustering
  - PCA

Recap: Risk, Empirical Risk

- Loss function

$$\ell(h, (x, y))$$

- Risk function

$$R(h, \mathcal{D}) = \mathbb{E}_{\mathcal{D}}[\ell(h, (x_i, y_i))]$$

- Empirical risk

$$\hat{R}(h, \text{Data}) = \frac{1}{n} \sum_{i=1}^{n} \ell(h, (x_i, y_i))$$

Recap: "One algorithm that rules them all" --- Empirical Risk Minimization

- ERM

$$\hat{h} = \arg\min_{h \in \mathcal{H}} \hat{R}(h, \{(x_i, y_i) | i \in [n]\})$$

- Regularized ERM

$$\hat{h} = \arg\min_{h \in \mathcal{H}} \hat{R}(h, \mathrm{Data}) + g(h)$$

This lecture

- Risk bound for general bounded loss functions
- Model selection
- Optimization methods for machine learning

Recap: Theorem for (fixed design) linear regression

Theorem: Assume (A1) and (A2), the ordinary least square estimator for linear regression satisfies:

$$\mathbb{E}[R(\hat{\theta})] - R(\theta^*) \leq \frac{d\sigma^2}{n}$$

- Observations:
  - No assumptions on the design matrix X
  - The bound is exactly tight:
    - Recall that $\mathbb{E}[R(\hat{\theta})] - R(\theta^*) = \mathbb{E}[\|\hat{\theta} - \theta^*\|^2_{X^T X}]$
    - When X = I, $\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} y_i \sim \mathcal{N}(\theta^*, \sigma^2 I_d)$

The result relies on strong assumptions on how the data is generated

- e.g., it does NOT apply to the case for fitting a polynomial to a noisy sine function we gave earlier!

- The **statistical learning problem**:
  - Assumption B1: iid samples
  - Assumption B2: Bounded loss function
  - Assumption B3: Finite hypothesis class

The goal again is to bound the **excess risk .** This time we want a high probability bound.

- With probability at least $1 - \delta$

$$R(\hat{h}) - R(h^*) \leq \epsilon$$

- Parameterize ϵ as a function of
  - Number of data points
  - Size of the hypothesis class
  - Boundedness of the loss
  - Failure probability

Introducing two powerful "hammers": Hammer 1. Hoeffding's inequality

Theorem D.2 (Hoeffding's inequality) *Let $X_1, \ldots, X_m$ be independent random variables with X_i taking values in $[a_i, b_i]$ for all $i \in [m]$. Then, for any $\epsilon > 0$, the following inequalities hold for $S_m = \sum_{i=1}^m X_i$:*

$$\mathbb{P}[S_m - \mathbb{E}[S_m] \geq \epsilon] \leq e^{-2\epsilon^2 / \sum_{i=1}^m (b_i - a_i)^2} \tag{D.4}$$

$$\mathbb{P}[S_m - \mathbb{E}[S_m] \leq -\epsilon] \leq e^{-2\epsilon^2 / \sum_{i=1}^m (b_i - a_i)^2} . \tag{D.5}$$

(see Appendix D.1 of FML textbook for a proof)

Roughly saying that the **empirical averages** of independent random variable converges to the **mean** at a O(1/sqrt(n)) rate, with high probability.

Introducing two powerful "hammers": Hammer 2. Union bound

Lemma (Union bound): For any probability distribution and any event E1, E2:

$$\mathbb{P}[E_1 \cup E_2] \leq \mathbb{P}[E_1] + \mathbb{P}[E_2]$$

Now let's apply these two hammers to solve statistical learning

1. For each hypothesis h, apply Hoeffding's inequality

2. Union bound over all hypothesis

Now let's apply these two hammers to solve statistical learning

Theorem: Assume (B1),(B2) and (B3), with probability at least $1-\delta$ (over the distribution of the data), ERM satisfies

$$R(\hat{h}) - R(h^*) = O\left(\sqrt{\frac{\log|\mathcal{H}| + \log(1/\delta)}{n}}\right)$$

Quiz 2: Application to decision tree classifier

- **d**-dimensional discrete feature (**L**-levels for each)
- **H**-layer decision tree, binary decision in one layer
- **K** Labels
- *Upper bound of the size of hypothesis class?*

Quiz 3: Application to generic classification (no restriction on the hypothesis class)

- **d**-dimensional discrete feature (**L**-levels for each)
- **K** labels
- ***Total number of unique classifiers?***

Computation-approximation tradeoff in the choice of hypothesis class

	model $p^*(y\|x)$	**Linear learners**	**Neural networks**
Computation	Depends on how complex p* is	Efficient	Not efficient in the worst case, but…
Approximation	No approximation	Large approx. error	Small approx. error
Statistical efficiency	Depends on how complex p* is	Need less data	Need more data

“All models are wrong, but some are useful.”

George Box
(1919 - 2013)

Checkpoint: Theory of learning

- Risk bounds for linear regression model

$$\mathbb{E}[R(\hat{\theta})] - R(\theta^*) \leq \frac{d\sigma^2}{n}$$

- Risk bounds for a general learning problem with bounded loss

$$R(\hat{h}) - R(h^*) = O\left(\sqrt{\frac{\log|\mathcal{H}| + \log(1/\delta)}{n}}\right)$$

- Observations:
  - Not directly comparable for several reasons
  - Strong assumption => Strong results
  - Weak assumption => Weak results

This lecture

- Risk bound for general bounded loss functions
- Model selection
- Optimization methods for machine learning

Typical problems in model selection

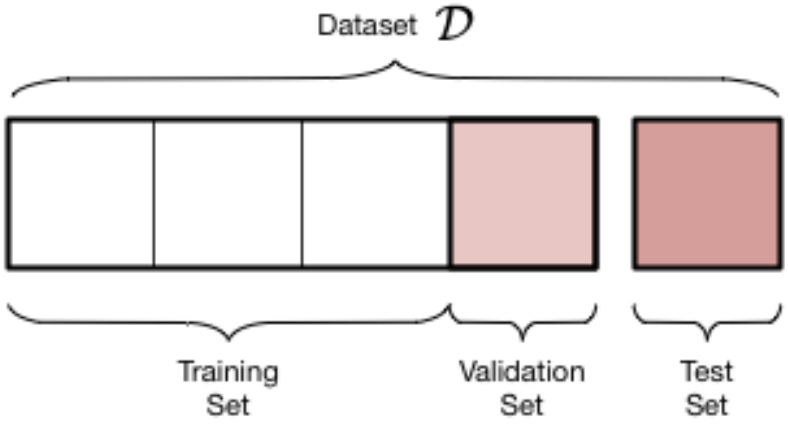
- Choose hypothesis class
  - Decision tree? Linear classifier? Or neural networks?
- Choose hyperparameters
  - Depth of decision tree
  - Regularization weights for Ridge / Lasso
- Choose which set of features of include

Model selection is challenging because we do not observe the actual *risk*!

- Empirical risk is often a poor surrogate due to the optimization bias
  - Example: 1-Nearest Neighbor classifier

- Two ideas for estimating the risk
  - Calculate or bound the actual risk in theory
  - Simulate the actual risk on a dataset not used for training.

Empirically measuring the *Risk* by splitting the data into: Training, Test, and Validation Sets



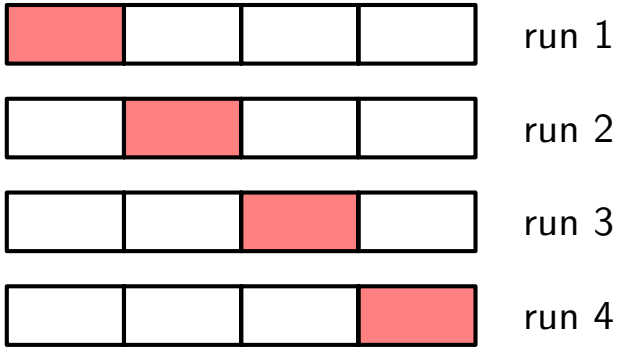
Validation set is used for model-selection:

- choosing decision tree vs. linear classifier
- Select features, tune hyperparameters

Test set is used only once to report the final results.

Cross-validation

Figure 1.18 The technique of S-fold cross-validation, illustrated here for the case of $S = 4$, involves taking the available data and partitioning it into S groups (in the simplest case these are of equal size). Then $S - 1$ of the groups are used to train a set of models that are then evaluated on the remaining group. This procedure is then repeated for all S possible choices for the held-out group, indicated here by the red blocks, and the performance scores from the S runs are then averaged.



- Pros:
  - No assumption on the data generating distributions, except iid.
  - Do not waste data, comparing to holdout.

- Cons:
  - It evaluates the model applying to (S-1)/S fraction of the data
  - Computation cost = O(S * number of models to select from)

Other approaches for model selection

- AIC (Akaike Information Criteria) / BIC (Bayesian information criteria)
  - (see PRML Section 1.3 and 4.4.1)

- Effective degree of freedom
  - Measuring the effective number of parameters
  - For fixed-design regression with square loss + Gaussian noise, any estimator:

$$R(\hat{h}) - \mathbb{E}[\hat{R}(\hat{h})] = \frac{2\sigma^2}{n} df(\hat{h})$$

So if one can estimate df, then can use it for model selection

Effective degree of freedom for Regularized Linear Regression

- Ridge regression

$$df(X\hat{\theta}) = \mathrm{tr}(X(X^T X + \lambda I)^{-1} X^T)$$

  - Number of parameters, if no regularization
  - Independent to data y, can be computed ahead of time

- Lasso $$df(X\hat{\theta}) = \mathbb{E}\left[\sum_{j\in[d]} \mathbb{I}(\hat{\theta}_j \neq 0)\right]$$

  - Expected number of non-zero weights -- Sparsity.
  - This is truly remarkable that we get this via L1-regularization

See e.g. : https://www.stat.cmu.edu/~ryantibs/papers/lassodf.pdf

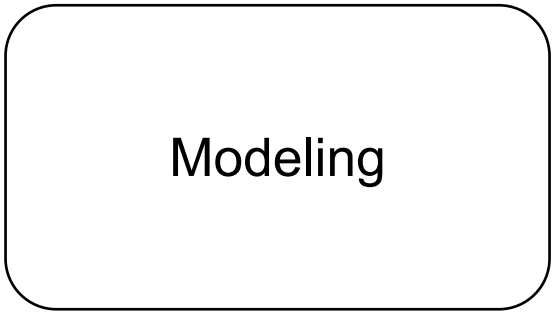
Checkpoint: model selection

- Three approaches for model selection
  - Holdout
  - Cross validation
  - Penalize information criteria
- Cross validation is what is most commonly used in practice.

Remainder of this lecture

- Risk bound for general bounded loss functions
- Model selection
- Optimization methods for machine learning

How do we solve ERM?



- Feature engineering
- Specify a family of classifiers

Apply the classifier to emails

Learning the best performing classifier

Recap: Linear classifiers

- Score(x) = $w_0 + w_1 * \mathbf{1}$(hyperlinks) + $w_2 * \mathbf{1}$(contact list) + w_3 * misspelling + w_4 * length

- A linear classifier: h(x) = 1 if Score(x) > 0 and 0 otherwise.

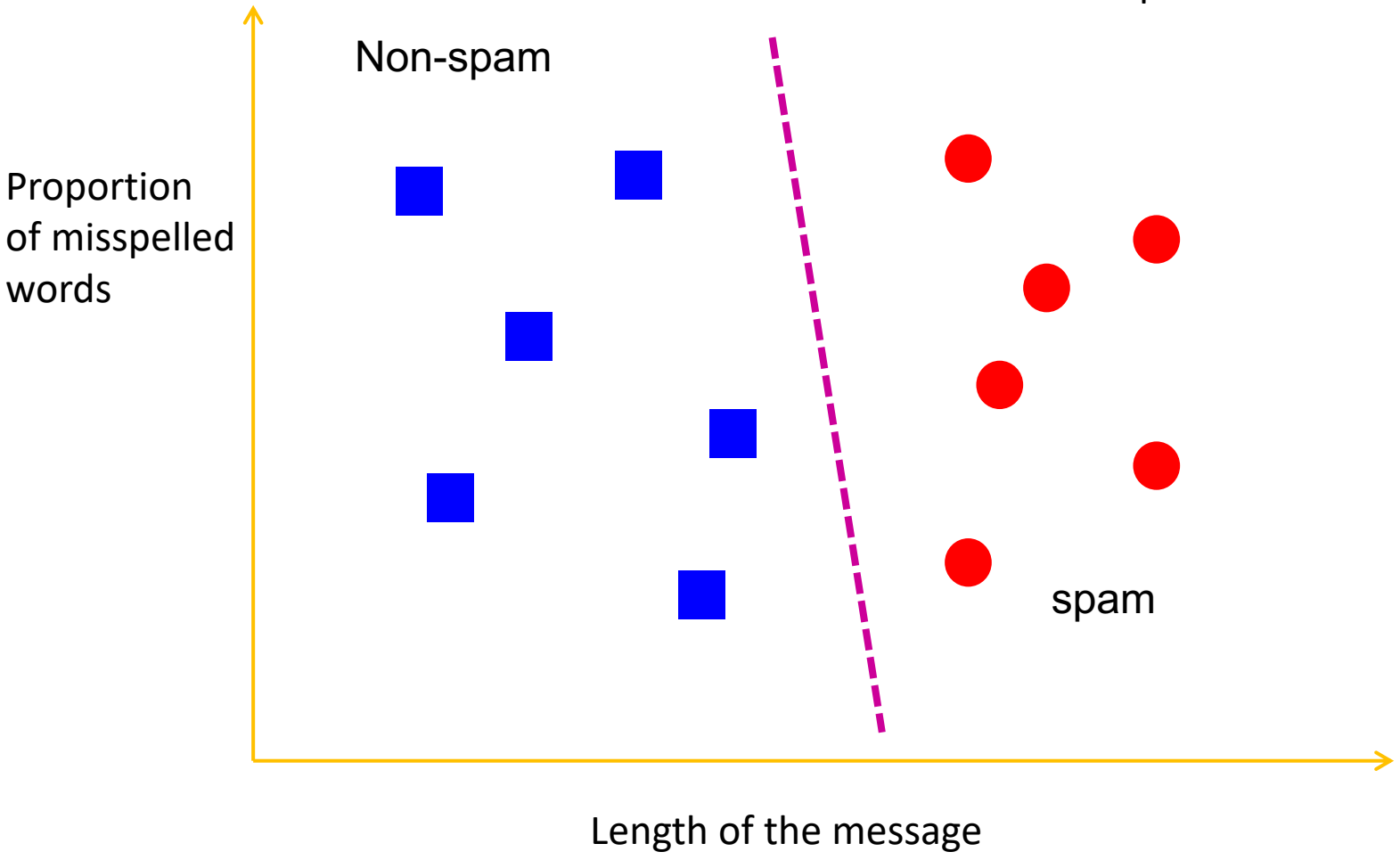
- Question: What are the "free-parameters" in a linear classifier?
  - If we redefine $\mathcal{Y} = \{-1, 1\}$
  - A compact representation:

$$h(x) = \text{sign}(w^T[1; x])$$

Recap: Geometric view: Linear classifier are “half-spaces”!

$\{ x \mid w_0 + w_1 * x1 + w_2 * x2 + w_3 * x3 + w_4 * x4 > 0 \}$

The set of all ”emails” that will be classified as “Spams”.



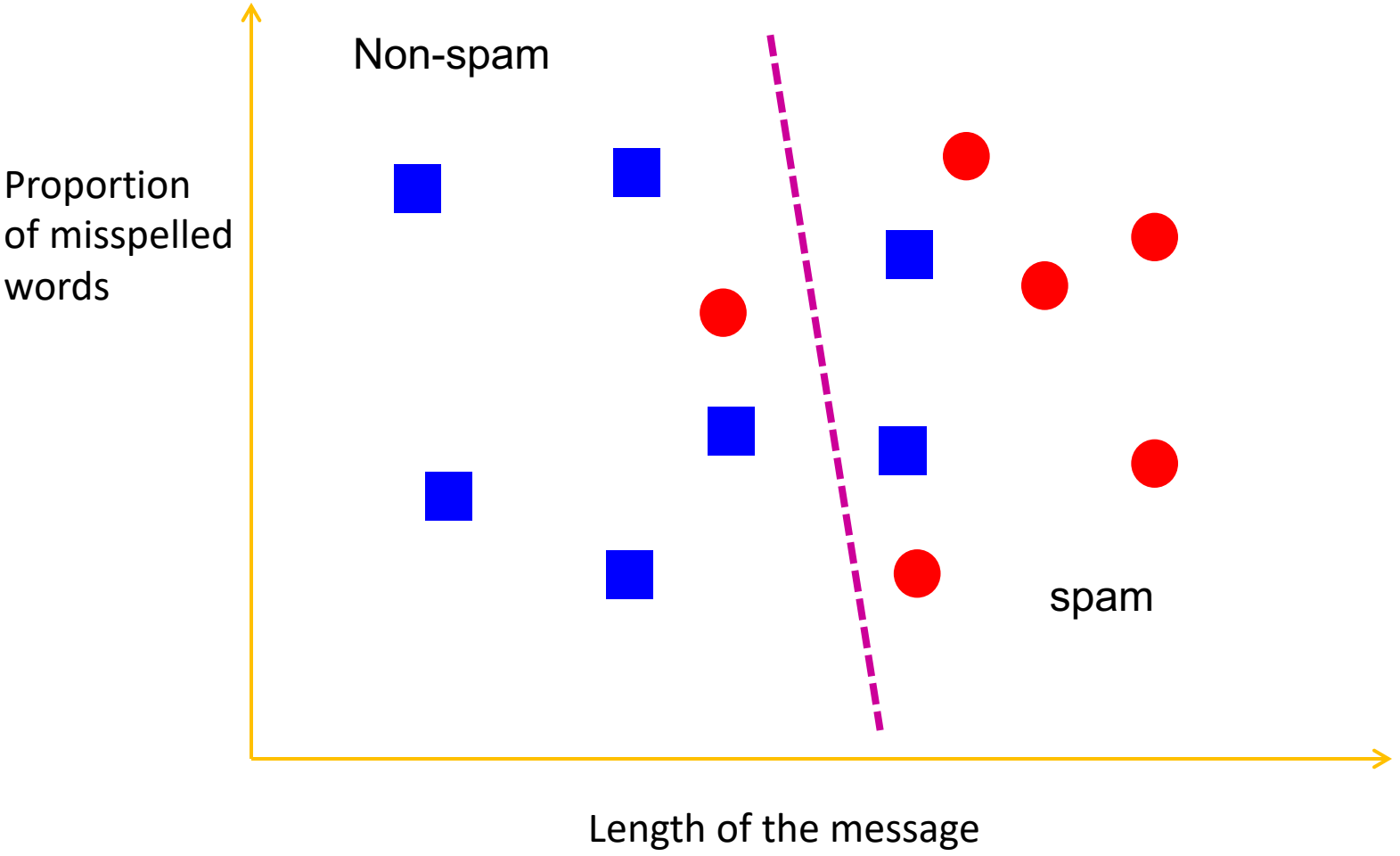
In the case when the training data is linearly separable, there is a polynomial time algorithm.

- Why?
  - Easiest way to see it is that it is a **linear program**.
  - Polynomial time algorithm exists for all LPs.

$$\begin{aligned}
&\text{find } w \in \mathbb{R}^d \\
&\text{subject to:} \\
&\quad w^T x_i > 0 \; \forall i \in \{1, 2, ..., n\} \text{ s.t.} y_i = 1 \\
&\quad w^T x_i \leq 0 \; \forall i \in \{1, 2, ..., n\} \text{ s.t.} y_i = -1
\end{aligned}$$

Also, check out the **Rosenblatt (1958)'s Perceptron algorithm** from PRML 4.1.7, as well as its "mistake" bound analysis in FML-8.3.1.

Best linear separator in general (linearly non-separable cases) is NP-hard.



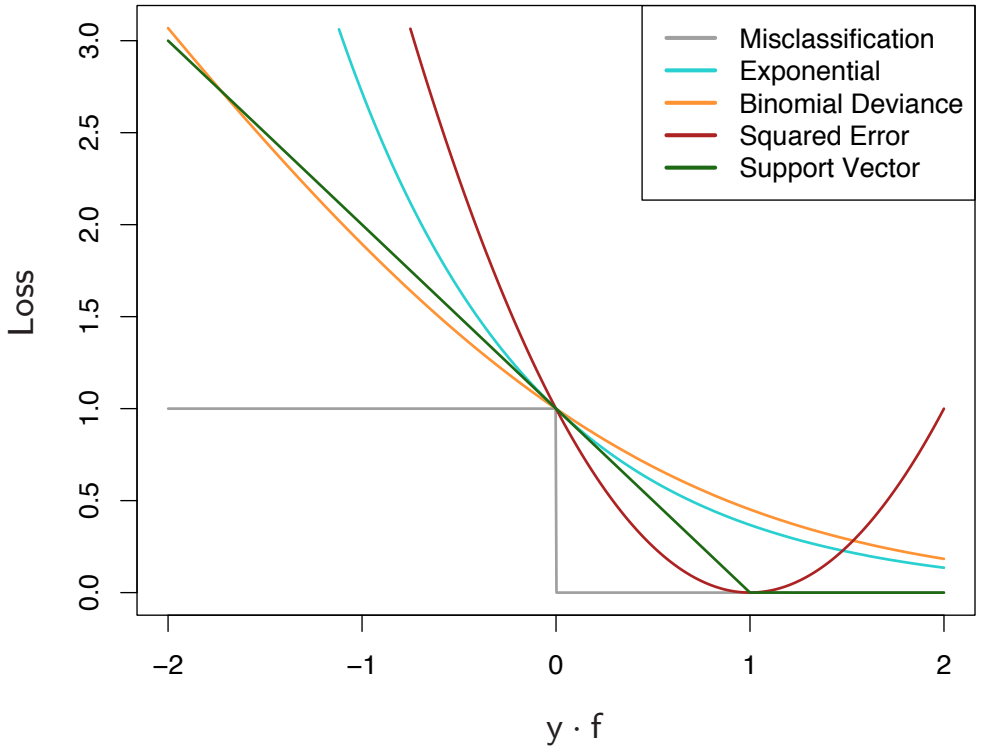
Just “relax”: relaxing a hard problem into an easier one

$$\min_{w \in \mathbb{R}^d} \text{Error}(w) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(\text{sign}(w^T x_i) \neq y_i)$$

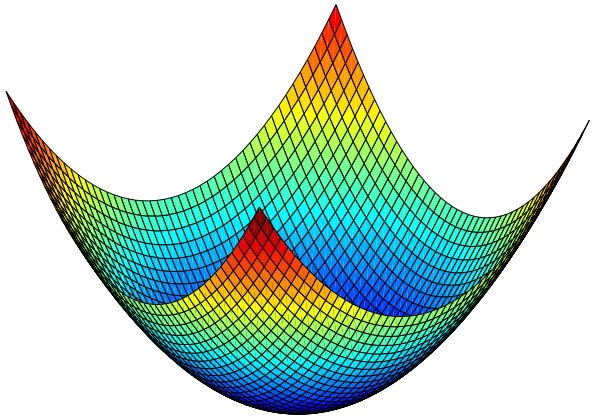
$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(w^T x_i, y_i).$$

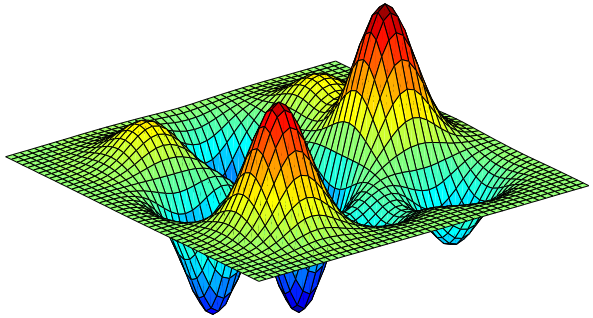
Why are “surrogate losses” easier to minimize?

- They are continuous.
- Differentiable (except hinge loss).
- Convex.



Convex vs Nonconvex optimization





- Unique optimum: global/local.

- Multiple local optima
- In high dimensions possibly exponential local optima

* Be careful: The surrogate loss being convex does not imply all ML problems using surrogate losses are convex. Linear classifiers are, but non-linear classifiers are usually not. Take “convex optimization” to know more.

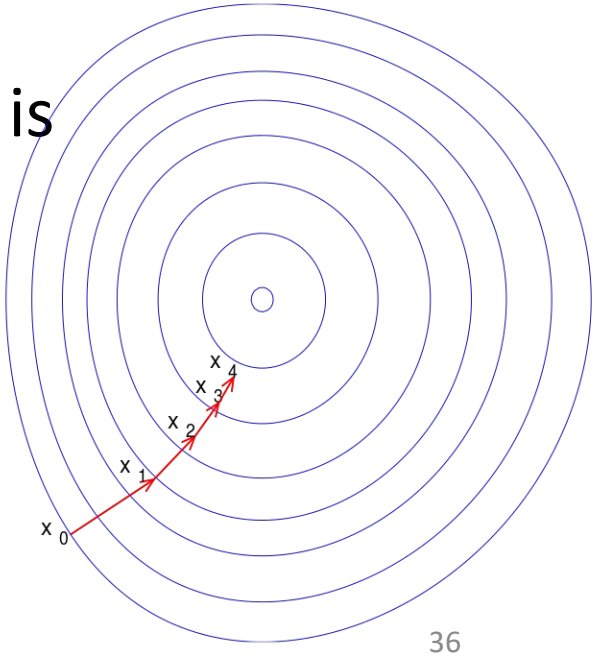
How do we optimize a continuously differentiable function in general?

- The problem: $\min_{\theta} f(\theta)$

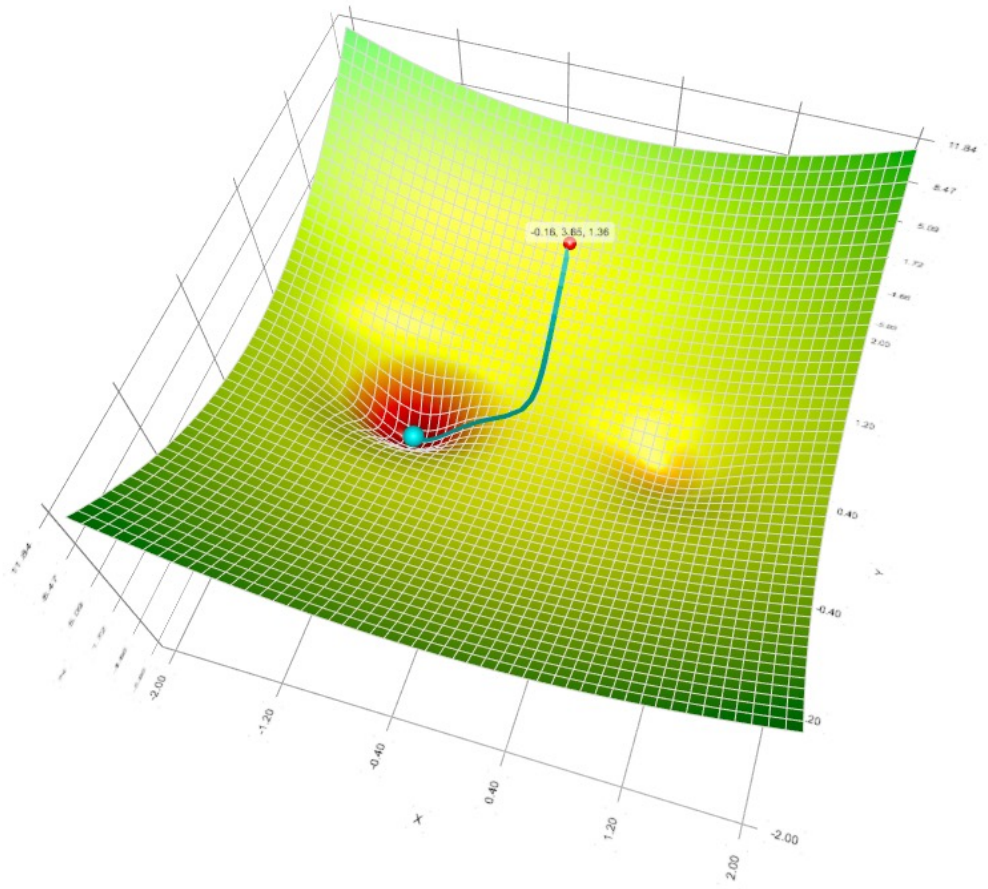
- Let's just optimize it anyway!
  - With gradient descent.

- Assumption: The objective function is differentiable almost everywhere.

$$\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t)$$



Gradient Descent Demo



- Play with this excellent tool yourself to build intuition:

https://github.com/lilipads/gradient_descent_viz

Gradient of logistic loss for learning a linear classifier

- The function to minimize is

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i \cdot x_i^T w))$$

- How to calculate the gradient?
  - Take out a piece of paper and work on it!
  - (you have 3 min)

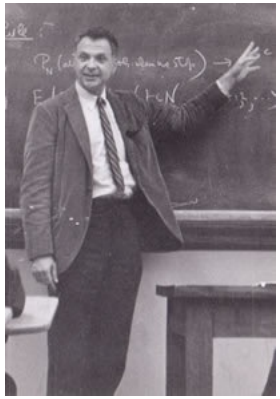
Hint:
- Apply the chain rule.
- d log(x) / dx = 1/x
- d exp(x) / dx = exp(x)

Gradient of logistic loss for learning a linear classifier

$$\nabla f(w) = \frac{1}{n} \sum_{i=1}^{n} \frac{\exp(-y_i \cdot x_i^T w)}{1 + \exp(-y_i \cdot x_i^T w)} (-y_i x_i)$$

- What is the time complexity of computing this gradient?

Stochastic Gradient Descent (Robbins-Monro 1951)



Herbert Robbins
1915 - 2001

- Gradient descent

$$\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t)$$

- Stochastic gradient descent

$$\theta_{t+1} = \theta_t - \eta_t \hat{\nabla} f(\theta_t)$$

- Using a stochastic approximation of the gradient:

$$\mathbb{E}[\hat{\nabla} f(\theta_t) | \theta_t] = \nabla f(\theta_t)$$

$$\mathbf{Var}[\hat{\nabla} f(\theta_t) | \theta_t] \leq \sigma^2$$

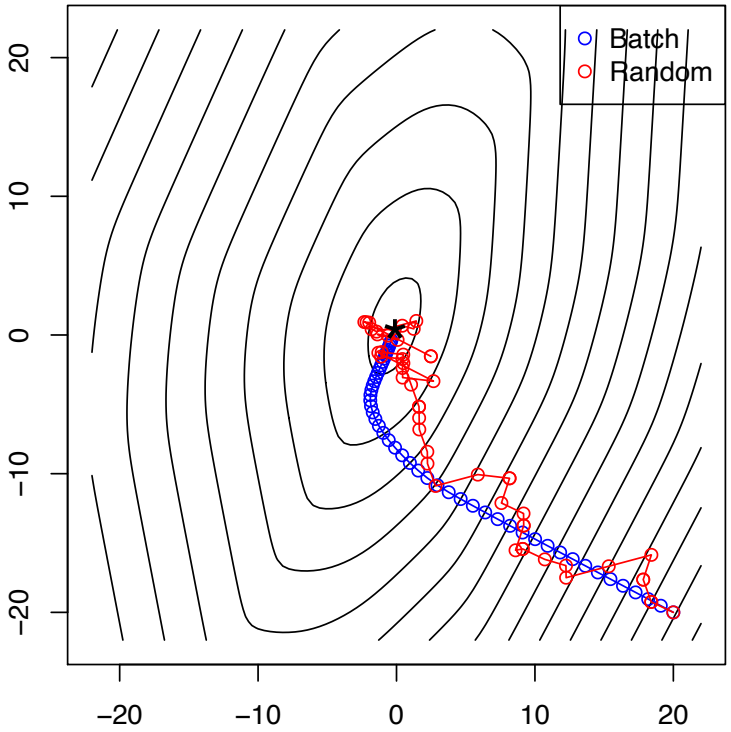
One natural stochastic gradient to consider in machine learning

- Recall that

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, (x_i, y_i))$$

- Pick a **single** data point i uniformly at random
  - Use $\nabla_\theta \ell(\theta, (x_i, y_i))$
  - Show that this is an unbiased estimator!

Illustration of GD vs SGD

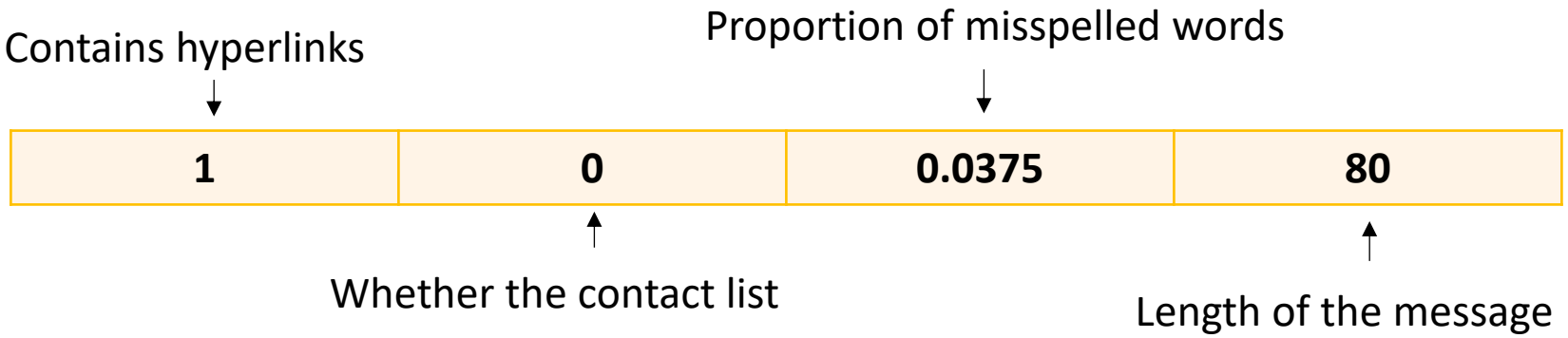


Rule of thumb for stochastic methods:

- generally thrive far from optimum
- generally struggle close to optimum

Observation: With the time gradient descent taking one step. SGD would have already moved many steps.

Intuition of the SGD algorithm on the "Spam Filter" example



- Score(x) = $w_0 + w_1 * 1(\text{hyperlinks}) + w_2 * 1(\text{contact list}) + w_3 * \text{misspelling} + w_4 * \text{length}$

- Meaning of these weight?
  - The more positive, the more we think the feature is associated with Spam email.
  - The more negative, the less that we think the feature is associated with Spam email

Intuition of the SGD algorithm on the "Spam Filter" example

$$\nabla \ell(w, (x_i, y_i)) = \frac{\exp(-y_i \cdot x_i^T w)}{1 + \exp(-y_i \cdot x_i^T w)} (-y_i x_i)$$

Scalar > 0:
≈ 0 if the prediction is correct
≈ 1 otherwise

Vector of dimension d: provides the direction of the gradient

If we receive an example [1, 0 , 0.0375, 80] like the one before.
And a label y = 1 saying that this is a spam.

How will the SGD update change the weight vector?

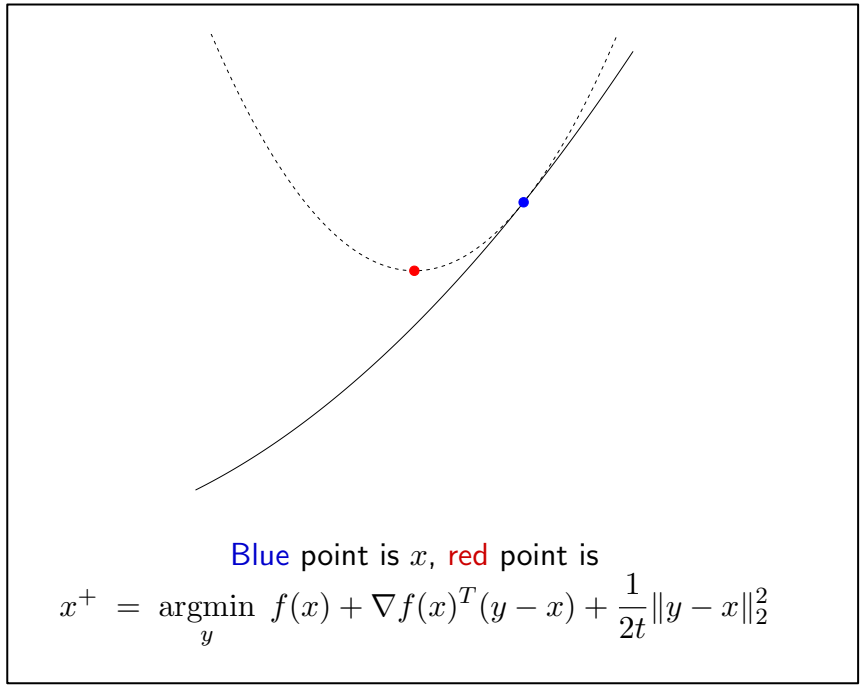
Then by moving w towards the negative gradient direction, we are changing the weight vector by increasing the weights. i.e., increasing the amount they contribute to the score function (if currently the classifier is making a mistake on this example)

Interpretation of Gradient Descent

1. Taylor approximation of the objective function:

$$f(y) \approx f(x) + \nabla f(x)^T (y - x) + \frac{1}{2t}\|y - x\|_2^2$$

2. Choose the next point by minimizing the expansion



Convergence analysis of GD for smooth & non-convex objective

- Problem setting: $\min_{\theta} f(\theta)$
  - Assumption: f is L-smooth (but not necessarily convex)

- How to measure success?
  - ε-stationary point:
  - Iteration complexity:

- Algorithm $\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t)$

Convergence analysis of GD for smooth & non-convex objective

- Descent lemma

- Telescoping

Convergence analysis of SGD for smooth & non-convex objective

- Expected Descent Lemma

- Telescoping

What happens if we assume the objective function is convex?

- One region of solutions, thus stronger goal possible:

- A summary of results (in iteration complexity)

	Convex + G-Lipschitz + Bounded domain	Convex + L-Smooth	μ-strongly convex + L-smooth
GD			
SGD			

How to choose the step sizes / learning rates in practice?

- In practice:
  - Use cross-validation on a subsample of the data.
  - Fixed learning rate for SGD is usually fine.
  - If it diverges, decrease the learning rate.
  - If for extremely small learning rate, it still diverges, check if your gradient implementation is correct.

The power of SGD

- Extremely general:
  - Specify an end-to-end differentiable score function, e.g., a complex neural network.
  - Beyond the context of machine learning
- Extremely simple: a few lines of code.
- Extremely scalable
  - Just a few pass of the data, no need to store the data
- People are continuing to discover that many methods are special cases of SGD.

Gradient Boosting

The Annals of Statistics
2001, Vol. 29, No. 5, 1189–1232

1999 REITZ LECTURE

GREEDY FUNCTION APPROXIMATION: A GRADIENT BOOSTING MACHINE[1]

BY JEROME H. FRIEDMAN

Stanford University

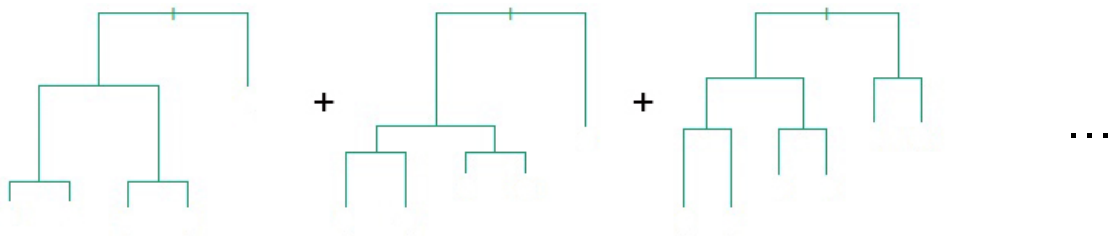
Function estimation/approximation is viewed from the perspective of numerical optimization in function space, rather than parameter space. A connection is made between stagewise additive expansions and steepest-descent minimization. A general gradient descent "boosting" paradigm is developed for additive expansions based on any fitting criterion. Specific algorithms are presented for least-squares, least absolute deviation, and Huber-M loss functions for regression, and multiclass logistic likelihood for classification. Special enhancements are derived for the particular case where the individual additive components are regression trees, and tools for interpreting such "TreeBoost" models are presented. Gradient boosting of regression trees produces competitive, highly robust, interpretable procedures for both regression and classification, especially appropriate for mining less than clean data. Connections between this approach and the boosting methods of Freund and Shapire and Friedman, Hastie and Tibshirani are discussed.

Widely-used packages: *Xgboost, LightGBM*

Gradient Boosting

- Choose your favorite loss function
  - Square loss, Huber loss, Cross-entropy, Hinge loss, etc.
- Score function is a **weighted sum of decision trees.**

$$u_i = \sum_{j=1}^{m} \beta_j \cdot T_j(x_i), \quad i = 1, \dots n$$



- ERM amounts to solving $\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} \ell \left(y_i, \sum_{j=1}^{M} \beta_j T_j(x_i) \right)$

Gradient Boosting as a "Projected" SGD algorithm

$$\min_f \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, f(x_i)\right)$$ with no restriction on f

Take the gradient with respect to the predictions of f

$$d_i = -\frac{1}{n}\left[\frac{\partial \ell(y_i, u_i)}{\partial u_i}\right]_{u_i = u_i^{(k-1)}}, \quad i = 1, ..., n$$

But f needs to be a linear combination of trees, so let's use a tree to approximate the gradient

$$\min_{\text{trees } T} \sum_{i=1}^{n} (d_i - T(x_i))^2$$

Not hard to (approximately) solve for a single tree

Update the current function by gradient descent:

$$f^{(k)} = f^{(k-1)} + \eta_t \cdot T_k$$

Summary

- Risk bound for general bounded loss functions
  - Hoeffding's inequality + Union bound argument
- Model selection
- Optimization methods for machine learning
  - Gradient Descent
  - Stochastic Gradient Descent
  - Convergence analysis
  - Gradient boosting as gradient descent.

Upcoming next: More discriminative modeling

- Thursday: Feedforward neural networks

- Next Tuesday: Convolutional neural networks